ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

LÝ THUYẾT VÀ BÀI TẬP

NGÔN NGỮ LẬP TRÌNH C

SINH VIÊN : HOÀNG VĂN TRỌNG
NGÀY SINH : 27/09/1990
QUÊ QUÁN : Giao Xuân – Giao Thủy – Nam Định
LỚP : K54 Địa Lý
ĐIỆN THOẠI : 0974 971 149
MAIL : hoangtronghus@yahoo.com.vn

Hà Nội 03/11/2013

Lời chia sẻ

Xin chào các bạn!

Trong chương trình đào tạo của các ngành đều có môn *Tin học cơ sở* với mục đích giúp chúng ta nắm được kiến thức cơ bản nhất về xử lý dữ liệu bằng máy tính. Một trong những dạng xử lý đó chính là lập trình để giải quyết một bài toán cụ thể.

Thuật ngữ "*lập trình*" có thể khiến một số bạn cảm thấy hơi trừu tượng và khó hiểu, cộng thêm được nghe đồn từ một số người đi trước là môn này khó học nên lại càng hoang mang. Thực ra, bản chất của lập trình chính là chúng ta viết ra một bản kế hoạch để thực hiện công việc A nào đó sao cho chúng ta có thể hiểu được các bước cần thực hiện được đề ra trên bản kế hoạch, đồng thời máy tính cũng phải hiểu được kế hoạch cần thực hiện những bước gì và cuối cùng cho ra kết quả mà ta mong muốn.

Vì sao chúng ta phải học lập trình? Thực tế chúng ta gặp phải rất nhiều các bài toán phức tạp và có khối lượng tính toán lớn. Nếu chuyển các bài toán này cho máy tính làm thì chỉ mất một đến vài giây trong khi nếu giải bằng tay thì có thể mất đến vài ngày thậm chí nhiều hơn nữa. Các chương trình ứng dụng như Word, Excel hay trang web tìm kiếm nổi tiếng Google.com,…thì cũng đều có bản chất chung là lập trình. Trong trường hợp ta muốn giải quyết bài toán chuyên môn đặc thù nhưng lại không có sẵn chương trình để thực hiện hoặc tiền mua phần mềm quá đắt thì chỉ còn cách phải tự xây dựng cho mình một chương trình để tính toán $\Rightarrow$ cần thiết phải học lập trình.

Tuy nhiên, hiện nay có rất nhiều ngôn ngữ lập trình (C, Java, Pascal, VBA, PHP, Android,…) cũng như có nhiều loại lập trình (lập trình hệ điều hành, lập trình web, lập trình cho điện thoại di động, tủ lạnh, điều hòa,…) thế thì *tại sao lại lựa chọn lập trình C?* Ngôn ngữ C được chọn cho người bắt đầu làm quen với lập trình vì một số lý do sau:

* *C là một ngôn ngữ thể hiện tính có cấu trúc khá rõ nét* (thực hiện theo trình tự, rẽ nhánh, vòng lặp), điều này cũng tương tự như ở Pascal.

* *C là một ngôn ngữ linh hoạ*t, một ký tự nào đó trong chương trình thì tùy vào ngữ cảnh mà nó có ý nghĩa khác nhau. Ví dụ dấu * thì có khi biểu thị phép tính nhân nhưng có khi là phép lấy giá trị,…

* *C có cấu trúc phân nhỏ thành các chương trình con* (hàm). Điều này làm cho chương trình chính trông mạch lạc hơn và dễ phát hiện lỗi để sửa. Một chương trình con có thể được sử dụng nhiều lần trong chương trình chính.

* *C có thể can thiệp khá sâu vào phần cứng của máy tính* (ví dụ RAM hoặc ổ đĩa cứng) nên nó được sử dụng rộng rãi để lập trình hệ điều hành, điển hình như Linux. C chạy tốt trên nhiều loại máy tính khác nhau.

* Cũng vì khả năng can thiệp sâu vào phần cứng mà C nổi trội hơn nhiều ngôn ngữ lập trình khác và *rất mạnh mẽ khi thao tác trên địa chỉ của dữ liệu* (người ta hay gọi là lập trình với biến con trỏ). Nếu bạn nào có ý định viết virus thì C cũng là một lựa chọn khá tốt (mình không khuyến khích viết virus đâu nhé!),…

Chính những lý do đó mà người bắt đầu làm quen với lập trình thì nên học ngôn ngữ C trước. Khi đã nắm vững ngôn ngữ C rùi thì đó là một thế mạnh và là nền tảng để sau này bạn tiếp tục học với ngôn ngữ bậc cao hơn (C#, Java, Android, IOS,…). Đối với một số bạn thì C là môn cơ sở cho các môn chuyên ngành khác: *Lý thuyết đồ thị, Vật lý tính toán*,…các môn này sử dụng ngôn ngữ C để thực hành.

♣ Chúng ta cần chuẩn bị những điều kiện gì để học tốt môn lập trình nói chung cũng như lập trình C nói riêng?

+ Thứ nhất và quan trọng nhất vẫn là niềm *đam mê*! Học lập trình để làm gì? Có bạn học lập trình với mục tiêu là tự mình viết một phần mềm ứng dụng, có bạn muốn học lập trình với mục đích thương mại,…Đối với mình thì mình thích học lập trình là do muốn tìm hiểu để viết một chương trình virus trêu bạn bè tí (không có ý xấu đâu). Mục tiêu và niềm đam mê sẽ tạo cho bạn một động lực mạnh mẽ nhất để bạn tìm hiểu và tự học một cách nghiêm túc và có hiệu quả, có thể có những nội dung thầy cô không dạy nhưng các bạn vẫn tìm hiểu vì nó hữu ích cho bạn và tất nhiên khi bạn nắm vững kiến thức thì chuyện thi cử lấy điểm cao sẽ là đơn giản.

Còn nếu mục đích chỉ đơn thuần là học để đi thi lấy điểm cao không thôi thì có thể điểm sẽ cao (thậm chí là 10.0 nếu thầy cô cho đề dễ) nhưng những gì bạn nhận được sau môn học này cũng bình thường thôi, có nhiều nội dung bạn không cần quan tâm vì nằm ngoài nội dung ôn thi, có nhiều lệnh đơn giản nhưng bạn không cần hiểu bản chất của nó,…Điều đó sẽ là một trở ngại lớn nếu sau này bạn muốn học thêm ngôn ngữ lập trình khác cũng như viết một chương trình ứng dụng để giải bài toán thực tế.

+ Thứ hai, do chương trình và máy tính có mối quan hệ mật thiết với nhau nên trước khi học lập trình bạn nên *xem lại phần cấu trúc máy tính cũng như nguyên tắc hoạt động của máy tính*. Xem lại phần này để biết được những đoạn mã lệnh mà ta viết ra thì khi vào máy tính nó sẽ biến đổi thành các trạng thái vật lý như thế nào và thông tin được điều khiển bởi bộ phận nào cũng như được xử lý tại đâu trong máy tính,…đặc biệt là các máy tính hiện nay đều làm việc theo nguyên lý Von Neumann (nguyên lý điều khiển bằng chương trình và truy cập, lưu trữ theo địa chỉ). Nguyên lý Von Neumann sẽ giúp bạn hiểu sâu sắc hơn khi học đến phần lập trình với địa chỉ (sử dụng biến con trỏ)…

+ Thứ ba, phải *nhớ các quy tắc sử dụng các kí tự* trong một ngôn ngữ lập trình cụ thể. Ngôn ngữ lập trình được cấu thành từ bộ kí tự nhất định và quy tắc sử dụng các kí tự trong bộ kí tự đó. Vì vậy khi học ngôn ngữ nào thì phải nhớ những kí tự được phép sử dụng trong ngôn ngữ đó cũng như nhớ cách kết hợp các kí tự đó với nhau thành một cấu trúc cụ thể. Đây chính là *ngữ pháp* của một ngôn ngữ lập trình.

+ Thứ tư, do máy tính sẽ thực hiện công việc mà chúng ta lập trình nên chương trình phải có tính chính xác và tính logic cao. *Tư duy logic* sẽ giúp bạn nhanh chóng nắm bắt được ý nghĩa của những dòng mã lệnh. Khi viết ra những dòng lệnh thì bạn hãy tưởng tượng xem những dòng lệnh này tương đương với ngôn ngữ đời thường là gì. Ví dụ khi viết là: `if (a>b) printf("a la so lon");` thì tương đương với ngôn ngữ đời thường là: "*nếu a lớn hơn b thì in ra màn hình dòng chữ: a là số lớn*"

Trước một yêu cầu của bài toán bạn nên nghĩ xem nếu giải bằng tay thì gồm những bước nào. Sau đó áp dụng vào một ngôn ngữ lập trình cụ thể rồi mình mới chuyển các

bước giải bằng tay thành các dòng lệnh tương ứng mà máy tính có thể hiểu được. Nếu học tốt môn Toán thì bạn sẽ có lợi thế trong học lập trình vì Toán vừa giúp bạn tư duy logic vừa giúp bạn nghĩ ra được cách giải khi làm bằng tay.

Ví dụ trước một yêu cầu giải bài toán tích phân trên đoạn [1, 3] của hàm số $y = 3x^2$ thì chúng ta phải biết được định nghĩa tích phân xác định mới có các bước giải.

+ Thứ năm, *chương trình bạn viết nên ngắn gọn và đơn giản* nhất có thể, nhưng cũng phải chính xác và khái quát được mọi trường hợp có thể xảy ra. Bạn đừng nghĩ là một chương trình phức tạp với nhiều dòng lệnh mới thể hiện được trình độ lập trình. Một chương trình ưu việt là ngắn gọn rõ ràng để khi ta đọc lại dễ phát hiện lỗi sai cũng như người khác có thể đọc được mã lệnh do mình viết. Chương trình gọn nhẹ là một tiêu chí để đánh giá thuật toán. Khi viết xong chương trình, bạn nên đọc lại mã lệnh và phải hiểu được ý nghĩa của từng lệnh. Đọc tới đâu lại chuyển sang ngôn ngữ đời thường tới đó.

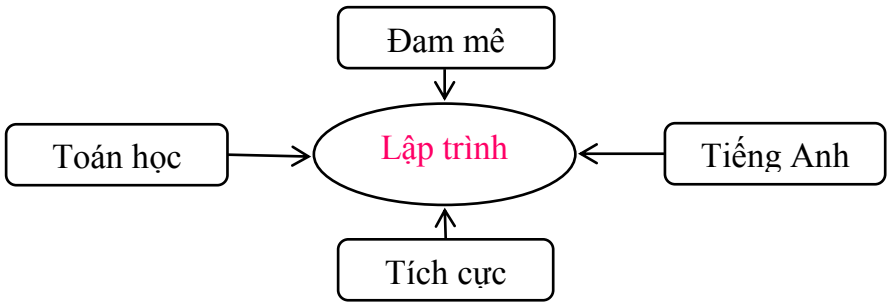
+ Thứ sáu, *chăm chỉ đọc tài liệu và học hỏi kinh nghiệm của những người đi trước*. Học tốt tiếng Anh sẽ giúp bạn tham khảo thêm được nhiều nguồn tài liệu quý báu mà những sách tiếng Việt không có hoặc chưa cập nhật. Tích cực trao đổi với bạn bè và thầy cô sẽ giúp bạn tìm ra được hướng giải quyết cho bài toán hoặc có thêm cách làm hay. Học hỏi trên mạng cũng là một giải pháp mà mình hay áp dụng.

Điều tuyệt đối không nên là sao chép y nguyên các dòng lệnh của người khác sang chương trình của mình mà không hiểu gì về ý nghĩa của từng lệnh. Mong rằng các bạn cũng chỉ tham khảo các dòng lệnh của mình để hiểu nó chứ đừng copy nguyên sang nhé!

+ Thứ bảy, *giúp đỡ người khác là giúp chính mình*. Khi coi vấn đề của người khác là vấn đề của mình thì tự nhiên mình sẽ tích cực hơn trong việc tìm tòi, học hỏi. Có thể ta chưa gặp dạng bài như của người khác thì khi tìm cách giải cho người khác sẽ là lúc ta làm thêm dạng bài mới mà trước đó chưa gặp.

+ Thứ tám, cũng như bao môn học khác là phải chịu khó *thực hành thật nhiều*. Đặc thù của môn này đòi hỏi làm nhiều trên máy thì mới có kinh nghiệm viết mã lệnh, kiểm tra lỗi, sửa lỗi. Làm nhiều bài tập sẽ giúp bạn nhạy bén hơn trong việc kiểm tra ngữ pháp cũng như xác định cách làm cho bài toán. Ban đầu, việc viết chương trình và sửa lỗi khá khó khăn nhưng dần dần chuyện đó không còn là trở ngại nữa, bạn chỉ cần nhìn qua là biết chương trình sai ở đâu.

Ngoài ra, còn nhiều phương pháp học khác tùy thuộc vào từng người. Trong quá trình viết mã lệnh, bạn sẽ ngộ ra nhiều cách học hay mà phù hợp với mình. Tóm lại, lập trình giỏi phụ thuộc vào các yếu tố cơ bản sau:



♣ Tài liệu tham khảo cho môn Lập trình C

+ Trước hết, phải có chương trình dịch ngôn ngữ C ra ngôn ngữ máy. Trường mình và nhiều trường khác sử dụng Dev – Cpp, link download phần mềm:

http://www.mediafire.com/download.php?n2tu8v135qyjo23

Sau khi download phần mềm về máy thì các bạn giải nén (Các bước giải nén: Chuột phải/ Extract files/ Ok) và copy thư mục Dev – Cpp vừa giải nén vào ổ cứng C. Tiếp theo, mở thư mục Dev – Cpp lên rồi kích chuột phải vào biểu tượng devcpp màu xanh/ Send to/ Desktop (create shortcut). Cuối cùng, ra ngoài Desktop rồi kích đúp vào biểu tượng màu xanh devcpp và cứ thế chọn OK cho tới khi cài đặt xong.

+ Quách Tuấn Ngọc, *Ngôn ngữ lập trình C* – NXB Thống kê. Quyển này rất thích hợp khi học phần lý thuyết, đọc dễ hiểu và có thể tự học.

+ Nguyễn Hữu Ngự, *Bài tập Lập trình cơ sở* – NXB Giáo dục. Quyển này rất thích hợp khi học thực hành. Thầy Ngự đã hướng dẫn khá chi tiết về sử dụng thuật toán (hướng dẫn từng bước giải) cho từng bài tập cụ thể, nhiệm vụ của chúng ta chỉ là chuyển nó sang ngôn ngữ C. Hơn nữa, ở phần cuối sách còn có những chương trình đã được viết sẵn để các bạn tham khảo, nó sẽ giúp bạn rèn luyện kỹ năng đọc hiểu mã lệnh do người khác viết.

+ Phạm Văn Ất, *Kỹ thuật lập trình C* – NXB Giao thông vận tải. Quyển này cũng được dùng để học lý thuyết, tác giả viết khá chi tiết. Các bạn có thể download giáo trình ở địa chỉ sau: http://www.mediafire.com/view/?bd25e82c6bmmlbh

+ W. Kernighan and M. Ritchie, *The C programming Language*. Giáo trình này do chính các tác giả của ngôn ngữ C viết ra, tuy bằng tiếng Anh nhưng cũng có thể tham khảo ở một số nội dung. Link download giáo trình:

http://www.mediafire.com/view/?22q84rykeh4bu0l

+ Để biết thêm thông tin về tài liệu tham khảo các bạn có thể xem và thảo luận trong page ĐỀ THI HUS – KHTN HÀ NỘI trên web facebook.com tại link sau:

https://www.facebook.com/media/set/?set=a.428919497180976.104662.420799524659640&type=3

+ Ngoài các giáo trình trên, các bạn có thể học trong tập bài giảng ở địa chỉ sau:

http://www.mediafire.com/download.php?ugxpje9q5sjwxh1

Đây là bài giảng Power Point của tác giả Trần Đăng Hưng mà một số thầy cô sử dụng làm bài giảng cho một số lớp. Học ở bài giảng này có thể nhanh chóng nắm được kiến thức cơ bản trong thời gian ngắn.

♣ Về cấu trúc của file này

Mình sẽ cố gắng đưa vào nhiều nội dung lý thuyết cũng như làm thật nhiều dạng bài tập với mục tiêu xây dựng thành một file lập trình mẫu để thuận tiện cho sau này học những ngôn ngữ lập trình khác. Do là kiến thức cơ sở nên việc hiểu sâu, hiểu đúng bản chất vấn đề sẽ rất cần thiết. File này gồm hai phần chính: *Lý thuyết* và *Bài tập*. Phần Lý thuyết là sự tổng hợp lại những kiến thức cơ bản theo từng chương mục.

Các bài tập mình lấy trong quá trình học cũng như trong giáo trình và bài toán thực tế thuộc các ngành chuyên môn khác nhau. Thường thì một bài sẽ có nhiều cách giải khác nhau nên bạn không nhất thiết phải làm theo cách này hay cách khác. Khi đã hiểu rõ hơn

một chút về lập trình thì bạn sẽ thấy yếu tố quan trọng nhất quyết định đến việc chúng ta có viết được chương trình hoàn chỉnh hay không chính là do hướng làm, là thuật toán. Sau mỗi một bài tập mình có viết thêm phần hướng dẫn cách làm cũng như giải thích ý nghĩa của các biến được sử dụng trong chương trình. Bố cục file bài tập từ đơn giản đến phức tạp và được sắp xếp tương tự như trong giáo trình *Ngôn ngữ lập trình C* của Quách Tuấn Ngọc, cụ thể như sau:

Chương I: Gồm các bài tập cơ bản khi mới làm quen với Lập trình C. Đó là thao tác với biến đơn và sử dụng cấu trúc rẽ nhánh if, switch; cấu trúc vòng lặp for, while,…Trong chương này cần thực hành nhiều hơn với vòng lặp for, while, do while để hiểu rõ cách dùng các vòng lặp đó phục vụ cho học những chương sau.

Chương II: Sử dụng chương trình con (hay còn gọi là *hàm*) để chia chương trình chính ra thành các công việc nhỏ hơn cho dễ kiểm soát lỗi cũng như làm cho chương trình chính mạch lạc hơn. Từ chương này về sau, mình sẽ ưu tiên cho việc sử dụng chương trình con nếu thấy thích hợp.

Chương III: Thao tác trên mảng một chiều với các phần tử đều có cùng kiểu số. Đây là nội dung bắt đầu tiếp cận với dữ liệu kiểu mảng, là việc xử lý một dãy các giá trị chứ không phải là một giá trị đơn thuần như các chương trước.

Chương IV: Thao tác trên mảng nhiều chiều mà chủ yếu là mảng 2 chiều (ma trận). Dạng dữ liệu kiểu ma trận rất hay gặp trong các bài toán chuyên môn, ví dụ để áp dụng vào môn *Đại số tuyến tính và Hình học giải tích*. Vì vậy, cần phải thực hành thật nhiều.

Chương V: Lập trình với biến địa chỉ (con trỏ). Vì đây là thế mạnh của C nên mình sẽ cố gắng làm nhiều bài tập liên quan tới nó. Mình thấy có nhiều bạn ngại học nội dung này vì chưa hiểu địa chỉ là gì, biến con trỏ là gì. Để hiểu bản chất của vấn đề các bạn nên xem lại nguyên lý làm việc của máy tính (nguyên lý Von Neumann). Khi ta khai báo một biến hay một mảng thì biến hay mảng đó sẽ được lưu vào một ô nhớ có chỉ số nhất định trên thanh RAM.

Chương VI: Thao tác với xâu kí tự nhưng thực chất là thao tác trên mảng một chiều mà các phần tử đều có kiểu kí tự.

Chương VII: Kiểu dữ liệu cấu trúc để thể hiện các biến có nhiều thành phần trong đó mỗi thành phần có thể có kiểu dữ liệu khác nhau. Bản chất chính là sự tổ hợp lại của các kiểu dữ liệu cơ bản như: kiểu số, kiểu kí tự, kiểu mảng,...

Chương VIII: Kiểu File rất tiện khi làm việc với khối lượng lớn dữ liệu mà không cần nhập từ bàn phím. Đặc biệt là khi muốn lưu kết quả để làm dữ liệu đầu vào cho bài toán khác. Các đề thi học sinh giỏi và Olympic đều phải thao tác với dữ liệu kiểu File bên cạnh những thuật toán khó.

Chương IX: Phần đồ họa giới thiệu cách sử dụng các hàm dùng để vẽ hình trong C. Nội dung này sẽ tạo ra giao diện đẹp hơn cho bài thực hành. Tuy nhiên, C không mạnh về đồ họa như nhiều ngôn ngữ lập trình khác nên mình chỉ giới thiệu qua và làm một vài bài tập.

Chương X: Một số đề thi đặc biệt như: thi học sinh giỏi, thi Olympic… Các đề thi này mình sưu tập ở các đề học sinh giỏi cấp 3 cũng như thi Olympic sinh viên. Tuy nhiên,

có nhiều đề yêu cầu viết chương trình bằng Pascal hoặc C++, còn mình thì chuyển hết sang viết bằng C.

Chương XI: Một số chương trình virus đơn giản để thấy được sức mạnh can thiệp vào phần cứng máy tính của C, tất nhiên là để hiểu về virus chứ không có mục đích xấu. Ở đây, mình chỉ giới thiệu nội dung của chương trình virus và không hướng dẫn cách cài đặt vào máy tính cũng như cách sử dụng chúng.

Chương XII: Ứng dụng lập trình C để giải một số bài toán thực tế trong các lĩnh vực khác nhau. Trong chương trình tin học ở mức cơ sở thì chúng ta chưa đủ kiến thức để viết phần mềm hệ thống (hệ điều hành) mà chúng ta chỉ có thể viết được những chương trình ứng dụng đơn giản. Khi gặp bất cứ bài toán nào thuộc chuyên môn của bạn mà bạn có ý tưởng chuyển nó sang lập trình thì có nghĩa là bạn đã tìm thấy sự hữu ích từ việc học môn này. Vì vậy, trong chương XII mình sẽ làm những bài toán đặc thù của ngành/ chuyên ngành chứ không làm các bài tương tự mà thầy cô cho thực hành.

♥ Trên đây là chút kiến thức ít ỏi mà mình muốn chia sẻ cùng các bạn. Mình viết file bài tập này trong và sau khi học xong môn THCS 3 nên kiến thức về lập trình C còn hạn chế mong mọi người thông cảm và cho ý kiến đóng góp để mình sửa lại chính xác hơn.

Các bạn có điều gì thắc mắc xin gửi về địa chỉ: hoangtronghus@yahoo.com.vn

Hoặc đăng ý kiến lên page: ĐỀ THI HUS – KHTN HÀ NỘI của web facebook.com để cùng trao đổi và thảo luận.

Hoàng Văn Trọng

In xong 03/11/2013

MỤC LỤC

Chú ý: Những bài đánh dấu * là những bài đã từng thi ở một số lớp khác nhau.

PHẦN A: LÝ THUYẾT

PHẦN B: BÀI TẬP

CHƯƠNG I. BIẾN ĐƠN VÀ CÁC CẤU TRÚC ĐIỀU KHIỂN: IF, SWITCH, FOR, WHILE, DO WHILE, BREAK, CONTINUE, GOTO,…

Bài 1: Viết chương trình in ra màn hình các câu chào khác nhau. Mỗi câu trên một dòng?

```
#include <stdio.h>
#include <conio.h>
main ()
{
    printf("\n  Eku. Chao may!\n");
    getch();
    printf("  Hichic! Chao chu em.\n");
    getch();
    printf("  Thich danh nhau ah?\n");
    getch();
    printf("  Choi luon, khong phai xoan...!\n");
    getch();
}
```

* *Giải thích:*

a) #include <stdio.h> và #include <conio.h> là các thư viện mà trước khi chạy hàm chính (hàm main() là hàm chính) thì chương trình sẽ đọc các thư viện này trước. Khi ta viết tên lệnh ra thì có nghĩa là các phép toán của lệnh đó được lấy trong thư viện tương ứng. Trong đó, ***include*** *nghĩa là đưa thêm vào,* ***stdio*** *nghĩa là standard input/output – vào/ ra chuẩn,* ***conio*** *nghĩa là console input/ output – chứa các hàm vào/ ra như getch(),* ***h*** *nghĩa là header – tiêu đề.*

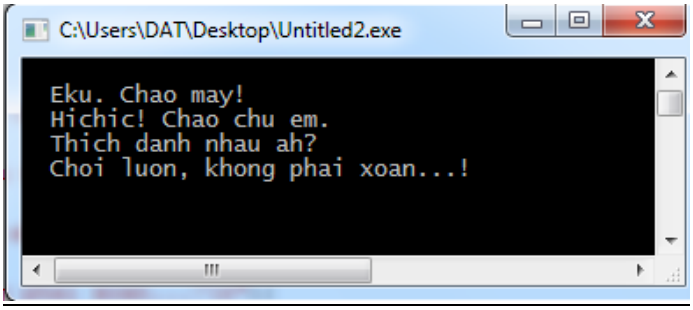
Ngoài các thư viện trên còn có nhiều các loại thư viện khác như: ***math.h*** *chứa các hàm tính toán,* ***string.h*** *chứa các hàm xử lý chuỗi,…*

b) Hàm main() là hàm chính của chương trình và lúc nào cũng phải có. Hàm này không cần thiết phải trả lại giá trị cho tên hàm nên ta không cần viết ***return*** *ở cuối hoặc là chỉ cần viết return 0. Nếu viết return main() thì hàm chính sẽ chạy lại từ đầu.*

c) Lệnh printf() là lệnh in ra màn hình. In những nội dung nằm trong 2 dấu ngoặc kép "", đặc biệt có ký hiệu ***\n*** *không được in ra màn hình mà để báo cho máy hiểu là hãy đưa con trỏ xuống dòng.*

d) Lệnh getch() là lệnh có tác dụng giữ lại màn hình đang hiện ra kết quả. Nếu không có lệnh này thì khi hiện kết quả xong, màn hình sẽ biến mất. Lệnh getch() được lấy ở thư viện conio.h và trong ví dụ trên thì lệnh getch() được xen kẽ giữa các lệnh printf() với ý nghĩa là mỗi lần nhấn Enter sẽ chỉ hiện ra một dòng chào nhau. Nếu không xen kẽ như vậy thì sau khi nhấn Ctrl + F10 để chạy chương trình thì nó sẽ hiện ra một lúc tất cả các dòng chào nhau, như thế nhìn khá rối mắt và không đẹp!

* *Màn hình kết quả như sau:*



Bài 2: Viết chương trình nhập vào một ký tự hoặc một chuỗi ký tự từ bàn phím và in ra màn hình ký tự hoặc chuỗi ký tự vừa nhập? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
    char ChuoiKyTu[20];
    printf("\n  Nhap vao mot ky tu hoac chuoi ky tu:  ");
    gets(ChuoiKyTu);
    printf("  => Ky tu hoac chuoi ky tu do la: %s\n",ChuoiKyTu);
    getch();
    return main();
}
```

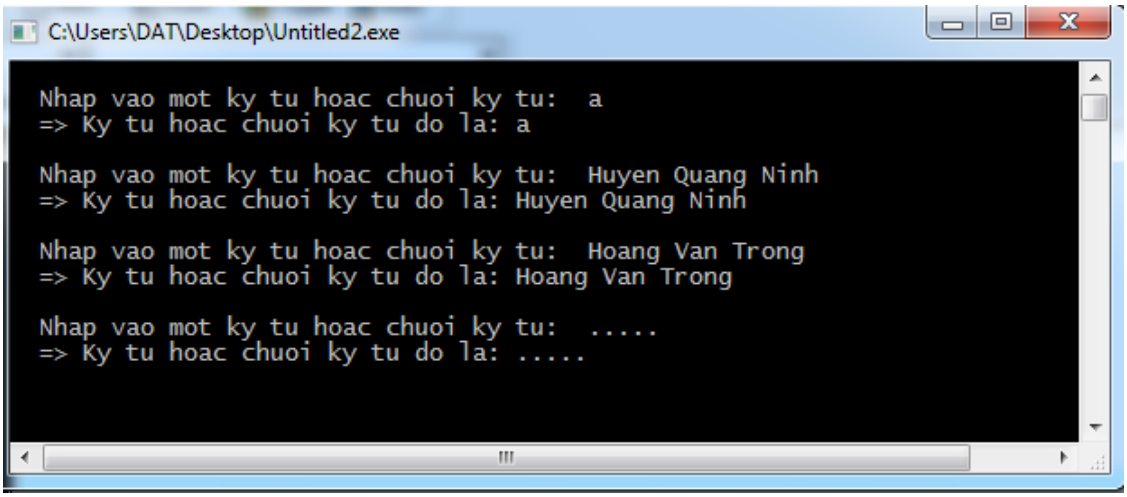
* *Giải thích:*

a) Các biến trong chương trình đều phải khai báo trước mới sử dụng được, đồng thời phải khai báo kiểu của biến đó. Trong bài tập này, biến ***ChuoiKyTu*** *thì phải dùng kiểu ký tự (****char*** *= character = ký tự) và giới hạn cung cấp bộ nhớ cho lưu trữ ChuoiKyTu là (20 + 1) byte, trong đó có 1 byte chứa kí tự kết thúc '\0'. Tùy vào trường hợp cụ thể mà người lập trình có thể định giới hạn bộ nhớ này miễn sao không quá nhỏ cũng không quá lớn.*

b) Lệnh gets() là lệnh đọc ký tự nhập vào từ bàn phím tương tự như lệnh đọc scanf() ở các bài tập sau. Tuy nhiên, có sự khác nhau giữa 2 lệnh này, lệnh gets() đọc được cả ký tự trắng hay dấu cách còn lệnh scanf() thì không. Giả sử trong bài này ta nhập vào chuỗi có tên là Huyện Quảng Ninh mà sử dụng lệnh gets() thì sau đó nó sẽ hiện ra dòng chữ "Huyện Quảng Ninh", nếu sử dụng lệnh scanf() thì nó sẽ chỉ hiện ra chữ "Huyện" mà thôi. Lệnh scanf() khi gặp ký tự trắng thì nó sẽ dừng lại.

*c) Lệnh **return main()** để quay lại hàm main() và tiếp tục nhập và in ra màn hình chuỗi ký tự mới. Nếu không dùng lệnh này thì khi muốn nhập chuỗi mới ta phải chạy lại chương trình, như thế rất bất tiện. Các bài về sau, mình sẽ dùng lệnh này với ý nghĩa như thế.*

* *Màn hình kết quả như sau:*



Bài 3: Viết chương trình nhập vào một kí tự bất kỳ từ bàn phím và in ra màn hình số thứ tự của ký tự đó trong bảng mã ASCII ? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
    char a;
    printf("\n  Nhap vao mot ky tu:  ");
    scanf("%c",&a);
    printf("  => Ma ASCII cua ky tu vua nhap la:  %d\n",a);
    getch();
    fflush(stdin);
    return main();
```

```
}
```

* *Giải thích:*

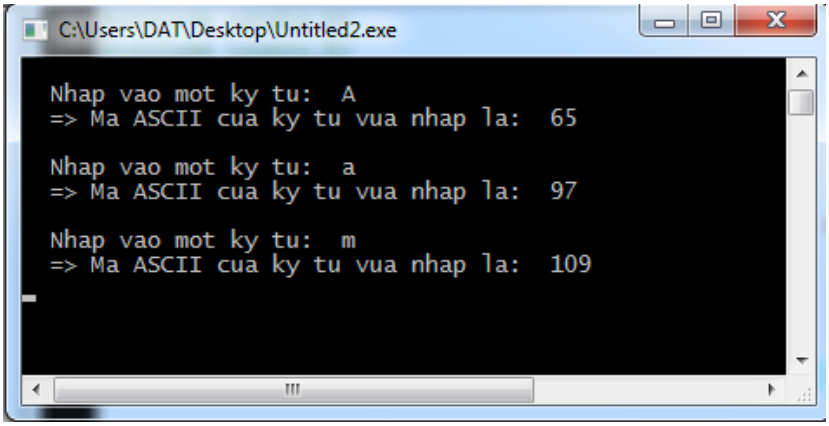
*a) Bảng chữ **ASCII** (American Standard Codes for Information Interchange) là bộ mã chuẩn của Mỹ để trao đổi thông tin. Trong bảng mã này, với dãy số nhị phân nào đó là một chữ tương tứng (có phân biệt chữ hoa và chữ thường) và thứ tự của chúng đi kèm. Ví dụ: với dãy mã nhị phân là 01011001 thì ta có ô thứ 89 và biểu diễn chữ Y. Trong bài này, giả sử nhập vào một ký tự Y và máy sẽ hiện ra là 89 chứ không hiện ra mã nhị phân của ký tự Y.*

Mọi người có thể tham khảo bảng mã ASCII trong giáo trình Tin học cơ sở - Đào Kiến Quốc (chủ biên), Bùi Thế Duy, NXB ĐHQGHN trang 86 hoặc bất kỳ giáo trình nào có giới thiệu bảng mã này.

b) Bài tập này yêu cầu nhập vào là ký tự và xuất ra là dạng số. Ta chỉ cần khai báo kiểu ký tự (char), đọc từ bàn phím kiểu ký tự (%c) và in ra màn hình với định dạng số nguyên (%d) là xong.

*c) Lệnh **fflush(stdin)** có tác dụng xóa sạch những ký tự Enter vì khi ta nhấn Enter ở trên thì chương trình sẽ coi đó là ký tự \n và gán thêm cho ký tự tiếp theo. Như thế khi chuyển sang ký tự khác thì mã ASCII sẽ hiển thị không chính xác. Nếu ta không dùng lệnh return main() thì cũng không cần dùng lệnh fflush(stdin).*

* *Màn hình kết quả như sau:*



Bài 4: Viết chương trình nhập vào 2 số nguyên dương rồi đưa ra tổng, hiệu, tích, thương của hai số đó?

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a,b,Tong,Hieu,Tich;
    float Thuong;
    printf("\n  Nhap vao cho anh hai so nguyen duong a va b:  ");
```

```
    scanf("%d%d",&a,&b);
    Tong=a+b;
    Hieu=a-b;
    Tich=a*b;
    Thuong=(float)a/b;
    printf("  Chu em! Tong cua hai so ne: %d\n",Tong);
        getch();
    printf("  Chu em! Hieu cua hai so ne: %d\n",Hieu);
        getch();
    printf("  Chu em! Tich cua hai so ne: %d\n",Tich);
        getch();
    printf("  Chu em! Thuong cua hai so ne: %f\n",Thuong);
        getch();
    printf("\n");
    return main();
}
```

* *Giải thích:*

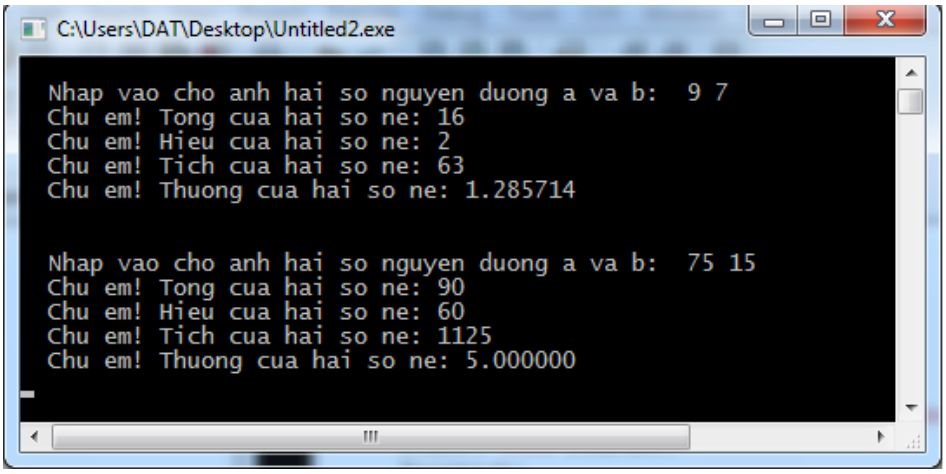
a) Các biến a, b, Tong, Hieu, Tich có kiểu là số nguyên (int = integer), còn biến Thuong thì có kiểu số thực (float = floating).

b) Lệnh scanf() là lệnh đọc ký tự nhập vào từ bàn phím. Dấu %d là thể hiện cách thức đọc, ở đây cách thức đọc là số nguyên. Dấu & thể hiện địa chỉ của a: &a được hiểu là đọc vào ký tự a và để ký tự này vào ô nhớ có địa chỉ &a. Tương tự đối với &b.

c) Điều đặc biệt trong thuật toán trên là công thức tính giá trị của biến Thuong:
Thuong = (float) a/b với ý nghĩa là lấy giá trị có kiểu số thực của phép chia trên. Nếu chỉ viết Thuong = a/b thì giá trị tính được sẽ là phần nguyên của phép chia đó. VD: Thuong = 3/5 thì giá trị trả về là 0, còn Thuong =(float) 3/5 thì giá trị trả về là 0,6.

d) Trong lệnh in ra kết quả: printf("Chu em! Tong cua hai so ne: %d\n",Tong) thì dấu %d vừa biểu thị vị trí của kết quả Tong, vừa biểu thị kiểu của kết quả Tong là kiểu số nguyên.

* *Màn hình kết quả như sau:*



Bài 5: Cho x = 10, y = 20 và z = 30. Tính giá trị các biểu thức và viết chương trình kiểm tra:

5.1 **A = 2* (x – y ++) + z*(++ z – x*y)**

5.2 **B = (--x + --y + z--)*2 + ++y*2**

5.3 **C = (x<<2) + (y|3 + z|8) + 2**

5.4 **D = ((x= =y) && (x!= z))**

Giải:

Câu 5.1: A = 2* (x – y ++) + z*(++ z – x*y) = 2*(10 – 20) + 31*(31 – 10*20) = -5259.

Chương trình là:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int x=10,y=20,z=30,a;
    a=2*(x-y++)+z*(++z-x*y);
    printf("\n  Ket qua la: %d",a);
    getch();
}
```

Câu 5.2: B = (--x + --y + z--)*2 + ++y*2 = (9 + 19 + 30)*2 + 20*2 = 156.

Chương trình là:

```
#include <stdio.h>
#include <conio.h>
```

```
main()
{
    int x=10,y=20,z=30,b;
    b=(--x+--y+z--)*2+(++y*2);
    printf("\n  Ket qua la: %d",b);
    getch();
}
```

Câu 5.3: C = (x<<2) + (y|3 + z|8) + 2 = 40 + (23 + 30)|8 + 2
= 40 + 61 + 2 = 103

Chương trình là:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int x=10,y=20,z=30,c;
    c=(x<<2)+(y|3+z|8)+2;
    printf("\n  Ket qua la: %d\a",c);
    getch();
}
```

Câu 5.4: D = ((x= =y) && (x!= z)) = 0 && 1 = 0

Chương trình là:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int x=10,y=20,z=30,d;
    d=((x==y)&&(x!=z));
    printf("\n  Ket qua la: %d\a\n",d);
    getch();
}
```

* *Giải thích:*

a) Nếu dấu "--" hoặc dấu "++" được đặt trước biến thì giá trị của biến giảm đi hoặc tăng lên 1 đơn vị rồi mới đưa vào biểu thức để tính. Hay nói cách khác là các dấu đó phải đặt trước biến thì mới ảnh hưởng đến kết quả của biểu thức. Nếu các dấu đó đặt sau biến thì chỉ ảnh hưởng đến kết quả của biến sau khi thực hiện xong biểu thức, còn giá trị của biểu thức vẫn tính theo giá trị của biến như ban đầu.

Dấu == là dấu so sánh bằng nhau, còn dấu != là dấu so sánh khác nhau. Đây là phép toán logic nên kết quả trả về là 1 hoặc 0. Nếu điều kiện đúng thì kết quả là 1, nếu kết quả sai thì kết là là 0.

b) Ký tự ***\a*** *là báo hiệu một tiếng chuông (đưa vào để cho sinh động ý mà).*

** Lưu ý: Ta có thể dùng cấu trúc rẽ nhiều nhánh* ***switch*** *(cấu trúc này sẽ được học ở nội dung của phần sau) để viết một thuật toán gộp cho cả 3 phần ở trên (không dùng witch cũng viết thuật toán gộp được nhưng kết quả hiện ra theo một thứ tự nhất định chứ không theo ý của người dùng, thích hiện kết quả của biểu thức nào thì nhập ký tự tương ứng của biểu thức đó vào) :*

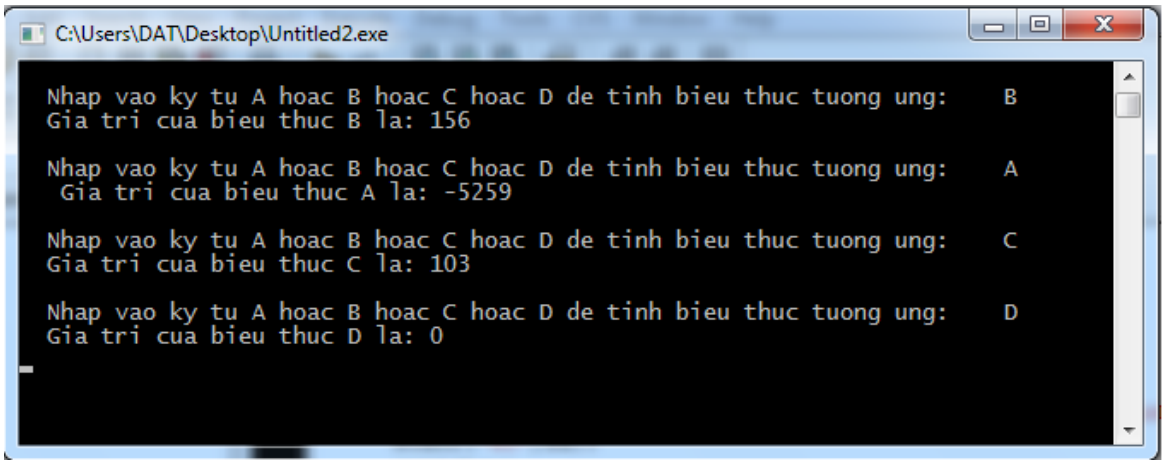
```
#include <stdio.h>
#include <conio.h>
main()
{
    int x=10,y=20,z=30;
    char a;
    printf("\n  Nhap vao ky tu A hoac B hoac C hoac D de tinh bieu thuc tuong ung:\t");
    scanf("%c",&a);
    switch(a)
    {
     case 'A':printf("   Gia tri cua bieu thuc A la: %d\n",2*(x-y++)+z*(++z-x*y));break;
     case 'B':printf("  Gia tri cua bieu thuc B la: %d\n",(--x+--y+z--)*2+(++y*2));break;
     case 'C':printf("  Gia tri cua bieu thuc C la: %d\n",(x<<2)+(y|3+z|8)+2);break;
     case 'D':printf("  Gia tri cua bieu thuc D la: %d\n",(x==y)&&(x!=z));break;
     default:printf("  Khong phai bieu thuc roi con oi!\n");
    }
    getch();
    fflush(stdin);
    return main() ;
}
```

Trong đó:

- *Kiểu của biến a là kiểu ký tự (char = character). Dấu %c thể hiện cách đọc kiểu ký tự.*

- Lệnh switch(a) là lệnh chọn một trong nhiều phương án, lệnh này xem biến a là trường hợp nào thì sẽ thực hiện lệnh tiếp theo tương ứng với trường hợp đó. Trong bài này, nếu a nhận ký tự là A thì thực hiện tiếp lệnh: 2(x-y++)+z*(++z-x*y). Còn lệnh break có tác dụng nhảy ra khỏi lệnh switch, lệnh default tương ứng với các trường hợp còn lại của lệnh switch mà không được liệt kê trong các dòng case.*

* *Màn hình kết quả như sau :*



Bài 6: Viết chương trình tính diện tích hình chữ nhật có chiều a, b được nhập vào từ bàn phím. In ra màn hình (có định dạng) kết quả? (X)

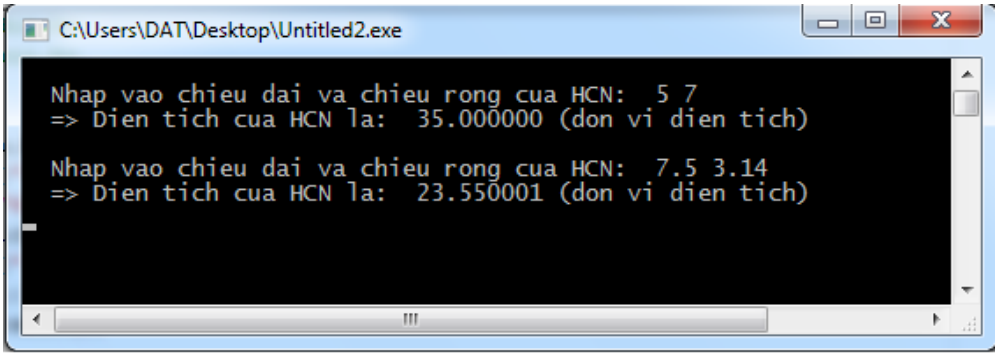
```
#include <stdio.h>
#include <conio.h>
main()
{
    float a,b,DienTichHCN;
    printf("\n  Nhap vao chieu dai va chieu rong cua HCN:  ");
    scanf("%f%f",&a,&b);
    DienTichHCN=a*b;
    printf("  => Dien tich cua HCN la:  %f (don vi dien tich)\n",DienTichHCN);
    getch();
    return main();
}
```

* *Giải thích:*

- Muốn tính diện tích HCN khi biết 2 chiều là a và b thì chỉ lấy tích a.b là xong. Ta khai báo 3 biến có kiểu số thực trong đó có 2 biến a, b tương ứng với 2 chiều của HCN và biến

DienTichHCN để tính diện tích HCN đó. Kết quả hiện ra màn hình là của biến DienTichHCN = a. b với kiểu số thực.

* *Màn hình kết quả như sau:*



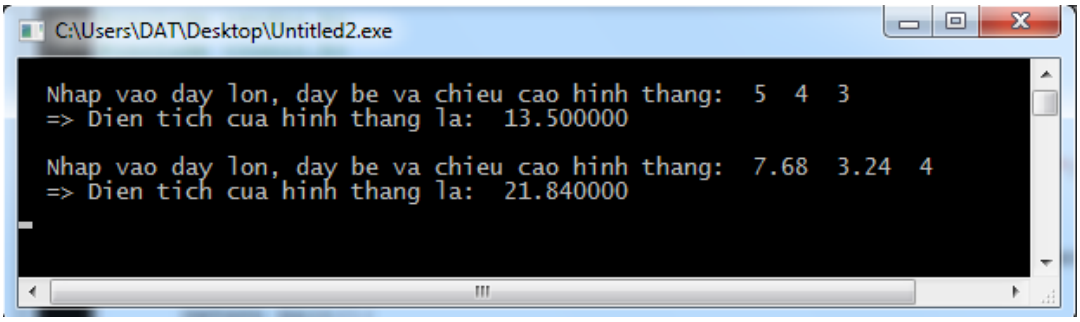
Bài 7: Viết chương trình tính diện tích hình thang có đáy lớn là a, đáy bé là b, chiều cao là h (a, b, h nhập vào từ bàn phím). Tính và in kết quả có định dạng? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
    float a,b,h,DienTichHinhThang;
    printf("\n  Nhap vao day lon, day be va chieu cao hinh thang:  ");
    scanf("%f%f%f",&a,&b,&h);
    DienTichHinhThang=(a+b)/2*h;
    printf("  => Dien tich cua hinh thang la:  %f\n",DienTichHinhThang);
    getch();
    return main();
}
```

* *Giải thích:*

- Diện tích hình thang khi biết đáy lớn là a, đáy nhỏ là b, chiều cao là h: $\frac{(a+b)}{2}.h$ *Kết quả trả về là của biến DienTichHinhThang. Bài này tương tự bài 6.*

* *Màn hình kết quả như sau:*



<u>Bài 8:</u> Hãy viết chương trình tìm max của 3 số nguyên a, b, c?

```
#include <stdio.h>
#include <conio.h>
main()
{
   int a,b,c,max;
   printf("\n  Nhap vao cho tao 3 so a, b, c:   ");
   scanf("%d%d%d",&a,&b,&c);
   if((a>=b)&&(a>=c))
      {printf("  => Gia tri lon nhat day bo:%5d\n",a);}
   else
       if((b>=a)&&(b>=c))
      {printf("  => Gia tri lon nhat day bo:%5d\n",b);}
       else
       {printf("  => Gia tri lon nhat day bo:%5d\n",c);}
   getch();
   return main();
}
```

* *<u>Giải thích:</u>*

a) Thuật toán tìm số lớn nhất trong 3 chữ số: Nếu 1 trong 3 số là lớn nhất thì nó phải lớn hơn hoặc bằng hai số còn lại. Trong đó, có tất cả 6 kiểu hoán vị (3! = 6) cho 3 số đó.

$a \geq b \geq c$ *(1)* $\quad$ $c \geq a \geq b$ *(5)*

$a \geq c \geq b$ *(2)* $\quad$ $c \geq b \geq a$ *(6)*

$b \geq a \geq c$ *(3)*

$b \geq c \geq a$ *(4)*

Nếu a là số lớn nhất thì phải thỏa mãn đồng thời 2 điều kiện (1) và (2), nếu b là số lớn nhất thì phải thỏa mãn đồng thời 2 điều kiện (3) và (4), nếu c là số lớn nhất thì phải thỏa mãn các điều kiện còn lại. Lệnh else với ý nghĩa nếu điều kiện ngược lại với điều kiện của hàm if () thì thực hiện công việc khác. Sau lệnh else thì điều kiện có thể chia nhỏ hơn nữa nên ta lại dùng hàm if () tiếp. Chính vì vậy các hàm if () có thể lồng nhau. Về thuật toán, có sự tương đồng giữa hàm if () trong C với hàm if () được sử dụng trong Excel, ví dụ: xếp loại học lực của một học sinh khi biết điểm tổng kết của học sinh đó nằm ở ô A1:

=IF(A1>=9,"Xuất sắc",IF(A1>=8,"Giỏi",IF(A1>=7,"Khá",IF(A1>=5,"TB","Yếu"))))

$\Rightarrow$ *Hàm IF đầu tiên bao hàm tất cả các hàm IF còn lại.*

Công thức trên có thể viết tương đương bằng ngôn ngữ lập trình C là:

```
if(A1>=9)
printf("Xuat sac");
else
        if(A1>=8)
        printf("Gioi");
        else
                if(A1>=7)
                printf("Kha");
                else
                        if(A1>=5)
                        printf("TB");
                        else
                                printf("Yeu");
```

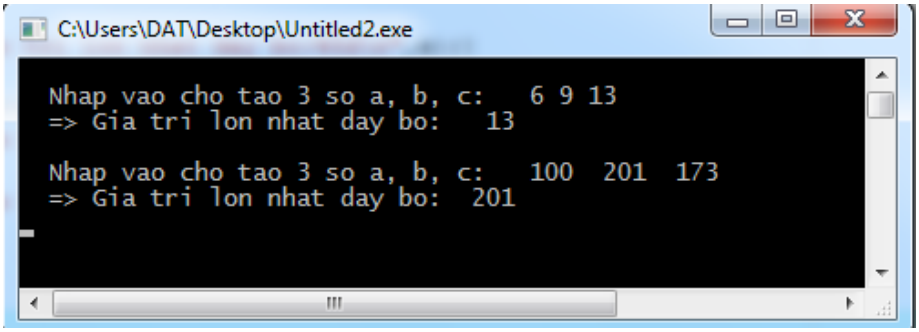
Chúng ta có thể lồng rất nhiều hàm if () như thế, tùy thuộc vào việc bạn muốn chia thành bao nhiêu khoảng giá trị.

b) Bài toán trên có thể giải bằng cách khác: đưa thêm biến ***max*** *vào và gán giá trị đầu tiên cho max. Sau đó so sánh max với biến thứ 2, nếu max ≤ biến thứ 2 thì max nhận giá trị mới bằng biến thứ 2. Cứ như thế cho tới biến cuối cùng. Cách này ngắn gọn và dễ hiểu hơn, đặc biệt khi áp dụng cho trường hợp có nhiều số thì thuật toán này thể hiện tính ưu việt hơn:*

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a,b,c,max;
    printf("\n  Nhap vao cac so a, b, c:   ");
    scanf("%d%d%d",&a,&b,&c);
```

```
    max=a;
    if(max<=b)
    max=b;
    if(max<=c)
    max=c;
    printf("  => So lon nhat la: %d\n",max);
    getch();
    return main();
}
```

* *Màn hình kết quả như sau:*



Bài 9: Viết chương trình nhập vào từ bàn phím bốn số a, b, c, d. Tìm giá trị nhỏ nhất và giá trị lớn nhất trong bốn số đó, in kết quả ra màn hình? (X)

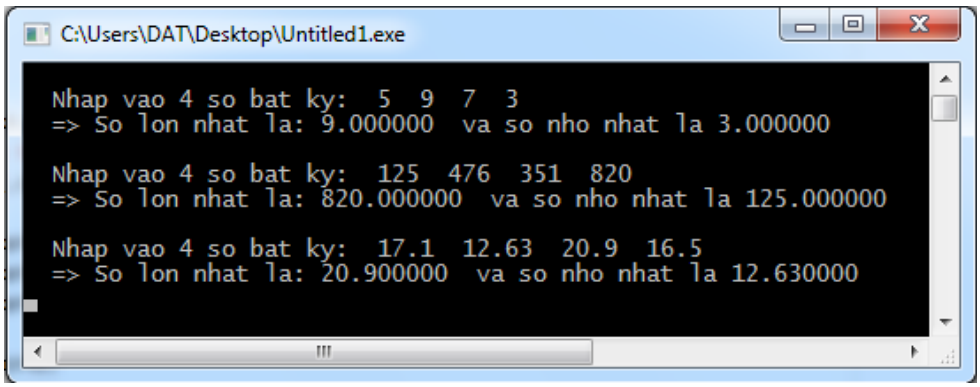
```
#include <stdio.h>
#include <conio.h>
main()
{
   float a,b,c,d,max,min;
   printf("\n  Nhap vao 4 so bat ky:  ");
   scanf("%f%f%f%f",&a,&b,&c,&d);
   max=a;
     if(max<=b)max=b;
     if(max<=c)max=c;
     if(max<=d)max=d;
   min=a;
```

```
    if(min>=b)min=b;
    if(min>=c)min=c;
    if(min>=d)min=d;
  printf("  => So lon nhat la: %f  va so nho nhat la %f\n",max,min);
  getch();
  return main();
}
```

* *Giải thích:*

- Tương tự như bài tìm max của 3 số, sử dụng và gán giá trị đầu tiên cho 2 biến max/min. Sau đó so sánh max/min với các số còn lại, sử dụng hàm if(). Kết quả in ra màn hình là max và min.

* *Màn hình kết quả như sau:*



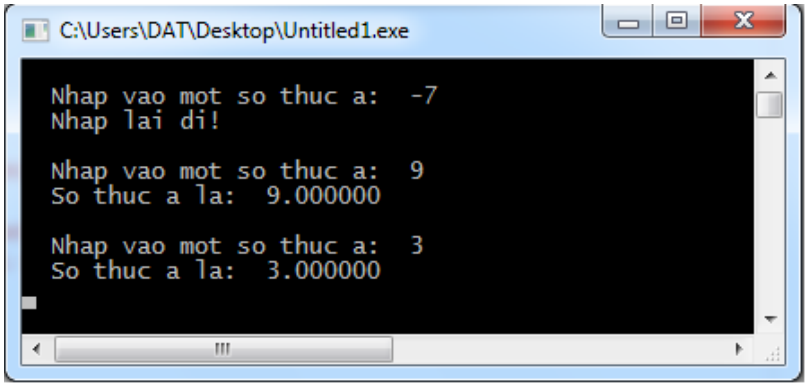
Bài 10: Viết chương trình nhập vào từ bàn phím một số thực a. Nếu a ≤ 0 thì nhập lại a, nếu a > 0 thì in ra màn hình số đó? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
  float a;
  printf("\n  Nhap vao mot so thuc a:  ");
  scanf("%f",&a);
  if(a>0)printf("  So thuc a la:  %f\n",a);
  else printf("  Nhap lai di!\n");
  getch();
```

```
    return main();
}
```

*- Bài này sử dụng hàm if() đơn giản hơn. Chỉ in ra số thực a nếu nó thỏa mãn điều kiện >0. Cũng có thể sử dụng lệnh nhảy **goto** để thực hiện tại bất kỳ vị trí nào mong muốn.*

** Màn hình kết quả như sau:*



Bài 11: Lập chương trình nhập vào tọa độ 3 điểm trên hệ trục tọa độ Oxy. Hãy cho biết điểm gần nhất với điểm O (0, 0)?

```
#include <stdio.h>
#include <conio.h>
main()
{
   int a1,a2,b1,b2,c1,c2;
   float OA,OB,OC;
   printf("\n  Nhap cho tao toa do diem A:  ");
   scanf("%d%d",&a1,&a2);
   printf("  Nhap cho tao toa do diem B:  ");
   scanf("%d%d",&b1,&b2);
   printf("  Nhap cho tao toa do diem C:  ");
   scanf("%d%d",&c1,&c2);
      OA=sqrt(a1*a1+a2*a2);
      OB=sqrt(b1*b1+b2*b2);
      OC=sqrt(c1*c1+c2*c2);
   if((OA<=OB)&&(OA<=OC))
```

```
    printf("  => Diem gan nhat la:  A (%d,%d) voi khoang cach:  %f (don vi do dai)\n",a1,a2,OA);
        else if((OB<=OA)&&(OB<=OC))
        printf("  => Diem gan nhat la:  B (%d,%d) voi khoang cach:  %f (don vi do dai)\n",b1,b2,OB);
          else
          printf("  => Diem gan nhat la:  C (%d,%d) voi khoang cach:  %f (don vi do dai)\n",c1,c2,OC);
    getch();
    return main();
}
```

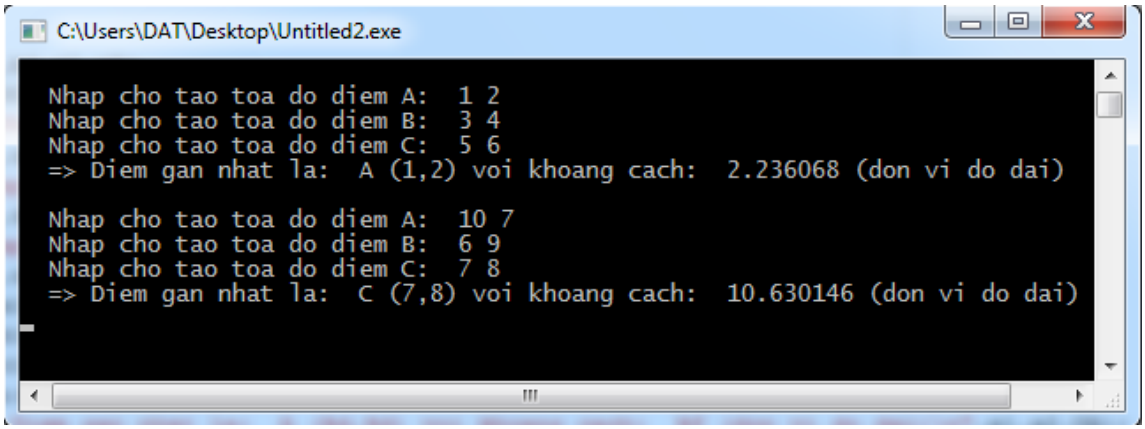
* *Giải thích:*

a) Điểm gần O (0,0) nhất chính là điểm mà khoảng cách từ nó đến O là nhỏ nhất. Điều kiện để một khoảng cách nhỏ nhất là khoảng cách đó phải nhỏ hơn tất cả các khoảng cách còn lại (có thể có nhiều khoảng cách cùng nhỏ nhất).

b) Hàm ***sqrt*** *(sqrt = square root) là hàm lấy căn bậc hai và được lấy ra từ thư viện chuẩn, hàm này được dùng để tính khoảng cách giữa hai điểm khi biết tọa độ của chúng.*

$$O\ (0,\ 0);\ A\ (a_1,\ a_2) \Rightarrow OA = \sqrt{(a_1-0)^2+(a_2-0)^2} = \sqrt{a_1^2+a_2^2} = sqrt\ (a_1^2+a_2^2)$$

* *Màn hình kết quả như sau:*

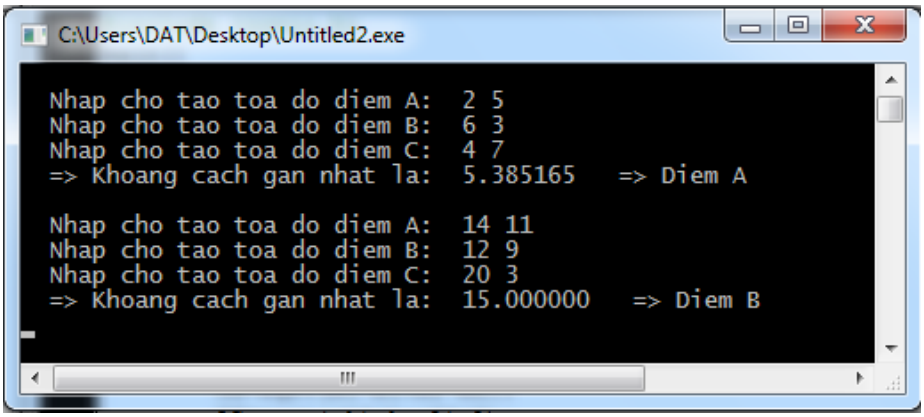


Hoặc có thể dùng thuật toán khác sẽ dễ hiểu hơn:

```
#include <stdio.h>
#include <conio.h>
main()
{
```

```
    int a1,a2,b1,b2,c1,c2;
    float OA,OB,OC,min;
    printf("\n  Nhap cho tao toa do diem A:  ");
    scanf("%d%d",&a1,&a2);
    printf("  Nhap cho tao toa do diem B:  ");
    scanf("%d%d",&b1,&b2);
    printf("  Nhap cho tao toa do diem C:  ");
    scanf("%d%d",&c1,&c2);
       OA=sqrt(a1*a1+a2*a2);
       OB=sqrt(b1*b1+b2*b2);
       OC=sqrt(c1*c1+c2*c2);
    min=OA;
    if(min>=OB)
      min=OB;
    if(min>=OC)
      min=OC;
    printf("  => Khoang cach gan nhat la:  %f",min);
    if(min==OA)
      printf("   => Diem A\n");
    if(min==OB)
      printf("   => Diem B\n");
    if(min==OC)
      printf("   => Diem C\n");
    getch();
    return main();
}
```

* *Màn hình kết quả như sau:*



Bài 12: Viết chương trình nhập vào tọa độ 3 điểm trên hệ trục tọa độ Oxy. Xét xem 3 điểm đó có tạo thành một tam giác hay không? Tính diện tích tam giác (nếu tạo thành)? (X)

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
main()
{
   float x1,y1,x2,y2,x3,y3,AB,AC,BC,p,S,a,b,c;
   printf("\n  Nhap toa do diem A:  ");
   scanf("%f%f",&x1,&y1);
   printf("  Nhap toa do diem B:  ");
   scanf("%f%f",&x2,&y2);
   printf("  Nhap toa do diem C:  ");
   scanf("%f%f",&x3,&y3);
   c=(x2-x1)*(x2-x1)+(y2-y1)*(y2-y1);AB=sqrt(c);
   b=(x3-x1)*(x3-x1)+(y3-y1)*(y3-y1);AC=sqrt(b);
   a=(x3-x2)*(x3-x2)+(y3-y2)*(y3-y2);BC=sqrt(a);
   if((AB+BC>AC)&&(AB+AC>BC)&&(BC+AC>AB))
     {
       printf("  => Ba diem tren tao thanh mot tam giac");
       if((AB==BC)&&(BC==AC))
         printf(" deu");
```

```
        else
           {
              if((a+b==c)||(a+c==b)||(b+c==a))
                printf(" vuong");
              if((AB==AC)||(BC==AB)||(AC==BC))
                printf(" can");
           }
        printf(".\n     Dien tich tam giac la:  ");
        p=(AB+AC+BC)/2;
        S=sqrt(p*(p-AB)*(p-AC)*(p-BC));
        printf("%0.4f\n",S);
      }
    else printf("  => Ba diem tren khong tao thanh mot tam giac.\n");
    getch();
    return main();
}
```

* *Giải thích:*

a) Nhập vào từng tọa độ các điểm và gán cho các biến tương ứng. Mỗi điểm được đặc trưng bởi hai giá trị là hoành độ x_i và tung độ y_i. Khi biết tọa độ của các điểm thì ta có thể tính được khoảng cách giữa hai điểm bất kì A, B theo công thức: $d(A,B)=\sqrt{(x_B-x_A)^2+(y_B-y_A)^2}$. *Gán khoảng cách tính được cho các biến tương ứng **AB**, **AC**, **BC**. Hàm **sqrt()** là hàm tính giá trị tuyệt đối của một số.*

b) Theo bất đẳng thức tam giác thì 3 điểm bất kì A, B, C muốn tạo thành tam giác thì phải thỏa mãn đồng thời 3 điều kiện:

$$\begin{cases} AB+BC>AC \\ AB+AC>BC \\ BC+AC>AB \end{cases}$$

Trường hợp thỏa mãn 3 điều kiện trên thì tam giác được tạo thành. Tiếp tục xét nếu thấy: AB = AC = BC thì kết luận đó là tam giác đều. Nếu chỉ có AB = AC hoặc AB = BC hoặc AC = BC thì kết luận là tam giác cân. Nếu $AB^2 + BC^2 = AC^2$ hoặc $AC^2 + BC^2 = AB^2$ hoặc $AB^2 + AC^2 = BC^2$ thì kết luận là tam giác vuông. Chắc mọi người đang thắc mắc là tại sao dùng thêm biến a, b, c (trong đó $a = BC^2$, $b = AC^2$ và $c = AB^2$); mình dùng thêm a, b, c bởi vì khi khai căn của số a thì bằng BC nhưng khi bình phương BC lên thì lại không bằng a (máy tính có một độ lệch nhất định) do đó điều kiện kiểm tra tính chất vuông của tam giác sẽ không đúng. Nếu không tin các bạn có thể không dùng biến a, b, c và thay điều kiện if((a+b==c)||(a+c==b)||(b+c==a)) bằng

điều kiện if((BC²+AC²=AB²)||(BC²+AB²=AC²)||(AC²+AB²=BC²)) xem chương trình có kiểm tra được tính chất vuông của tam giác hay không? (lấy 3 điểm đặc biệt là (0, 0); (3, 0) và (0,5)).

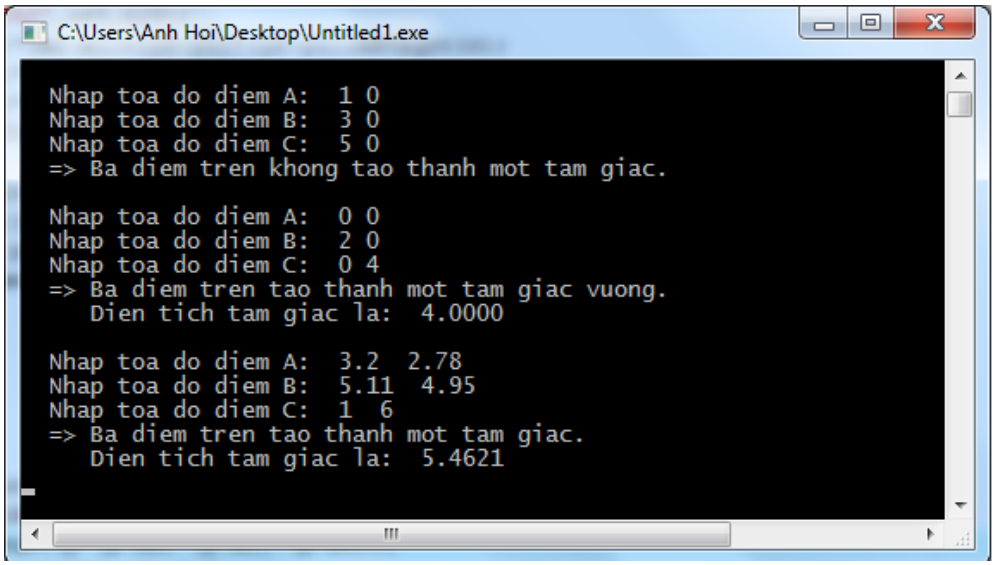
c) Diện tích tam giác được tính dựa vào công thức Heron:

$$p = \sqrt{p(p-AB).(p-BC).(p-AC)}$$

Trong đó, p là nửa chu vi của tam giác: $p = \dfrac{AB+BC+AC}{2}$

Có nhiều cách tính diện tích tam giác: có thể dùng phương pháp hình học, lượng giác, vectơ nhưng trong trường hợp tổng quát thì sử dụng công thức Heron là đơn giản nhất.

* *Màn hình kết quả như sau:*



<u>Bài 13:</u> Viết chương trình tìm nghiệm của hệ hai phương trình bậc nhất 2 ẩn:

$$\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases} \quad \textbf{(X)}$$

```
#include <stdio.h>
#include <conio.h>
main()
{
    float a1,b1,c1,a2,b2,c2,D,Dx,Dy,x,y;
    printf("\n  Nhap vao he so a1, b1, c1 cua phuong trinh thu nhat:  ");
    nhaplai1:scanf("%f%f%f",&a1,&b1,&c1);
    if(a1*a1+b1*b1==0)
```

```
    {printf("  Nhap lai a1, b1, c1:  ");goto nhaplai1;}
  printf("  Nhap vao he so a2, b2, c2 cua phuong trinh thu hai:  ");
  nhaplai2:scanf("%f%f%f",&a2,&b2,&c2);
  if(a2*a2+b2*b2==0)
    {printf("  Nhap lai a2, b2, c2:  ");goto nhaplai2;}
  D=a1*b2-a2*b1;
  Dx=c1*b2-c2*b1;
  Dy=a1*c2-a2*c1;
  if(D!=0)
    {
      x=Dx/D;y=Dy/D;
      printf("  => He phuong trinh co nghiem duy nhat:\n\t(x, y) = (%0.3f,
%0.3f).\n",x,y);
    }
  if(D==0)
    {
      if(Dx!=0||Dy!=0)
        printf("  => He phuong trinh vo nghiem.\n");
      if(Dx==Dy&&Dx==0)
        printf("  => He phuong trinh co vo so nghiem.\n");
    }
  getch();
  return main();
}
```

* *Giải thích:*

a) Để cho hệ trên là hệ hai phương trình bậc nhất hai ẩn số thì các hệ số: a_1 và b_1 không đồng thời bằng 0 ($a_1^2 + b_1^2 \neq 0$); a_2 và b_2 không đồng thời bằng 0 ($a_2^2 + b_2^2 \neq 0$). Nếu xảy ra trường hợp đồng thời bằng 0 thì tiến hành nhập lại hệ số, sử dụng lệnh nhảy vô điều kiện ***goto****. Bài này mình chỉ làm với hệ bậc nhất hai ẩn (tức là các hệ số phải không đồng thời bằng 0), còn nếu muốn làm tổng quát về giải và biện luận hệ phương trình thì phải chia nhỏ điều kiện ra nữa.*

b) Sau khi nhập xong hệ số thì tính các định thức:

$$D = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} = a_1 b_2 - a_2 b_1$$

$$D_X = \begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix} = c_1 b_2 - c_2 b_1$$

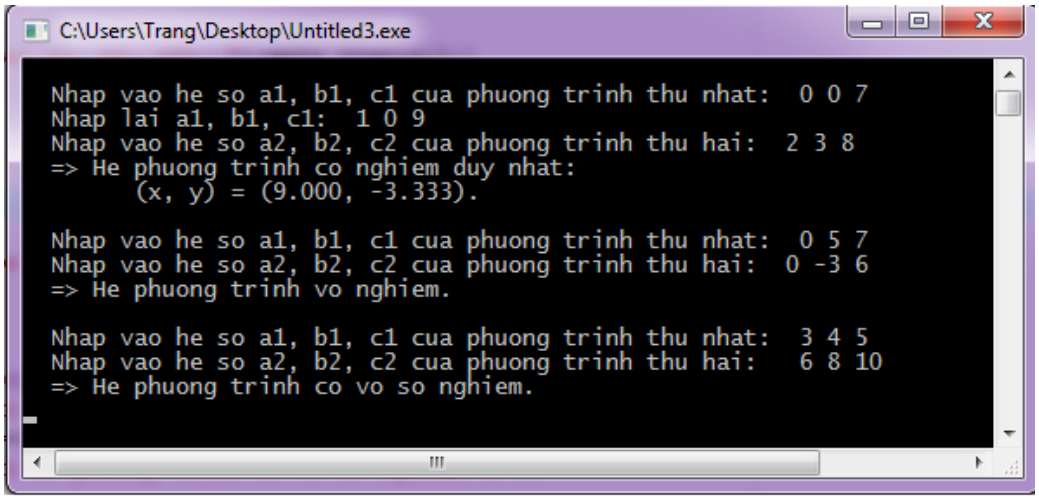
$$D_Y = \begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix} = a_1 c_2 - a_2 c_1$$

+ Nếu $D \neq 0$ thì hệ phương trình có nghiệm duy nhất: $x = \frac{D_X}{D}$; $y = \frac{D_Y}{D}$

+ Nếu $D = 0$ và ($D_X \neq 0$ hoặc $D_Y \neq 0$) thì hệ phương trình vô nghiệm.

+ Nếu $D = 0$ và $D_X = D_Y = 0$ thì hệ phương trình có vô số nghiệm.

* *Màn hình kết quả như sau:*



<u>Bài 14:</u> Viết chương trình tìm nghiệm của phương trình: $ax^2 + bx + c = 0$ với a, b, c nhập vào từ bàn phím. In ra màn hình nghiệm của phương trình đó? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
   float a,b,c,Del,x1,x2,d,e;
   printf("\n  Nhap vao cac he so a, b, c:\t");
   scanf("%f%f%f",&a,&b,&c);
   if((a==0)&&(b==0)&&(c==0))
     printf("  => Phuong trinh luon dung voi moi x!\n");
   if((a==0)&&(b==0)&&(c!=0))
```

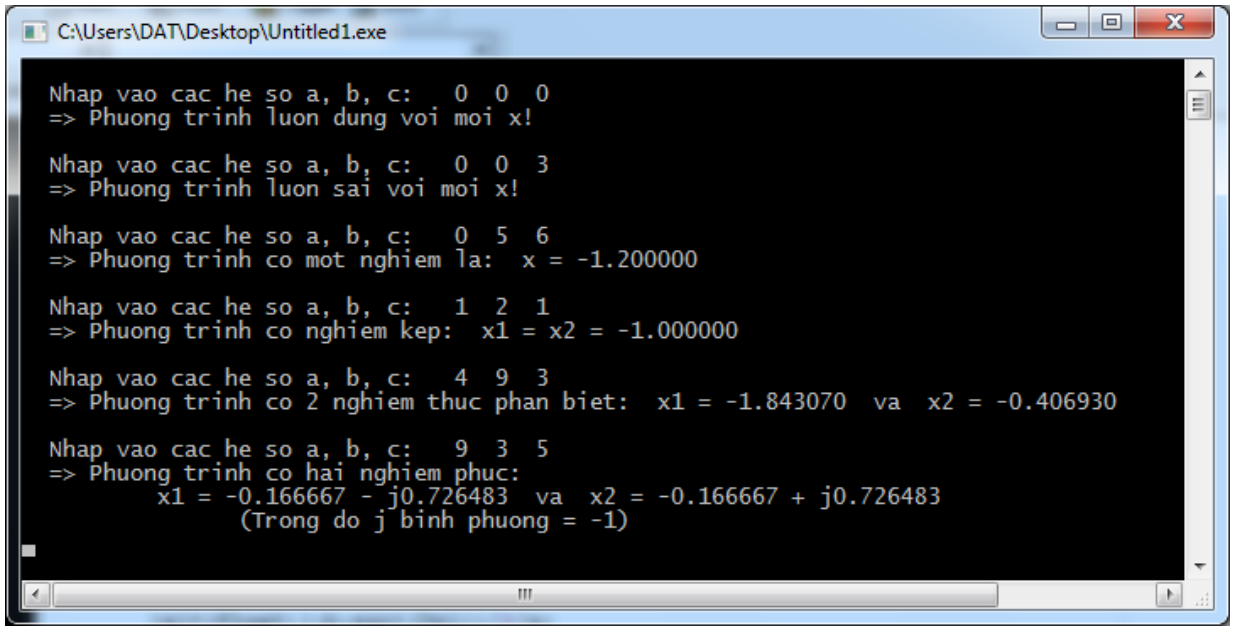
```
        printf("  => Phuong trinh luon sai voi moi x!\n");
    if((a==0)&&(b!=0))
        printf("  => Phuong trinh co mot nghiem la:  x = %f\n",-c/b);
    if(a!=0)
    {
        Del=b*b-4*a*c;
        if(Del>0)
        {x1=(float)(-b-sqrt(Del))/2/a;
        x2=(float)(-b+sqrt(Del))/2/a;
        printf("  => Phuong trinh co 2 nghiem thuc phan biet:  x1 = %f  va  x2 =
%f\n",x1,x2);}
        else if(Del==0)
        printf("  => Phuong trinh co nghiem kep:  x1 = x2 = %f\n",(float)(-b/2/a));
        else
        {
            d=-b/2/a;
            e=sqrt(fabs(Del))/2/a;
            printf("  => Phuong trinh co hai nghiem phuc:\n\t  x1 = %f - j%f  va  x2
= %f + j%f\n",d,e,d,e);
            printf("  \t\t(Trong do j binh phuong = -1)\n");
        }
    }
    getch();
    return main();
}
```

* *Giải thích:*

- Ta phải phân chia thành nhiều trường hợp: Nếu a = b = c = 0 thì hiển nhiên phương trình luôn đúng $\forall x$*, nếu a = b = 0 và c* $\neq 0$ *thì phương trình luôn sai* $\forall x$*, nếu a = 0 và b* $\neq$ *0 thì* $x = \frac{-c}{b}$*, nếu a* $\neq$ *0 thì phương trình trở thành phương trình bậc 2 đối với x. Giải phương trình bậc hai, đầu tiên phải tính* $\Delta = b^2 - 4ac$ *rồi sau đó dựa vào từng trường hợp của* Δ *để tính nghiệm của phương trình, sử dụng hàm if(). Trong trường hợp* $\Delta < 0$ *ta vẫn tìm nghiệm mở rộng của phương trình trên tập số phức:*

$$x_1 = \frac{-b - i\sqrt{|\Delta|}}{2a} \quad và \quad x_2 = \frac{-b + i\sqrt{|\Delta|}}{2a} \quad (trong\ đó\ i^2 = -1)$$

* *Màn hình kết quả như sau:*



<u>Bài 15:</u> Viết chương trình nhập vào tháng m và năm y. In ra màn hình số ngày của tháng m trong năm y đó? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
    int m,y;
    printf("\n  Nhap vao thang m va nam y:  ");
    scanf("%d%d",&m,&y);
    if((m<1)||(m>12)||(y<1))printf("  => Thang hoac nam khong dung!\n");
    else
     {
       if(m==2)
         {
           if(((y%4==0)&&(y%100!=0))||((y%100==0)&&(y%400==0)))
             printf("  => Thang 2 nam %d co 29 ngay (vi la nam nhuan)\n",y);
```

```
            else printf ("  => Thang 2 nam %d co 28 ngay\n",y);
          }
        if((m==4)||(m==6)||(m==9)||(m==11))
           printf("  => Thang %d nam %d co 30 ngay\n",m,y);
        if((m==1)||(m==3)||(m==5)||(m==7)||(m==8)||(m==10)||(m==12))
           printf("  => Thang %d nam %d co 31 ngay\n",m,y);
      }
      getch();
      return main();
  }
```

* <u>*Giải thích:*</u>

- Khi nhập vào tháng m và năm y thì xảy ra các trường hợp sau:

+ Nếu m = 4, 6, 9, 11 thì tháng m có 30 ngày đối với tất cả các năm y.

+ Nếu m = 1, 3, 5, 7, 8, 10, 12 thì tháng m có 31 ngày đối với tất cả các năm y.

+ Đặc biệt nếu m = 2 thì tháng m có thể 28 ngày hoặc 29 ngày (đối với năm nhuận). Vấn đề là xác định xem năm nào là năm nhuận, vậy phải dựa vào số y. Những năm nhuận là những năm mà chia hết cho 4 và nếu năm đó chia hết cho 100 thì nó phải chia hết cho 400 nữa mới thỏa mãn là năm nhuận. Chắc mọi người đang thắc mắc tại sao lại phải thêm điều kiện chia hết cho 400 đối với những năm tận cùng là 00? Mình xin giải thích như sau:

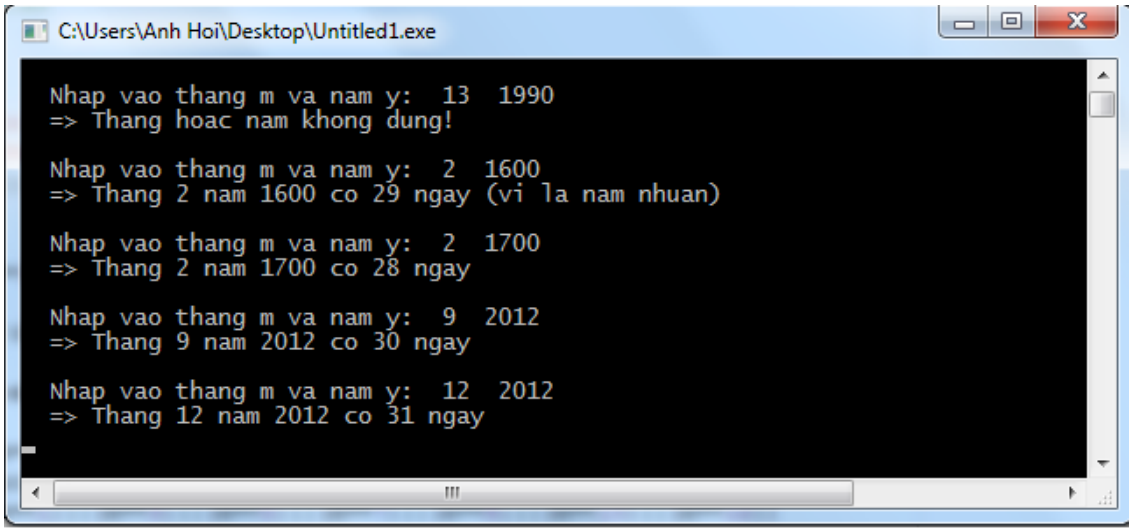
Bình thường nếu ta coi Trái Đất chuyển động quanh Mặt Trời một năm mất $365\frac{1}{4}$ *ngày thì có nghĩa là 4 năm phải thừa ra một ngày và 4 năm thì có 1 năm nhuận (366 ngày). Tuy nhiên, thực tế Trái Đất chuyển động quanh Mặt Trời chỉ có 365 ngày 5 giờ 48 phút 46 giây = 365,2421991 ngày mà thôi (<* $365\frac{1}{4}$*). Vì thế, nếu cứ 4 năm tính nhuận một lần thì dần dần sẽ thừa thời gian ra. Do đó, người ta hiệu chỉnh bằng cách không tính nhuận cho những năm tận cùng là 00 mà không chia hết cho 400 (hay nói cách khác là cứ 2000 năm thì không tính nhuận cho 15 lần).*

- Bài này chỉ cần dùng hàm if() để phân ra các trường hợp là tính được thôi. Nếu là tháng 2 của năm nhuận thì điều kiện là:

$$\left[\begin{array}{l}\begin{cases} y \vdots 4 \\ y \text{ không} \vdots 100 \end{cases} \\ \begin{cases} y \vdots 100 \\ y \vdots 400 \end{cases}\end{array}\right. \Leftrightarrow ((y\%4==0)\&\&(y\%100!=0))||((y\%100==0)\&\&(y\%400==0))$$

Và bài này chỉ áp dụng cho những năm sau công nguyên (y ≥ 1).

* <u>*Màn hình kết quả như sau:*</u>



<u>Bài 16:</u> Hãy viết chương trình đổi giá trị 2 biến có kiểu số cho nhau mà không được sử dụng biến trung gian?

```
#include <stdio.h>
#include <conio.h>
main()
{
   int a,b;
   printf("\n  Nhap cho tao 2 so a, b:  ");
   scanf("%d%d",&a,&b);
   a=a+b;
   b=a-b;
   a=a-b;
   printf("  => Hai so a va b la: %d   %d\n",a,b);
   getch();
   return main();
}
```

* *<u>Giải thích:</u>*

Thuật toán đổi 2 biến có kiểu số cho nhau mà không sử dụng biến trung gian: Giá trị của biến được cập nhật liên tục theo từng lệnh. Đầu tiên a = a + b tức là gán giá trị a + b cho a, hay nói cách khác là a đã tăng lên một lượng là b đơn vị. Sau đó, b = a – b mà a đã tăng lên b đơn vị thì sau phép toán này b = a +b – b = a (giá trị a ban đầu nhập vào). Cuối cùng, a = a – b mà a

đã tăng lên b đơn vị, b đã biến thành a nên ⇒ a = a + b – a = b. Vậy hai biến đã đổi giá trị cho nhau.

Ví dụ: Ban đầu nhập vào là a = 5, b = 7

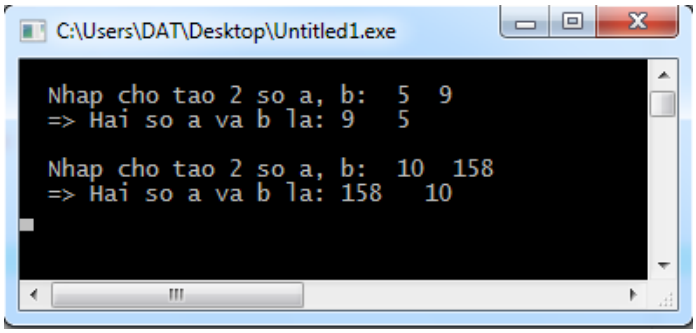
a = a + b = 5 + 7 = 12

b = a – b = 12 – 7 = 5 (*)

a = a – b = 12 – 5 = 7 (*)

⇒ Sau 3 bước trên giá trị của hai biến a và b đã hoán đổi cho nhau.

* *Màn hình kết quả như sau:*



Bài 17: Giả sử rằng các số 1, 2, 3, 4, 5, 6, 7 tương ứng là các ngày chủ nhật, thứ hai, thứ ba, thứ tư, thứ năm, thứ sáu, thứ bảy trong tuần. Viết chương trình nhập vào từ bàn phím một số nguyên dương nằm trong đoạn [1, 7]. In ra ngày trong tuần tương ứng với số vừa nhập? (X)

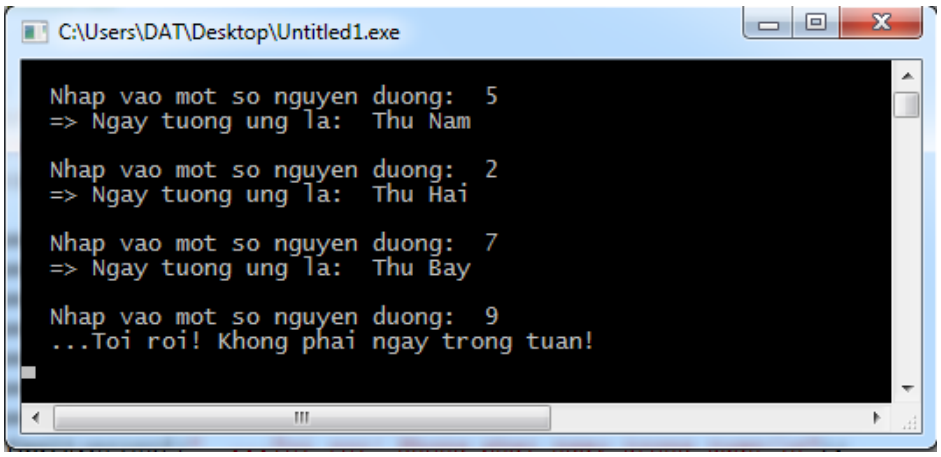
```
#include <stdio.h>
#include <conio.h>
main()
{
   int a;
   printf("\n  Nhap vao mot so nguyen duong:  ");
   scanf("%d",&a);
   switch(a)
   {
      case 1:printf("  => Ngay tuong ung la:  Chu Nhat\n");break;
      case 2:printf("  => Ngay tuong ung la:  Thu Hai\n");break;
      case 3:printf("  => Ngay tuong ung la:  Thu Ba\n");break;
      case 4:printf("  => Ngay tuong ung la:  Thu Tu\n");break;
```

```
        case 5:printf("  => Ngay tuong ung la:  Thu Nam\n");break;
        case 6:printf("  => Ngay tuong ung la:  Thu Sau\n");break;
        case 7:printf("  => Ngay tuong ung la:  Thu Bay\n");break;
        default:printf("  ...Toi roi! Khong phai ngay trong tuan!\n");
    }
    getch();
    return main();
}
```

* *Giải thích:*

*- Bài này dùng cấu trúc rẽ nhiều nhánh **switch** như đã giải thích ở bài tập tính giá trị các biểu thức A, B, C, D. Tương ứng với số nhập vào cụ thể như trên dòng case thì chương trình sẽ thực hiện nhiệm vụ nhất định và thoát khỏi lệnh switch nhờ lệnh break. Các trường hợp còn lại không được liệt kê trên các dòng case thì thực hiện công việc theo nhánh default.*

* *Màn hình kết quả như sau:*



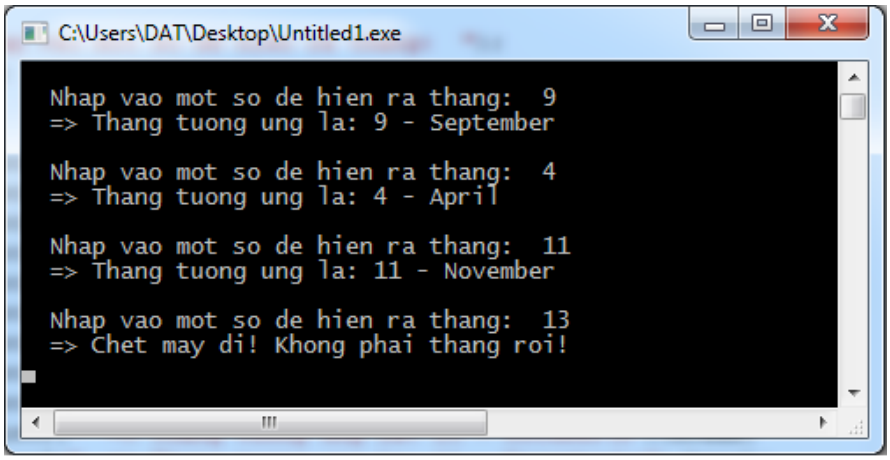
Bài 18: Viết chương trình nhập vào một số và đưa ra tên của tháng đó trong năm?

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a;
    printf("\n  Nhap vao mot so de hien ra thang:  ");
    scanf("%d",&a);
    switch(a)
```

```
    {
       case 1:printf("  => Thang tuong ung la: 1 - January\n");break;
       case 2:printf("  => Thang tuong ung la: 2 - February\n");break;
       case 3:printf("  => Thang tuong ung la: 3 - March\n");break;
       case 4:printf("  => Thang tuong ung la: 4 - April\n");break;
       case 5:printf("  => Thang tuong ung la: 5 - May\n");break;
       case 6:printf("  => Thang tuong ung la: 6 - June\n");break;
       case 7:printf("  => Thang tuong ung la: 7 - July\n");break;
       case 8:printf("  => Thang tuong ung la: 8 - August\n");break;
       case 9:printf("  => Thang tuong ung la: 9 - September\n");break;
       case 10:printf("  => Thang tuong ung la: 10 - October\n");break;
       case 11:printf("  => Thang tuong ung la: 11 - November\n");break;
       case 12:printf("  => Thang tuong ung la: 12 - December\n");break;
       default:printf("  => Chet may di! Khong phai thang roi!\n");
    }
    getch();
    return main();
}
```

* *Giải thích: Tương tự bài tập In ra các thứ tương ứng trong tuần, chỉ thay ngày trong tuần bằng tháng trong năm!*

* *Màn hình kết quả như sau:*



<u>Bài 19:</u> Nhập vào hai số x, y và nhập 1 trong 4 phép toán +, - , *, /. Thực hiện phép tính tương tứng với phép toán vừa nhập?

(Câu 1 trong đề thi cuối kỳ lớp THCS 3, ĐHKHTN – ĐHQGHN, Thứ 7 ngày 22/12/2012, Kỳ I năm học 2012 – 2013, thời gian làm bài: 30 phút/ 2 câu)

CHƯƠNG II. CHƯƠNG TRÌNH CON – HÀM

<u>Bài 19:</u> Tính tổng: S = 1.3.5 + 3.5.7 + … + n.(n+2).(n+4) với n được nhập vào từ bàn phím?

```
#include <stdio.h>
#include <conio.h>
main()
{
    long i,n,S=0;
    printf("\n  Nhap vao so n de thay xem cho:  ");
    scanf("%ld",&n);
    for(i=1;i<=n;i=i+2)
    S=S+i*(i+2)*(i+4);
    printf("  => Diem Vuong bao S bang:  %ld\n",S);
    getch();
    return main();
}
```

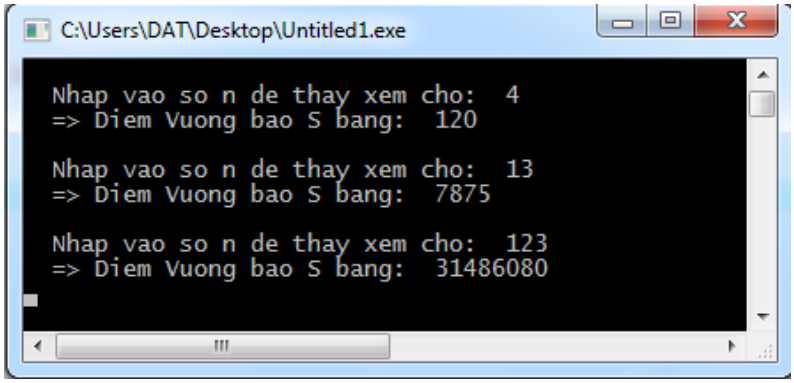
* *<u>Giải thích:</u>*

*a) Vòng lặp xác định **for** bắt đầu với giá trị khởi tạo i = 1, nếu thỏa mãn điều kiện i ≤ n mà ta nhập vào thì sẽ tiếp tục công việc tính S=S+i.(i+2).(i+4). Sau đó quay lại vòng thứ hai và tăng i lên 2 đơn vị, tiếp tục kiểm tra điều kiện i ≤ n nếu thỏa mãn thì tính S. Cứ như thế cho tới khi điều kiện không thỏa mãn thì dừng lại và in ra màn hình kết quả của S.*

b) Trong bài này thì các giá trị i nhận đều là số lẻ nên khi ta nhập n là số lẻ hoặc số chẵn ngay liền sau nó thì kết quả S nhận được là giống nhau. Ví dụ: nhập n bằng 5 hay bằng 6 thì kết quả S vẫn là 435.

*Vì kết quả của phép tính là lớn nếu n có 3 hoặc 4 chữ số trở lên, do đó mình để kiểu **long**. Nếu nhập số lớn quá thì kết quả hiện không đúng (thường ra số âm).*

* *Màn hình kết quả như sau:*



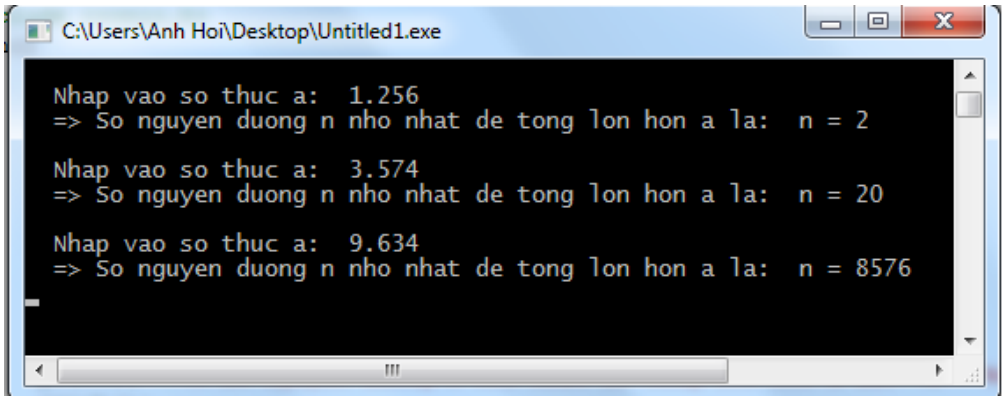
Bài 20: Nhập số thực a. Tìm số nguyên n nhỏ nhất để tổng $S = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n}$ có giá trị lớn hơn a? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
    int n;
    float a,Tong=0;
    printf("\n  Nhap vao so thuc a:  ");
    scanf("%f",&a);
    for(n=1;;n++)
      {
        Tong=Tong+(float)1/n;
        if(Tong>a)break;
      }
    printf("  => So nguyen duong n nho nhat de tong lon hon a la:  n =
%d\n",n);
    getch();
    return main();
}
```

* *Giải thích:*

*- Thuật toán xác định giá trị của n: Ta thấy n chạy từ 1 đến $+\infty$ nên ban đầu chưa thể biết vòng lặp for thực hiện bao nhiêu lần. Vì vậy sẽ không có điều kiện trong vòng lặp for, n chạy tới đâu thì biến **Tong** được cộng dồn thêm một lượng bằng 1/n với giá trị khởi tạo bằng 0. Tính giá trị của Tong xong thì thực hiện so sánh, nếu Tong > a thì thoát khỏi vòng lặp và giá trị cần tìm là giá trị của n ngay tại thời điểm cuối trước khi thoát vòng lặp.*

* *Màn hình kết quả như sau:*



Bài 21: Cho số nguyên n (n < 2 000 000 000). Hãy tính tổng các chữ số của n?

```
#include <stdio.h>
#include <conio.h>
main()
{
    int Tong=0;
    long n;
    printf("\n  So nguyen duong n la gi the ku?:  ");
    scanf("%ld",&n);
    if(n<2000000000)
      {
        while(n>0)
         {
           Tong=Tong+n%10;
           n=n/10;
         }
        printf("  => Tong cac chu so trong n la:  %d\n",Tong);
      }
```

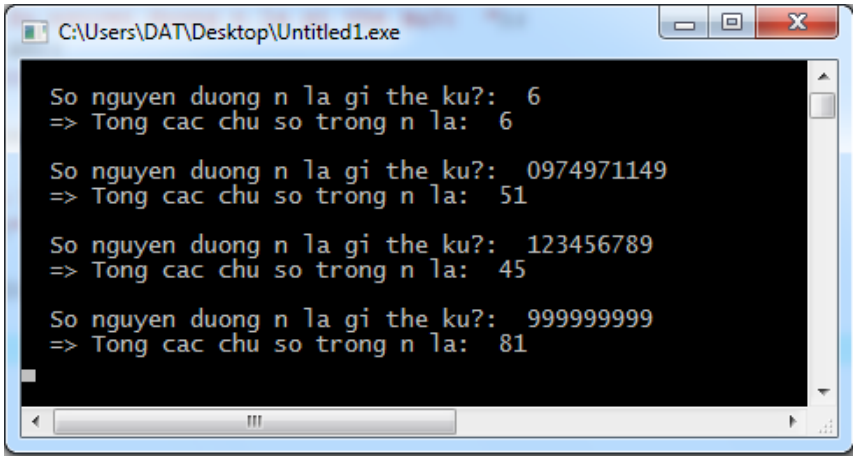
```
        else printf("   => So nay khong hop le...\n");
        getch();
        return main();
    }
```

* *Giải thích:*

a) Vì n < 2 tỉ nên phải dùng khai báo kiểu ***long*** *mới đủ dung lượng bộ nhớ. Kiểu long chiếm 4 byte (hay 32 bit) và biểu diễn giá trị từ -2 147 483 648 đến 2 147 483 648. Dấu %ld thể hiện kiểu long.*

b) Nếu số n nhập vào mà thỏa mãn < 2 000 000 000 thì biến Tong bằng số nằm ở phần đơn vị của n được tính bằng phép chia lấy dư n%10, còn lại phần nguyên được tính bằng phép chia lấy phần nguyên n/10 thì tiếp tục quay lại vòng lặp và Tong lại được cộng dồn. Cứ như thế cho tới khi n = 0 thì dừng lại và in kết quả Tong ra màn hình. VD: nhập vào số 279 thì sau vòng lặp thứ nhất Tong = 9, sau vòng thứ hai thì Tong = 9 + 7, sau vòng thứ ba thì Tong = 16 +2

* *Màn hình kết quả như sau:*



Bài 22*: Nhập vào một số nguyên dương N (N > 10 000). Hãy đưa ra màn hình số chữ số của N và tích các chữ số của N?

(Đề kiểm tra giữa kỳ lớp THCS 3, ĐHKHTN – ĐHQGHN, Thứ 4 ngày 24/10/2012, Kỳ I năm học 2012 – 2013, thời gian làm bài: 30 phút)

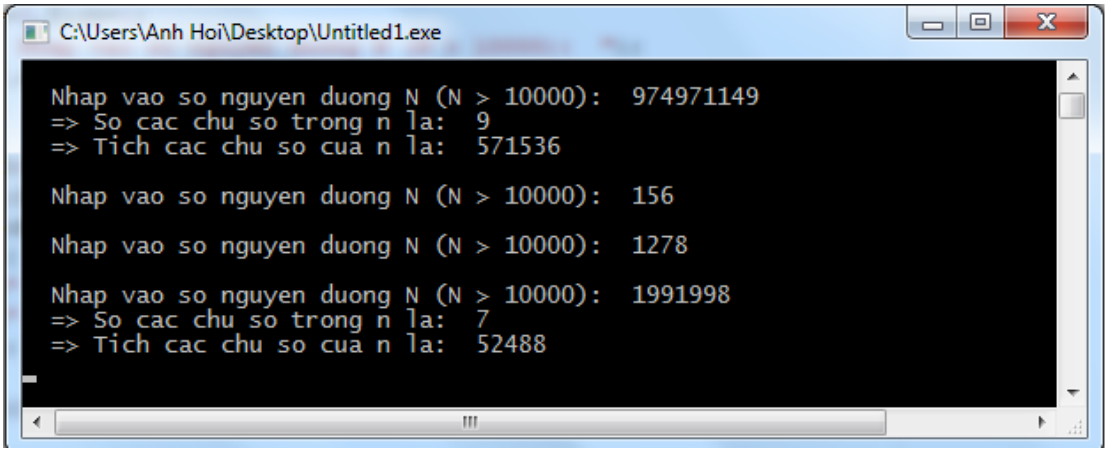
```
#include <stdio.h>
#include <conio.h>
main()
{
    long N,i,So=0,Tich=1;
    printf("\n  Nhap vao so nguyen duong N (N > 10000):  ");
```

```
        scanf("%ld",&N);
        if(N>10000)
        {
           while(N>0)
           {i=N%10;So=So+1;
           Tich=Tich*i;
           N=N/10;}
           printf("  => So cac chu so trong n la:  %ld\n",So);
           printf("  => Tich cac chu so cua n la:  %ld\n",Tich);
           getch();
        }
        return main();
    }
```

* *Giải thích:*

- Số chữ số của N được xác định bằng số lượng phép chia lấy dư của N cho 10, có bao nhiêu phép chia lấy dư của N cho 10 thì trong N có bấy nhiêu chữ số. Giá trị tích các chữ số trong N được nhân liên tiếp với giá trị khởi tạo bằng 1. Nếu xuất hiện ít nhất 1 chữ số 0 trong N thì Tích = 0. Lưu ý là kiểu khai báo dữ liệu cho trường hợp này phải là kiểu long.

* *Màn hình kết quả như sau:*



Bài 23: Tìm tất cả các số đối xứng có 5 chữ số?

```
#include <stdio.h>
#include <conio.h>
main()
```

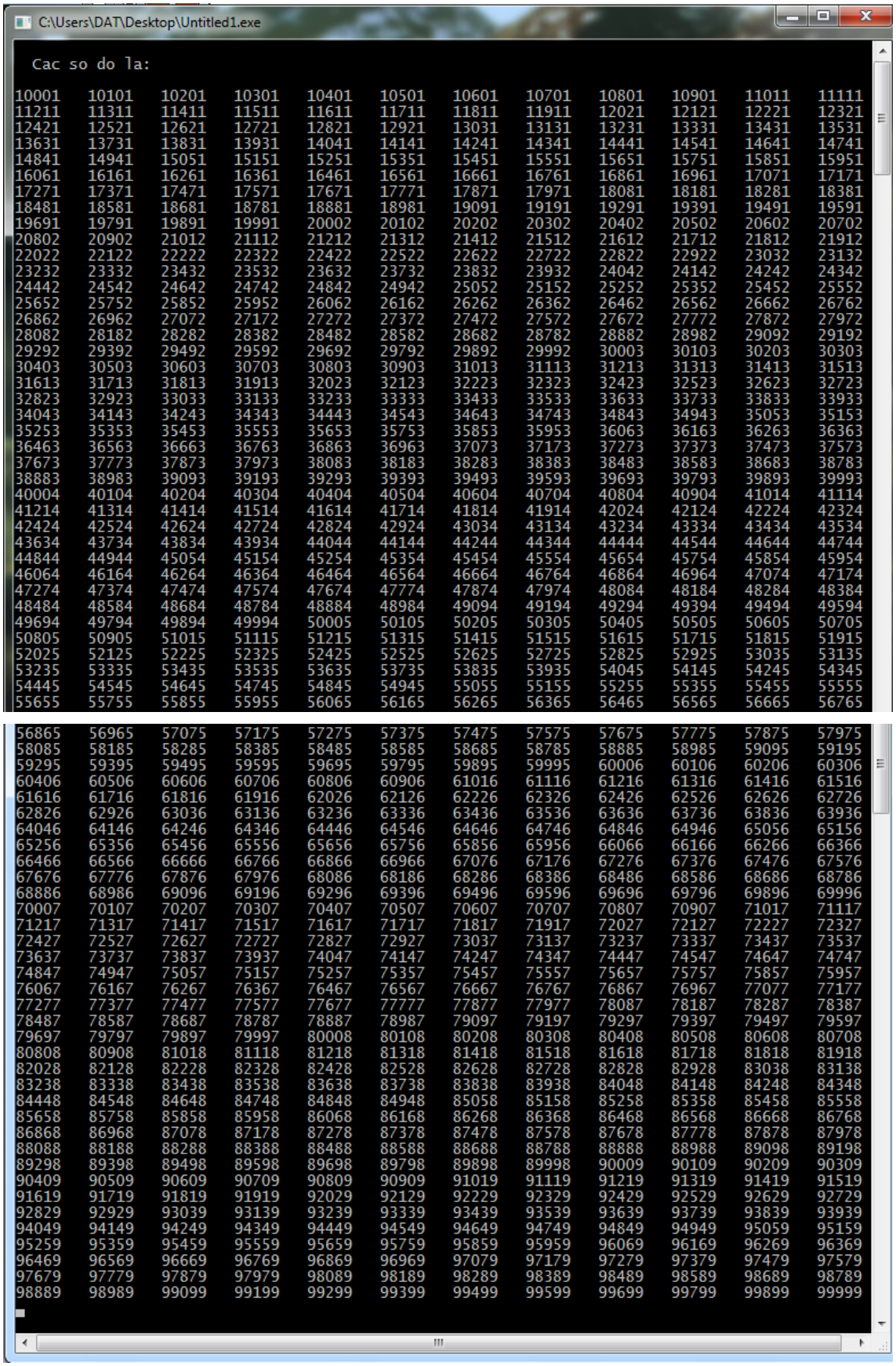
```
{
    int a,b,c;
    printf("\n  Cac so do la:\t\n\n");
    for(a=1;a<=9;a++)
      for(b=0;b<=9;b++)
        for(c=0;c<=9;c++)
    printf("%d%d%d%d%d\t",a,b,c,b,a);
    getch();
}
```

* *Giải thích:*

a) Số đối xứng là số mà các chữ số cách đều hai đầu mút là giống hệt nhau. Nếu số có 5 chữ số thì nó có dạng ***abcba****, trong đó* $a = \overline{1,9}$ *và* $b,c = \overline{0,9}$. *Theo đại số tổ hợp thì có: 9.10.10 = 900 (số đối xứng).*

b) Ở đây sử dụng các hàm for lồng nhau, ý nghĩa của nó là: với mỗi giá trị của a thì có 10 giá trị của b, với mỗi giá trị của b thì có 10 giá trị của c. Vì a là chữ số đầu tiên nên nó phải > 0 và chỉ có 9 giá trị của a. Sau mỗi một vòng lặp thì kết quả được in ra màn mình là một số đối xứng cụ thể. Hai số cạnh nhau được ngăn cách bởi dấu ***\t*** *(t = tab).*

* *Màn hình kết quả như sau:*


C:\Users\DAT\Desktop\Untitled1.exe
Cac so do la:
10001 10101 10201 10301 10401 10501 10601 10701 10801 10901 11011 11111
11211 11311 11411 11511 11611 11711 11811 11911 12021 12121 12221 12321
12421 12521 12621 12721 12821 12921 13031 13131 13231 13331 13431 13531
13631 13731 13831 13931 14041 14141 14241 14341 14441 14541 14641 14741
14841 14941 15051 15151 15251 15351 15451 15551 15651 15751 15851 15951
16061 16161 16261 16361 16461 16561 16661 16761 16861 16961 17071 17171
17271 17371 17471 17571 17671 17771 17871 17971 18081 18181 18281 18381
18481 18581 18681 18781 18881 18981 19091 19191 19291 19391 19491 19591
19691 19791 19891 19991 20002 20102 20202 20302 20402 20502 20602 20702
20802 20902 21012 21112 21212 21312 21412 21512 21612 21712 21812 21912
22022 22122 22222 22322 22422 22522 22622 22722 22822 22922 23032 23132
23232 23332 23432 23532 23632 23732 23832 23932 24042 24142 24242 24342
24442 24542 24642 24742 24842 24942 25052 25152 25252 25352 25452 25552
25652 25752 25852 25952 26062 26162 26262 26362 26462 26562 26662 26762
26862 26962 27072 27172 27272 27372 27472 27572 27672 27772 27872 27972
28082 28182 28282 28382 28482 28582 28682 28782 28882 28982 29092 29192
29292 29392 29492 29592 29692 29792 29892 29992 30003 30103 30203 30303
30403 30503 30603 30703 30803 30903 31013 31113 31213 31313 31413 31513
31613 31713 31813 31913 32023 32123 32223 32323 32423 32523 32623 32723
32823 32923 33033 33133 33233 33333 33433 33533 33633 33733 33833 33933
34043 34143 34243 34343 34443 34543 34643 34743 34843 34943 35053 35153
35253 35353 35453 35553 35653 35753 35853 35953 36063 36163 36263 36363
36463 36563 36663 36763 36863 36963 37073 37173 37273 37373 37473 37573
37673 37773 37873 37973 38083 38183 38283 38383 38483 38583 38683 38783
38883 38983 39093 39193 39293 39393 39493 39593 39693 39793 39893 39993
40004 40104 40204 40304 40404 40504 40604 40704 40804 40904 41014 41114
41214 41314 41414 41514 41614 41714 41814 41914 42024 42124 42224 42324
42424 42524 42624 42724 42824 42924 43034 43134 43234 43334 43434 43534
43634 43734 43834 43934 44044 44144 44244 44344 44444 44544 44644 44744
44844 44944 45054 45154 45254 45354 45454 45554 45654 45754 45854 45954
46064 46164 46264 46364 46464 46564 46664 46764 46864 46964 47074 47174
47274 47374 47474 47574 47674 47774 47874 47974 48084 48184 48284 48384
48484 48584 48684 48784 48884 48984 49094 49194 49294 49394 49494 49594
49694 49794 49894 49994 50005 50105 50205 50305 50405 50505 50605 50705
50805 50905 51015 51115 51215 51315 51415 51515 51615 51715 51815 51915
52025 52125 52225 52325 52425 52525 52625 52725 52825 52925 53035 53135
53235 53335 53435 53535 53635 53735 53835 53935 54045 54145 54245 54345
54445 54545 54645 54745 54845 54945 55055 55155 55255 55355 55455 55555
55655 55755 55855 55955 56065 56165 56265 56365 56465 56565 56665 56765
56865 56965 57075 57175 57275 57375 57475 57575 57675 57775 57875 57975
58085 58185 58285 58385 58485 58585 58685 58785 58885 58985 59095 59195
59295 59395 59495 59595 59695 59795 59895 59995 60006 60106 60206 60306
60406 60506 60606 60706 60806 60906 61016 61116 61216 61316 61416 61516
61616 61716 61816 61916 62026 62126 62226 62326 62426 62526 62626 62726
62826 62926 63036 63136 63236 63336 63436 63536 63636 63736 63836 63936
64046 64146 64246 64346 64446 64546 64646 64746 64846 64946 65056 65156
65256 65356 65456 65556 65656 65756 65856 65956 66066 66166 66266 66366
66466 66566 66666 66766 66866 66966 67076 67176 67276 67376 67476 67576
67676 67776 67876 67976 68086 68186 68286 68386 68486 68586 68686 68786
68886 68986 69096 69196 69296 69396 69496 69596 69696 69796 69896 69996
70007 70107 70207 70307 70407 70507 70607 70707 70807 70907 71017 71117
71217 71317 71417 71517 71617 71717 71817 71917 72027 72127 72227 72327
72427 72527 72627 72727 72827 72927 73037 73137 73237 73337 73437 73537
73637 73737 73837 73937 74047 74147 74247 74347 74447 74547 74647 74747
74847 74947 75057 75157 75257 75357 75457 75557 75657 75757 75857 75957
76067 76167 76267 76367 76467 76567 76667 76767 76867 76967 77077 77177
77277 77377 77477 77577 77677 77777 77877 77977 78087 78187 78287 78387
78487 78587 78687 78787 78887 78987 79097 79197 79297 79397 79497 79597
79697 79797 79897 79997 80008 80108 80208 80308 80408 80508 80608 80708
80808 80908 81018 81118 81218 81318 81418 81518 81618 81718 81818 81918
82028 82128 82228 82328 82428 82528 82628 82728 82828 82928 83038 83138
83238 83338 83438 83538 83638 83738 83838 83938 84048 84148 84248 84348
84448 84548 84648 84748 84848 84948 85058 85158 85258 85358 85458 85558
85658 85758 85858 85958 86068 86168 86268 86368 86468 86568 86668 86768
86868 86968 87078 87178 87278 87378 87478 87578 87678 87778 87878 87978
88088 88188 88288 88388 88488 88588 88688 88788 88888 88988 89098 89198
89298 89398 89498 89598 89698 89798 89898 89998 90009 90109 90209 90309
90409 90509 90609 90709 90809 90909 91019 91119 91219 91319 91419 91519
91619 91719 91819 91919 92029 92129 92229 92329 92429 92529 92629 92729
92829 92929 93039 93139 93239 93339 93439 93539 93639 93739 93839 93939
94049 94149 94249 94349 94449 94549 94649 94749 94849 94949 95059 95159
95259 95359 95459 95559 95659 95759 95859 95959 96069 96169 96269 96369
96469 96569 96669 96769 96869 96969 97079 97179 97279 97379 97479 97579
97679 97779 97879 97979 98089 98189 98289 98389 98489 98589 98689 98789
98889 98989 99099 99199 99299 99399 99499 99599 99699 99799 99899 99999

<u>Bài 24:</u> Tìm tất cả các số nguyên tố nhỏ hơn số N cho trước?

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int N,i,j;
    printf("\n  Nhap cho em so N kai:  ");
    scanf("%d",&N);
    printf("  => Bo em bao cac so nguyen to nho hon N la:   ");
    for(i=2;i<N;i++)
    {
    for(j=2;i%j!=0;j++);
    if(j==i)printf("%d  ",i);
    }
    printf("\n");
    getch();
    return main();
    getch();
}
```

* <u>*Giải thích:*</u>

a) Số nguyên tố là những số tự nhiên mà chỉ chia hết cho 1 và chính nó. Theo quy ước thì hai số 0 và 1 không phải là các số nguyên tố. Vì vậy ta chỉ cần kiểm tra các số từ 2 đến < N.

b) Dùng các hàm for lồng nhau để kiểm tra tính nguyên tố của i (i có giá trị từ 2 đến N – 1). Ứng với mỗi giá trị của i ta xem i có chia hết cho j hay không (j có giá trị từ 2 đến i). Vì bất kì số i nào cũng chia hết cho 1 và chính nó (i ≥ 2) nên trong vòng for thứ hai ta dùng điều kiện i không chia hết cho j. Khi i chia hết cho j thì vòng for thứ hai dừng lại và kiểm tra điều kiện tiếp theo, nếu i = j thì có nghĩa là i không chia hết cho số nào nhỏ hơn i mà chỉ chia hết cho chính nó hay i chính là số nguyên tố và ta in ra màn hình.

Kết thúc việc kiểm tra i thì tiếp tục quay lại kiểm tra giá trị i tiếp theo ở vòng for thứ nhất.

c) Thuật toán kiểm tra một số i có phải là số nguyên tố hay không thì ta không nhất thiết phải chia i cho các số từ 2 đến i – 1 mà chỉ cần kiểm tra xem i có chia hết cho ít nhất một số nào đó từ 2 đến phần nguyên của căn bậc hai của i: [$\sqrt{i}$]. Làm như thế thì chương trình phải kiểm tra ít hơn rất nhiều lần và sẽ chạy nhanh hơn. Có thể chứng minh điều này bằng toán học như sau:

Giả sử số n có hai ước tương ứng là a và b (n = a. b). Ta cần chứng minh cho $a \le \sqrt{n}$ *hoặc* $b \le \sqrt{n}$ *. Dùng phương pháp chứng minh phản chứng:*

Giả sử $a > \sqrt{n}$ *và* $b > \sqrt{n} \Rightarrow a.b > \sqrt{n} . \sqrt{n}$

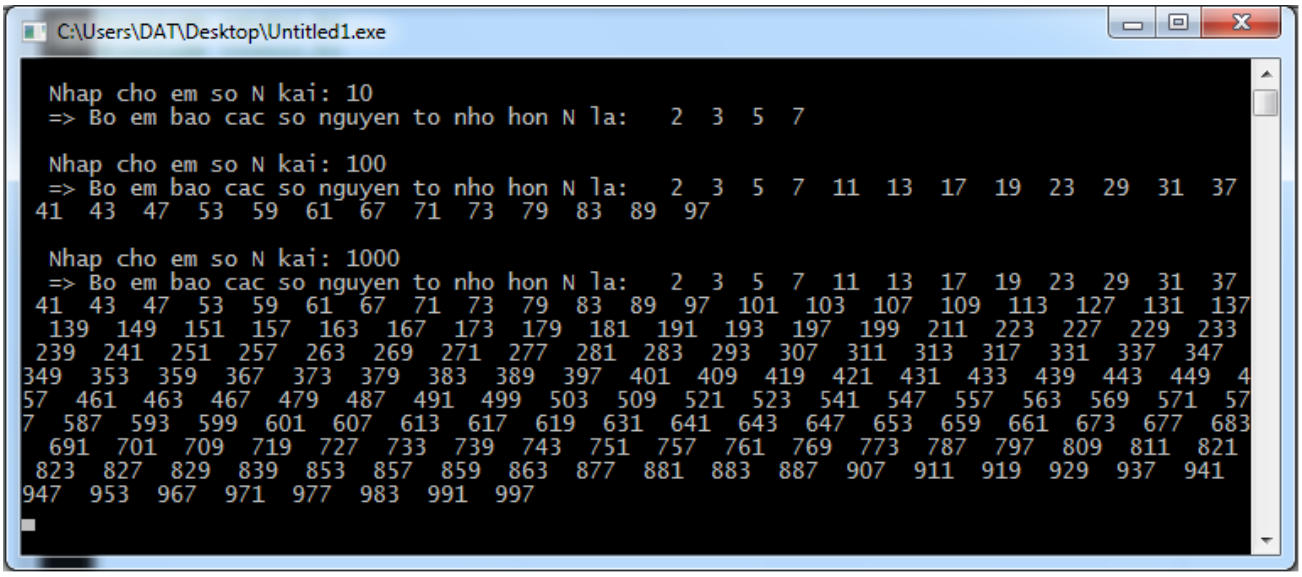
$\Leftrightarrow$ *a.b > n (điều này trái với giả thiết a.b = n)*

Vậy phải có một số a hoặc b sao cho $a \le \sqrt{n}$ *hoặc* $b \le \sqrt{n}$ *. Do đó, nếu n không phải là số nguyên tố thì nó phải chia hết cho một số mà số đó phải* $\le \sqrt{n}$ *. Vì* $\sqrt{n}$ *có thể là số hữu tỉ nên ta làm tròn thành* $a \le [\sqrt{n}]$ *hoặc* $b \le [\sqrt{n}]$ *(giới hạn đến phần nguyên căn bậc hai của n).*

$\Rightarrow$ *Thuật toán cho trường hợp này là:*

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int N,i,j,m;
    printf("\n  Nhap cho em so N kai: ");
    scanf("%d",&N);
    printf("  => Bo em bao cac so nguyen to nho hon N la:   ");
    for(i=2;i<N;i++)
    {
     m=1;
     for(j=2;j<=floor(sqrt(i));j++)
        if(i%j==0)m=0;
        if(m==1) printf("%d  ",i);
    }
    printf("\n");
    getch();
    return main();
}
```

* *Màn hình kết quả như sau:*



<u>Bài 25:</u> Viết chương trình tính tổng bình phương các số nguyên tố nằm trong đoạn [N1, N2]. Với N1 và N2 là các số nguyên và được nhập từ bàn phím? (X)

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int N1,N2,i,j,S=0,max=2;
    printf("\n  Nhap vao hai so N1 va N2:  ");
    scanf("%d%d",&N1,&N2);
    if(max<=N1)max=N1;
        for(i=max;i<=N2;i++)
        {
        for(j=2;i%j!=0;j++);
        if(j==i)
        S=S+i*i;
        }
        printf("  => Tong binh phuong cac so nguyen to nam trong doan [N1, N2] la:  %d\n",S);
    getch();
    return main();
```

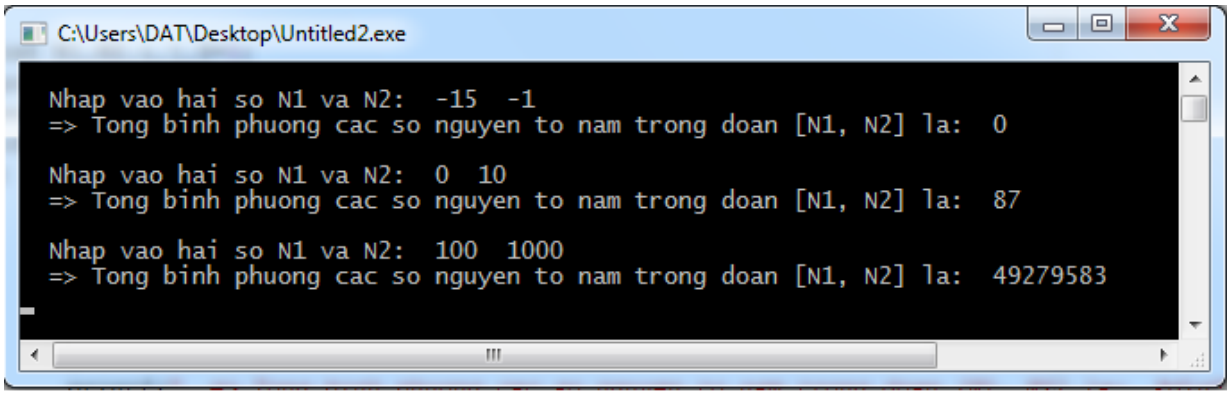
```
}
```

* *Giải thích:*

a) Vì số 0 và 1 không phải là số nguyên tố theo quy ước, do đó ta chỉ kiểm tra các số từ 2 tới N2. Ta phải phân ra làm 2 trường hợp, nếu N1 ≥ 2 thì kiểm tra tính nguyên tố của số i từ N1 đến N2, còn nếu N1 < 2 thì kiểm tra số i từ 2 đến N2. Hay nói cách khác là ta kiểm tra từ số lớn nhất trong 2 số (N1 và 2) tới N2. Tất nhiên nếu xảy ra trường hợp cả N1 và N2 đều nhỏ hơn 2 (N1 < N2 <2) thì sẽ không có số nguyên tố nào trong đoạn [N1, N2] và tổng bình phương = 0.

b) Ban đầu, kiểm tra tính nguyên tố của số i, nếu i thỏa mãn thì tổng bình phương S được cộng dồn. Kết quả được in ra màn hình là giá trị của S.

* *Màn hình kết quả như sau:*



Bài 26: Cho một số tự nhiên n. In ra màn hình n số nguyên tố đầu tiên? (X)

```
#include <stdio.h>
#include <conio.h>
int NguyenTo(int a);
main()
{
    int n,i,Dem=0;
    printf("\n  Nhap vao mot so tu nhien n:  ");
    scanf("%d",&n);
    printf("  => Suy ra %d so nguyen to dau tien la:\n\t",n);
    for(i=2;;i++)
      {
        if(NguyenTo(i)==1)
          {
```

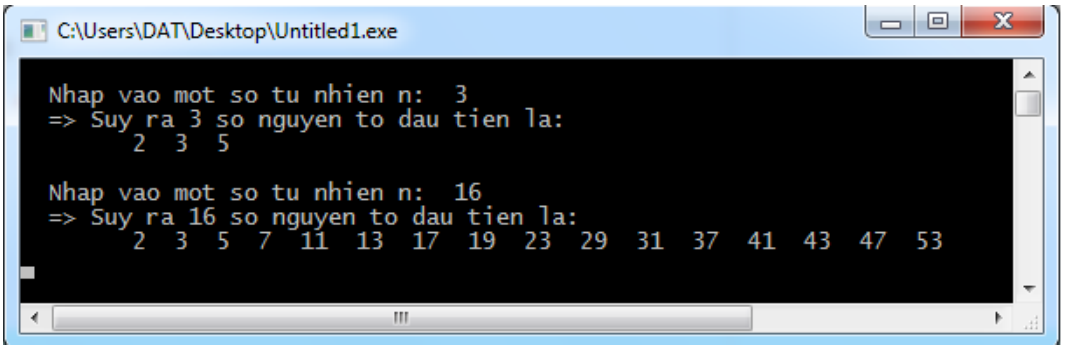
```
                printf("%d  ",i);
                Dem++;
              }
            if(Dem==n)break;
          }
        printf("\n");
        getch();
        return main();
    }
    int NguyenTo(int a)
    {
       int i,b=0;
       if(a>=2)
         {  for(i=2;a%i!=0;i++);
            if(i==a)b=1;
         }
       return (b);
    }
```

* *Giải thích:*

- In ra n số nguyên tố đầu tiên, bắt đầu từ số 2 (vì số 0 và 1 không được coi là số nguyên tố): Vòng lặp for với biến i chạy từ 2 đến vô tận, tức là không có điều kiện lặp. Khi kiểm tra một số i nào đó mà thỏa mãn là số nguyên tố thì in số đó ra màn hình và tăng biến ***Dem*** *lên 1 đơn vị, dùng hàm* ***NguyenTo()*** *để kiểm tra tính nguyên tố của một số.*

- Vì đề bài chỉ yêu cầu hiện ra n số nguyên tố đầu tiên nên khi thấy Dem = n thì phải dừng vòng lặp ngay, dùng lệnh thoát ***break****.*

* *Màn hình kết quả như sau:*



<u>Bài 27:</u> Nhập vào một số nguyên dương n, kiểm tra xem n có phải là số nguyên tố hay không? Nếu n không phải là số nguyên tố thì có thể tách n thành tổng 2 số nguyên tố được không? (X)

```
#include <stdio.h>
#include <conio.h>
int NguyenTo(int a);
main()
{
   int n,i;
   printf("\n  Nhap vao so nguyen duong n:  ");
   scanf("%d",&n);
   if(NguyenTo(n)==1)printf("  => So %d la so nguyen to.\n",n);
   else
     {
       printf("  => So %d khong phai so nguyen to.\n",n);
       printf("  => Tach so %d thanh tong 2 so nguyen to la:\n");
       for(i=2;i<=n/2;i++)
         if(NguyenTo(i)==1&&NguyenTo(n-i)==1)
           printf("\t%d = %d + %d\n",n,i,n-i);
     }
   getch();
   return main();
}
int NguyenTo(int a)
{
  int i,b=0;
  if(a>=2)
    {
      for(i=2;a%i!=0;i++);
      if(i==a)b=1;
    }
```

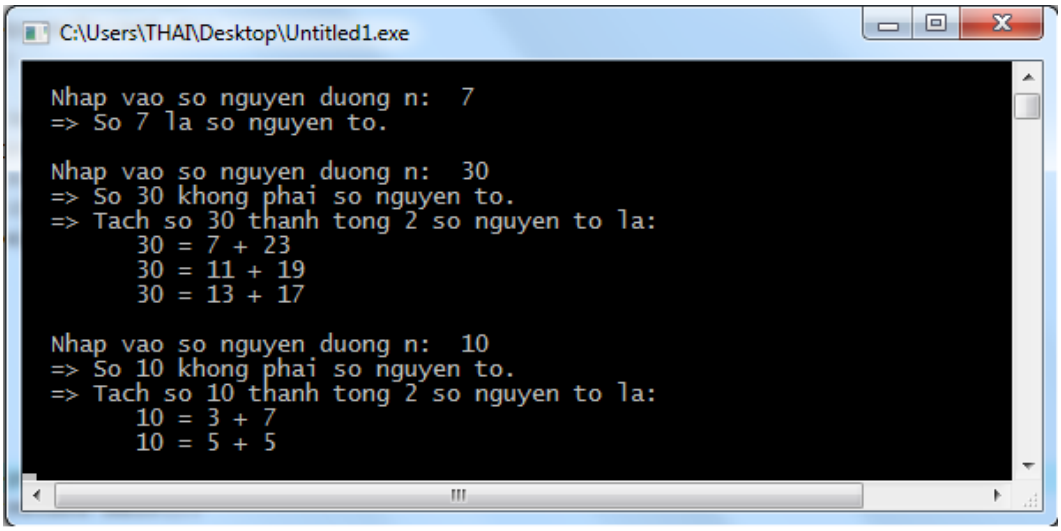
```
    return (b);
}
```

* *Giải thích:*

a) Định nghĩa thêm hàm kiểm tra xem một số nguyên dương n có phải là số nguyên tố hay không. Trong trường hợp n không phải là số nguyên tố thì ta mới tiến hành tách n thành tổng 2 số nguyên tố (vì một số nguyên tố không bao giờ tách được thành tổng 2 số nguyên tố khác).

b) Để tách số n thành tổng 2 số nguyên tố ta dùng vòng lặp for với biến i duyệt từ giá trị 2 đến giá trị n/2. Nếu giá trị i và giá trị n – i đều là các số nguyên tố thì in các giá trị đó ra màn hình trong biểu thức khai triển n thành tổng 2 số i và (n – i): n = i + (n – i).

* *Màn hình kết quả như sau:*



Bài 28: Số hoàn thiện (hay số hoàn hảo, hoàn chỉnh) là số nguyên dương có tổng các ước số nguyên dương bé hơn nó thì bằng chính nó. Hãy viết chương trình tìm tất cả các số hoàn thiện nhỏ hơn số n cho trước? (X)

```
#include <stdio.h>
#include <conio.h>
int HoanThien(int a);
main()
{
    int n,i;
    printf("\n  Nhap vao so nguyen duong n:  ");
    scanf("%d",&n);
    printf("  => Cac so hoan thien nho hon %d la:  ",n);
```

```
    for(i=1;i<n;i++)
      if(HoanThien(i)==1)printf("%d  ",i);
    printf("\n");
    getch();
    return main();
}
int HoanThien(int a)
{
    int j,k=0,Tong=0;
    for(j=1;j<a;j++)
      if(a%j==0)Tong=Tong+j;
    if(Tong==a)k=1;
    return (k);
}
```

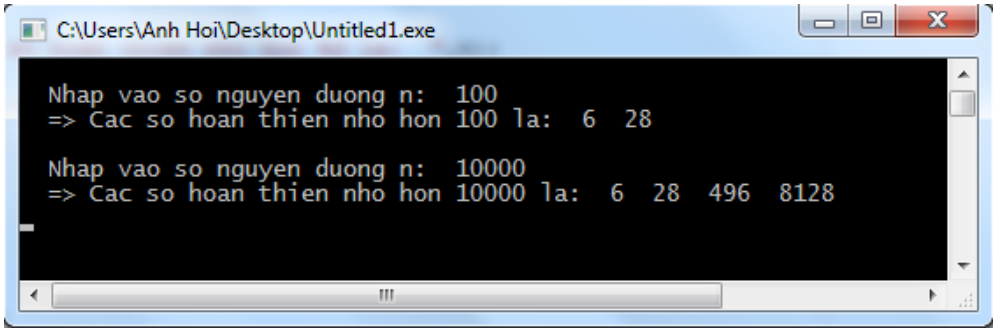
* *Giải thích:*

a) Hàm kiểm tra tính hoàn thiện của một số a: Dùng biến j duyệt từ giá trị 1 đến giá trị a – 1, nếu a chia hết cho j thì biến ***Tong*** *được cộng dồn một lượng bằng j với khởi tạo bằng 0. Xong vòng lặp nếu thấy Tong = a thì kết luận số a là số hoàn thiện (gán k = 1), ngược lại thì số a không phải số hoàn thiện (k = 0).*

b) Tìm tất cả các số hoàn thiện nhỏ hơn số n cho trước: Duyệt từ giá trị 1 đến giá trị n – 1 bằng vòng lặp for với biến i, nếu thấy i là số hoàn thiện thì in ra màn hình giá trị của i.

Các bạn có thể xem thêm về số hoàn thiện trên Wikipedia!

* *Màn hình kết quả như sau:*



Bài 29: **Viết chương trình giải bài toàn sau:**

"Vừa gà vừa chó

Bó lại cho tròn,

36 con, 100 chân chẵn".

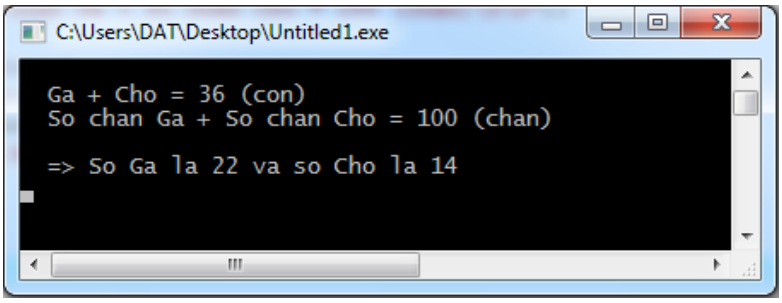
Tìm số gà, số chó? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
   int x,y;
   printf("\n  Ga + Cho = 36 (con)\n");
   printf("  So chan Ga + So chan Cho = 100 (chan)\n\n");
   for(x=1;x<36;x++)
     for(y=1;y<36;y++)
       if((x+y==36)&&(2*x+4*y==100))
          printf("  => So Ga la %d va so Cho la %d\n",x,y);
   getch();
}
```

* *Giải thích:*

- Đây là dạng bài: giải bài toán bằng cách lập hệ phương trình. Gọi số Gà là x và số Chó là y. Theo đề bài ta có: x + y = 36 (*). Mà gà có 2 chân và chó có 4 chân ⇒ 2x + 4y = 100 (**). Dùng hai vòng lặp for lồng nhau với điều kiện lặp là cả x và y phải nhỏ hơn 36 (vì x + y =36). Sau mỗi vòng lặp, nếu thỏa mãn đồng thời 2 điều kiện (*) và (**) thì hiện kết quả của x và y ra màn hình. Tất nhiên, bài này chỉ có nghiệm duy nhất là (x, y) = (22, 14).

* *Màn hình kết quả như sau:*



<u>Bài 30:</u> Hãy lập chương trình tính điểm trung bình các số dương nhập vào từ bàn phím. Số lượng điểm số cho vào là do người lập trình quyết định và việc nhập điểm này kết thúc khi gõ vào một số âm? (X)

(Bài tập 2 trang 152 Giáo trình "Ngôn ngữ lập trình C" – Quách Tuấn Ngọc)

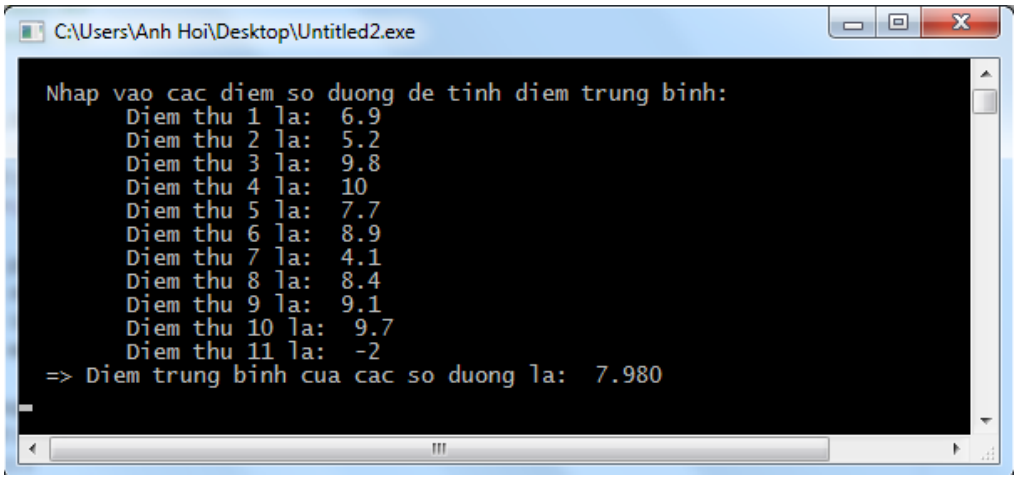
```
#include <stdio.h>
#include <conio.h>
main()
{
    int i,Dem=0;
    float n,Tong=0,TB;
    printf("\n  Nhap vao cac diem so duong de tinh diem trung binh:\n");
    for(i=1;;i++)
      {
        printf("\tDiem thu %d la:  ",i);
        scanf("%f",&n);
        if(n>=0){Tong=Tong+n;Dem++;}
        else break;
      }
    TB=Tong/Dem;
    printf("  => Diem trung binh cua cac so duong la:  %0.3f\n",TB);
    getch();
    return main();
}
```

* *<u>Giải thích:</u>*

*- Dùng vòng lặp **for()** để nhập các giá trị điểm, tính tổng các điểm dương và số lượng điểm dương. Vòng for không có điều kiện nên nó lặp vô hạn nếu như các số nhập vào đều > 0, khi nhập giá trị ≤ 0 thì thực hiện lệnh nhảy vô điều kiện **break** để thoát khỏi vòng lặp. Khi còn trong vòng lặp thì tính tổng các điểm dương sử dụng biến **Tong** với khởi tạo bằng 0 và đếm số lượng điểm dương sử dụng biến **Dem.***

*- Khi thoát khỏi vòng lặp thì tính giá trị trung bình bằng cách lấy **Tong** chia cho **Dem**. Kết quả in ra màn hình được định dạng lấy 3 chữ số sau dấu phẩy ngăn cách phần nguyên và phần thập phân.*

* *<u>Màn hình kết quả như sau:</u>*



<u>Bài 31:</u> Viết chương trình tính bảng nhân từ 1 đến 10 và in ra màn hình? (X)

(Bài tập 5 trang 153 Giáo trình "Ngôn ngữ lập trình C" – Quách Tuấn Ngọc)

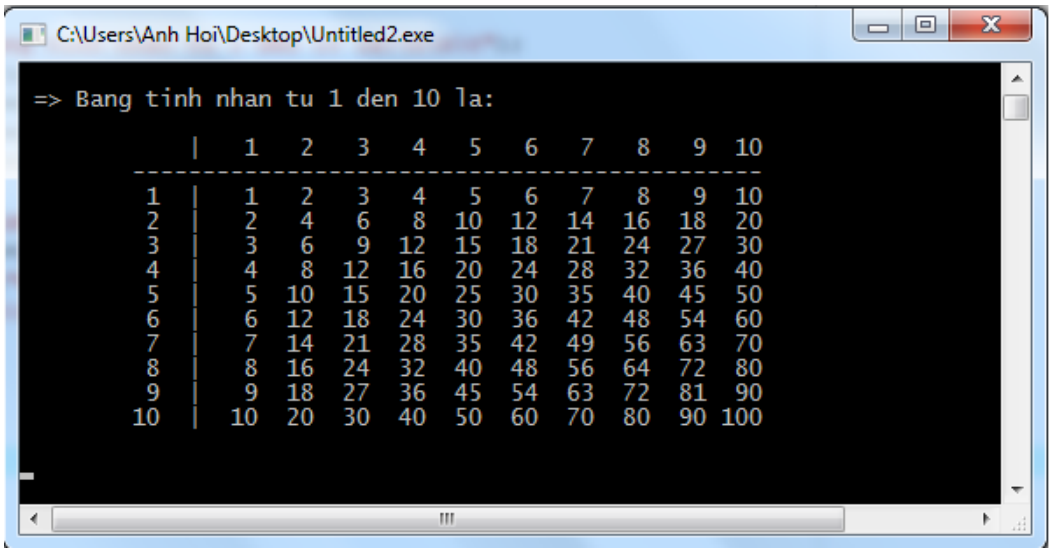
```
#include <stdio.h>
#include <conio.h>
main()
{
    int i,j;
    printf("\n => Bang tinh nhan tu 1 den 10 la:\n\n\t");
    printf("%5c",'|');
    for(i=1;i<=10;i++)printf("%4d",i);printf("\n\t");
    for(i=1;i<=45;i++)printf("-");printf("\n\t");
    for(i=1;i<=10;i++)
      {
        printf("%2d%3c",i,'|');
        for(j=1;j<=10;j++)
          printf("%4d",i*j);
        printf("\n\t");
      }
    printf("\n");
    getch();
}
```

* *<u>Giải thích:</u>*

a) Hai hàng đầu tiên chỉ là cho hiển thị ra màn hình dãy số từ 1 đến 10 và dãy kí tự "-". Vòng lặp for đầu tiên với i chạy từ 1 đến 10 và biến i chạy đến đâu thì hiển thị ra màn hình giá trị i đến đó và dành ra 4byte để hiển thị cho i. Vòng for thứ hai để hiển thị ra dãy kí tự "-" và xác định tổng số kí tự hiển thị ra sao cho dãy kí tự dài bằng dãy số ở hàng thứ nhất (xác định chủ yếu bằng phương pháp thử – sai).

*b) Từ hàng thứ 3 trở đi ta mới thực hiện tính toán bảng cửu chương, dùng vòng lặp for thứ ba với biến i chạy từ 1 đến 10, chạy đến đâu thì in ra màn hình giá trị i và kí tự '|' với định dạng cần thiết. Ứng với mỗi giá trị của i thì dùng vòng lặp for với biến j và j chạy đến đâu thì in ra màn hình có định dạng giá trị i*j đến đó. Hết mỗi giá trị của i thì dùng lệnh xuống dòng "\n" để tạo một hàng mới cho biến i tiếp theo.*

* *Màn hình kết quả như sau:*



Bài 32: Dùng vòng FOR để viết các số từ 0 đến 99 thành các hàng khác nhau sao cho mỗi hàng có 10 số? (X)

(Bài tập 6 trang 153 Giáo trình "Ngôn ngữ lập trình C" – Quách Tuấn Ngọc)

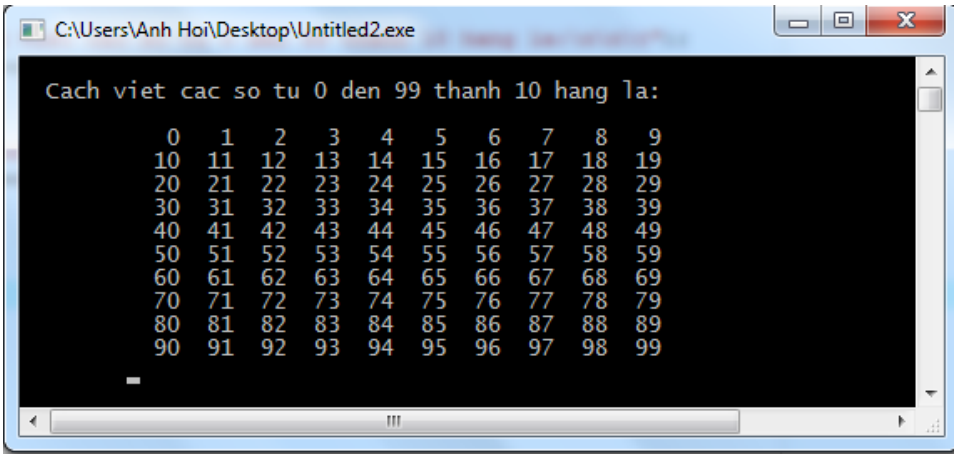
```
#include <stdio.h>
#include <conio.h>
main()
{
    int i,Dem=0;
    printf("\n  Cach viet cac so tu 0 den 99 thanh 10 hang la:\n\n\t");
    for(i=0;i<=99;i++)
      {
        Dem++;
```

```
            printf("%4d",i);
            if(Dem%10==0)printf("\n\t");
          }
       getch();
    }
```

* *Giải thích:*

- Dùng vòng lặp for() với biến i chạy từ 0 đến 99 và hiện ra thành các hàng sao cho mỗi hàng có 10 số, i chạy đến đâu thì hiện ra giá trị của i đến đó. Một điều rất quan trọng là phải xác định được vị trí xuống dòng để viết cho hàng mới. Ta thấy rằng, mỗi hàng có 10 số cho nên khi kết thúc mỗi hàng thì ta được số lượng các số là một giá trị chia hết cho 10. Sử dụng biến ***Dem*** *để đếm số lượng các số; khi vòng for chạy đến giá trị i nào đó, nếu Dem chia hết cho 10 thì thực hiện lệnh xuống dòng và cách đầu dòng 1 khoảng bằng tab.*

* *Màn hình kết quả như sau:*



Bài 33: Dùng dấu * để vẽ hình chữ nhật đặc? (X)

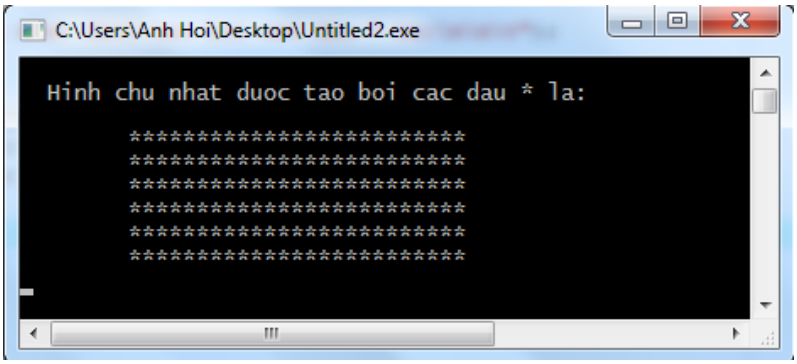
```
#include <stdio.h>
#include <conio.h>
main()
{
   int i,j;
   printf("\n  Hinh chu nhat duoc tao boi cac dau * la:\n\n\t");
   for(i=1;i<=6;i++)
     {
       for(j=1;j<=25;j++)
```

```
            printf("*");
          printf("\n\t");
        }
      printf("\r");
      getch();
  }
```

* *Giải thích:*

*- Dùng dấu * để vẽ hình chữ nhật: có tất cả 6 dòng và mỗi dòng chứa 25 dấu * tạo nên hình chữ nhật có cạnh 6x25. Dùng 2 vòng lặp for lồng nhau, với mỗi giá trị của biến i thì biến j chạy từ 1 đến 25 và chạy tới đâu in ra màn hình dấu * đến đó, kết thúc vòng for bên trong thì thực hiện lệnh xuống dòng để hiện tiếp dấu * ở các dòng còn lại. Hay nói cách khác, biến i đại diện cho dòng còn biến j đại diện cho cột. Dấu* **\r** *là lệnh đưa con chuột nhấp nháy về đầu dòng.*

* *Màn hình kết quả như sau:*



Bài 34: Dùng dấu * để vẽ các hình tam giác khác nhau:

a) Tam giác vuông với góc vuông ở phía dưới bên trái.

b) Tam giác vuông với góc vuông ở phía trên bên trái.

c) Tam giác cân với đáy ở phía dưới. (X)

(Bài tập 7 trang 153 Giáo trình "Ngôn ngữ lập trình C" – Quách Tuấn Ngọc)

```
#include <stdio.h>
#include <conio.h>
main()
{
   int i,j,Trang;
   printf("\n  Dung dau * de ve cac hinh tam giac khac nhau:\n");
   printf("  => Tam giac vuong voi goc vuong o phia duoi ben trai:\n\n\t");
```

```
    for(i=1;i<=6;i++)
      {
        for(j=1;j<=i;j++)
          printf("*");
        printf("\n\t");
      }
    printf("\n\r  => Tam giac vuong voi goc vuong o phia tren ben trai:\n\n\t");
    for(i=1;i<=6;i++)
      {
        for(j=1;j<=6-i+1;j++)
          printf("*");
        printf("\n\t");
      }
    printf("\n\r  => Tam giac can voi day o phia duoi la:\n\n");
    for(i=0;i<=5;i++)
     {
       Trang=13-i;
       for(j=1;j<=Trang;j++)
         printf(" ");
       for(j=0;j<2*i+1;j++)
         printf("*");
       printf("\n");
     }
    getch();
}
```

* *<u>Giải thích:</u>*

*a) Dùng dấu * vẽ tam giác vuông với góc vuông ở phía dưới bên trái: Giả sử tam giác gồm 6 hàng và số dấu sao của mỗi hàng bằng số thứ tự của hàng theo chiều từ trên xuống dưới (nghĩa là hàng thứ nhất có 1 dấu *, hàng thứ hai có 2 dấu *,..., hàng thứ sáu có 6 dấu *). Biến i đại diện cho hàng còn biến j đại diện cho số dấu * của hàng i. Với mỗi biến i thì biến j chạy từ 1 đến i và j chạy đến đâu thì in ra màn hình dấu * đến đó, xong mỗi hàng thì thực hiện lệnh xuống dòng "\n".*

b) Vẽ tam giác vuông với góc vuông ở phía trên bên trái: Tương tự câu a, tuy nhiên điều kiện chạy của biến j không phải từ 1 đến i mà từ 1 đến (6 – i + 1). Với i = 1 thì dòng thứ nhất có 6 dấu

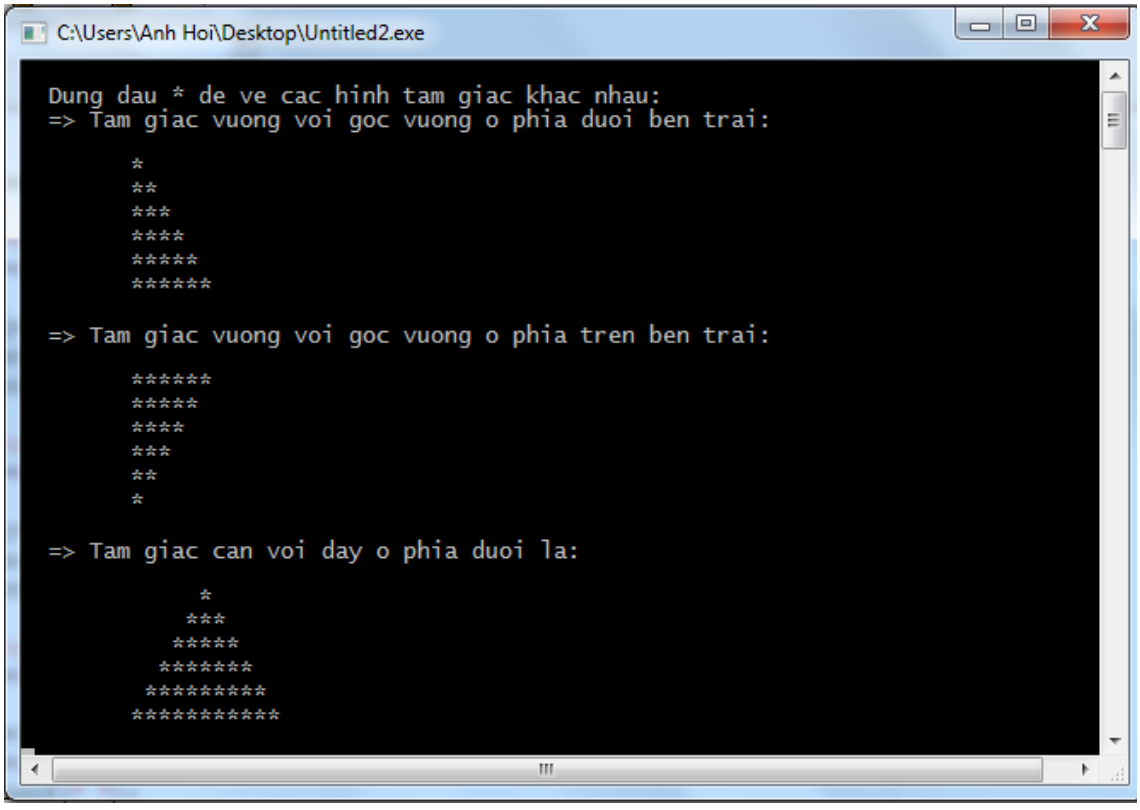
**, với i = 2 thì dòng thứ hai có 5 dấu *,..., với i = 6 thì dòng thứ sáu có 1 dấu *. In xong dấu * ở mỗi hàng thì cũng dùng lệnh xuống dòng "\n".*

*c) Vẽ tam giác cân với đáy ở phía dưới: Trường hợp này hơi khác so với hai trường hợp trên. Ta thấy số dấu * trên các hàng đều là số lẻ (2*i +1 với i =0, 1, 2, 3, 4, 5). Chính vì vậy, ở vòng for bên ngoài ta cho i chạy từ 0 đến 5 và số dấu * trên mỗi dòng được xác định bằng công thức: 2*i+1. Nhưng ở mỗi hàng thì trước khi in dấu * ta phải in kí tự trắng trước và số kí tự trắng trên mỗi hàng cũng khác nhau. Vòng for(j=1;j<=Trang;j++) printf(" ") là để in ra số kí tự trắng của hàng i, dòng sau thì ít hơn dòng trước 1 kí tự trắng. Số kí tự trắng của hàng i được xác định bằng biến* ***Trang****.*

*Giả sử khởi tạo ban đầu cho biên Trang = 13 – i, sau đó thực hiện công việc trong vòng lặp for(j=1;j<=Trang;j++) printf(" "); nếu i = 0 thì sẽ in ra 13 kí tự trắng rồi in tiếp 1 dấu * ; nếu i = 1 thì sẽ in ra 12 kí tự trắng rồi in tiếp 3 dấu *,..., nếu i = 5 thì in ra 8 kí tự trắng rồi in tiếp 11 dấu *.*

Ta có thể khởi tạo ban đầu cho biến Trang khác nhau (mình để số 13 thì nó hiện ra một tam giác cân cách mép trái của màn hình một khoảng vừa phải, đảm bảo tính thẩm mỹ và cân xứng với 2 tam giác vuông ở câu a và b). Nếu khởi tạo Trang = 5 – i thì tam giác cân hiện sát mép trái màn hình, nếu khởi tạo Trang < 5 – i thì tam giác cân hiện ra không chính xác vì khi i = 5 thì điều kiện của vòng lặp for(j=1;j<=Trang;j++) printf(" ") không được thỏa mãn và đáy của tam giác cân bị lệch sang phải.

** Màn hình kết quả như sau:*



Bài 35: Viết chương trình nhập một số nguyên dương n. In ra màn hình tam giác Pascal chiều cao n? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
   int i,j,n,a[100][100],Trang;
   printf("\n  Nhap vao mot so nguyen duong n:  ");
   scanf("%d",&n);
   for(i=1;i<=n;i++)
     {
       a[i][1]=1;
       a[i][i]=1;
     }
   for(i=3;i<=n;i++)
     for(j=2;j<i;j++)
       a[i][j]=a[i-1][j-1]+a[i-1][j];
   printf("  => Tam giac Pascal chieu cao %d la:\n\n",n);
   for(i=1;i<=n;i++)
     {
       Trang=20-2*i;
       for(j=1;j<=Trang;j++)
         printf(" ");
       for(j=1;j<=i;j++)
         printf("%4d",a[i][j]);
       printf("\n");
     }
   getch();
   return main();
}
```

* *Giải thích:*

a) Các phần tử của tam giác Pascal có dạng khối nên dùng thêm một mảng 2 chiều để lưu các phần tử này. Vấn đề quan trọng nhất của bài toán này chính là xác định được giá trị của các phần tử tương ứng. Ta thấy rằng trong tam giác Pascal thì trên dòng thứ i sẽ có i phần tử và phần tử đầu tiên cũng như phần tử cuối cùng của dòng thứ i đều bằng 1. Chính vì vậy, ta sử dụng vòng for với biến i chạy từ 1 đến n, chạy tới dòng nào thì gán giá trị đầu tiên và giá trị cuối cùng của dòng đó bằng 1 (a[i][1] = a[i][i] = 1).

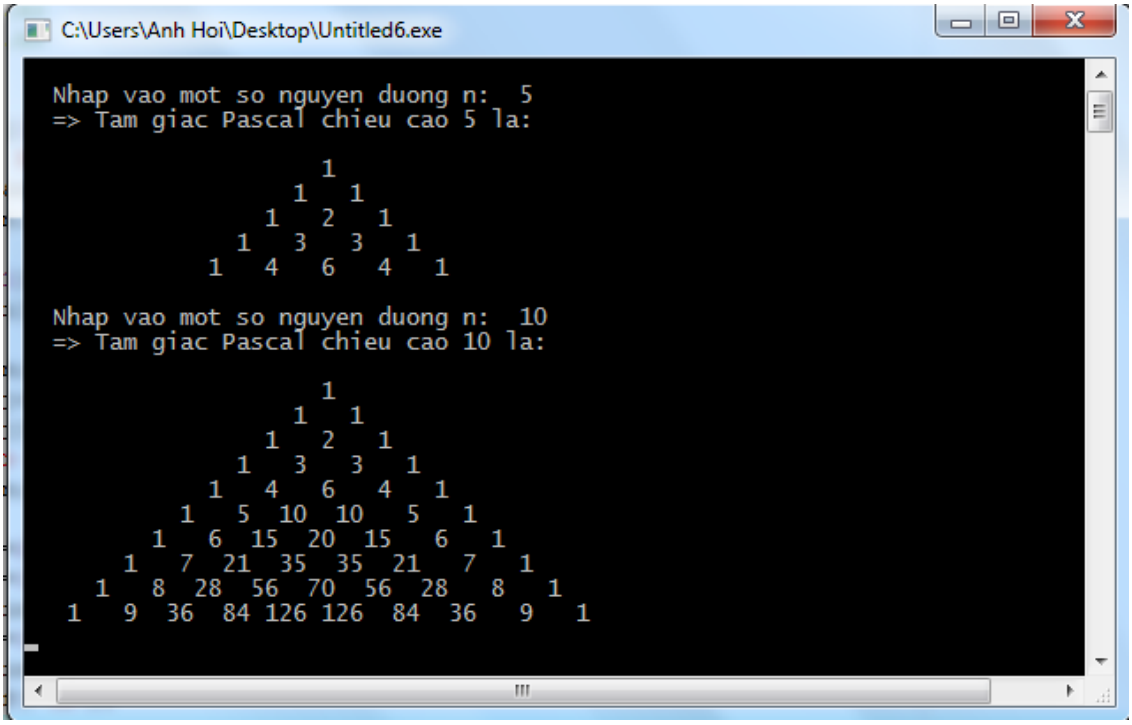
Tiếp đến là các giá trị bên trong của mỗi dòng i thì sử dụng công thức tổ hợp để tính:

$$C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$$

Hay: a[i][j] = a[i-1][j-1] + a[i-1][j] với biến j chạy từ 2 đến i – 1.

*b) Để in ra dưới dạng tam giác cân thì phải chèn thêm các kí tự trắng vào đằng trước các dòng. Dòng đầu chèn nhiều kí tự trắng nhất, dòng 2 chèn ít kí tự trắng hơn dòng 1 và dòng 3 chèn ít hơn dòng 2,...Biến **Trang** là để xác định số kí tự trắng cần chèn vào cho mỗi dòng, mình khởi tạo bằng 20 (các bạn có thể khởi tạo bằng 30, 40 tùy ý miễn sao đảm bảo tính thẩm mỹ cho hình tam giác là được) và dòng sau được chèn ít hơn dòng trước nó 2 kí tự trắng.*

* *Màn hình kết quả như sau:*



Bài 36: In ra các cách để có 200 đồng với 3 loại giấy bạc: 5đ, 2đ và 1đ. (X)

(Bài tập 14 trang 154 Giáo trình "Ngôn ngữ lập trình C" – Quách Tuấn Ngọc)

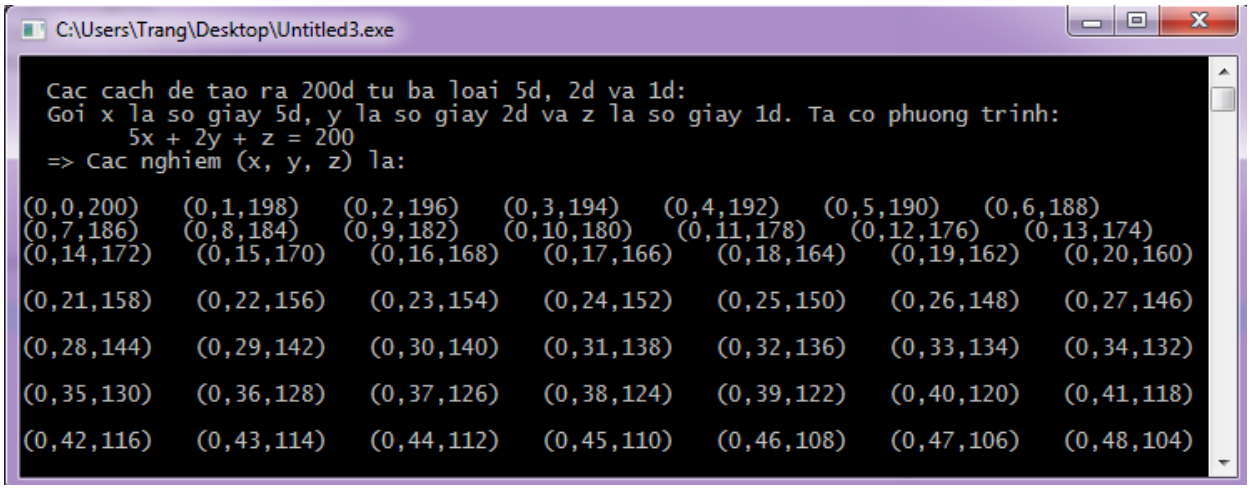
```
#include <stdio.h>
#include <conio.h>
main()
```

```
{
    int x,y,z,Dem=0;
    printf("\n  Cac cach de tao ra 200d tu ba loai 5d, 2d va 1d:\n");
    printf("  Goi x la so giay 5d, y la so giay 2d va z la so giay 1d. Ta co
phuong trinh:\n");
    printf("\t5x + 2y + z = 200\n");
    printf("  => Cac nghiem (x, y, z) la:\n\n");
    for(x=0;x<=40;x++)
      for(y=0;y<=100;y++)
        for(z=0;z<=200;z++)
          if(5*x+2*y+z==200)
            {
              Dem++;
              printf("(%d,%d,%d)   ",x,y,z);
              if(Dem%7==0)printf("\n");
            }
    getch();
}
```

* *Giải thích:*

*- Thực chất của bài này là tìm nghiệm của phương trình: 5x + 2y + z = 200 với x, y, z lần lượt là số tờ giấy bạc 5đ, 2đ và 1đ. Ta thấy, x chỉ nhận các giá trị từ 0 → 40, y chỉ nhận các giá trị từ 0 → 100 và z chỉ nhận các giá trị từ 0 → 200. Vì vậy, ta dùng 3 vòng for lồng nhau với các biến x, y, z tương ứng, nếu thấy 5*x + 2*y + z = 200 thì in ra các bộ số (x, y, z) là các nghiệm của phương trình. Sử dụng thêm biến* ***Dem*** *để sao cho mỗi dòng chỉ hiện ra 7 nghiệm (nhìn cho đẹp).*

* *Màn hình kết quả như sau: (do phương trình có tới 2081 nghiệm nên mình chỉ lấy một phần nghiệm để in ra màn hình)*



<u>Bài 37:</u> In ra trên màn hình, mỗi dòng chứa một chữ số (từ '0' đến '9') sau đó là dấu hai chấm và mã ASCII của nó. Thí dụ vài dòng đầu:

'0' : 48

'1' : 49

'2' : 50 (X)

(Bài tập 15 trang 154 Giáo trình "Ngôn ngữ lập trình C" – Quách Tuấn Ngọc)

```
#include <stdio.h>
#include <conio.h>
main()
{
    char i;
    printf("\n  => Ma ASCII cua cac chu so tu 0 den 9 la:\n\n\t");
    for(i='0';i<='9';i++)
        printf("'%c' : %d\n\t",i,i);
    getch();
}
```

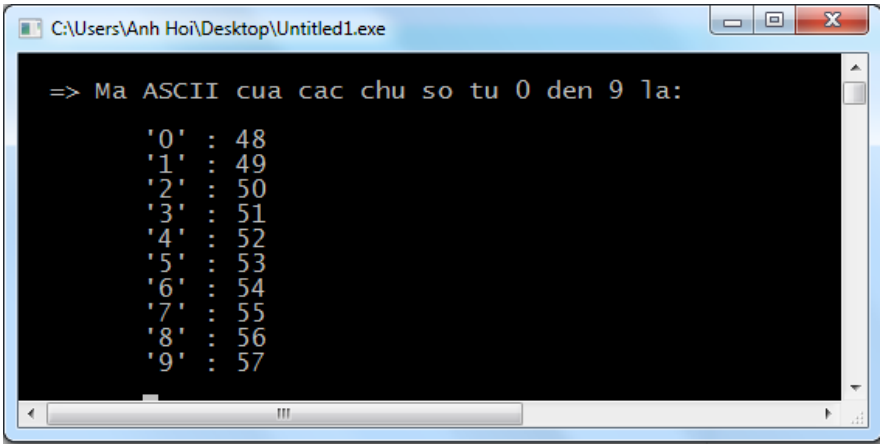
* *<u>Giải thích:</u>*

- Mã ASCII (American Standard Codes of Information Interchange) là một bộ mã chuẩn của Mỹ dùng để trao đổi thông tin. Các chữ cái, chữ số, kí tự đặc biệt và hay sử dụng thường ngày được mã hóa dưới dạng nhị phân. Đây là bộ mã xây dựng từ 8 bit nên chỉ mã hóa tối đa 2^8=256 kí tự.

- Khai báo biến i có kiểu kí tự. Ở đây ta đi tìm mã ASCII của các kí tự từ 0 đến 9 chứ không phải đi tìm mã ASCII cho các số từ 0 đến 9. Vòng lặp for duyệt từ kí tự '0' đến hết kí tự '9', duyệt

tới kí tự nào thì cho in ra kí tự đó với bên trái là định dạng kiểu kí tự và bên phải là định dạng kiểu số. Cột bên phải là mã ASCII tương ứng với các kí tự ở cột bên trái.

* *Màn hình kết quả như sau:*



Bài 38: Cho số vốn ban đầu, số tháng gửi t và lãi suất (tính theo %). Hãy tính số tiền thu được sau t tháng gửi? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i;
    float SoTien,r,t;
    printf("\n  Nhap vao so tien ban dau:  ");
    scanf("%f",&SoTien);
    printf("  Nhap vao lai suat (%%):  ");
    scanf("%f",&r);
    printf("  Nhap vao so thang can tinh:  ");
    scanf("%f",&t);
    for(i=1;i<=t;i++)
      SoTien=SoTien*(1+r/100);
    printf("  => So tien thu duoc sau %1.0f thang gui la:  %1.4f\n",t,SoTien);
    getch();
    return main();
```

```
}
```

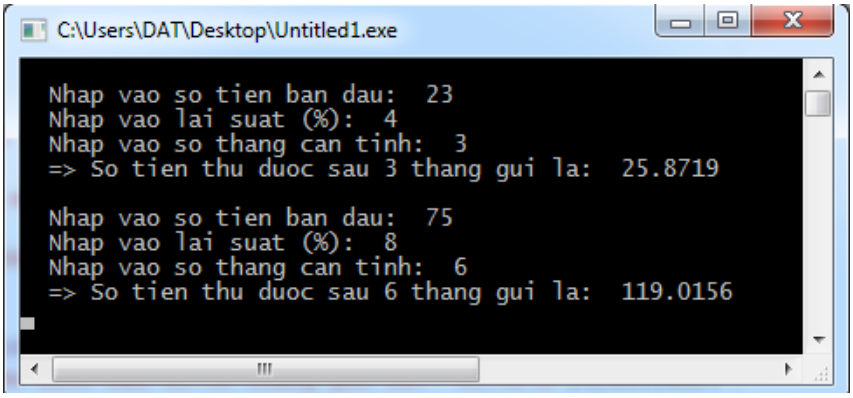
* *Giải thích:*

*a) Số tiền ban đầu nhập vào được lưu vào địa chỉ cho biến **SoTien**, rồi sau đó nhập thêm các thông tin khác như lãi suất và số tháng cần tính. Giả sử lãi suất là r % thì sau năm thứ nhất số tiền thu được cả gốc lẫn lãi là: SoTien + r %. SoTien,..., sau t năm thì số tiền cả gốc lẫn lãi thu được là SoTien. $(1 + r\ \%)^t$ Bài này tính lãi suất theo tháng chứ không phải theo năm như thực tế vẫn tính dựa vào lãi suất ngân hàng.*

b) Vòng lặp for với biến i chạy từ 1 đến t, chạy tới đâu thì tính dồn giá trị của biến SoTien đến đó và cộng dồn thêm một lượng bằng r %. SoTien

Kết thúc vòng lặp for dùng lệnh in giá trị của biến SoTien ra màn hình và đây chính là kết quả mà ta mong muốn.

* *Màn hình kết quả như sau:*



Bài 39: Bài toán lãi suất tiết kiệm: T là số tiền gửi, S là số tiền nhận được sau t tháng gửi, k là lãi suất (%). Tính và đưa ra kết quả ứng với các trường hợp sau:

- Gửi vào số tiền T ban đầu, với lãi suất k% một tháng thì sau t tháng sẽ nhận được số tiền S là bao nhiêu?

- Gửi vào số tiền T ban đầu, với lãi suất k% một tháng, để nhận được số tiền S thì phải gửi trong bao nhiêu tháng?

- Tính số tiền T gửi ban đầu với lãi suất k% một tháng để sau t tháng nhận được số tiền là S? (X)

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
main()
{
   float T,S,t,k;
```

```
    printf("\n  Truong hop 1: Cho T, k, t. Tinh S?\n");
    printf("  Nhap vao so tien ban dau T, lai suat k va so thang gui t:  ");
    scanf("%f%f%f",&T,&k,&t);
    printf("     => So tien S nhan duoc sau %.0f thang gui la: %.4f\n",t,T*pow(1+k,t));
    printf("\n  Truong hop 2: Cho T, k, S. Tinh t?\n");
    printf("  Nhap vao so tien ban dau T, lai suat k va so tien nhan S:  ");
    scanf("%f%f%f",&T,&k,&S);
    printf("     => De nhan duoc so tien tren thi phai gui so thang la: %.4f\n",log(S/T)/log(1+k));
    printf("\n  Truong hop 3: Cho k, t, S. Tinh T?\n");
    printf("  Nhap vao lai suat k, so thang gui t va so tien nhan S:  ");
    scanf("%f%f%f",&k,&t,&S);
    printf("  => So tien T phai gui ban dau la:  %.4f\n",S/pow(1+k,t));
    getch();
    return main();
}
```

* *Giải thích:*

*- Đây là bài toán cho biểu thức quan hệ giữa 4 đại lượng, sau đó cho 3 đại lượng và tìm đại lượng còn lại. Có thể giải quyết các ý nhỏ của bài toán bằng cách sử dụng vòng lặp nhưng có cách đơn giản hơn là ta có thể tìm được biểu thức tính trực tiếp giá trị chưa biết dựa vào quan hệ giữa chúng. Vì có sử dụng thêm các hàm toán học như tính logarit cơ số a của x, tính lũy thừa của cơ số a với số mũ b nên phải khai báo thêm thư viện toán học <**math.h**>*

a) TH1: Cho biết số tiền gửi ban đầu, lãi suất gửi theo tháng và số tháng gửi. Tính số tiền nhận được (Cho T, k, t và tính S):

Giả sử số tiền gửi ban đầu là T thì:

Sau 1 tháng số tiền nhận được sẽ là: $S_1 = T + T.k = T(1+k)$

Sau 2 tháng số tiền nhận được sẽ là:

$$\begin{aligned} S_2 &= (T + T.k) + k(T + kT) \\ &= T + 2kT + k^2T^2 \\ &= T(1+k)^2 \end{aligned}$$

Tổng quát, sau t tháng số tiền nhận được sẽ là: $S = T(1+k)^t$

*Vậy chỉ cần dùng hàm **pow**(1 + k, t) để tính lũy thừa bậc t của biểu thức (1 + k) rồi sau đó nhân với T là ra kết quả cần tính.*

b) TH2: Cho biết số tiền gửi ban đầu, lãi suất gửi theo tháng và số tiền nhận được. Tính số tháng gửi (Cho T, k, S và tính t):

Theo câu a ta có công thức: $S = T(1+k)^t$

Từ đó suy ra: $(1+k)^t = \frac{S}{T} \Rightarrow t = \log_{1+k}\left(\frac{S}{T}\right)$

$$\Leftrightarrow t = \frac{\log(S/T)}{\log(1+k)}$$ *(Công thức đổi sang cơ số 10)*

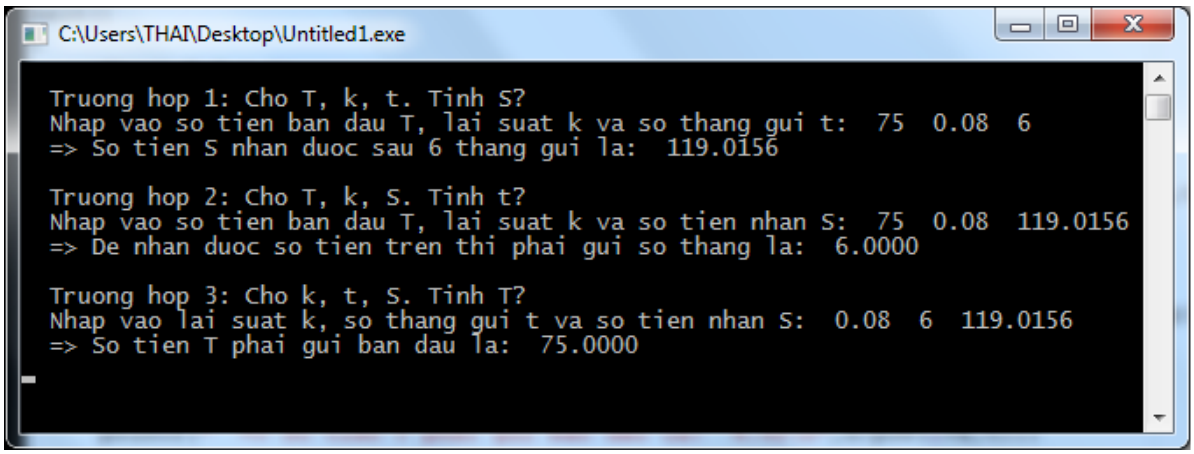
Hàm tính log10(x) được lấy trong thư viện <math.h>

c) TH3: Cho biết lãi suất gửi theo tháng, số tháng gửi và số tiền nhận được. Tính số tiền gửi ban đầu (Cho k, t, S và tính T):

Theo câu a ta có công thức: $S = T(1+k)^t$

Suy ra: $T = \frac{S}{(1+k)^t}$

* *Màn hình kết quả như sau:*



Bài 40: Nhập vào từ bàn phím một số nguyên dương n. Đưa ra màn hình biểu diễn dưới dạng nhị phân (hay dạng bit) của số nguyên dương đó? (X)

(Bài này thuộc lớp cô Huyền dạy lý thuyết vào chiều thứ 2 và thực hành sáng thứ 4)

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a,b[100],i=0,j,k;
```

```
    printf("\n  Nhap vao mot so nguyen duong a:  ");
    scanf("%d",&a);
    k=a;
    do
      {
        b[i]=a%2;
        a=a/2;
        i++;
      }
    while (a>0);
    printf("  => Ma nhi phan cua so %d la:  ",k);
    for(j=i-1;j>=0;j--)
      printf("%d",b[j]);
    printf("\n");
    getch();
    return main();
}
```

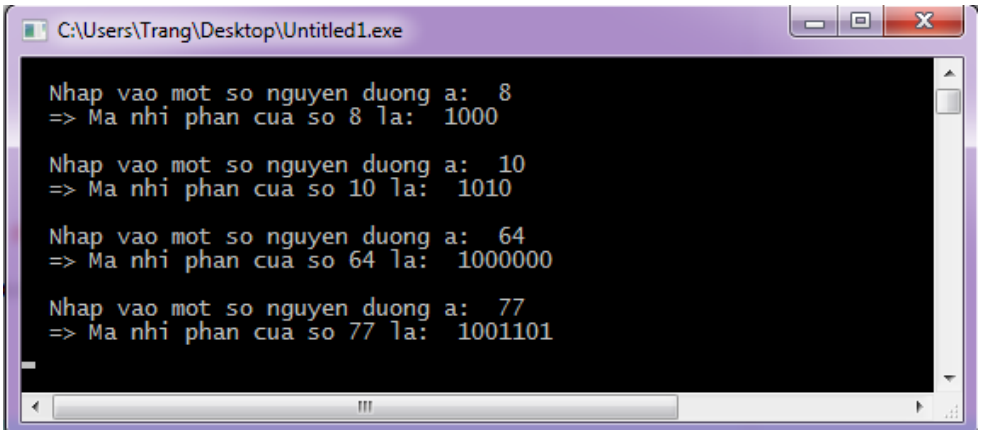
* *Giải thích:*

a) Muốn tìm mã nhị phân của một số a thì ta thực hiện phép chia a cho 2 và lấy các số dư. Sau đó, viết ngược các số dư từ cuối lên đầu sẽ được mã nhị phân của số cần tìm. Ví dụ, muốn tìm mã nhị phân của số 8 thì ta thực hiện chia 8 cho 2 bằng 4 dư 0; 4 chia 2 bằng 2 dư 0; 2 chia 2 bằng 1 dư 0; phép tính cuối cùng là 1 chia 2 bằng 0 dư 1. Vậy mã nhị phân của số 8 là các số dư tính theo chiều từ cuối lên đầu là: 1000

b) Các kết quả của phép chia lấy dư a cho 2 chính là các phần tử của mã nhị phân và ta gán chúng bằng các phần tử của mảng b nào đó. Ban đầu, thực hiện theo chiều xuôi b[i] = phần dư của phép chia a cho 2 và a bằng phần nguyên của a chia cho 2; rồi tăng biến i lên 1 đơn vị. Sau đó, kiểm tra điều kiện nếu a > 0 thì thực hiện tiếp vòng lặp sau, không thỏa mãn thì thoát khỏi vòng lặp.

c) Cuối cùng in ra các phần tử của mảng b mà ta vừa gán nhưng in theo chiều ngược lại của phép chia bên trên. Giá trị bắt đầu in không phải là giá trị b[i] mà là giá trị b[i-1] cho tới giá trị b[0].

* *Màn hình kết quả như sau:*



<u>Bài 41:</u> Viết chương trình:

- Lập hàm tính trung bình cộng của của ba số nguyên x, y, z.

- Nhập vào 3 số nguyên a, b, c. Gọi hàm vừa lập vào chương trình chính, tính và in ra trung bình cộng của 3 số nguyên a, b, c. (X)

```
#include <stdio.h>
#include <conio.h>
float TBC(int x, int y, int z)
{
   return (x+y+z)/3.0;
}
main()
{
   int a,b,c;
   printf("\n  Nhap vao 3 so nguyen a, b, c:  ");
   scanf("%d%d%d",&a,&b,&c);
   printf("  => Trung binh cong cua 3 so vua nhap la:  %0.4f\n",TBC(a,b,c));
   getch();
   return main();
}
```

* *<u>Giải thích:</u>*

*a) Hàm **TBC(x, y, z)** là hàm tính trung bình cộng của 3 số và có kiểu trả về là số thực (float). Các biến x, y, z trong hàm TBC() đều là các biến cục bộ, tức là chỉ sử dụng được trong hàm này mà không sử dụng được ở các hàm khác như hàm chính main(). Giá trị trả về cho hàm là*

(a+b+c)/3.0 và phải để mẫu số có dạng 3.0 để lấy số thực nếu không kết quả sẽ chỉ có phần nguyên. Nếu không để như thế thì phải ép kiểu cho nó như sau: return (float) (x + y + z)/3;

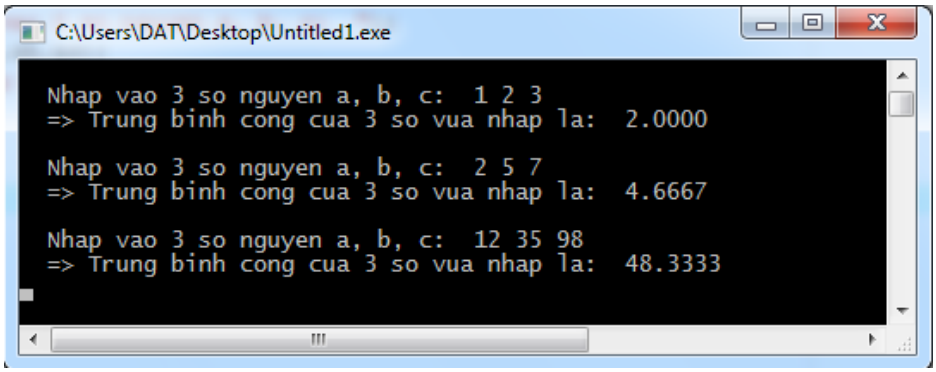
Hàm con có thể được viết trước hoặc sau hàm chính main(). Nếu hàm con được viết sau hàm chính thì nên khai báo prototype (nguyên mẫu) trước để tránh trường hợp lỗi chương trình (đối với những chương trình có nhiều hơn 1 hàm con). Khai báo prototype là khai báo tên của hàm con được sử dụng trong chương trình chính và phải để trước main. Ví dụ:

```
#include <stdio.h>
float TBC(int x, int y, int z);
main()  {…….}
```

*b) Trong chương trình chính **main()** thì hàm TBC() được gọi vào để tính trung bình cộng cho 3 số a, b, c và để vào vị trí cần hiện ra trong lệnh printf(). Dấu "%0.4f" biểu thị rằng chỉ lấy đến 4 chữ số sau dấu phẩy, nếu chỉ dùng "%f" thì máy sẽ mặc định lấy 6 chữ số sau dấu phẩy ngăn cách phần nguyên và phần thập phân.*

Bản chất của hàm trong C cũng giống như hàm trong toán học thông thường. Như hàm trên thì tương đương với hàm $u = f(x,y,z) = \frac{x+y+z}{3}$. *Khi ta thay x, y, z lần lượt bằng a, b, c thì qua ánh xạ f sẽ được một giá trị cụ thể cho hàm u. Công việc thay x, y, z bằng a, b, c được gọi là truyền tham số a, b, c cho hàm.*

* *Màn hình kết quả như sau:*



Bài 42: Viết chương trình:

- Nhập 3 số a, b, c. Kiểm tra nếu a = 0 thì nhập lại cho đến khi a ≠ 0.

- Lập hàm tìm nghiệm thực của phương trình bậc hai $ax^2 + bx + c = 0$, gọi vào chương trình để tìm nghiệm của một phương trình bất kì. In ra màn hình có định dang kết quả vừa tìm được. (X)

```
#include <stdio.h>
#include <conio.h>
void GiaiPTBH(float a,float b,float c,float *x1,float *x2,float *CoNghiem)
```

```
{
   int Del;
   Del=b*b-4*a*c;
   if(Del>0)
     {
       *CoNghiem=1;
       *x1=(-b-sqrt(Del))/2/a;
       *x2=(-b+sqrt(Del))/2/a;
     }
   else if(Del==0)
     {*CoNghiem=1;*x1=-b/2/a;*x2=-b/2/a;}
   else
      *CoNghiem=0;
}
main()
{
   float a,b,c,x1,x2,CoNghiem;
   printf("\n  Nhap vao cac he so a, b, c de giai phuong trinh bac hai:  ");
   nhaplai:scanf("%f%f%f",&a,&b,&c);
   if(a!=0)
     {
       GiaiPTBH(a,b,c,&x1,&x2,&CoNghiem);
       if(CoNghiem==1)
          printf("  => Phuong trinh co nghiem thuc la:  x1 = %f   va x2 =
%f\n",x1,x2);
       else printf("  => Phuong trinh khong co nghiem thuc!\n");
     }
   else
     {printf("  Nhap lai cac he so:  ");goto nhaplai;}
   getch();
   return main();
}
```

* *Giải thích:*

*Đối với bài này, mình sử dụng biến con trỏ để làm vì hàm **GiaiPTBH** có nhiều hơn một giá trị trả về. Nếu không sử dụng biến con trỏ để lấy nghiệm của phương trình thì trong thân hàm con ta phải cho hiện luôn ra nghiệm x_1 và x_2*

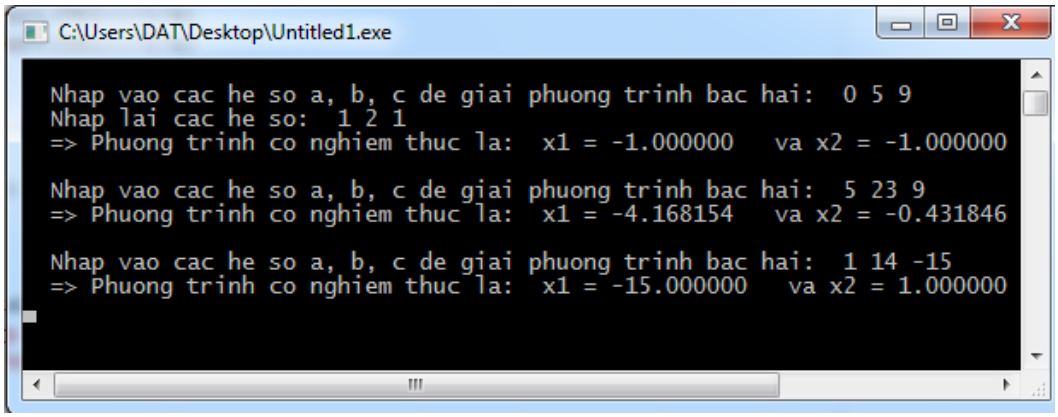
*Thực hiện nhập hệ số a, b, c mà thấy a ≠ 0 thì mới làm các công việc sau đó cũng như gọi chương trình con vào chương trình chính. Nếu a = 0 thì tiến hành nhập lại, sử dụng lệnh nhảy vô điều kiện **goto** để nhảy tới chỗ gán nhãn **nhaplai**.*

*a) Hàm **GiaiPTBH()** có 3 tham trị là a, b, c và 3 tham biến là x1, x2, CoNghiem. Hàm này thao tác trên 3 tham trị và trả về kết quả cho 3 tham biến. Nếu thấy Δ ≥ 0 thì gán giá trị *CoNghiem = 1 và thực hiện tính các giá trị *x1 cũng như *x2, ngược lại nếu Δ < 0 thì ta gán giá trị *CoNghiem = 0 và không tính *x1 cũng như *x2*

Các tham số hình thức của hàm thì mình chọn giống với các tham số thực tế để cho dễ hiểu hơn.

b) Trở lại chương trình chính: Sau khi thực hiện xong hàm GiaiPTBH mà thấy CoNghiem = 1 (tức là phương trình có nghiệm) thì in ra màn hình các nghiệm tương ứng, còn nếu thấy CoNghiem = 0 (tức là phương trình vô nghiệm) thì kết luận "Phương trình đã cho không có nghiệm thực".

* *Màn hình kết quả như sau:*



Bài 43: Tính giai thừa các số từ 1 đến 10, có sử dụng hàm?

```
#include <stdio.h>
#include <conio.h>
int GiaiThua(int i);
main()
{
    int i;
    printf("  \n   => Giai thua cua cac so tu 1 den 10 la: \n\n");
```

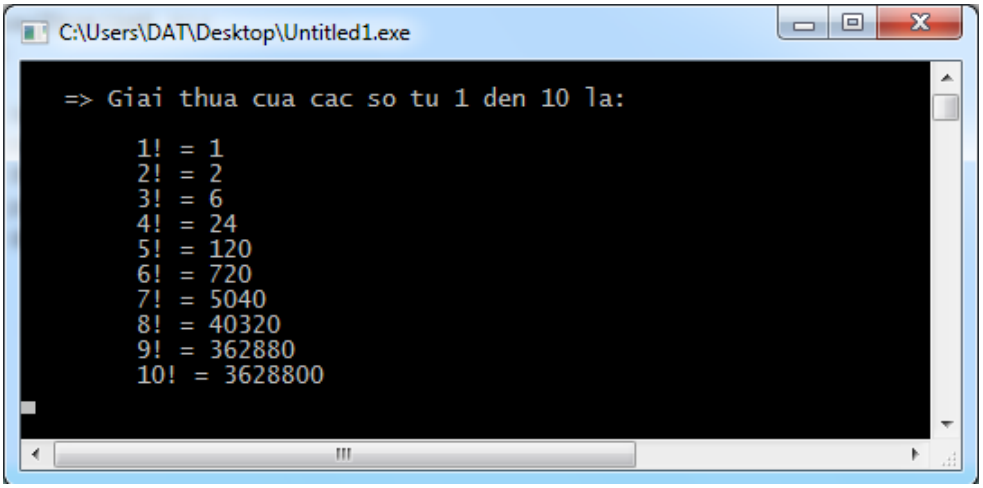
```
    for(i=1;i<=10;i++)
    printf("\t%d! = %d\n",i,GiaiThua(i));
    getch();
}
int GiaiThua(int i)
{
    if(i<=1)return 1;
    else return i*GiaiThua(i-1);
}
```

* *Giải thích:*

a) Tính và in ra màn hình giai thừa của các số từ 1 đến 10 thì ta dùng vòng lặp xác định for. Kết quả trả về là GiaiThua(i) với i là một số nào đó nằm trong đoạn [1, 10] và ***GiaiThua()*** *là một hàm mà ta định nghĩa trước khi sử dụng.*

b) Hàm GiaiThua() là hàm tính giai thừa của số i. Khi i bằng 0 hoặc 1 thì giai thừa của i là 1. Khi i > 1 thì ta có: i! = i. (i – 1)! và tiếp tục quay lại tính giai thừa của i – 1 (đệ quy gọi lại chính hàm đó. Đệ quy gọi lại hàm cũng gần tương tự như vòng lặp for quay lại thực hiện chính mình khi điều kiện còn thỏa mãn).

* *Màn hình kết quả như sau:*



<u>Bài 44:</u> Viết hàm tính giá trị giai thừa của một số nguyên. Sử dụng hàm để tính các giá trị của bài toán đếm tổ hợp sau (với n và k được nhập từ bàn phím):

- Hoán vị của n phần tử: $P_n = n!$

- Hoán vị vòng quanh của n phần tử: $Q_n = (n+1)!$

- Chỉnh hợp không lặp chập k của n phần tử: $A_n^k = \frac{n!}{(n-k)!}$

- Chỉnh hợp lặp chập k của n phần tử: $F_n^k = n^k$

- Tổ hợp chập k của n phần tử: $C_n^k = \frac{n!}{k!\,(n-k)!}$ **(X)**

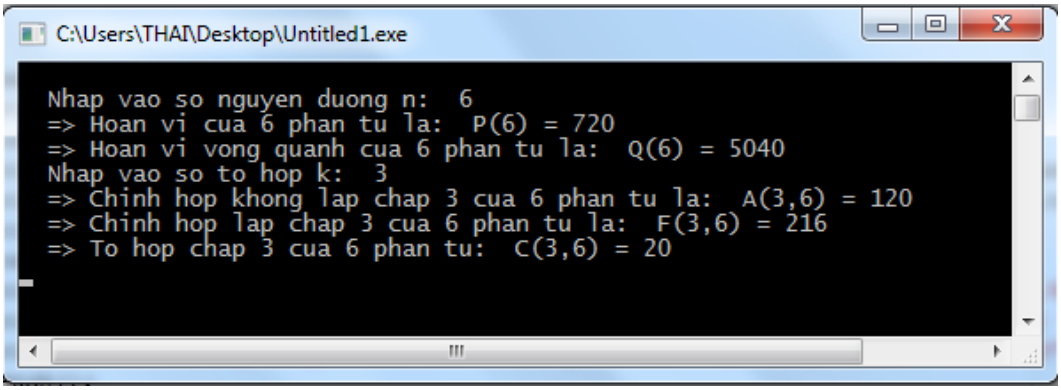
```
#include <stdio.h>
#include <conio.h>
double GiaiThua(int n);
main()
{
    double n,k;
    printf("\n  Nhap vao so nguyen duong n:  ");
    scanf("%lf",&n);
    printf("  => Hoan vi cua %.0lf phan tu la:  P(%.0lf) =
%.0lf\n",n,n,GiaiThua(n));
    printf("  => Hoan vi vong quanh cua %.0lf phan tu la:  Q(%.0lf) =
%.0lf\n",n,n,GiaiThua(n+1));
    printf("  Nhap vao so to hop k:  ");
    scanf("%lf",&k);
    printf("  => Chinh hop khong lap chap %.0lf cua %.0lf phan tu la:
A(%.0lf,%.0lf) = %.0lf\n",k,n,k,n,GiaiThua(n)/GiaiThua(n-k));
    printf("  => Chinh hop lap chap %.0lf cua %.0lf phan tu la:  F(%.0lf,%.0lf)
= %.0lf\n",k,n,k,n,pow(n,k));
    printf("  => To hop chap %.0lf cua %.0lf phan tu:  C(%.0lf,%.0lf) =
%.0lf\n",k,n,k,n,GiaiThua(n)/GiaiThua(k)/GiaiThua(n-k));
    getch();
    return main();
}
double GiaiThua(int n)
```

```
{
    if(n<=1)return 1;
    else return n*GiaiThua(n-1);
}
```

* *Giải thích:*

*- Viết thêm chương trình con (hàm) để tính giai thừa của một số, vì giá trị giai thừa tăng lên rất nhanh nên để kiểu **double** để tránh trường hợp máy tính không đủ dung lượng bộ nhớ cho việc lưu trữ kết quả tính được. Sau đó, gọi chương trình con vào để tính giá trị cho các biểu thức tương ứng với yêu cầu của đề bài.*

* *Màn hình kết quả như sau:*



Bài 45: Tính giá trị n!! với n được nhập từ bàn phím, có sử dụng hàm? (X)

```
#include <stdio.h>
#include <conio.h>
double GiaiThua(int n);
main()
{
    int n;
    printf("\n  Nhap vao so n:  ");
    scanf("%d",&n);
    GiaiThua(n);
    printf("  => Gia tri n!! la:  %lf\n",GiaiThua(n));
    getch();
    return main();
}
```

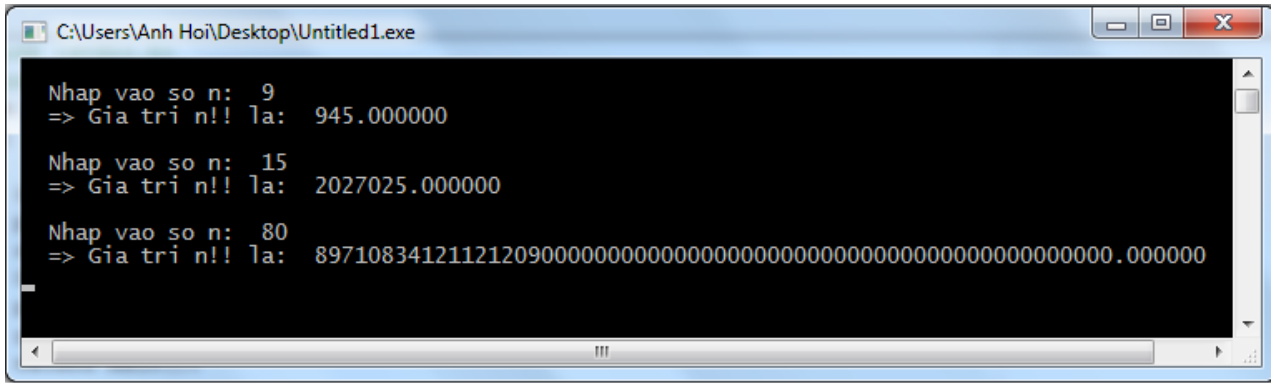
```
double GiaiThua(int n)
{
        if(n<=1)
        return 1;
        else return n*GiaiThua(n-2);
}
```

* *Giải thích:*

- *Ta có:* $n!! = \begin{cases} 2.4.6.8...n & (n\ chan) \\ 1.3.5.7...n & (n\ le) \end{cases}$

Ta thấy rằng: n!! = n.(n-2).(n-4)…. và cứ thế lặp lại cho tới khi n = 0 hoặc n = 1 thì dừng lại. Bài này gần giống bài tính giai thừa các số từ 1 đến 10, chỉ cần thay (n-1)! bằng (n-2)!. Phải để khai báo kiểu ***double*** *vì giá trị n!! tăng lên rất nhanh.*

* *Màn hình kết quả như sau:*



Bài 46: Cho số nguyên dương N, tìm chữ số lớn nhất của N?

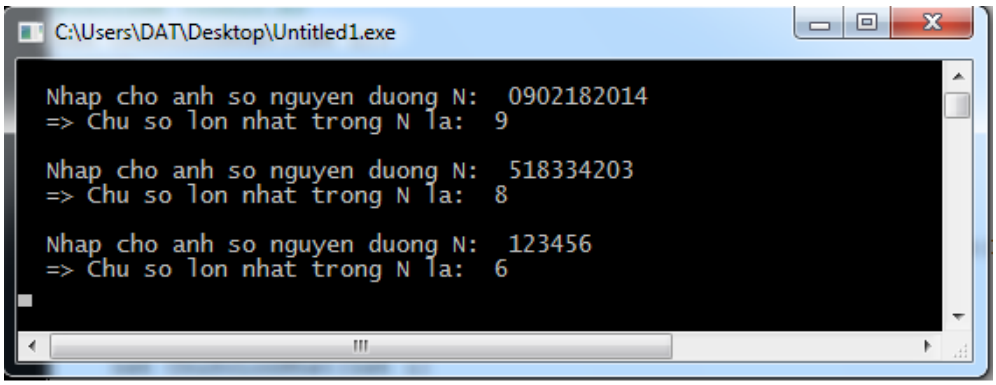
```
#include <stdio.h>
#include <conio.h>
int ChuSoLonNhat(int i);
main()
{
    int N;
    printf("\n  Nhap cho anh so nguyen duong N:  ");
    scanf("%d",&N);
    ChuSoLonNhat(N);
```

```
    printf("  => Chu so lon nhat trong N la:  %d\n",ChuSoLonNhat(N));
    getch();
    return main();
}
  int ChuSoLonNhat(int i)
   {
     int j,max=0;
     while (i>0)
     {
          j=i%10;
          if(max<=j)max=j;
          i=i/10;
     }
     return (max);
   }
```

* *Giải thích:*

*- Đưa thêm hàm **ChuSoLonNhat(i)** vào trong hàm chính main(). Hàm mới định nghĩa ChuSoLonNhat(i) có nhiệm vụ tìm ra chữ số nào là lớn nhất trong tất cả các chữ số bằng vòng lặp không xác định while. Sau mỗi phép chia lấy dư ta đều so sánh với biến max được gán giá trị khởi tạo là 0, nếu biến max ≤ kết quả của phép chia lấy dư thì giá trị của max lại được cập nhật mới. Kết quả max cuối cùng được trả về cho hàm mà ta định nghĩa.*

* *Màn hình kết quả như sau:*



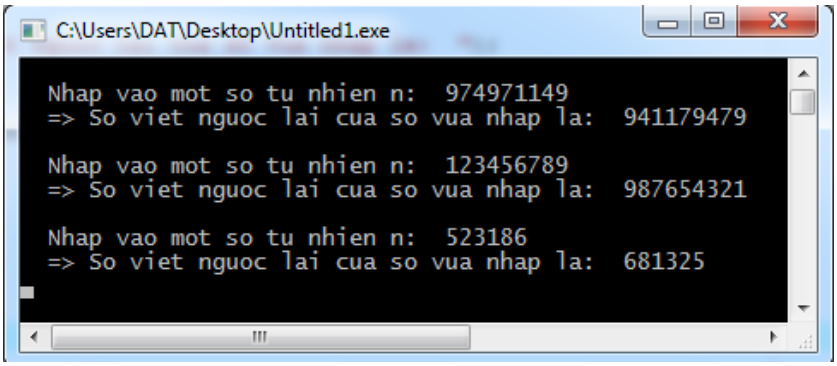
Bài 47: Cho một số tự nhiên, in ra số có các chữ số theo thứ tự ngược lại của số đã cho? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
   int n;
   printf("\n  Nhap vao mot so tu nhien n:  ");
   scanf("%d",&n);
   printf("  => So viet nguoc lai cua so vua nhap la:  ");
   while (n>0)
     {
       printf("%d",n%10);
       n=n/10;
     }
   printf("\n");
   getch();
   return main();
}
```

* *Giải thích:*

- Số mà có các chữ số theo thứ tự ngược lại của số nhập vào: Ta phải tách các chữ số của số đã nhập, dùng phép chia lấy phần dư của n cho 10 và kết quả nhận được sẽ cho hiển thị luôn ra màn hình. Sau đó gán n bằng phần nguyên của phép chia n cho 10 và tiếp tục tách tiếp các chữ số còn lại. Tách đến đâu in ra màn hình đến đó nên ta được một số mới có các chữ số viết theo chiều ngược lại của số đã cho.

* *Màn hình kết quả như sau:*



Bài 48: Tìm tất cả các số có ba chữ số $\overline{abc}$ thỏa mãn: $a^2 + b^2 = c^2$ (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i;
    printf("\n  => Cac so co 3 chu so dang abc thoa man a^2 + b^2 = c^2
la:\n\t");
    for(i=100;i<=999;i++)
      if(Pitago(i)==1) printf("%d  ",i);
    getch();
}
int Pitago(int n)
{
    int i,k=0,x,y,Tong=0;
    y=n%10;
    n=n/10;
    while(n>0)
      {
        x=n%10;
        Tong=Tong+x*x;
        n=n/10;
      }
    if(y*y==Tong) k=1;
    return (k);
}
```

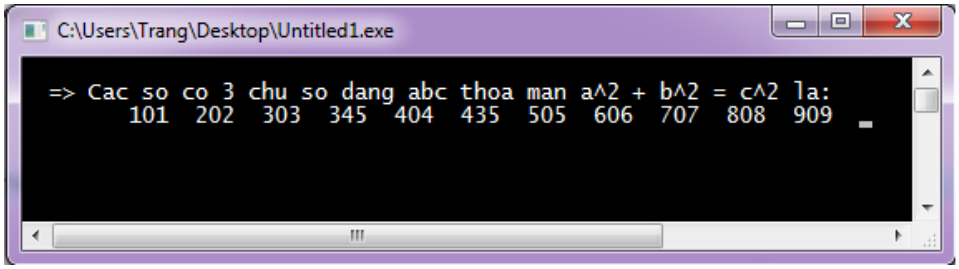
* *Giải thích:*

*- Ta kiểm tra tất cả các số có 3 chữ số dạng $\overline{abc}$ từ 100 đến 999, nếu số nào thỏa mãn tổng bình phương của 2 chữ số đầu bằng bình phương của chữ số thứ 3 thì in ra màn hình số đó, sử dụng hàm **Pitago()** để kiểm tra điều này. Nếu thấy $a^2 + b^2 = c^2$ (1) thì trả về giá trị 1 cho hàm Pitago, ngược lại trả về giá trị 0.*

- Trong hàm Pitago, ban đầu tách chữ số thứ 3 ra bằng phép chia lấy phần dư của n cho 10 và gán cho giá trị của biến y. Tiếp theo n được gán bằng phần nguyên của phép chia n cho 10 để được một số chỉ gồm chữ số thứ nhất và chữ số thứ 2. Ta lại tách tiếp 2 chữ số còn lại và tách tới đâu thì bình phương số đã tách lên rồi cộng thêm giá trị của biến ***Tong****. Cuối cùng thực hiện so sánh, nếu thấy Tong = y thì kết luận số đã kiểm tra thỏa mãn tính chất (1) và gán k=1.*

Trả về giá trị k cho tên hàm.

* *Màn hình kết quả như sau:*



Bài 49: Tìm tất cả các số có 3 chữ số $\overline{abc}$ thỏa mãn: $\overline{abc} = a^3 + b^3 + c^3$ (VD: 153)

```
#include <stdio.h>
#include <conio.h>
int TongLapPhuong(int i);
main()
{
    int i;
    printf("\n  => Tat ca cac so co 3 chu so thoa man dieu kien abc = a^3 + b^3
+ c^3 la:  \n\t");
    getch();
    for(i=100;i<=999;i++)
    {TongLapPhuong(i);
    if(i==TongLapPhuong(i))
    printf("  %d  ",i);}
    getch();
}
int TongLapPhuong(int i)
  {
    int S=0,k;
```

```
        while(i>0)
        {
          k=i%10;
          S=S+k*k*k;
          i=i/10;
        }
      return (S);
    }
```

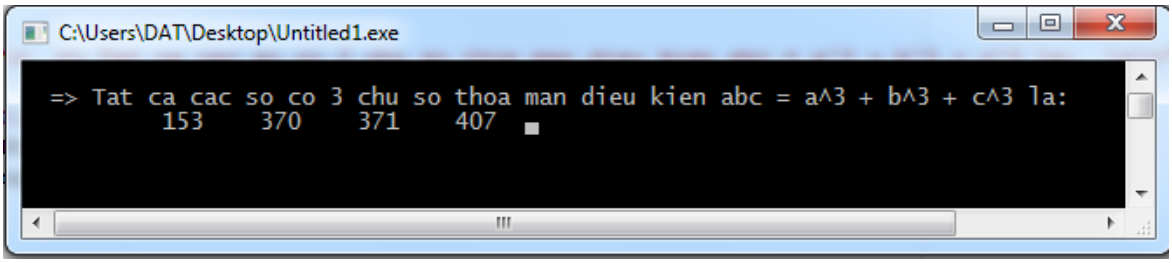
* *Giải thích:*

a) Kiểm tra điều kiện $\overline{abc} = a^3 + b^3 + c^3$ *với các số có 3 chữ số từ 100 đến hết 999, dùng vòng lặp for để kiểm tra từng số trong đoạn [100, 999]. Ứng với mỗi số i, ta kiểm tra điều kiện trên thông qua hàm* ***TongLapPhuong(i).***

b) Hàm TongLapPhuong(i) có nhiệm vụ tính tổng lập phương các chữ số trong i và trả về kết quả cho hàm này. Cách tính tổng lập phương các chữ số trong i tương tự như cách tính tổng các chữ số ở bài tập trước.

* *Lưu ý: Đây chính là bài toán tìm số Amstrong cho trường hợp có 3 chữ số. Là một trường hợp riêng của bài "Tìm tất cả các số Amstrong nhỏ hơn N với N được nhập vào từ bàn phím".*

* *Màn hình kết quả như sau:*



Bài 50: Một số nguyên dương N có k chữ số được gọi là Amstrong nếu nó bằng tổng các lũy thừa bậc k của các chữ số trong N. Cho N, tìm tất cả các số Amstrong < N?

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
int Amstrong(int a);
main()
{
    int N,i;
```

```
        printf("\n  Nhap vao so nguyen duong N:  ");
        scanf("%d",&N);
          printf("  => Cac so Amstrong nho hon N la:  ");
          for(i=1;i<N;i++)
          {
           Amstrong(i);
           if(i==Amstrong(i))
           printf("%d  ",i);
          }
        printf("\n");
        getch();
        return main();
    }
int Amstrong(int a)
    {
        int b,c,k=0,S=0;
        c=a;
        while (c>0)
          {
             c=c/10;
             k++;
          }
        while (a>0)
          {
           b=a%10;
           S=S+(float)pow(b,k);
           a=a/10;
          }
        return(S);
    }
```

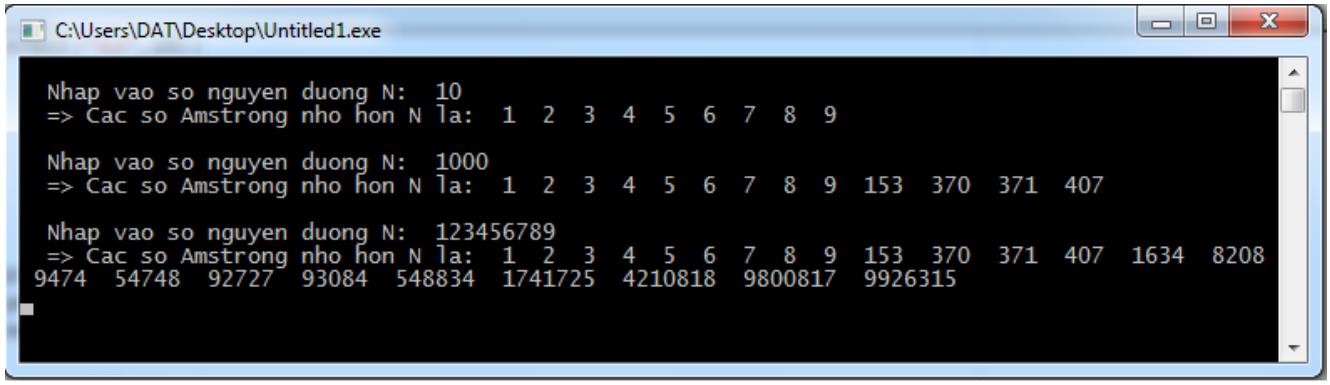
* *Giải thích:*

a) Trong hàm chính, ta kiểm tra các số nguyên dương i nằm trong nửa khoảng [1, N) bằng vòng lặp for. Nếu số i có giá trị bằng tổng lũy thừa bậc k của các chữ số (k là số chữ số của i) thì ta đưa số đó hiển thị ra màn hình.

*b) Trong hàm được định nghĩa thêm **Amstrong(a)**, ta viết mã để nó thực hiện nhiệm vụ tính tổng lũy thừa bậc k của các chữ số. Ban đầu, phải xác định được số i có mấy chữ số thông qua vòng lặp while với biến c lấy giá trị của biến i (không được sử dụng trực tiếp biến i vì nó sẽ ảnh hưởng đến kết quả của lệnh tiếp theo). Có bao nhiêu phép chia lấy dư thì có bấy nhiêu chữ số trong i. Tổng S được cộng dồn với giá trị khởi tạo bằng 0. Lệnh **pow(b,k)** dùng để tính lũy thừa bậc k của cơ số b và phải được để kết quả ở dạng số thực. Kết quả của S được trả về cho hàm Amstrong(a).*

Lưu ý: Hàm được định nghĩa thêm là Amstrong(a) nhưng khi kiểm tra cho giá trị i thì i sẽ thay thế vị trí của a. Chúng ta có thể định nghĩa hàm là Amstrong(i) cho đỡ nhầm lẫn vì sử dụng ít biến hơn và vẫn ra kết quả đúng nhưng về bản chất toán học của hàm thì không chính xác.

* *Màn hình kết quả như sau:*



Bài 51: Cho hai số nguyên dương M, N. Viết chương trình tìm ƯCLN của (M, N)?

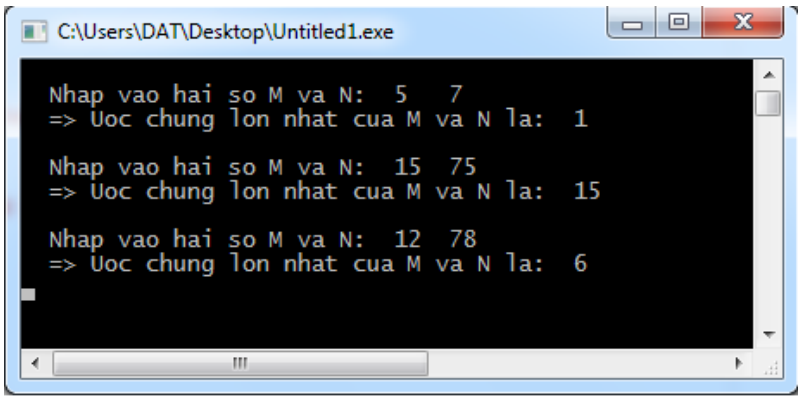
```
#include <stdio.h>
#include <conio.h>
int UCLN(int a, int b);
main()
{
    int a,b;
    printf("\n  Nhap vao hai so M va N:  ");
    scanf("%d%d",&a,&b);
    UCLN(a,b);
    printf("  => Uoc chung lon nhat cua M va N la:  %d\n",UCLN(a,b));
    getch();
```

```
    return main();
}
int UCLN(int a, int b)
   {
     while(a-b!=0)
     {
     a=abs(a-b);
     b=abs(b-a);
     }
     return (a);
   }
```

* *Giải thích:*

- Điều mấu chốt ở đây là thuật toán tìm ƯCLN của hai số. ƯCLN của hai số M, N là số lớn nhất mà cả hai số đều chia hết cho nó. Nếu hai số là các số nguyên tố thì ƯCLN của chúng bằng 1. Ta sử dụng phương pháp "trừ dần" cho tới khi hiệu hai số bằng 0 thì ƯCLN chính bằng 1 trong 2 số tại thời điểm trừ lần cuối cùng. Hàm ***abs()*** *là hàm lấy giá trị tuyệt đối của số nguyên và được lấy trong thư viện stdio.h, sử dụng hàm này để tránh trường hợp a – b < 0.*

* *Màn hình kết quả như sau:*



Bài 52: Cho 3 số nguyên dương a, b, c. Viết chương trình tìm ƯCLN của (a, b, c)? (X)

```
#include <stdio.h>
#include <conio.h>
int UCLN(int a, int b);
main()
```

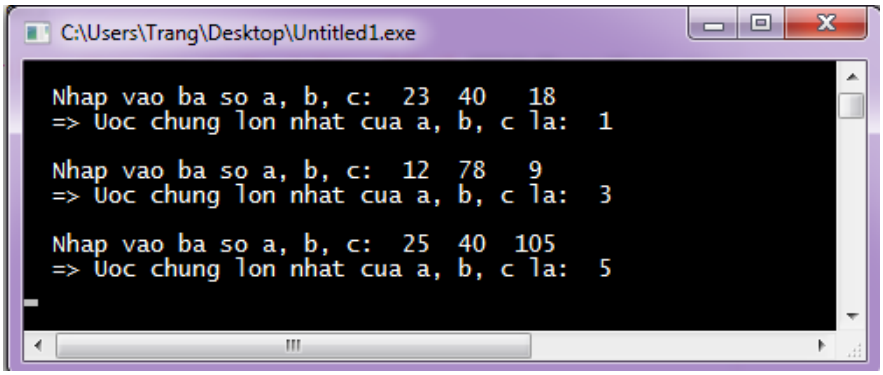
```
{
    int a,b,c,TrungGian;
    printf("\n  Nhap vao ba so a, b, c:  ");
    scanf("%d%d%d",&a,&b,&c);
    TrungGian=UCLN(a,b);
    printf("  => Uoc chung lon nhat cua a, b, c la:
%d\n",UCLN(TrungGian,c));
    getch();
    return main();
}
int UCLN(int a, int b)
{
    while(a-b!=0)
      {
        a=abs(a-b);
        b=abs(b-a);
      }
    return (a);
}
```

* *Giải thích:*

- Thuật toán tìm ước chung lớn nhất của nhiều số là dạng tổng quát của việc tìm ước chung lớn nhất của 2 số. Ở đây, cách tìm ƯCLN bằng phương pháp trừ dần, có thể làm bằng phương pháp chia dần.

*- Để tìm ƯCLN của 3 số a, b, c thì đầu tiên tìm ƯCLN của hai số a và b, kết quả được gán cho biến **TrungGian**. Sau đó tìm ƯCLN của biến TrungGian và biến c ta được kết quả cuối cùng là ƯCLN của ba biến a, b, c.*

* *Màn hình kết quả như sau:*



Bài 53: **Dãy Fibonacci được định nghĩa như sau:**

F[0] = F[1] = 1

F[n] = F[n – 1] + F[n – 2]

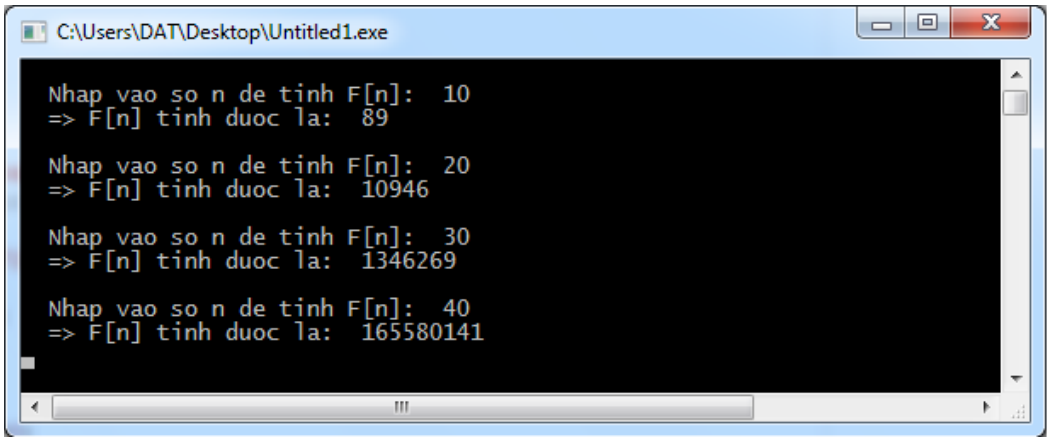
Cho số nguyên dương n, tính và đưa ra màn hình F[n]?

```
#include <stdio.h>
#include <conio.h>
int FIB (int a);
main()
{
    int n;
    printf("\n  Nhap vao so n de tinh F[n]:  ");
    scanf("%d",&n);
    FIB(n);
    printf("  => F[n] tinh duoc la:  %d\n",FIB(n));
    getch();
    return main();
}
int FIB (int a)
   {
     if(a==1||a==0)
     return 1;
     else return FIB(a-1)+FIB(a-2);
   }
```

* *Giải thích:*

- Ở hàm chính main() thì F[n] được tính và đưa ra màn hình. Hàm FIB để tính F[n] được định nghĩa và viết code sau hàm main() và khai báo prototype trước hàm main(). Dùng phương pháp đệ quy để tính cho hàm FIB(n), khi n >1 thì tính tiếp FIB (n-1) + F(n-2), cứ thế cho tới khi n = 1 thì trả về giá trị là 1 và kết thúc FIB(). Vì giá trị của F[n] tăng rất nhanh nên nếu muốn nhập số n lớn thì phải thay đổi kiểu khai báo cho n.

* *Màn hình kết quả như sau:*



Bài 54: Viết chương trình giải phương trình $ax^2 + bx + c = 0$, có sử dụng hàm?

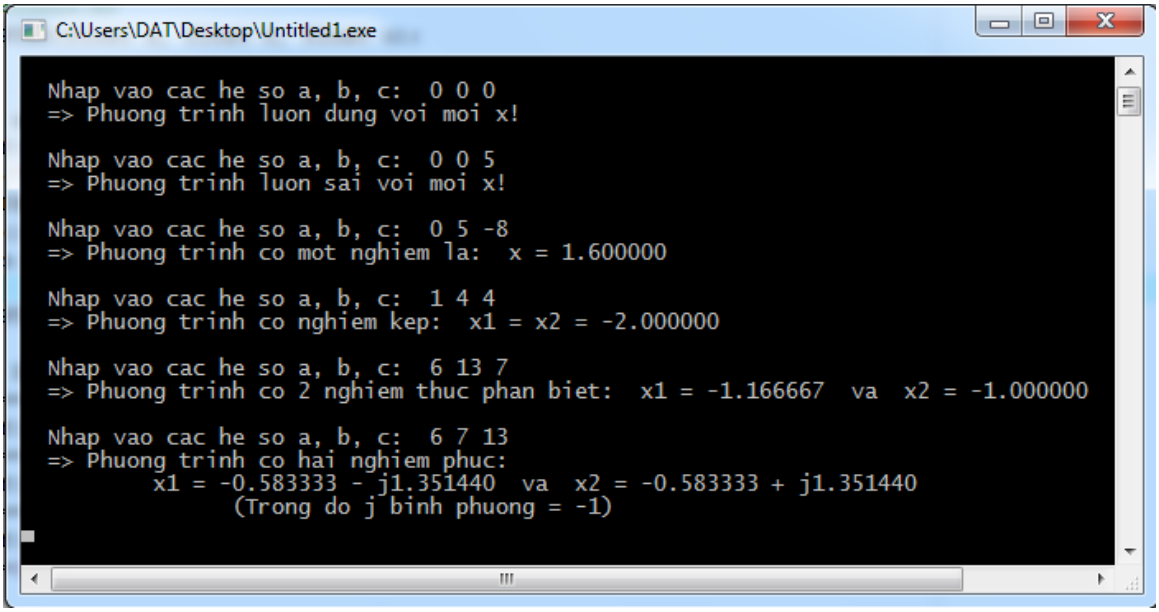
```
#include <stdio.h>
#include <conio.h>
void GiaiPT(float a, float b, float c);
main()
{
   float a,b,c;
   printf("\n  Nhap vao cac he so a, b, c:  ");
   scanf("%f%f%f",&a,&b,&c);
   GiaiPT(a,b,c);
   getch();
   return main();
}
void GiaiPT(float a, float b, float c)
{
   float Del,x1,x2,d,e;
   if((a==0)&&(b==0)&&(c==0))
```

```
    printf("  => Phuong trinh luon dung voi moi x!\n");
    if((a==0)&&(b==0)&&(c!=0))
    printf("  => Phuong trinh luon sai voi moi x!\n");
    if((a==0)&&(b!=0))
    printf("  => Phuong trinh co mot nghiem la:  x = %f\n",-c/b);
    if(a!=0)
    {
       Del=b*b-4*a*c;
       if(Del>0)
       {x1=(float)(-b-sqrt(Del))/2/a;
       x2=(float)(-b+sqrt(Del))/2/a;
       printf("  => Phuong trinh co 2 nghiem thuc phan biet:  x1 = %f  va  x2
= %f\n",x1,x2);}
          else if(Del==0)
          printf("  => Phuong trinh co nghiem kep:  x1 = x2 = %f\n",(float)(-
b/2/a));
          else
           {
             d=-b/2/a;
             e=sqrt(fabs(Del))/2/a;
             printf("  => Phuong trinh co hai nghiem phuc:\n\t  x1 = %f - j%f
va  x2 = %f + j%f\n",d,e,d,e);
             printf("  \t\t(Trong do j binh phuong = -1)\n");
           }
    }
}
```

* <u>*Giải thích:*</u>

- Bài này gần như bài – Tìm nghiệm của phương trình $ax^2 + bx + c = 0$ với a, b, c được nhập vào từ bàn phím. Chỉ cần tách phần thuật giải ra ngoài để tạo thành hàm khác, sau đó đưa hàm đó vào hàm chính main(). Cô giáo chỉ yêu cầu làm với phương trình bậc hai (tức là chỉ nhập $a \neq 0$) nhưng bài này mình viết cho dạng tổng quát để giải với hệ số a, b, c bất kì được nhập từ bàn phím. Nếu ở dạng phương trình bậc hai mà xảy ra trường hợp $\Delta < 0$ thì phương trình vẫn có nghiệm trên tập số phức.

* <u>*Màn hình kết quả như sau:*</u>



<u>Bài 55:</u> Tìm tất cả các số chính phương nhỏ hơn số N cho trước với N được nhập vào từ bàn phím (N < 2 000 000 000), có sử dụng hàm?

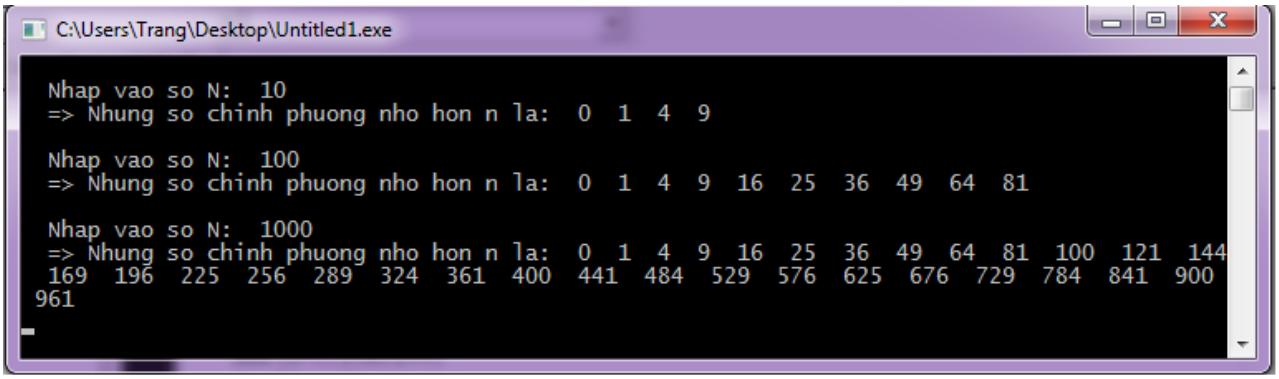
```
#include <stdio.h>
#include <conio.h>
int ChinhPhuong(int i);
main()
{
   int N,i,m;
   printf("\n  Nhap vao so N:  ");
   scanf("%d",&N);
   printf("  => Nhung so chinh phuong nho hon n la:  ");
   for(i=0;i<N;i++)
   {m=ChinhPhuong(i);
   if(m==i)printf("%d  ",i);}
   printf("\n");
   getch();
   return main();
}
int ChinhPhuong(int i)
```

```
{
   int j;
   for(j=0;j<=i;j++)
   if(j*j==i)return (i);
}
```

* *Giải thích:*

- Số chính phương là số bằng bình phương của một số nguyên, như: 0, 1, 4, 9, 16,…(số 0 vẫn được coi là số chính phương). Muốn kiểm tra xem i có phải là số chính phương hay không thì ta thử từng số j (j chạy từ 0 đến i), nếu $j^2 = i$ thì i là số chính phương và in ra màn hình số i. Cứ kiểm tra như thế cho tới khi i = N thì dừng lại.

* *Màn hình kết quả như sau:*



Bài 56*: Tính tổng: S = 1.3.5 + 3.5.7 + …+ (2n – 1).(2n + 1).(2n + 3) với n được nhập vào từ bàn phím (n > 5) và kiểm tra xem tổng S có phải là số chính phương hay không?

(Đề thi cuối kỳ lớp THCS 3, ĐHKHTN – ĐHQGHN, Thứ 4 ngày 19/12/2012, Kỳ I năm học 2012 – 2013, thời gian làm bài: 30 phút)

```
#include <stdio.h>
#include <conio.h>
int ChinhPhuong (int n);
main()
{
    long n,i,S=0;
    printf("\n  Nhap vao so nguyen duong n:  ");
    scanf("%ld",&n);
    while (n<=5)
```

```
            {
             printf("  Nhap lai n:  ");
             scanf("%ld",&n);
            }
          for(i=1;i<=n;i++)
            S=S+(2*i-1)*(2*i+1)*(2*i+3);
          printf("  => Tong S la:  %ld\n",S);
          if(ChinhPhuong(S)==1)printf("  => S là so chinh phuong.\n");
          else printf("  => S khong la so chinh phuong.\n");
          getch();
          return main();
      }
      int ChinhPhuong (int n)
      {
          int i,k=0;
          if(n==1)k=1;
          if(n>1)
            {
             for(i=1;i<=floor(sqrt(n));i++)
             if(i*i==n)k=1;
            }
          return (k);
      }
```

* Giải thích:

Bài 56: Viết chương trình tính e^x theo công thức khai triển Taylor – Mac laurin sau:

$e^x \approx 1+\frac{x}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+.....+\frac{x^n}{n!}$ **với x được nhập từ bàn phím và sai số 0.00001 (X)**

```
#include <stdio.h>
#include <conio.h>
double GiaiThua (int a);
main()
{
    double x,k=1,n=0,S=0;
    printf("\n  Nhap vao so x de tinh e^x:  ");
    scanf("%lf",&x);
    while(k>0.00001)
    {
      k=pow(x,n)/GiaiThua(n);
      S=S+k;n++;
    }
    printf("  => Gia tri xap xi cua e^x la:  %lf\n",S);
    getch();
    return main();
}
double GiaiThua (int a)
{
   if(a<=1)return 1;
   else return a*GiaiThua(a-1);
}
```

* *Giải thích:*

a) Công thức khai triển của e^x : Theo định lý mở rộng của định lý Lagrange :

- Nếu hàm số f(x) xác định, có đạo hàm đến cấp n và liên tục trong đoạn [a, b], có đạo hàm đến cấp (n+1) trong khoảng (a, b) thì với bất kì $x_0 \in (a, b)$ ta luôn có:

$$f(x)=f(x_0)+\frac{f'(x_0)}{1!}(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+....+\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n+\frac{f^{(n+1)}(\overline{x_0})}{(n+1)!}(x-x_0)^{n+1}$$

(với $\overline{x_0}$ là một số nằm giữa x và x_0).

Công thức trên gọi là công thức khai triển Taylor của hàm số f(x).

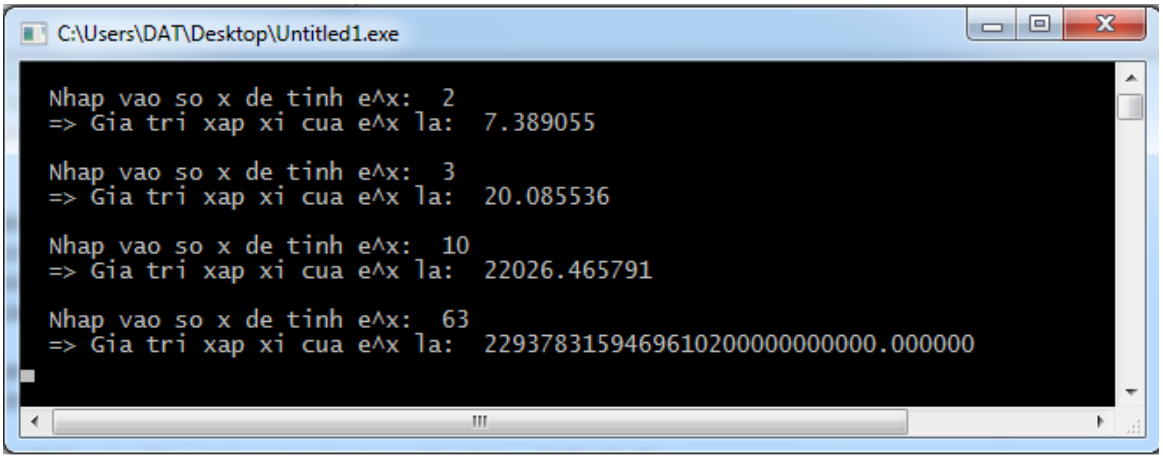
- Đặc biệt khi thay f(x) = e^x và thay $x_0 = 0$ thì ta có khai triển Mac – Laurin của e^x như sau:

$e^x=1+\frac{x}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+.....+\frac{x^n}{n!}+0(x^n)$ *trong đó $0(x^n)$ là vô cùng bé của x^n và đây chính là sai số để chúng ta viết thuật toán tính e^x của bài này.*

b) Ta thấy số hạng tổng quát thứ k của khai triển trên là $\frac{x^k}{k!}$ *và do đó* $e^x=\sum_{k=0}^{n}\frac{x^k}{k!}$. *Điều kiện để vòng lặp while thực hiện chính là giá trị thứ* $k=\frac{x^k}{k!}$ *phải lớn hơn sai số cho phép (k > 0.00001), vì giá trị thứ k chính là sai số của khai triển trên. Khi k = 0.00001 thì vòng lặp dừng lại và giá trị S được đưa ra màn hình.*

*- Bài này mình định nghĩa thêm hàm **GiaiThua(a)** để nhìn cho thoáng và tránh nhầm lẫn! Và vì giá trị e^x tăng lên rất nhanh khi tăng x nên cần phải để kiểu khai báo là **double** hoặc **long double**.*

** <u>Màn hình kết quả như sau:</u>*



<u>Bài 57:</u> Viết chương trình tính sin(x) theo công thức khai triển Taylor – Mac laurin sau: $\sin(x)\approx x-\frac{x^3}{3!}+\frac{x^5}{5!}-.....+(-1)^{n-1}\frac{x^{2n-1}}{(2n-1)!}$ **với x được nhập từ bàn phím và sai số cho phép là 0.00001 (X)**

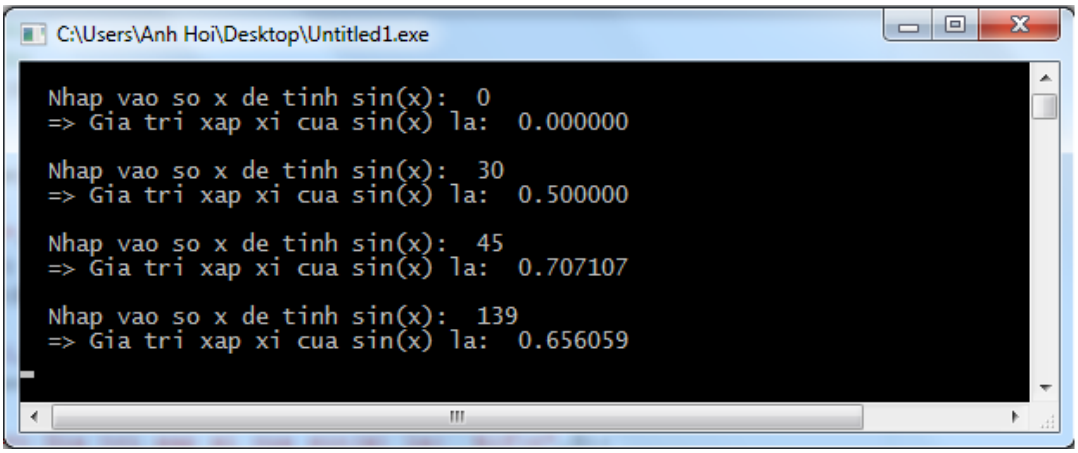
```
#include <stdio.h>
```

```
#include <conio.h>
double GiaiThua (int a);
main()
{
    double x,k=1,n=1,S=0,pi=3.14159265358979;
    printf("\n  Nhap vao so x de tinh sin(x):  ");
    scanf("%lf",&x);
    x=x*pi/180;
    while(fabs(k)>0.00001)
    {
      k=pow(-1,n-1)*pow(x,2*n-1)/GiaiThua(2*n-1);
      S=S+k;n++;
    }
    printf("  => Gia tri xap xi cua sin(x) la:  %lf\n",S);
    getch();
    return main();
}
double GiaiThua (int a)
{
   if(a<=1)return 1;
   else return a*GiaiThua(a-1);
}
```

* *Giải thích:*

- Tương tự bài tính xấp xỉ e^x, bài này áp dụng cho khai triển Mac – Laurin của sin(x). Hàm ***fabs()*** *là hàm tính giá trị tuyệt đối của số thực (dùng hàm này vì k có thể âm). Trong bài này, miền giá trị cần biểu diễn là rất lớn do đó khi nhập x từ 1500 trở lên thì giá trị tính được sẽ thiếu chính xác hoặc ngoài khả năng tính của máy. Lưu ý là giá trị nhập vào của x là độ.*

* *Màn hình kết quả như sau:*



** Để khắc phục hạn chế vì giá trị của các bước trung gian quá lớn không đủ khả năng tính toán của máy, ta nhận xét rằng: sin(x) = sin(x – m.360) với m là số nguyên, vì vậy ta dùng phép tính giảm giá trị của x xuống. Với cách này ta có thể tính được sin của góc trên 2 tỉ độ. Thuật toán như sau:*

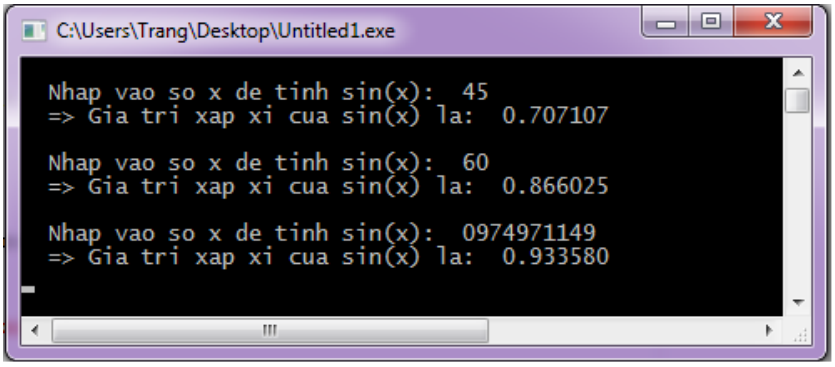
```
#include <stdio.h>
#include <conio.h>
double GiaiThua (int a);
main()
{
    double x,k=1,n=1,S=0,m,pi=3.14159265358979;
    printf("\n  Nhap vao so x de tinh sin(x):  ");
    scanf("%lf",&x);
    if(x>=1500)
      {
        m=(int)(x/360);
        x=(float)(x-m*360);
      }
    x=x*pi/180;
    while(fabs(k)>0.00001)
    {
      k=pow(-1,n-1)*pow(x,2*n-1)/GiaiThua(2*n-1);
      S=S+k;n++;
    }
```

```
        printf("  => Gia tri xap xi cua sin(x) la:  %lf\n",S);
        getch();
        return main();
    }
    double GiaiThua (int a)
    {
       if(a<=1)return 1;
       else return a*GiaiThua(a-1);
    }
```

* *Giải thích:*

- Nếu x ≥ 1500 thì ta tiến hành giảm x xuống để máy đủ khả năng tính toán. Cách giảm như sau: khai báo thêm biến m và biến pi sau đó gán cho pi = 3.14159265358979. Tính m bằng phần nguyên của phép chia $\frac{x}{360}$ *và vì m lấy phần nguyên nên* $\Rightarrow x > m.360$ *Giảm x xuống một lượng m.360 (x = x – m.360). Xong bước này thì ta thấy x đủ nhỏ để nằm trong vùng tính toán được của máy và sin(x) = sin(x – m.360)*

* *Màn hình kết quả như sau:*



Bài 58: Viết chương trình tính cos(x) theo công thức khai triển Taylor – Mac laurin sau: $\cos(x) \approx 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - .....+ (-1)^n \frac{x^{2n}}{(2n)!}$ **với giá trị x và sai số epsilon được nhập từ bàn phím? (X)**

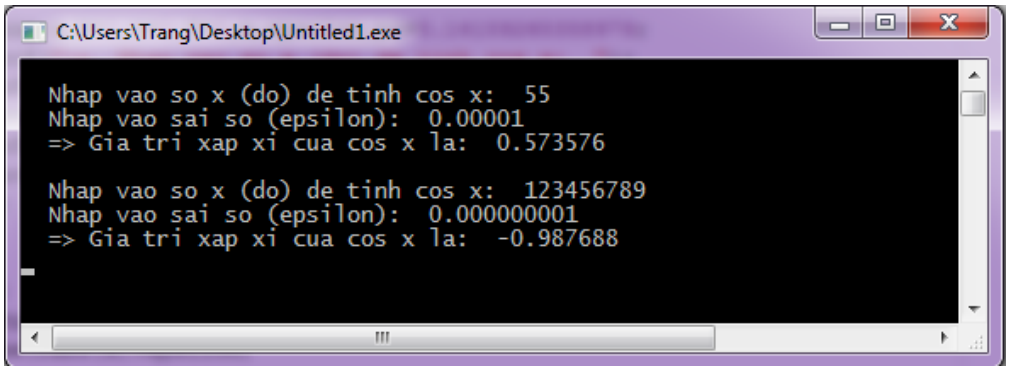
```
#include <stdio.h>
#include <conio.h>
double GiaiThua (int a);
main()
```

```
{
    double x,k=1,n=0,S=0,m,epsilon,pi=3.14159265358979;
    printf("\n  Nhap vao so x (do) de tinh cos x:  ");
    scanf("%lf",&x);
    printf("  Nhap vao sai so (epsilon):  ");
    scanf("%lf",&epsilon);
    if(x>=1500)
      {
        m=(int)(x/360);
        x=(float)(x-m*360);
      }
    x=x*pi/180;
    while(fabs(k)>epsilon)
     {
       k=pow(-1,n)*pow(x,2*n)/GiaiThua(2*n);
       S=S+k;n++;
     }
    printf("  => Gia tri xap xi cua cos x la:  %lf\n",S);
    getch();
    return main();
}
double GiaiThua (int a)
{
   if(a<=1)return 1;
   else return a*GiaiThua(a-1);
}
```

* *Giải thích:*

- Tương tự như bài tính xấp xỉ sin(x) theo khai triển Taylor – Mac Laurin, chỉ khác ở số hạng tổng quát của khai triển: $\cos(x) \approx \sum_{n=0}^{k} (-1)^n . \frac{x^{2n}}{(2n)!}$ *(k là một số nào đó sao cho số hạng thứ k nhỏ hơn hoặc bằng epsilon)*

* *Màn hình kết quả như sau:*



<u>Bài 59:</u> Giải phương trình y = f(x) trong đoạn [a, b] với độ chính xác epsilon cho trước bằng phương pháp chia đôi liên tiếp với giả thiết f(x) liên tục trên đoạn [a, b] và có giá trị trái dấu tại hai đầu mút? (X)

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
float HamSo(float x);
main()
{
    float x1,a,b,e,k;
    printf("\n  Nhap vao can tren a va can duoi b (a < b):  ");
    scanf("%f%f",&a,&b);
    printf("  Nhap vao sai so epsilon:  ");
    scanf("%f",&e);
    do
      {
        x1=(b+a)/2;
        if(HamSo(x1)==0)goto end;
        if(HamSo(a)*HamSo(x1)<0)b=x1;
        if(HamSo(b)*HamSo(x1)<0)a=x1;
        k=b-a;
      }
    while (fabs(k)>e);
```

```
        end:printf("  => Nghiem xap xi cua phuong trinh la:  %.3f\n",x1);
        getch();
        return main();
    }
    float HamSo(float x)
    {
        return x*x*x+3*x*x-2;
    }
```

* <u>*Giải thích:*</u>

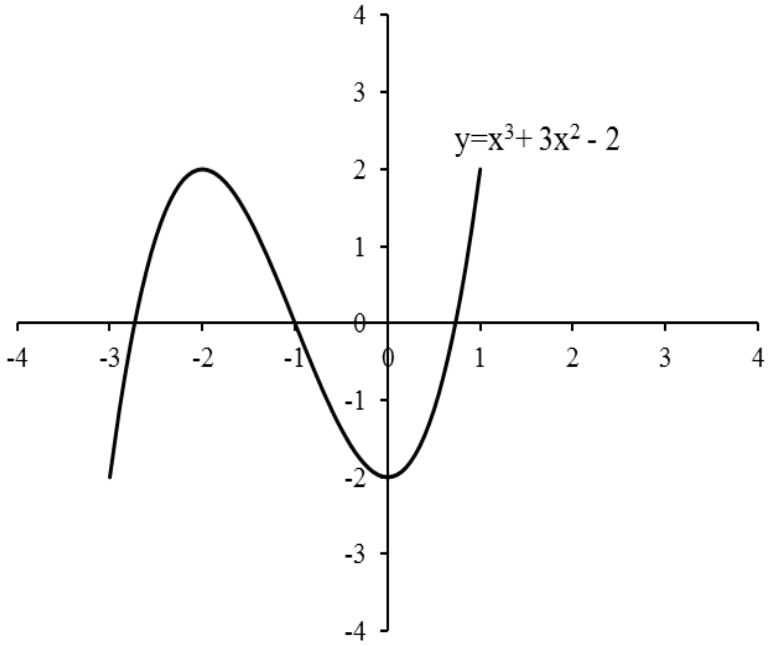
a) Sử dụng phương pháp chia đôi (các định lý về giá trị trung bình) để tìm nghiệm. Thực chất của phương pháp này là thực hiện một dãy hữu hạn các vòng lặp cho tới khi sai số nằm trong giới hạn cho phép thì in ra nghiệm của phương trình phi tuyến. Điều kiện cần để làm là hàm số f(x) phải liên tục trên đoạn [a, b] và chỉ đơn điệu tăng hoặc chỉ đơn điệu giảm trên đoạn đó. Vì vậy, ta chỉ tìm được một nghiệm của phương trình trên [a, b] mà thôi. Nếu muốn tìm được tất cả các nghiệm thì phải phân nhỏ đoạn [a, b] thành các đoạn con sao cho trên mỗi đoạn con hàm số chỉ đơn điệu tăng hoặc chỉ đơn điệu giảm (sử dụng tính chất đổi dấu của đạo hàm cấp 2).

b) Giả sử giải phương trình $x^3 + 3x^2 - 2 = 0$ trên đoạn [a, b] với a và b được nhập từ bàn phím. Đầu tiên gán nghiệm $x_1 = \frac{a+b}{2}$ là trung bình cộng của 2 đầu mút. Nếu xảy ra trường hợp đặc biệt mà $f(x_1) = 0$ thì kết luận x_1 chính là nghiệm của phương trình và đưa kết quả ra màn hình.

Nếu thấy $f(a).\ f(x_1) < 0$ tức là nghiệm của phương trình nằm trong đoạn $[a, x_1]$ thì gán lại giá trị $b = x_1$; ngược lại nếu thấy $f(b).\ f(x_1) < 0$ tức là nghiệm của phương trình nằm trong đoạn $[x_1, b]$ thì gán lại giá trị $a = x_1$. Cứ lặp lại như thế cho tới khi thỏa mãn sai số của phép tính nhỏ hơn hoặc bằng sai số cho phép. Sai số của phép tính là: $k = (b - a)$.

Dưới đây là đồ thị minh họa phương pháp chia đôi của bài toán giải phương trình:

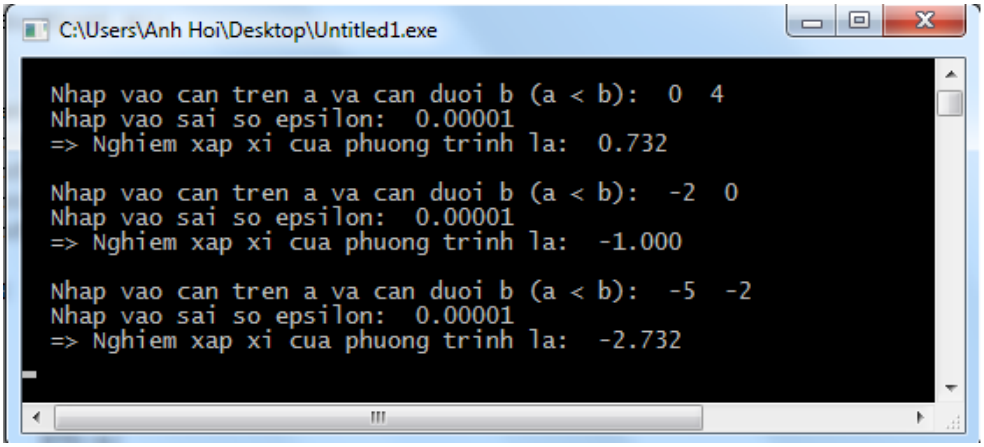
$$x^3 + 3x^2 - 2 = 0$$



*Ví dụ nhập a = 0 và b = 4 thì sau vòng lặp **do while** lần thứ nhất giá trị x là 2. Thực hiện kiểm tra thấy rằng f(0). f(2) < 0 thì gán b = 2. Đến vòng lặp thứ 2 sẽ có a, b mới là a = 0 và b = 2 suy ra x = 1. Cứ tính liên tiếp như thế cho tới khi a – b ≤ epsilon thì dừng lại và in kết quả x ra màn hình. Nghiệm của phương trình trên đoạn [0, 4] là 0,732*

Nhược điểm của phương pháp này là ta phải đoán các giá trị a và b nhập vào để trong đoạn [a, b] phương trình có 1 nghiệm. Nên vẽ đồ thị của hàm số để nhìn cho dễ.

* *Màn hình kết quả như sau:*



Bài 60: Viết chương trình nhập vào hai giá trị thực a, b (sao cho a < b) và một số nguyên dương n. Tính giá trị tích phân xác định $\int_a^b x^2 dx$ bằng định nghĩa (chia nhỏ

hình thang cong ban đầu thành n hình thang con có chiều cao Δx_i với $\Delta x_i = \frac{b-a}{n}$)?

(X)

```
#include <stdio.h>
#include <conio.h>
float HamSo(float x);
main()
{
    float a,b,i,h,TichPhan=0;
    int n;
    printf("\n  Nhap vao hai so a va b la hai can cua tich phan (a<b):  ");
    nhaplai1:scanf("%f%f",&a,&b);
    if(a>=b){printf("  Nhap lai a va b:  ");goto nhaplai1;}
    printf("  Nhap vao so khoang can chia (n>0):  ");
    nhaplai2:scanf("%d",&n);
    if(n<=0){printf("  Nhap lai n:  ");goto nhaplai2;}
    h=(b-a)/n;
    for(i=0;i<n;i++)
      TichPhan+=(HamSo(a+(i+1)*h)+HamSo(a+i*h))/2*h;
    printf("  => Gia tri tich phan cua ham so tren doan [%1.0f,%1.0f] la:
%0.4f\n",a,b,TichPhan);
    getch();
    return main();
}
float HamSo(float x)
{
    return (x*x);
}
```

* *Giải thích:*

*- Định nghĩa thêm hàm số $y = x^2$ ở bên ngoài hàm main() và trả về giá trị x^2 cho hàm **HamSo()**. Vì a và b là cận lấy tích phân cho nên a < b và n là số khoảng cần chia cho nên n > 0. Nếu nhập vào mà thấy $a \geq b$ cũng như $n \leq 0$ thì phải nhập lại, dùng lệnh nhảy vô điều kiện **goto** để nhảy tới vị trí cần nhập lại.*

*a) Ta dùng **phương pháp số** để tính dựa vào định nghĩa tích phân xác định của hàm một biến. Nếu trên đoạn [a, b] mà giá trị hàm số không âm thì tích phân chính là diện tích của hình phẳng giới hạn bởi trục Ox, đường thẳng x = a, đường thẳng x = b và đồ thị hàm số $y = x^2$. Trong trường hợp hàm số nhận giá trị âm trên đoạn [a, b] thì giá trị tích phân cũng nhận giá trị âm.*

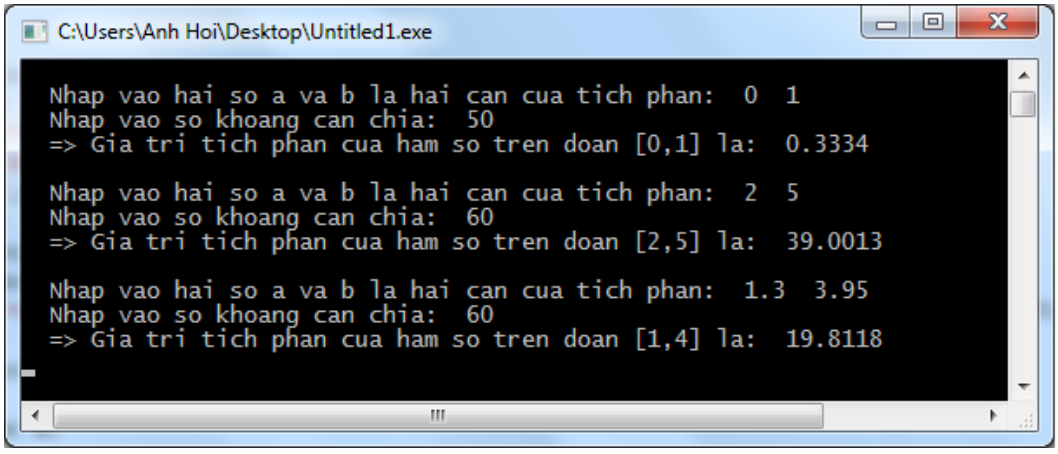
+ Giả sử muốn tính tích phân của hàm số trên đoạn [a, b] ta chia đoạn [a, b] thành n đoạn con và khoảng cách của mỗi đoạn con được gán cho biến h $\left(h = \frac{b-a}{n}\right)$*. Từ hình thang ban đầu bây giờ ta được n hình thang con và mỗi hình thang con có chiều cao là h. Gọi* $\sigma = \sum_{i=1}^{n} f(\theta)h$ *là tổng diện tích của các hình thang con đó,* θ *là điểm nằm giữa* x_i *và* x_{i+1}*, i chạy từ 0 đến n. Vậy theo định nghĩa, tích phân của hàm số f(x) trên [a, b] chính là* $\lim \sum_{i=1}^{n} f(\theta).h$ *khi n tiến dần đến vô cùng và giới hạn này không phụ thuộc cách chọn* θ*. Vì thế mình chọn* $\theta = \frac{x_i = x_{i+1}}{2}$ *(các bạn có thể chọn* θ *tùy ý miễn là nằm giữa* x_i *và* x_{i+1}

*b) Khởi tạo giá trị ban đầu cho biến **TichPhan** bằng 0, với mỗi biến i (i chạy từ 0 đến n – 1) thì giá trị của biến TichPhan được cộng dồn một lượng bằng diện tích hình thang thứ i (hình thang này có chiều cao h và trung bình cộng của 2 đáy là giá trị trung bình của hàm số ở hai đầu mút của hình thang đó).*

Bài này tính tích phân cho hàm $y = x^2$ tuy nhiên với thuật toán như trên có thể tính tích phân cho bất kì hàm số một biến nào trên bất kì cận nào miễn sao hàm số xác định và liên tục trên đoạn đó. Nếu muốn tính giá trị tích phân cho hàm khác thì chỉ cần chỉnh sửa lại một chút trong giá trị trả về của hàm HamSo().

*Mọi người có thể xem thêm phần định nghĩa tích phân xác định để biết thuật toán làm bài này. Xem trong giáo trình "**Toán học cao cấp tập 2**" – Phép tính giải tích một biến số – Nguyễn Đình Trí (chủ biên), Tạ Văn Đĩnh, Nguyễn Hồ Quỳnh – Trang 249*

* *Màn hình kết quả như sau:*



Bài 61: Xây dựng một thư viện gồm các hàm:

- Tìm max của 3 số.

- Tìm min của 3 số.

- Kiểm tra một số có là nguyên tố hay không.

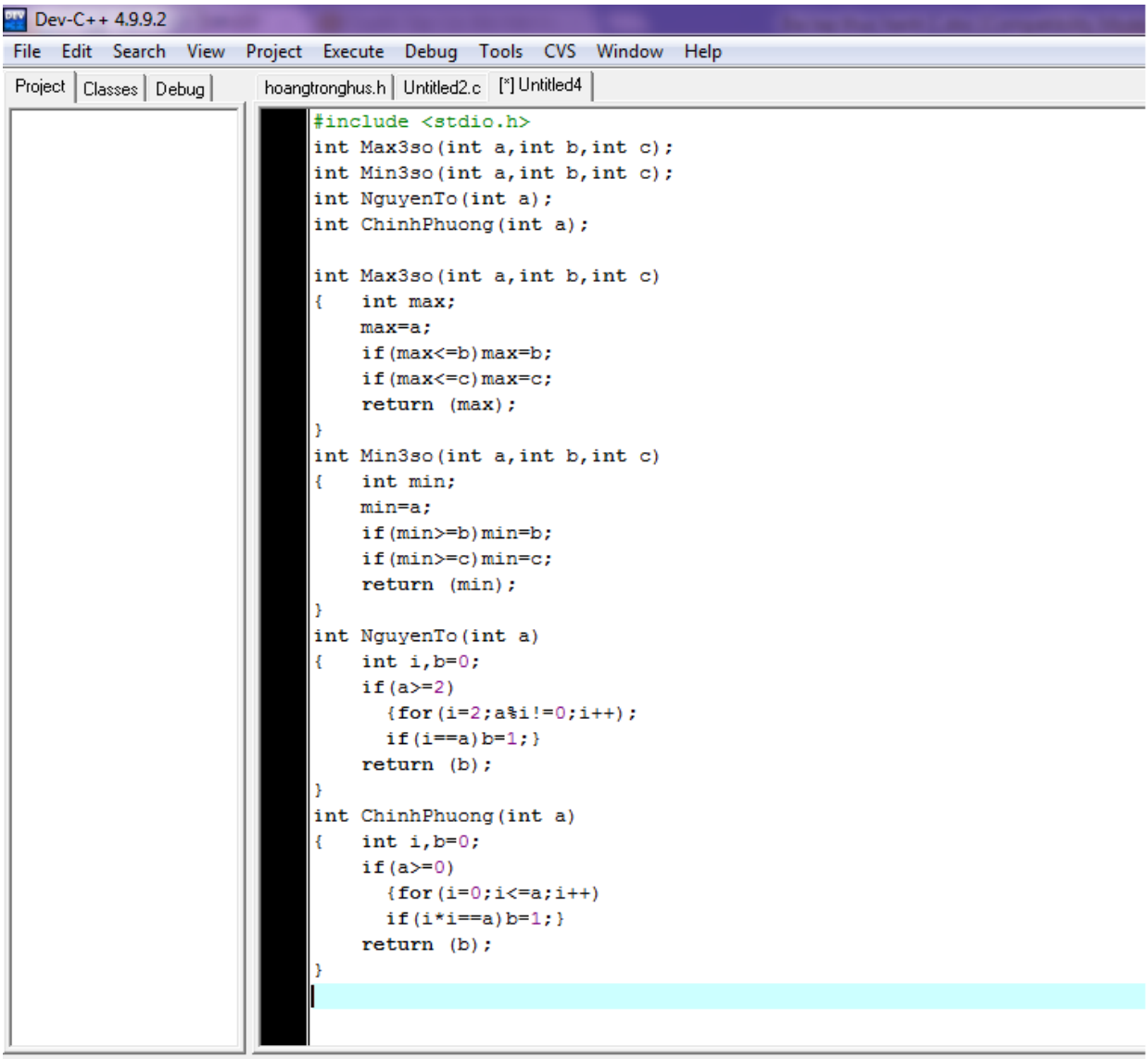
- Kiểm tra một số chính phương.

Lưu trữ các hàm trên vào file: *.h và viết chương trình sử dụng các hàm trong thư viện này?**

Giải:

⇒ Việc tạo thư viện chỉ cần thao tác chứ không phải suy luận hay tư duy logic gì cả!!! Ta nên tạo thêm thư viện để tránh viết lại hay copy các hàm đã có sẵn, với nhu cầu sử dụng nhiều lần. Và có nhiều cách tạo thư viện, mình xin giới thiệu một cách khá đơn giản gồm những bước như sau:

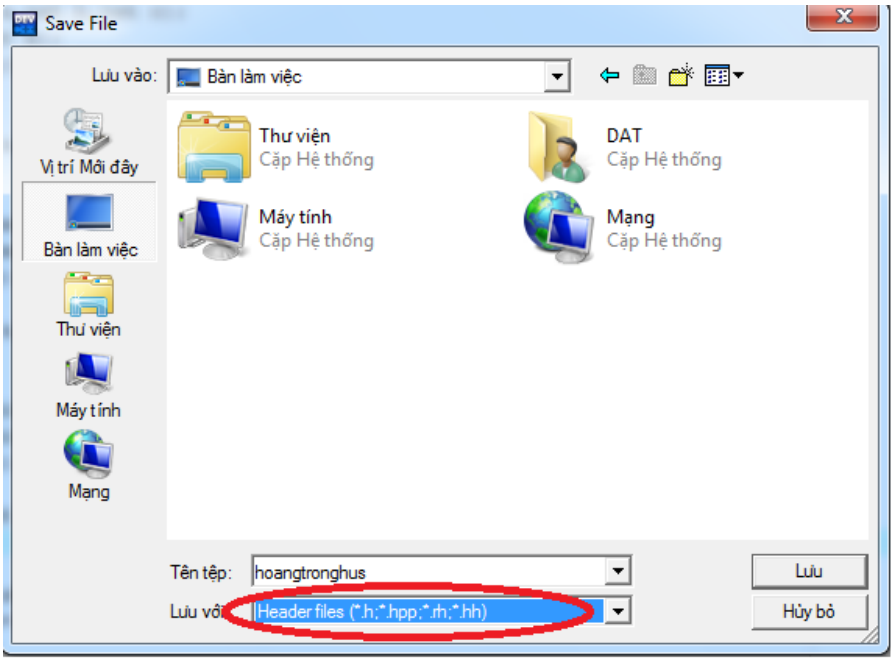
Bước 1: Viết ra các hàm mà mình muốn đưa vào thư viện, không cần viết hàm main() nhưng cần phải khai báo prototype (*nguyên mẫu*). Ví dụ mình sẽ viết cho 4 hàm trên như hình sau đây:



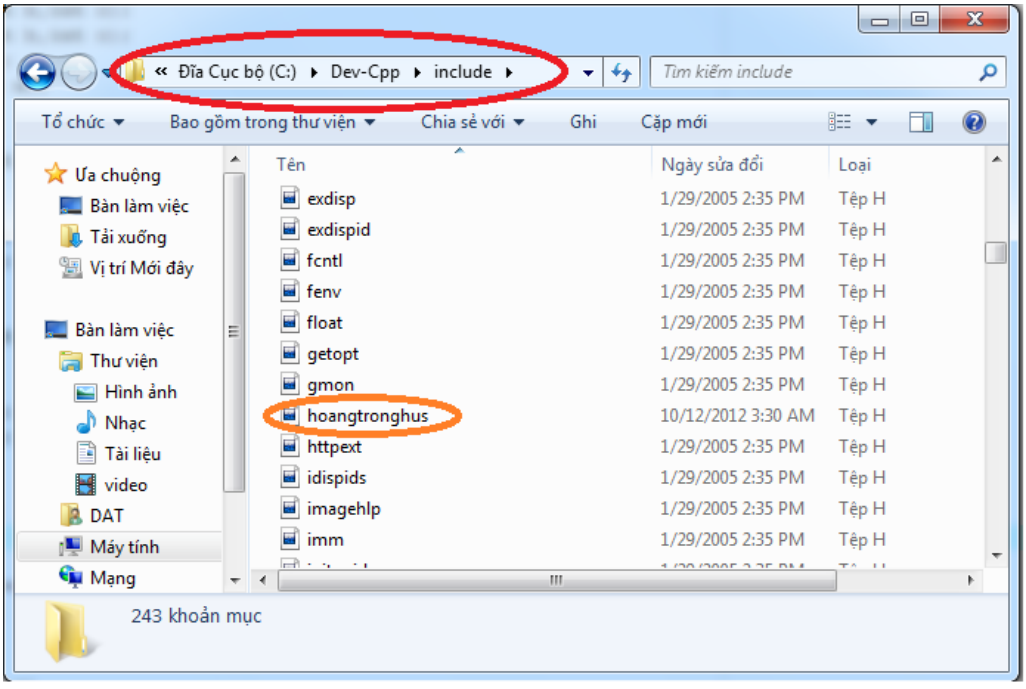

```
#include <stdio.h>
int Max3so(int a,int b,int c);
int Min3so(int a,int b,int c);
int NguyenTo(int a);
int ChinhPhuong(int a);

int Max3so(int a,int b,int c)
{  int max;
   max=a;
   if(max<=b)max=b;
   if(max<=c)max=c;
```

```
    return (max);
}
int Min3so(int a,int b,int c)
{   int min;
    min=a;
    if(min>=b)min=b;
    if(min>=c)min=c;
    return (min);
}
int NguyenTo(int a)
{   int i,b=0;
    if(a>=2)
      {for(i=2;a%i!=0;i++);
      if(i==a)b=1;}
    return (b);
}
int ChinhPhuong(int a)
{   int i,b=0;
    if(a>=0)
      {for(i=0;i<=a;i++)
      if(i*i==a)b=1;}
    return (b);
}
```

Bước 2: Lưu file vừa tạo với đuôi ***.h (Ví dụ mình lưu với file có tên là hoangtronghus.h):



Bước 3: Copy file hoangtronghus.h vừa tạo (mình để ở desktop) vào ổ C theo đường dẫn sau: C/ Dev-Cpp/ include. Cái này còn tùy khi cài đặt bạn chọn lưu Dev-Cpp ở đâu, miễn sao copy vào đúng thư mục include trong Dev-Cpp là được:



Bước 4: Bây giờ thì có thể sử dụng thư viện vừa tạo được rồi! Mình giả sử làm một bài tập thế này:

Viết chương trình yêu cầu thực hiện một trong các công việc sau dựa vào tùy chọn được nhập từ bàn phím: Tìm max của 3 số, tìm min của 3 số, kiểm tra tính nguyên tố hoặc tính chính phương của một số?

```
#include <stdio.h>
#include <conio.h>
#include "hoangtronghus.h"
main()
{
   int a,b,c,d,e,f,g,k,n,z;
   printf("\n  Nhap vao so tuong ung voi cac bai toan sau:\n\t1-Tim Max cua 3 so a, b, c\n\t2-Tim Min cua 3 so a, b, c\n\t3-Kiem tra tinh nguyen to cua so n\n\t4-Kiem tra tinh chinh phuong cua so n\n  ");
   vao:scanf("%d",&k);
   switch(k)
     {
       case 1:{chay1:printf("  Nhap vao so a,b,c de tim max:
");scanf("%d%d%d",&a,&b,&c);
            d=Max3so(a,b,c);printf("  => Gia tri lon nhat trong 3 so la:  %d\n",d);
            printf("  Nhap 1 de tiep tuc ham nay:
");scanf("%d",&z);if(z==1){printf("\n");goto chay1;}else goto tudau;}
       case 2:{chay2:printf("  Nhap vao so a,b,c de tim min:
");scanf("%d%d%d",&a,&b,&c);
            e=Min3so(a,b,c);printf("  => Gia tri nho nhat trong 3 so la:  %d\n",e);
            printf("  Nhap 1 de tiep tuc ham nay:
");scanf("%d",&z);if(z==1){printf("\n");goto chay2;}else goto tudau;}
       case 3:{chay3:printf("  Nhap vao mot so n de xac dinh tinh nguyen to:
");scanf("%d",&n);
            f=NguyenTo(n);printf("  => Tinh nguyen to cua so n la:  %d\n",f);
            printf("  Nhap 1 de tiep tuc ham nay:
");scanf("%d",&z);if(z==1){printf("\n");goto chay3;}else goto tudau;}
       case 4:{chay4:printf("  Nhap vao mot so n de xac dinh tinh chinh phuong:
");scanf("%d",&n);
            g=ChinhPhuong(n);printf("  => Tinh chinh phuong cua so n la:
%d\n",g);
```

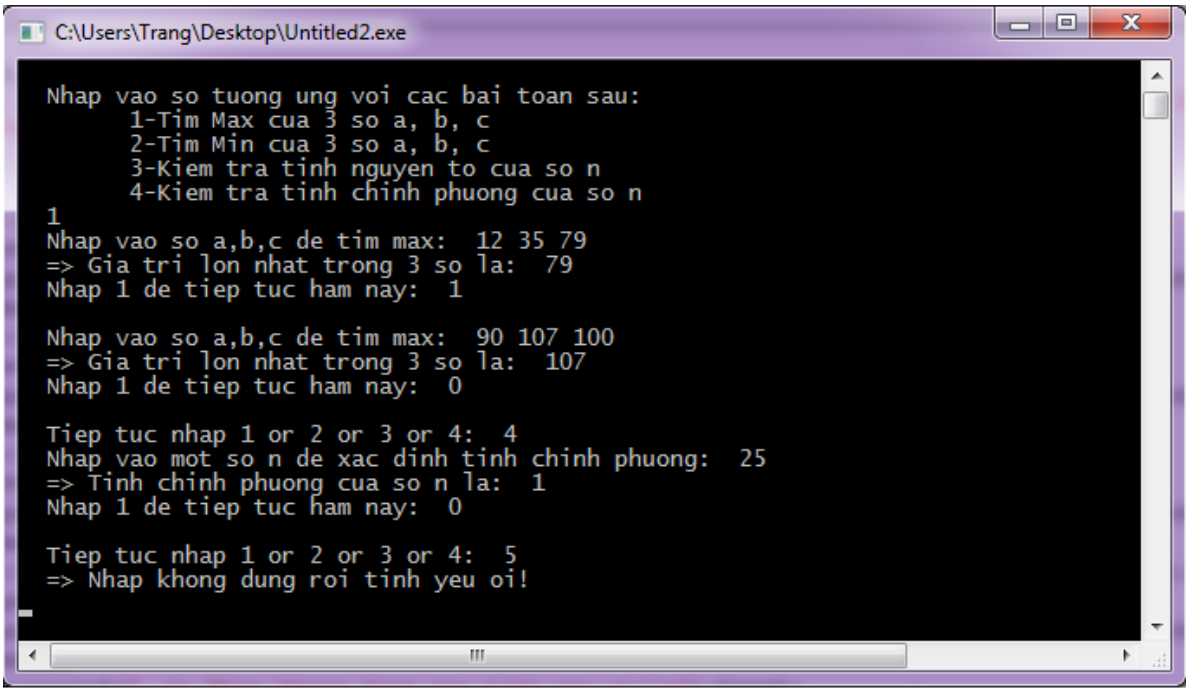
```
                printf("  Nhap 1 de tiep tuc ham nay:
");scanf("%d",&z);if(z==1){printf("\n");goto chay4;}else goto tudau;}
                 default:printf("  => Nhap khong dung roi tinh yeu oi!\n");
                }
                getch();
                tudau:printf("\n  Tiep tuc nhap 1 or 2 or 3 or 4:  ");
                goto vao;
            }
```

* *Giải thích:*

*a) Sau khi đã tạo thư viện cần phải khai báo thư viện đó thì mới dùng được và khai báo có dạng: #include "***.h" chứ không được khai báo kiểu: #include <***.h>*

*b) Dùng cấu trúc rẽ nhánh **switch** để phân ra các trường hợp nhập vào, nếu nhập vào là 1 thì sử dụng hàm tính giá trị lớn nhất, nếu nhập vào là 2 thì sử dụng hàm tính giá trị nhỏ nhất,... Để tiện lợi khi chạy chương trình nên mình dùng thêm lệnh nhảy vô điều kiện **goto** (xem phần lệnh nhảy vô điều kiện **break, continue, goto** trang 149 giáo trình Ngôn ngữ lập trình C – Quách Tuấn Ngọc để hiểu rõ hơn). Lệnh này rất thuận lợi khi ta muốn chương trình thực hiện tiếp tại bất kỳ vị trí nào mà ta chỉ định.*

* *Màn hình kết quả như sau: (quy ước 1 là có tính chất nguyên tố hoặc chính phương và 0 là không có tính chất trên).*



Trên đây là cách tạo một thư viện đơn giản và ứng dụng để giải các bài toán mà không cần viết lại hay copy lại mã. Nếu muốn tạo một thư viện đồ sộ cỡ như stdio.h hay conio.h thì cần phải hiểu thêm về kiến thức tiền xử lý nữa. Khi nào tạo được thì mình sẽ cập nhật sau...ok

CHƯƠNG III. MẢNG MỘT CHIỀU

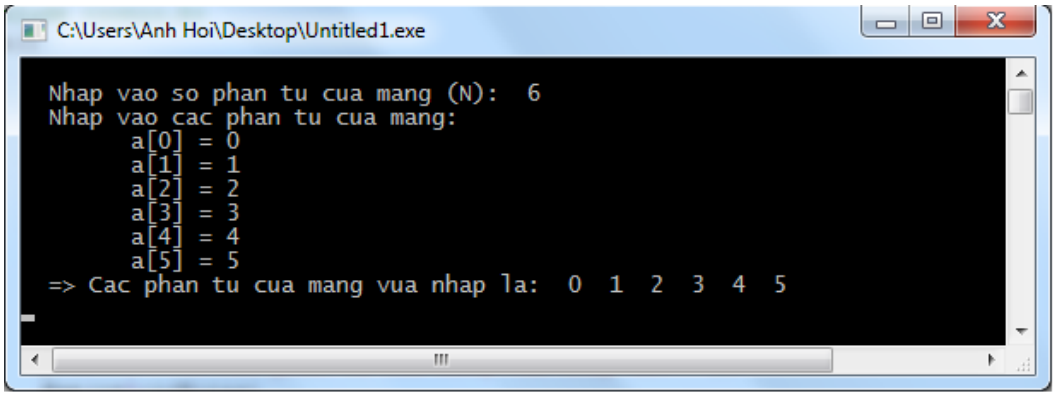
Bài 62: Viết chương trình nhập vào từ bàn phím mảng a có N phần tử. In ra giá trị các phần tử của mảng a vừa nhập? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
    int N,i;
    printf("\n  Nhap vao so phan tu cua mang (N):  ");
    scanf("%d",&N);
    int a[N];
    printf("  Nhap vao cac phan tu cua mang:\n");
    for(i=0;i<N;i++)
    { printf("\ta[%d] = ",i);
      scanf("%d",&a[i]);
    }
    printf("  => Cac phan tu cua mang vua nhap la:  ");
    for(i=0;i<N;i++)
     printf("%d  ",a[i]);
    printf("\n");
    getch();
    return main();
}
```

* *Giải thích:*

- Dùng lệnh for() để lặp biến i (i chạy từ 1 đến N – 1). Biến i chạy đến đâu ta nhập giá trị a[i] đến đó. Tương tự, khi muốn hiển thị ra màn hình, biến i chạy đến đâu thì ta cho a[i] hiện ra màn hình đến đó.

* *Màn hình kết quả như sau:*



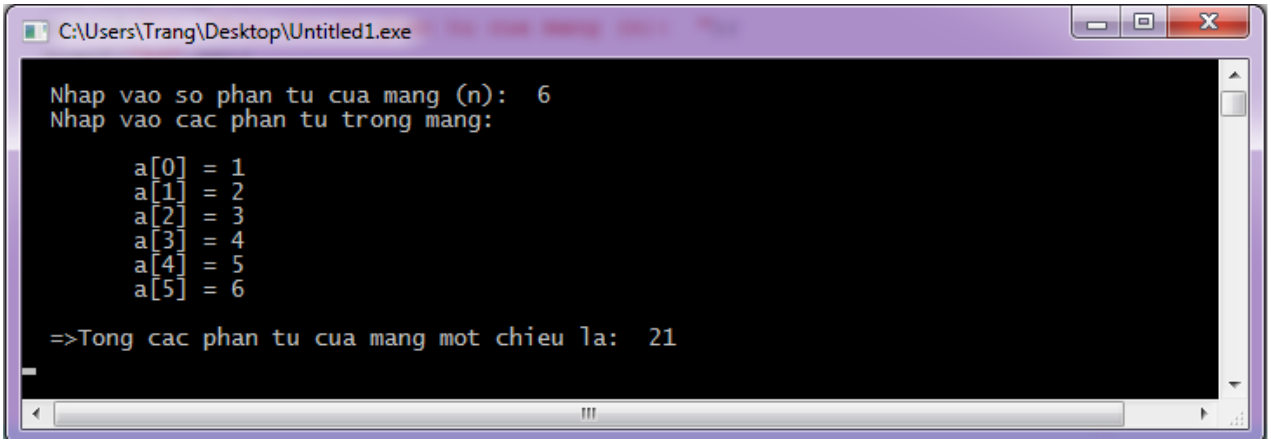
<u>Bài 63:</u> Nhập vào mảng một chiều và in ra màn hình tổng các phần tử của mảng đó?

```
#include <stdio.h>
#include <conio.h>
main()
{
   int n,i,Tong=0;
   printf("\n  Nhap vao so phan tu cua mang (n):  ");
   scanf("%d",&n);
   int a[n];
   printf("  Nhap vao cac phan tu trong mang:\n\n");
   for(i=0;i<n;i++)
      {
       printf("\ta[%d] = ",i);
       scanf("%d",&a[i]);
       Tong=Tong+a[i];
      }
   printf("\n  =>Tong cac phan tu cua mang mot chieu la:  %d\n",Tong);
   getch();
   return main();
}
```

* *<u>Giải thích:</u>*

- Các giá trị a[i] được nhập từ bàn phím và nhập đến đâu thì tính tổng đến đó. Cho tới khi nhập đủ n phần tử của mảng thì tổng được hiển thị lên màn hình.

* *<u>Màn hình kết quả như sau:</u>*



Bài 64: Viết chương trình nhập vào một mảng 1 chiều gồm n số nguyên, tính giá trị trung bình của các số chẵn (hoặc số lẻ) trong mảng?

** Đối với trường hợp số chẵn:*

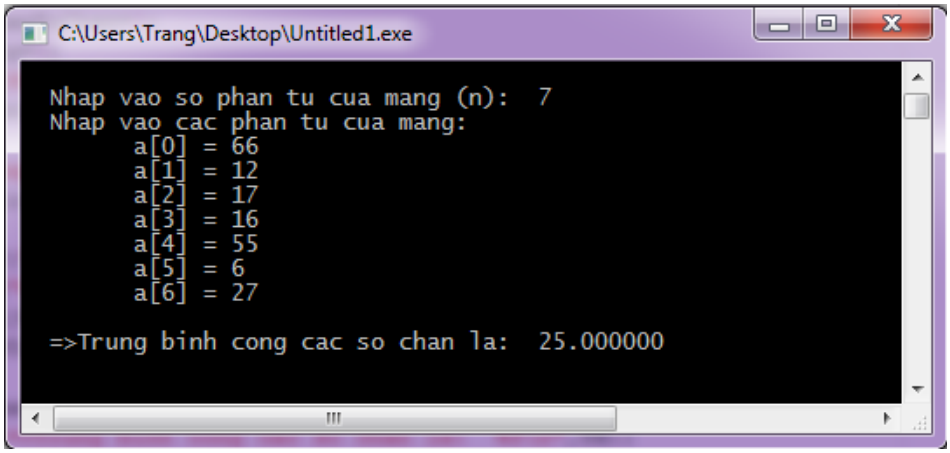
```
#include <stdio.h>
#include <conio.h>
main()
{
   int n,i,SoPhanTuChan=0;
   float TB,Tong=0;
   printf("\n  Nhap vao so phan tu cua mang (n):  ");
   scanf("%d",&n);
   int a[n];
   printf("  Nhap vao cac phan tu cua mang:  \n");
   for(i=0;i<n;i++)
      {
       printf("\ta[%d] = ",i);
       scanf("%d",&a[i]);
       if(a[i]%2==0)
       {Tong=Tong+a[i];SoPhanTuChan+=1;}
      }
   TB=Tong/SoPhanTuChan;
   printf("\n  =>Trung binh cong cac so chan la:  %f\n",TB);
```

```
    getch();
    return main();
}
```

* *Giải thích:*

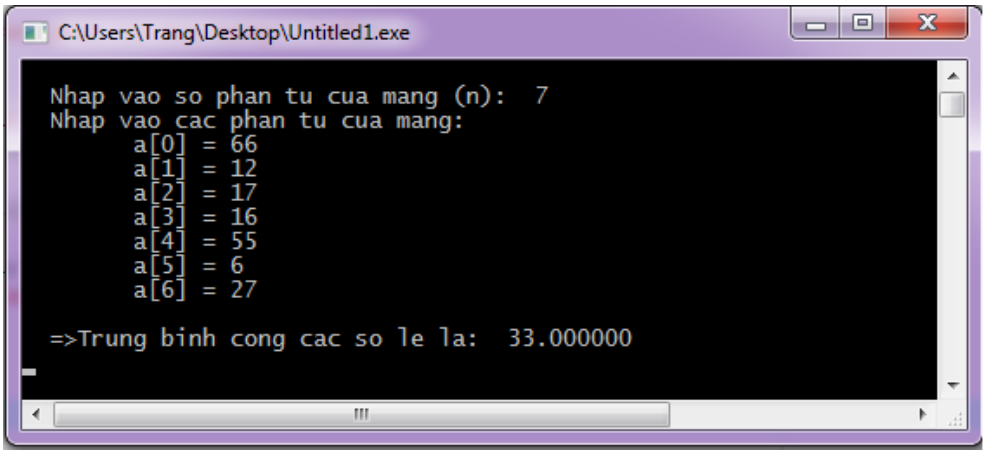
*- Giá trị **Tong** chỉ được cộng dồn đối với những số nhập vào thỏa mãn là số chẵn. Và khi có một số chẵn thỏa mãn thì **SoPhanTuChan** lại tăng lên 1 đơn vị với giá trị khởi tạo bằng 0. Giá trị trung bình được tính bằng tổng các số chẵn chia cho số phần tử chẵn.*

* *Màn hình kết quả như sau:*



* *Đối với trường hợp số lẻ:* Chỉ cần sửa một chút điều kiện của lệnh if() từ: if(a[i]%2==0) thành if(a[i]%2!=0) và sửa chuỗi ký tự hiển thị trong lệnh printf() từ: *số chẵn* thành *số lẻ.*

* *Màn hình kết quả như sau:*



<u>Bài 65:</u> Viết chương trình nhập vào từ bàn phím mảng a có N phần tử. Tính:

- Tổng các phần tử trong mảng a.

- Giá trị trung bình cộng của các phần tử dương.

- Giá trị lớn nhất, nhỏ nhất trong mảng.

In ra màn hình các giá trị vừa tìm được? (X)

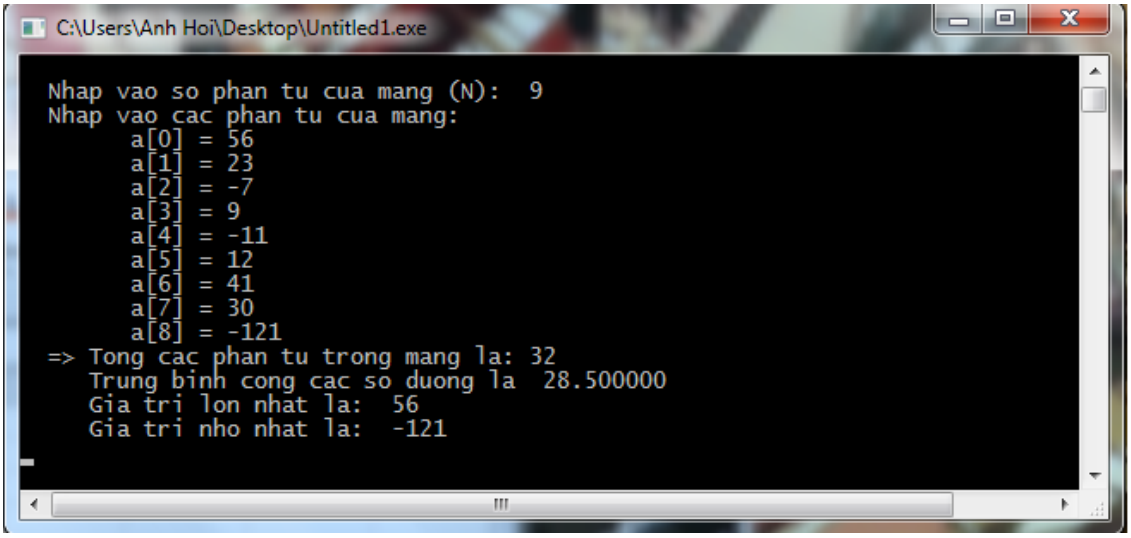
```
#include <stdio.h>
#include <conio.h>
main()
{
    int N,i,j,TongToanBo=0,TongSoDuong=0,So=0,max,min;
    float TB;
    printf("\n  Nhap vao so phan tu cua mang (N):  ");
    scanf("%d",&N);
    int a[N];
    printf("  Nhap vao cac phan tu cua mang:\n");
    for(i=1;i<=N;i++)
     {
      printf("\ta[%d] = ",i);
      scanf("%d",&a[i]);
      TongToanBo+=a[i];
      if(a[i]>0){TongSoDuong+=a[i];So=So+1;TB=(float)TongSoDuong/So;}
     }
    max=a[1];min=a[1];
    for(i=1;i<=N;i++)
     {
       if(max<=a[i])max=a[i];
       if(min>=a[i])min=a[i];
     }
    printf("  => Tong cac phan tu trong mang la: %d\n     Trung binh cong cac
so duong la  %f\n     Gia tri lon nhat la:  %d\n     Gia tri nho nhat la:
%d\n",TongToanBo,TB,max,min);
    getch();
```

```
    return main();
}
```

* *Giải thích:*

- Tổng các phần tử được cộng liên tục sau khi nhập vào giá trị a[i], giá trị trung bình được tính bằng tổng các phần tử dương chia cho số phần tử dương, GTLN và GTNN được tìm bằng cách gán giá trị khởi tạo bằng giá trị ban đầu của mảng và so sánh nó với các các giá trị còn lại để cập nhật giá trị mới.

* *Màn hình kết quả như sau:*



Bài 66: Viết chương trình nhập vào một dãy số nguyên và tìm số chẵn nhỏ nhất trong dãy đó? (X)

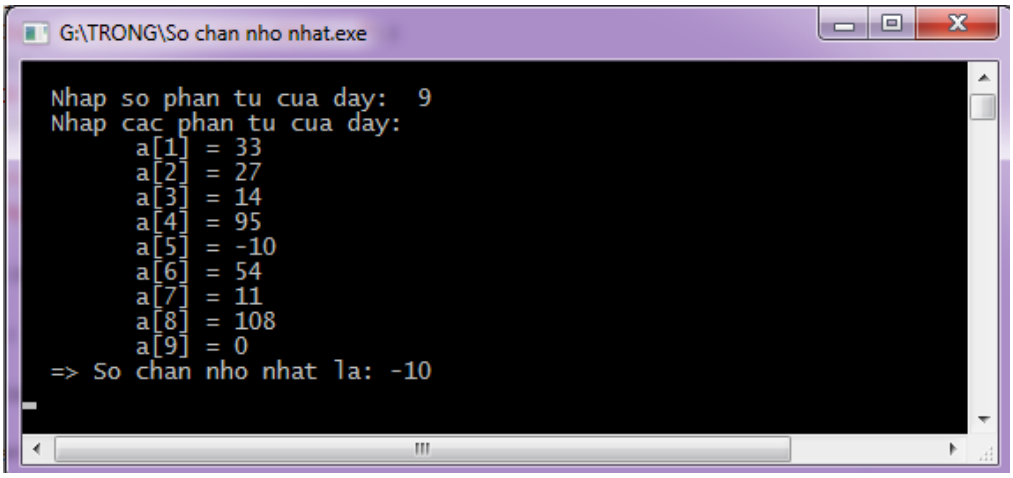
```
#include <stdio.h>
#include <conio.h>
void SoChanNhoNhat(int a[],int n);
main()
{
    int a[100],n,i;
    printf("\n  Nhap so phan tu cua day:  ");
    scanf("%d",&n);
    printf("  Nhap cac phan tu cua day:\n");
    for(i=1;i<=n;i++)
     {
```

```
            printf("\ta[%d] = ",i);
            scanf("%d",&a[i]);
          }
        SoChanNhoNhat(a,n);
        getch();
        return main();
    }
    void SoChanNhoNhat(int a[],int n)
    {
        int i,min,dem=0;
        for(i=1;i<=n;i++)
          if(a[i]%2==0){dem++;min=a[i];}
        if(dem==0)printf("  => Khong co so chan nao!\n");
        if(dem!=0)
            {
              for(i=1;i<=n;i++)
              if((a[i]%2==0)&&(a[i]<=min))min=a[i];
              printf("  => So chan nho nhat la: %d\n",min);
            }
    }
```

* *Giải thích:*

*- Tìm số chẵn nhỏ nhất của dãy số (đối với dãy nhập vào có xuất hiện số chẵn): Gán giá trị nhỏ nhất **min** là một số chẵn bất kỳ trong dãy. Sau đó, so sánh min với các số chẵn khác, nếu min nhỏ hơn số chẵn nào thì min được cập nhật mới bằng giá trị số chẵn đó.*

* *Màn hình kết quả như sau:*



<u>Bài 67:</u> Nhập vào một mảng gồm n số nguyên; in mảng vừa nhập ra màn hình; in ra các số âm lẻ trong mảng và cho biết giá trị trung bình của chúng? (X)

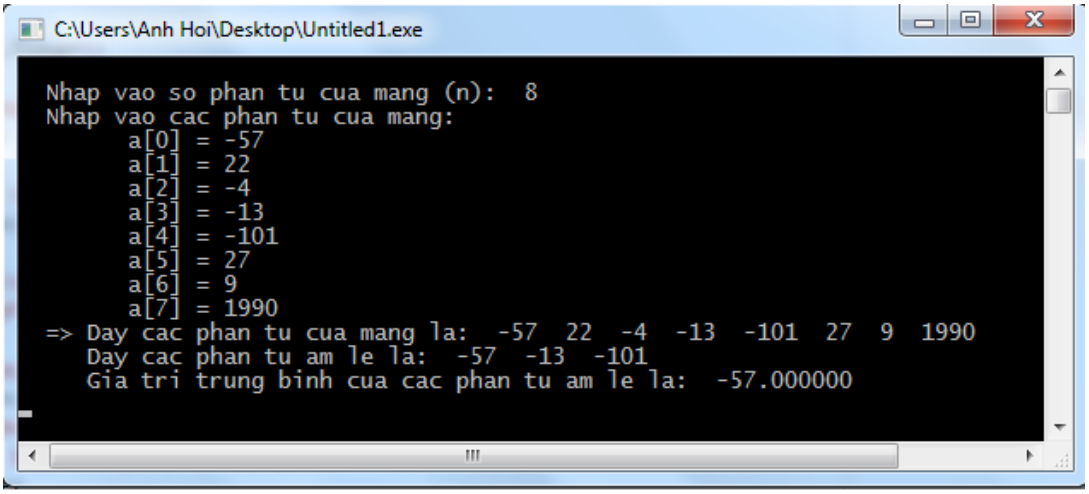
```
#include <stdio.h>
#include <conio.h>
main()
{
   int n,i,AmLe=0,Tong=0;
   float TB;
   printf("\n  Nhap vao so phan tu cua mang (n):  ");
   scanf("%d",&n);
   int a[n];
   printf("  Nhap vao cac phan tu cua mang:  \n");
   for(i=0;i<n;i++)
      {
       printf("\ta[%d] = ",i);
       scanf("%d",&a[i]);
      }
   printf("  => Day cac phan tu cua mang la:  ");
   for(i=0;i<n;i++)
       printf("%d  ",a[i]);
   printf("\n     Day cac phan tu am le la:  ");
   for(i=0;i<n;i++)
```

```
            if((a[i]<0)&&(a[i]%2!=0))
              {
                printf("%d  ",a[i]);
                Tong=Tong+a[i];
                AmLe++;TB=(float)Tong/AmLe;
              }
          printf("\n     Gia tri trung binh cua cac phan tu am le la:  %f\n",TB);
          getch();
          return main();
        }
```

* *Giải thích:*

- Phần tử âm lẻ phải thỏa mãn đồng thời 2 điều kiện: vừa phải nhỏ hơn 0 vừa không chia hết cho 2. Khi kiểm tra số a[i] mà thỏa mãn là số âm lẻ thì ***Tong*** *=* ***Tong****+a[i] và số các số âm lẻ* ***(AmLe)*** *được tăng lên 1 đơn vị. Giá trị trung bình được tính là:* ***TB = Tong/AmLe***

* *Màn hình kết quả như sau:*



Bài 68: Viết chương trình nhập vào n số nguyên. In ra màn hình có bao nhiêu số dương, bao nhiêu số âm, bao nhiêu số bằng 0? (X)

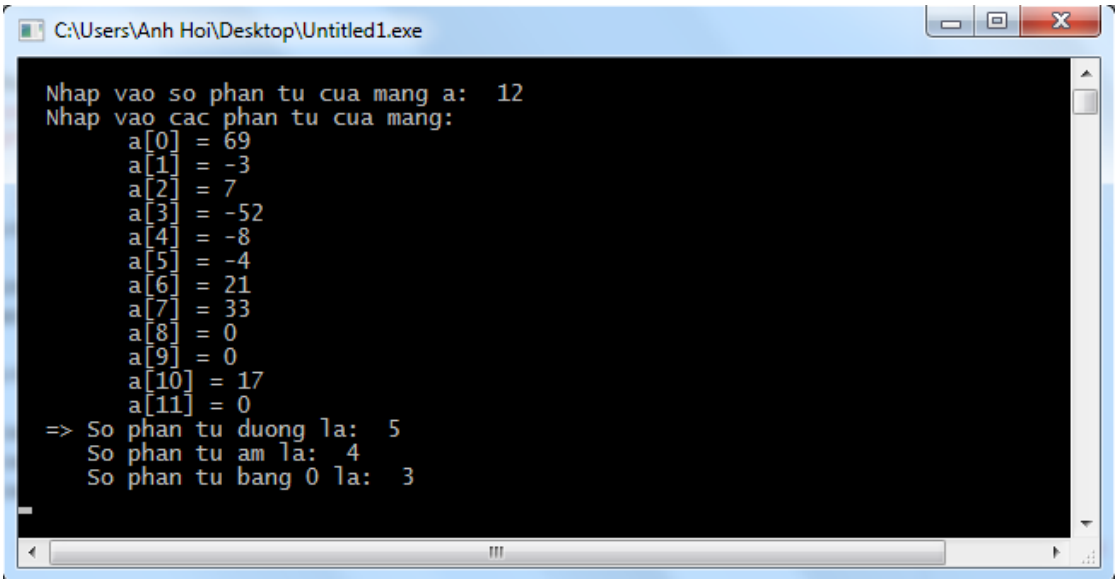
```
#include <stdio.h>
#include <conio.h>
main()
{
   int n,i,Duong=0,Am=0,BangKhong=0;
```

```
    printf("\n  Nhap vao so phan tu cua mang a:  ");
    scanf("%d",&n);
    int a[n];
    printf("  Nhap vao cac phan tu cua mang:  \n");
    for(i=0;i<n;i++)
       {
        printf("\ta[%d] = ",i);
        scanf("%d",&a[i]);
       }
    for(i=0;i<n;i++)
       {
        if(a[i]>0)Duong=Duong+1;
        if(a[i]<0)Am=Am+1;
        if(a[i]==0)BangKhong=BangKhong+1;
       }
    printf("  => So phan tu duong la:  %d\n     So phan tu am la:  %d\n     So
phan tu bang 0 la:  %d\n",Duong,Am,BangKhong);
    getch();
    return main();
}
```

* *Giải thích:*

- Mỗi lần kiểm tra nếu a[i] thỏa mãn lớn hơn 0 thì biến đếm số dương ***Duong*** *được tăng lên 1 đơn vị. Tương tự với biến đếm số âm* ***Am*** *và biến đếm số bằng 0* ***BangKhong****.*

* *Màn hình kết quả như sau:*



<u>Bài 69:</u> Viết chương trình nhập vào n số nguyên. In ra màn hình các số nguyên tố? (X)

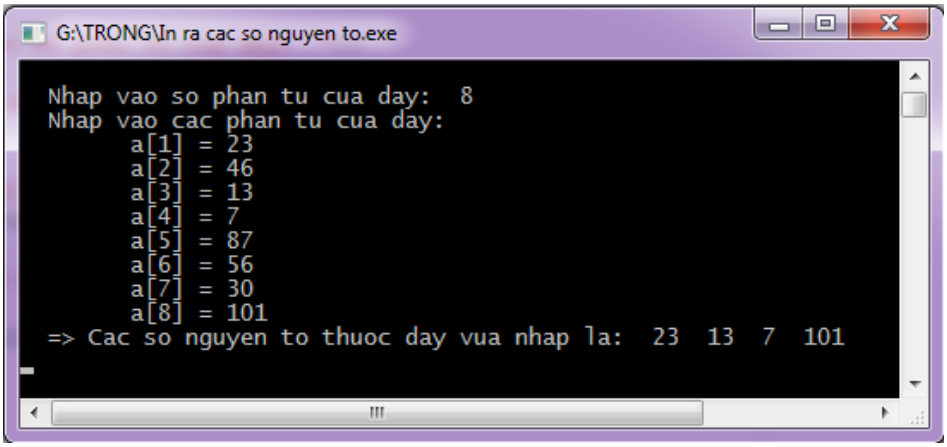
```
#include <stdio.h>
#include <conio.h>
int NguyenTo(int a);
main()
{
    int a[100],i,n;
    printf("\n  Nhap vao so phan tu cua day:  ");
    scanf("%d",&n);
    printf("  Nhap vao cac phan tu cua day:\n");
    for(i=1;i<=n;i++)
      {
       printf("\ta[%d] = ",i);
       scanf("%d",&a[i]);
      }
    printf("  => Cac so nguyen to thuoc day vua nhap la:  ");
    for(i=1;i<=n;i++)
       if(NguyenTo(a[i])==1)printf("%d  ",a[i]);
    printf("\n");
```

```
    getch();
    return main();
}
int NguyenTo(int a)
{  int i,b=0;
   if(a>=2)
    {for(i=2;a%i!=0;i++);
    if(i==a)b=1;}
   return (b);
}
```

* *Giải thích:*

- Kiểm tra từng số nhập vào trong n số. Nếu hàm kiểm tra tính nguyên tố trả về giá trị là 1 thì số đó thỏa mãn là số nguyên tố và in ra màn hình.

* *Màn hình kết quả như sau:*



Bài 70: Viết chương trình nhập vào một dãy gồm n số nguyên dương. Tìm số nguyên tố lớn nhất trong dãy? (X)

```
#include <stdio.h>
#include <conio.h>
int NguyenTo(int a);
main()
{
    int a[100],i,n,Max,Dem=0;
```

```
    printf("\n  Nhap vao so phan tu cua day:  ");
    scanf("%d",&n);
    printf("  Nhap vao cac phan tu cua day:\n");
    for(i=1;i<=n;i++)
      {
       printf("\ta[%d] = ",i);
       scanf("%d",&a[i]);
      }
    for(i=1;i<=n;i++)
      if(NguyenTo(a[i])==1)
        {Max=a[i];Dem++;}
    if(Dem==0)printf("  => Khong co so nguyen to nao trong day vua
nhap.\n");
    else
      {
        for(i=1;i<=n;i++)
          if(NguyenTo(a[i])==1&&a[i]>Max)Max=a[i];
        printf("  => So nguyen to lon nhat trong day la:  %d",Max);
      }
    printf("\n");
    getch();
    return main();
}
int NguyenTo(int a)
{
    int i,b=0;
    if(a>=2)
      {
        for(i=2;a%i!=0;i++);
        if(i==a)b=1;
      }
    return (b);
```

}

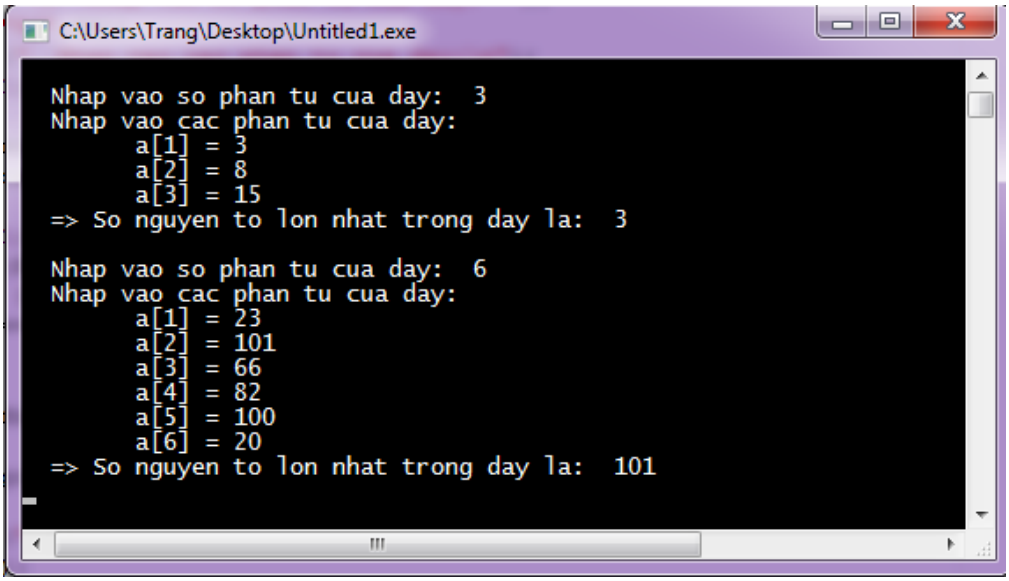
* *Giải thích:*

*a) Duyệt từng phần tử của mảng, nếu phần tử nào thỏa mãn là số nguyên tố thì gán biến **Max** bằng giá trị của phần tử đó, đồng thời tăng biến **Dem** lên 1 đơn vị. Bước làm này là để gán biến Max bằng một giá trị số nguyên tố nào đó có trong dãy. Trường hợp nếu không tìm thấy số nguyên tố nào thì biến Max sẽ không có giá trị khởi đầu và biến Dem = 0, khi ấy thực hiện lệnh xuất ra màn hình dòng chữ "Không có số nguyên tố nào trong dãy vừa nhập".*

*b) Sau khi kiểm tra thấy trong dãy có số nguyên tố thì bắt đầu đi tìm số nguyên tố lớn nhất. Biến **Max** đã được gán giá trị là số nguyên tố cuối cùng mà nó tìm thấy. Duyệt từng phần tử của mảng, nếu phần tử nào vừa thỏa mãn là số nguyên tố vừa có giá trị lớn hơn biến Max thì biến Max được cập nhật mới bằng giá trị của phần tử đó.*

Xuất giá trị của biến Max ra màn hình.

* *Màn hình kết quả như sau:*



Bài 71: Viết chương trình nhập vào một dãy gồm n số nguyên dương. Tính trung bình cộng của các số nguyên tố có trong dãy? (X)

```
#include <stdio.h>
#include <conio.h>
int NguyenTo(int a);
main()
{
    int a[100],i,n,Dem=0,Tong=0;
    printf("\n  Nhap vao so phan tu cua day:  ");
```

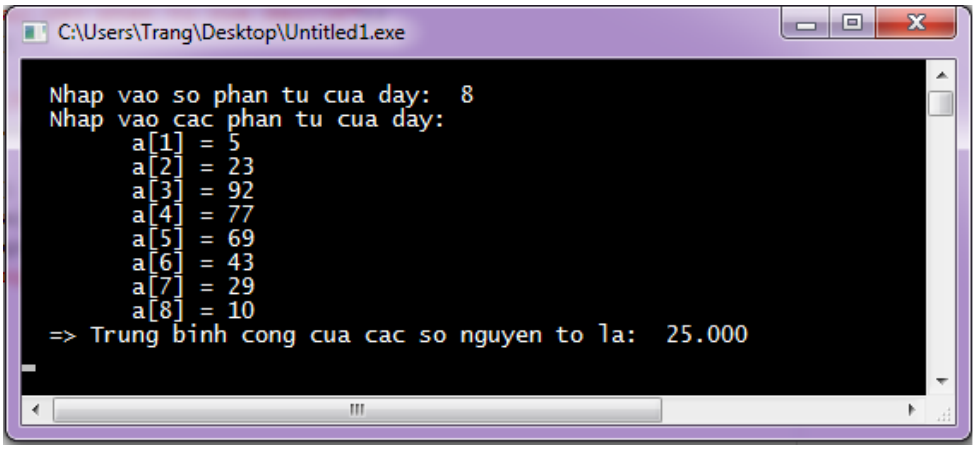
```
    scanf("%d",&n);
    printf("  Nhap vao cac phan tu cua day:\n");
    for(i=1;i<=n;i++)
      {
       printf("\ta[%d] = ",i);
       scanf("%d",&a[i]);
      }
    for(i=1;i<=n;i++)
       if(NguyenTo(a[i])==1)
          {Tong=Tong+a[i];Dem++;}
    printf("  => Trung binh cong cua cac so nguyen to la:
%0.3f\n",(float)Tong/Dem);
    getch();
    return main();
}
int NguyenTo(int a)
{
    int i,b=0;
    if(a>=2)
      {
        for(i=2;a%i!=0;i++);
        if(i==a)b=1;
      }
    return (b);
}
```

* *Giải thích:*

*- Bài này tương tự bài tìm các số nguyên tố trong dãy số nguyên nhập vào. Kiểm tra từng phần tử của dãy, nếu số nào mà thỏa mãn là số nguyên tố thì biến **Tong** được cộng dồn thêm một lượng bằng số nguyên tố đó với khởi đầu bằng 0, đồng thời tăng biến **Dem** lên 1 đơn vị. Biến Dem đại diện cho số lượng số nguyên tố có trong dãy.*

- Tính trung bình cộng thì chỉ việc lấy giá trị biến Tong chia cho giá trị biến Dem. Lưu ý phải ép kiểu float cho kết quả vì kiểu của biến Tong và biến Dem là kiểu nguyên còn giá trị trung bình thì thường là số thực.

* *Màn hình kết quả như sau:*



<u>Bài 72:</u> Dãy Fibonacci là dãy vô hạn các số tự nhiên bắt đầu bằng hai phần tử 0 và 1 được định nghĩa như sau:

$$F[n] = \begin{cases} 0 & khi\ n = 0 \\ 1 & khi\ n = 1 \\ F[n-1] + F[n-2] & khi\ n > 1 \end{cases}$$

Viết chương trình nhập vào số nguyên dương n, tính và in ra các phần tử của dãy Fibonacci? (X)

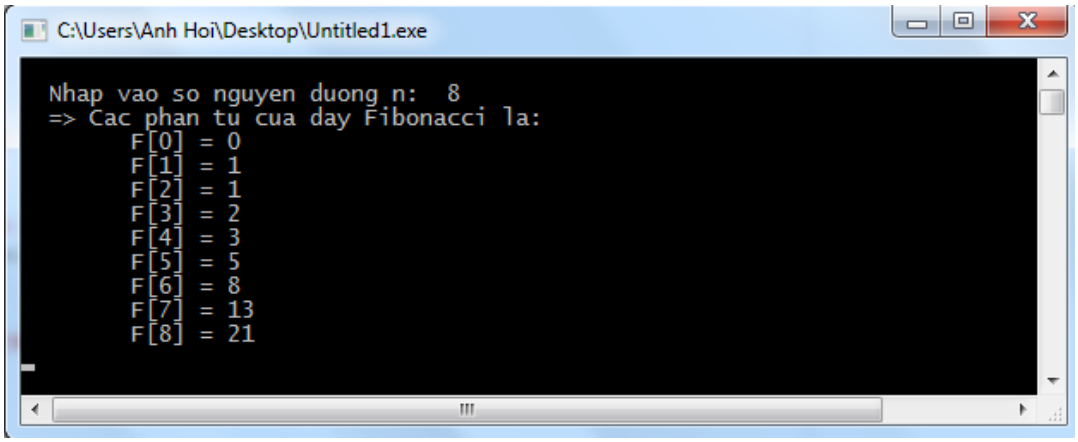
```
#include <stdio.h>
#include <conio.h>
int FIB (int a);
main()
{
    int n,i;
    printf("\n  Nhap vao so nguyen duong n:  ");
    scanf("%d",&n);
    int a[n+1];
    printf("  => Cac phan tu cua day Fibonacci la:\n");
    for(i=0;i<=n;i++)
     {
       FIB(i);
       printf("\tF[%d] = %d\n",i,FIB(i));
     }
```

```
        getch();
        return main();
    }
    int FIB (int a)
    {
        if(a==0)return 0;
        else if(a==1)return 1;
          else return FIB(a-1)+FIB(a-2);
    }
```

* *Giải thích:*

- Vòng for lặp biến i (i chạy từ 0 đến n). Lặp đến đâu thì in ra màn hình F[i] đến đó. Hàm FIB(i) là hàm tính giá trị Fibonacci của số i và được định nghĩa bên ngoài hàm main().

* *Màn hình kết quả như sau:*



Bài 73: Cho một dãy gồm n số nguyên, hãy sắp xếp dãy trên theo chiều không giảm. Đưa ra màn hình dãy ban đầu và dãy đã sắp xếp?

```
#include <stdio.h>
#include <conio.h>
main()
{
   int n,i,j,TrungGian;
   printf("\n  Nhap vao so phan tu cua chuoi (n):  ");
   scanf("%d",&n);
```

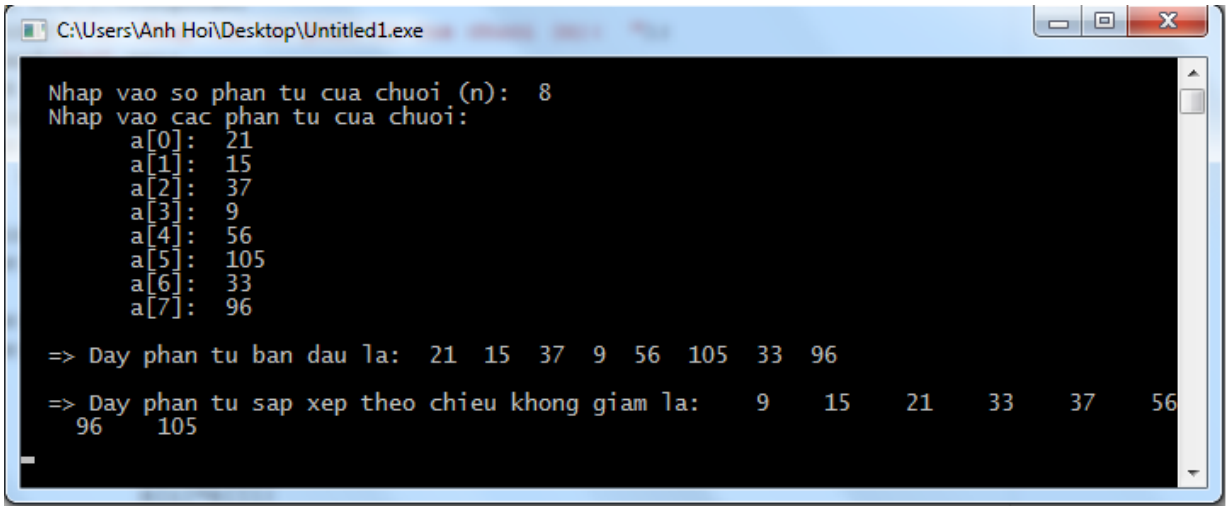
```
        int a[n];
        printf("  Nhap vao cac phan tu cua chuoi:\n");
        for(i=0;i<n;i++)
          {printf("\ta[%d]:  ",i);
           scanf("%d",&a[i]);}
        printf("\n  => Day phan tu ban dau la:  ");
        for(i=0;i<n;i++)
           printf("%d  ",a[i]);
        for(i=0;i<n-1;i++)
        for(j=i+1;j<n;j++)
        {
            if(a[i]>=a[j])
            {
                  TrungGian=a[i];
                  a[i]=a[j];
                  a[j]=TrungGian;
            }
        }
        printf("\n\n  => Day phan tu sap xep theo chieu khong giam la:  ");
        for(i=0;i<n;i++)
        printf("  %d  ",a[i]);
        printf("\n");
        getch();
        return main();
    }
```

* *Giải thích:*

- Thuật toán sắp xếp dãy số theo chiều không giảm: Giả sử dãy nhập vào có n số thì số đầu tiên trong dãy đã sắp xếp được xác định bằng giá trị nhỏ nhất trong các số từ a[0] đến a[n-1], số thứ hai được xác định bằng giá trị nhỏ nhất trong các số từ a[1] đến a[n-1], số thứ 3 được xác định bằng giá trị nhỏ nhất trong các số từ a[2] đến a[n-1].....số thứ n – 1 được xác định bằng giá trị nhỏ nhất trong hai số n – 2 và n – 1. Sử dụng hai lệnh for lồng nhau với ý nghĩa như vậy.

- Tiếp theo là đưa ra màn hình dãy ban đầu nhập vào và dãy đã sắp xếp, dùng lệnh for() bao hàm lệnh printf().

* *Màn hình kết quả như sau:*



<u>Bài 74:</u> Cho một dãy gồm n số thực. Tìm số có giá trị lớn thứ k trong dãy (k được nhập từ bàn phím)? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
   float TrungGian,a[100];
   int n,i,j,k;
   printf("\n  Nhap vao tong so phan tu cua day (n):  ");
   scanf("%d",&n);
   printf("  Nhap vao cac phan tu cua day:  ");
   for(i=0;i<n;i++)
     scanf("%f",&a[i]);
   for(i=0;i<n-1;i++)
     for(j=i+1;j<n;j++)
       if(a[i]<a[j])
          {
            TrungGian=a[i];
            a[i]=a[j];
            a[j]=TrungGian;
```

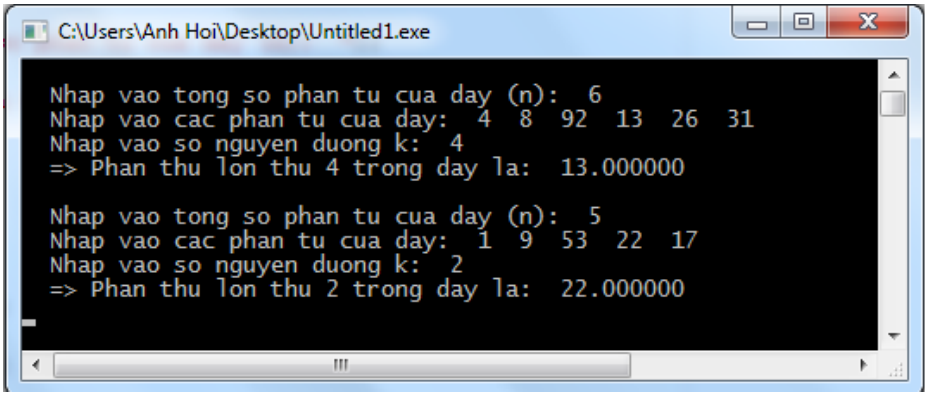
```
        }
    printf("  Nhap vao so nguyen duong k:  ");
    scanf("%d",&k);
    printf("  => Phan thu lon thu %d trong day la:  %f\n",k,a[k-1]);
    getch();
    return main();
}
```

* *Giải thích:*

- Để tìm được phần tử lớn thứ k thì trước hết phải sắp xếp dãy đã cho theo chiều giảm dần. Trong dãy đã sắp xếp theo chiều giảm dần thì phần tử thứ i được xác định bằng cách lấy phần tử lớn nhất trong các phần tử từ phần tử thứ i đến phần tử cuối cùng của dãy chưa sắp xếp. Sử dụng biến ***TrungGian*** *để đổi giá trị cho nhau giữa các phần tử để được dãy sắp xếp giảm dần như mong muốn.*

- Từ đó suy ra phần tử lớn thứ k trong dãy là phần tử a[k – 1].

* *Màn hình kết quả như sau:*



Bài 75: Viết chương trình:

- Lập hàm tính tổng bình phương của các phần tử dương trong mảng a.

- Lập hàm sắp xếp mảng a theo chiều tăng dần.

- Từ chương trình chính nhập mảng a có n phần tử, gọi hai hàm vừa lập ở trên thực hiện sắp xếp lại mảng, tính tổng bình phương của các phần tử dương trong mảng a. In ra màn hình kết quả? (X)

```
#include <stdio.h>
#include <conio.h>
int TongBinhPhuong(int a[],int n)
{
```

```
   int i,Tong=0;
   for(i=1;i<=n;i++)
     if(a[i]>0)Tong+=a[i]*a[i];
   return (Tong);
}
void SapXep(int a[],int n)
{
   int i,j,TrungGian;
   for(i=1;i<n;i++)
     for(j=i+1;j<=n;j++)
       if(a[i]>a[j])
          {
            TrungGian=a[i];
            a[i]=a[j];
            a[j]=TrungGian;
          }
   for(i=1;i<=n;i++)
     printf("%d  ",a[i]);
}
main()
{
   int a[100],i,n;
   printf("\n  Nhap vao so phan tu cua mang:  ");
   scanf("%d",&n);
   printf("  Nhap vao cac phan tu cua mang:\n\t");
   for(i=1;i<=n;i++)
     scanf("%d",&a[i]);
   printf("  => Cac phan tu sau khi sap xep tang dan la:\n\t");
   SapXep(a,n);
   printf("\n  => Tong binh phuong cua cac phan tu duong la:  %d\n",
TongBinhPhuong(a,n));
   getch();
```

```
    return main();
}
```

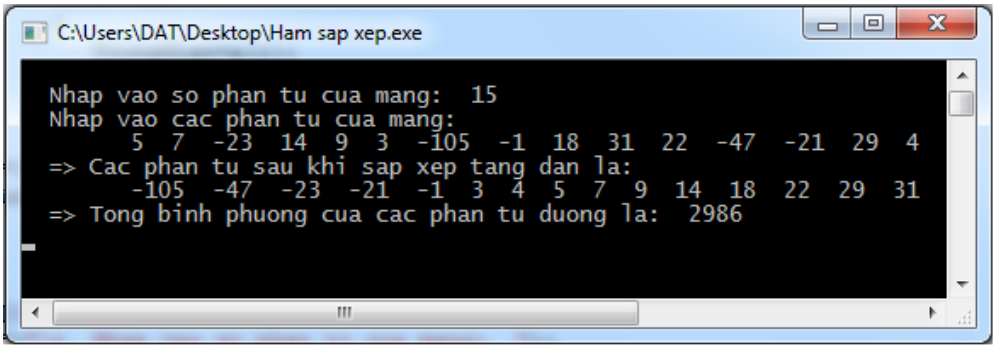
* *Giải thích:*

Khi truyền mảng vào cho hàm thì bản thân mảng chính là tham biến, có nghĩa là khi ra khỏi hàm thì các giá trị của mảng đã thay đổi. Chỉ cần truyền tên của mảng và kích cỡ mảng là được.

*a) Hàm tính tổng bình phương: Vòng lặp for duyệt qua từng phần tử, nếu phần tử nào > 0 thì biến **Tong** được cộng dồn bằng một lượng bình phương của a[i] với khởi tạo Tong = 0. Kết quả của biến Tong được trả về cho hàm TongBinhPhuong(a,n).*

b) Hàm sắp xếp lại các giá trị của mảng theo chiều tăng dần: biến i chạy từ 1 đến n – 1, phần tử thứ i được so sánh với các phần tử từ phần tử thứ j = i + 1 đến phần tử thứ n, nếu a[i] > a[j] thì tiến hành đổi chỗ cho nhau. Sau đó in luôn ra các phần tử đã sắp xếp tăng dần ngay trong chương trình con (hoặc có thể in ra ở chương trình chính cũng được).

* *Màn hình kết quả như sau:*



<u>Bài 76:</u> Viết chương trình nhập vào từ bàn phím mảng a có N phần tử. Lọc các số dương đưa vào mảng b, lọc các số âm đưa vào mảng c. In các phần tử của hai mảng b và c vừa tạo ra? (X)

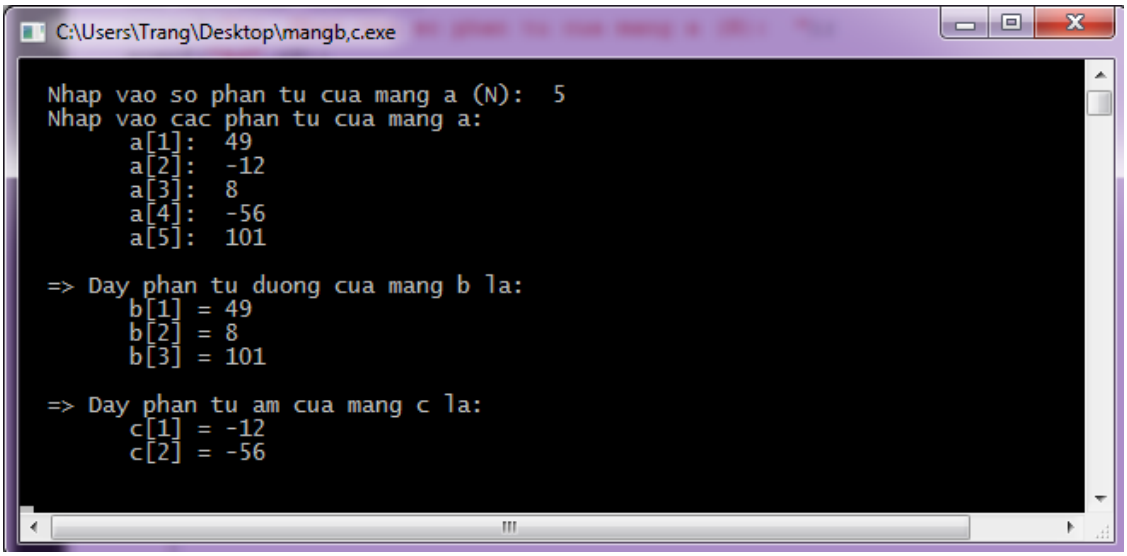
```
#include <stdio.h>
#include <conio.h>
main()
{
   int N,i,demb=0,demc=0;
   printf("\n  Nhap vao so phan tu cua mang a (N):  ");
   scanf("%d",&N);
   int a[1000],b[1000],c[1000];
   printf("  Nhap vao cac phan tu cua mang a:\n");
```

```
    for(i=1;i<=N;i++)
      {
       printf("\ta[%d]:  ",i);
       scanf("%d",&a[i]);
       if(a[i]>0)
          {
          demb+=1;
          b[demb]=a[i];
          }
       if(a[i]<0)
          {
          demc+=1;
          c[demc]=a[i];
          }
      }
    printf("\n  => Day phan tu duong cua mang b la:\n");
    for(i=1;i<=demb;i++)
       printf("\tb[%d] = %d\n",i,b[i]);
    printf("\n  => Day phan tu am cua mang c la:\n");
    for(i=1;i<=demc;i++)
       printf("\tc[%d] = %d\n",i,c[i]);
    printf("\n");
    getch();
    return main();
}
```

* *Giải thích:*

- In ra các phần tử của mảng b và mảng c: Nếu thỏa mãn điều kiện a[i] > 0 thì số a[i] được đưa vào mảng b, nếu a[i] < 0 thì số a[i] được đưa vào mảng c. Dùng vòng for để lặp cho các giá trị của biến i. Biến ***demb*** *và biến* ***demc*** *dùng để đếm số lượng giá trị của mảng b và mảng c. Ở đây, mình định nghĩa cho phần tử đầu tiên là a[1] cho dễ hiểu (thay vì phần tử a[0]).*

* *Màn hình kết quả như sau:*



<u>Bài 77:</u> Viết chương trình nhập vào dãy số và kiểm tra xem dãy đó có phải là dãy tăng, giảm, đối xứng, đan dấu, cấp số cộng, cấp số nhân, một đoạn của dãy Fibonacci hay không? (X)

```
#include <stdio.h>
#include <conio.h>
int DayTang(int a[],int n);
int DayGiam(int a[],int n);
int DayDoiXung(int a[],int n);
int DayDanDau(int a[],int n);
int CapSoCong(int a[],int n);
int CapSoNhan(int a[],int n);
int DoanDayFibonacci(int a[],int n);
main()
{
    int a[100],n,i,k,m;
    printf("\n  Nhap vao so phan tu cua mang:  ");
    scanf("%d",&n);
    printf("  Nhap vao cac phan tu cua mang:\n");
    for(i=1;i<=n;i++)
      {
       printf("\ta[%d] = ",i);
```

```
        scanf("%d",&a[i]);
      }
    printf("  Nhap mot so tuong ung voi bai toan sau:\n\t1-Kiem tra xem co phai day tang.\n\t2-Kiem tra xem co phai day giam.\n\t3-Kiem tra xem co phai day doi xung.\n\t4-Kiem tra xem co phai day dan dau.\n\t5-Kiem tra xem day co phai la cap so cong.\n\t6-Kiem tra xem day co phai la cap so nhan.\n\t7-Kiem tra xem day co phai doan con cua day Fibonacci.\n\t\t   ");
    vao:scanf("%d",&k);
    switch(k)
      {
        case 1:{printf("\t=> Tinh chat tang cua day tren la: %d\n",DayTang(a,n));getch();printf("\tNhap phim 1 de tiep tuc va nhap phim khac de quay lai ban dau:
");scanf("%d",&m);if(m==1){printf("\tNhap 1 or 2 or 3 or 4 or 5 or 6 or 7:
");goto vao;}else goto end;}
        case 2:{printf("\t=> Tinh chat giam cua day tren la: %d\n",DayGiam(a,n));getch();printf("\tNhap phim 1 de tiep tuc va nhap phim khac de quay lai ban dau:
");scanf("%d",&m);if(m==1){printf("\tNhap 1 or 2 or 3 or 4 or 5 or 6 or 7:
");goto vao;}else goto end;}
        case 3:{printf("\t=> Tinh chat doi xung cua day tren la: %d\n",DayDoiXung(a,n));getch();printf("\tNhap phim 1 de tiep tuc va nhap phim khac de quay lai ban dau:
");scanf("%d",&m);if(m==1){printf("\tNhap 1 or 2 or 3 or 4 or 5 or 6 or 7:
");goto vao;}else goto end;}
        case 4:{printf("\t=> Tinh chat dan dau cua day tren la: %d\n",DayDanDau(a,n));getch();printf("\tNhap phim 1 de tiep tuc va nhap phim khac de quay lai ban dau:
");scanf("%d",&m);if(m==1){printf("\tNhap 1 or 2 or 3 or 4 or 5 or 6 or 7:
");goto vao;}else goto end;}
        case 5:{printf("\t=> Tinh chat cap so cong cua day tren la: %d\n",CapSoCong(a,n));getch();printf("\tNhap phim 1 de tiep tuc va nhap phim khac de quay lai ban dau:
");scanf("%d",&m);if(m==1){printf("\tNhap 1 or 2 or 3 or 4 or 5 or 6 or 7:
");goto vao;}else goto end;}
        case 6:{printf("\t=> Tinh chat cap so nhan cua day tren la: %d\n",CapSoNhan(a,n));getch();printf("\tNhap phim 1 de tiep tuc va nhap phim khac de quay lai ban dau:
```

```
");scanf("%d",&m);if(m==1){printf("\tNhap 1 or 2 or 3 or 4 or 5 or 6 or 7:
");goto vao;}else goto end;}
        case 7:{printf("\t=> Tinh chat doan Fibonacci cua day tren la:
%d\n",DoanDayFibonacci(a,n));getch();printf("\tNhap phim 1 de tiep tuc va
nhap phim khac de quay lai ban dau:
");scanf("%d",&m);if(m==1){printf("\tNhap 1 or 2 or 3 or 4 or 5 or 6 or 7:
");goto vao;}else goto end;}
        default:printf("\tNhap lai!\n");printf("\tNhap 1 or 2 or 3 or 4 or 5 or 6
or 7: ");goto vao;
      }
    getch();
    end:return main();
}

int DayTang(int a[],int n)
{
    int i,d=1;
    for(i=1;i<n;i++)
    if(a[i]>=a[i+1])d=0;
    return (d);
}
int DayGiam(int a[],int n)
{
    int i,d=1;
    for(i=1;i<n;i++)
    if(a[i]<=a[i+1])d=0;
    return (d);
}
int DayDoiXung(int a[],int n)
{
    int i,d=1;
    for(i=1;i<n;i++)
    if(a[i]!=a[n-i+1])d=0;
```

```
    return (d);
}
int DayDanDau(int a[],int n)
{
    int i,d=1;
    for(i=1;i<n;i++)
    if(a[i]*a[i+1]>=0)d=0;
    return (d);
}
int CapSoCong(int a[],int n)
{
    int i,d,k=1;
    d=a[2]-a[1];
    for(i=1;i<n;i++)
    if(a[i+1]-a[i]!=d)k=0;
    return (k);
}
int CapSoNhan(int a[],int n)
{
    int i,k=1;
    float q;
    q=(float)a[2]/a[1];
    for(i=1;i<n;i++)
    if((float)a[i+1]/a[i]!=q)k=0;
    return (k);
}
int DoanDayFibonacci(int a[],int n)
{
    int i,j,k=1,m=0,x,y,b[100];
    if((a[1]==1)&&(a[2]==1)||(a[1]==1)&&(a[2]==2))m=1;
    if(a[1]>1)
```

```
        {
          b[0]=1;b[1]=1;
          for(i=2;i<=a[1];i++)
            {
              b[i]=b[i-1]+b[i-2];
              if(a[1]==b[i]){x=1;j=b[i-1];}
            }
          if(a[2]==a[1]+j)y=1;
          if((x==1)&&(y==1))m=1;
        }
      if(m==0)k=0;
      if(m==1)
        {
          for(i=3;i<=n;i++)
          {if(a[i]!=a[i-1]+a[i-2])k=0;}
        }
      return (k);
  }
```

* *Giải thích:*

- Bài này định nghĩa thêm hàm để viết tất cả các phép toán tương ứng với từng ý nhỏ: kiểm tra dãy tăng, giảm, đối xứng,...Sau đó, muốn sử dụng ý nào (hàm nào) thì nhập tùy chọn từ bàn phím theo hướng dẫn hiện ra trên màn hình. Hàm chính main() sử dụng tùy chọn nhập từ bàn phím để sử dụng hàm tương ứng, dùng cấu trúc rẽ nhánh switch(). Lệnh nhảy vô điều kiện ***goto*** *được dùng để tạo sự thuận tiện trong việc điều khiển công việc, có thể dùng lệnh này hoặc không dùng cũng không sao (cô giáo không dạy mà!!). Vì bài này là tổng hợp rất nhiều hàm nên hơi dài một chút, mọi người chịu khó vậy nhé!*

a) Đối với dãy tăng: Dãy tăng là dãy mà phần tử a[i+1] lớn hơn phần tử a[i] với i chạy từ 1 đến n – 1. Ban đầu giả sử rằng dãy trên là tăng (d = 1), khi kiểm tra mà phát hiện ra một trường hợp a[i] ≥ a[i+1] thì lập tức kết luận rằng dãy không tăng (gán d = 0).

b) Đối với dãy giảm thì ngược lại so với dãy tăng: Lúc đầu cũng giả sử dãy nhập vào là dãy giảm (khởi tạo d = 1), khi kiểm tra mà thấy 1 trường hợp a[i] ≤ a[i+1] thì kết luận dãy không giảm (gán d = 0).

c) Dãy đối xứng là dãy mà các phần tử cách đều 2 đầu mút thì giống hệt nhau, ví dụ dãy số sau: 1, 2, 3, 4, 5, 4, 3, 2, 1 là một dãy đối xứng. Để kiểm tra dãy nào đó có tính đối xứng hay không ta kiểm tra xem a[i] có bằng a[n – i +1] không, với n là số phần tử của dãy. Nếu có một trường hợp xảy ra a[i] ≠ a[n – i +1] thì dãy đã cho không phải là dãy đối xứng.

*d) Dãy đan dấu: là dãy mà phần tử thứ i có dấu ngược với dấu của phần tử thứ i +1. Nếu a[i] và a[i+1] cùng mang dấu dương hoặc cùng mang dấu âm thì dãy nhập vào không có tính đan dấu (gán d = 0). Nếu a[i] và a[i+1] cùng dấu thì tích a[i]*a[i+1] > 0.*

e) Dãy là cấp số cộng: Là dãy mà phần tử a[i+1] = a[i] +d trong đó d là công sai của cấp số cộng (gán d = a[2] – a[1]) và $i = \overline{1,\ n-1}$. *Nếu hiệu a[i+1] – a[i] ≠ d thì dãy số đó không phải là cấp số cộng (gán k = 0).*

g) Dãy số là cấp số nhân: Tương tự như cấp số cộng, nếu kết quả của phép chia $\frac{a[i+1]}{a[i]} \neq q$ *(trong đó q là công bội và gán* $q = \frac{a[2]}{a[1]}$ *) thì dãy số không thỏa mãn cấp số nhân (gán k = 0).*

h) Xét xem dãy nhập vào có phải là đoạn con của dãy Fibonacci hay không. Sử dụng định nghĩa sau về dãy Fibonacci: (tại vì có 2 định nghĩa khác nhau về dãy Fibonacci nên mình dùng định nghĩa này)

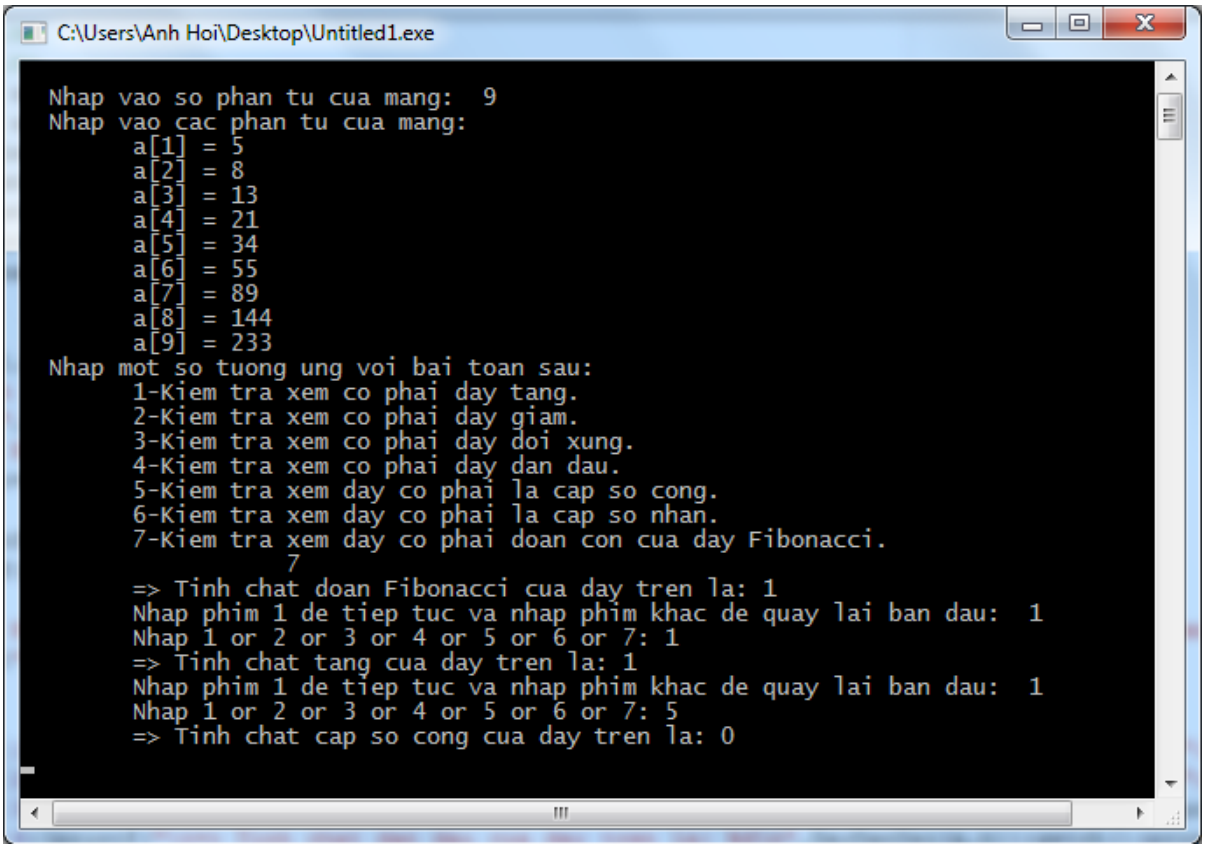
$$F[n] = \begin{cases} 1 & khi\ n = 0 \quad or \quad n = 1 \\ F[n-1] + F[n-2] & khi\ n > 1 \end{cases}$$

Ở đây, mình dùng cách làm tổng quát cho mọi đoạn con của dãy Fibonacci chứ không làm trường hợp riêng (kiểm tra có phải là dãy Fibonacci với 2 phần tử đầu tiên bằng 1), chính vì thế đoạn mã khá nhiều biến trung gian và gây khó hiểu.

+ ***Bước 1:*** *Kiểm tra xem a[1] và a[2] có phải là hai số Fibonacci liên tiếp nhau hay không. Nếu a[1] và a[2] không phải hai số Fibonacci liên tiếp thì kết luận cả dãy nhập vào không phải là đoạn con của dãy Fibonacci, nếu thỏa mãn 2 số Fibonacci liên tiếp thì làm tiếp bước 2. Thuật toán xác định xem a[1] và a[2] có phải là 2 số Fibonacci liên tiếp hay không thì còn phải chia ra làm nhiều trường hợp và sử dụng các biến trung gian khác (biến m, x, y, j và mảng phụ b[]).*

+ ***Bước 2:*** *Khi hai phần tử a[1] và a[2] đã thỏa mãn là hai số Fibonacci liên tiếp thì chưa thể kết luận dãy nhập vào là đoạn con của dãy Fibonacci được mà phải kiểm tra tiếp điều kiện xem a[i] = a[i – 1] + a[i – 2] hay không (i chạy từ 3 tới n). Nếu thỏa mãn điều kiện này nữa thì dãy nhập vào mới là đoạn con của dãy Fibonacci, còn không thỏa mãn điều kiện đủ này thì dãy nhập vào vẫn không là đoạn con của dãy Fibonacci (gán k = 0).*

** <u>Màn hình kết quả như sau:</u> (quy ước 1 là dãy có tính chất tăng, giảm, đối xứng, đan dấu…và 0 là không có các tính chất trên)*



<u>Bài 78*:</u> Nhập vào một dãy số nguyên. In ra dãy đó, tính trung bình cộng các số nguyên dương trong dãy và kiểm tra xem có phải là dãy đối xứng hay không? (X)

(Đề kiểm tra giữa kỳ lớp THCS 3, ĐHKHTN – ĐHQGHN, Thứ 5 ngày 01/11/2012, Kỳ I năm học 2012 – 2013, thời gian làm bài: 30 phút)

```
#include <stdio.h>
#include <conio.h>
int DayDoiXung(int a[],int n);
main()
{
    int n,a[100],Dem=0,Tong=0,i;
    printf("\n  Nhap vao so phan tu cua day:  ");
    scanf("%d",&n);
    printf("  Nhap vao cac phan tu cua day:\n");
    for(i=1;i<=n;i++)
      {
        printf("\ta[%d] = ",i);
```

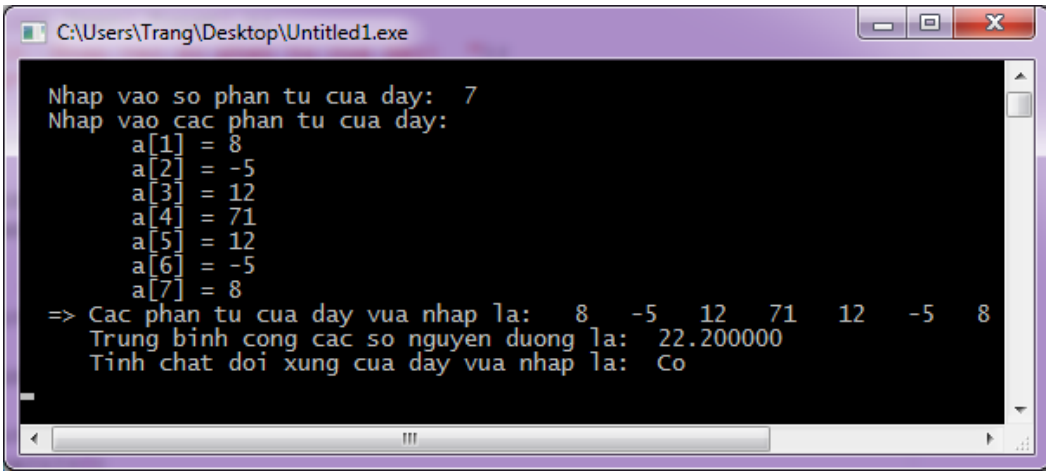
```
        scanf("%d",&a[i]);
      }
    printf("  => Cac phan tu cua day vua nhap la:");
    for(i=1;i<=n;i++)
        printf("   %d",a[i]);
    for(i=1;i<=n;i++)
        if(a[i]>0)
          {
            Tong=Tong+a[i];
            Dem++;
          }
    printf("\n     Trung binh cong cac so nguyen duong la:
%f\n",(float)Tong/Dem);
    printf("     Tinh chat doi xung cua day vua nhap la:  ");
    if(DayDoiXung(a,n)==1)printf("Co\n");else printf("Khong\n");
    getch();
    return main();
}

int DayDoiXung(int a[],int n)
{
    int i,d=1;
    for(i=1;i<n;i++)
    if(a[i]!=a[n-i+1])d=0;
    return (d);
}
```

* *Giải thích:*

*- Đưa vào biến **Tong** để tính tổng các số nguyên dương và biến **Dem** để đếm số lượng các số nguyên dương. Trung bình cộng được tính bằng **Tong/Dem**.*

- Dãy đối xứng là dãy mà các phần tử cách đều 2 đầu mút là giống hệt nhau.

* *Màn hình kết quả như sau:*



<u>Bài 79:</u> Viết chương trình nhập vào một dãy gồm n số nguyên. Tìm đoạn con không giảm dài nhất có trong dãy?

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a[100],i,j,k,n,Dem=1,Max=1;
    printf("\n  Nhap vao so luong phan tu: ");
      scanf("%d",&n);
    printf("  Nhap vao cac phan tu cua day:  ");
    for(i=1;i<=n;i++)
      scanf("%d",&a[i]);
    for(i=1;i<n;i++)
      {
       for(j=i;j<n;j++)
         {
          if(a[j]<=a[j+1])Dem++;
          if(a[j]>a[j+1]||j+1==n)
            {
             if(Max<Dem)Max=Dem;
             Dem=1;
             break;
```

```
            }
          }
        }
    if(Max==1)printf("  => Khong co doan con khong giam nao ca!\n");
    else
    {
     printf("  => Doan con khong giam dai nhat co so phan tu la:  %d\n",Max);
     printf("  => Cac doan con khong giam do la:\n\t");
     for(i=1;i<n;i++)
       {
        for(j=i;j<n;j++)
          {
           if(a[j]<=a[j+1])Dem++;
           if(a[j]>a[j+1]||j+1==n)
             {
              if(Max==Dem)
                {
                 if(j+1==n&&a[j]<=a[j+1])j++;
                 for(k=j-Dem+1;k<=j;k++)
                    printf("%d  ",a[k]);
                 printf("\n\t");
                }
              Dem=1;
              break;
             }
          }
       }
    }
    getch();
    return main();
}
```

* Giải thích:

Bài 80*: Nhập vào dãy n số (n > 10). Cho biết số phần tử dương trong dãy, sắp xếp dãy theo chiều giảm dần và đưa ra số nhỏ nhất có trong dãy?

(Đề thi cuối kỳ lớp THCS 3, ĐHKHTN – ĐHQGHN, Thứ 4 ngày 19/12/2012, Kỳ I năm học 2012 – 2013, thời gian làm bài: 30 phút)

Bài 81*: Viết chương trình nhập vào một dãy N số nguyên (N > 10):

- Tính và đưa ra màn hình giá trị trung bình của các phần tử chẵn trong dãy.

- Liệu có nhiều hơn 2 phần tử có cùng giá trị hay không.

- Dãy vừa nhập có phải là dãy Fibonacci hay không?

(Đề thi cuối kỳ lớp THCS 3, ĐHKHTN – ĐHQGHN, Thứ 6 ngày 21/12/2012, Kỳ I năm học 2012 – 2013, thời gian làm bài: 40 phút)

CHƯƠNG IV. MẢNG NHIỀU CHIỀU

Bài 79: Viết chương trình nhập và đưa mảng 2 chiều ra màn hình (đưa ra màn hình ma trận có dạng khối)?

```
#include <stdio.h>
#include <conio.h>
main()
{
    int m,n,i,j;
    printf("\n  Nhap mang 2 chieu co m dong, n cot:  ");
    scanf("%d%d",&m,&n);
    int a[100][100];
    for(i=1;i<=m;i++)
     {
```

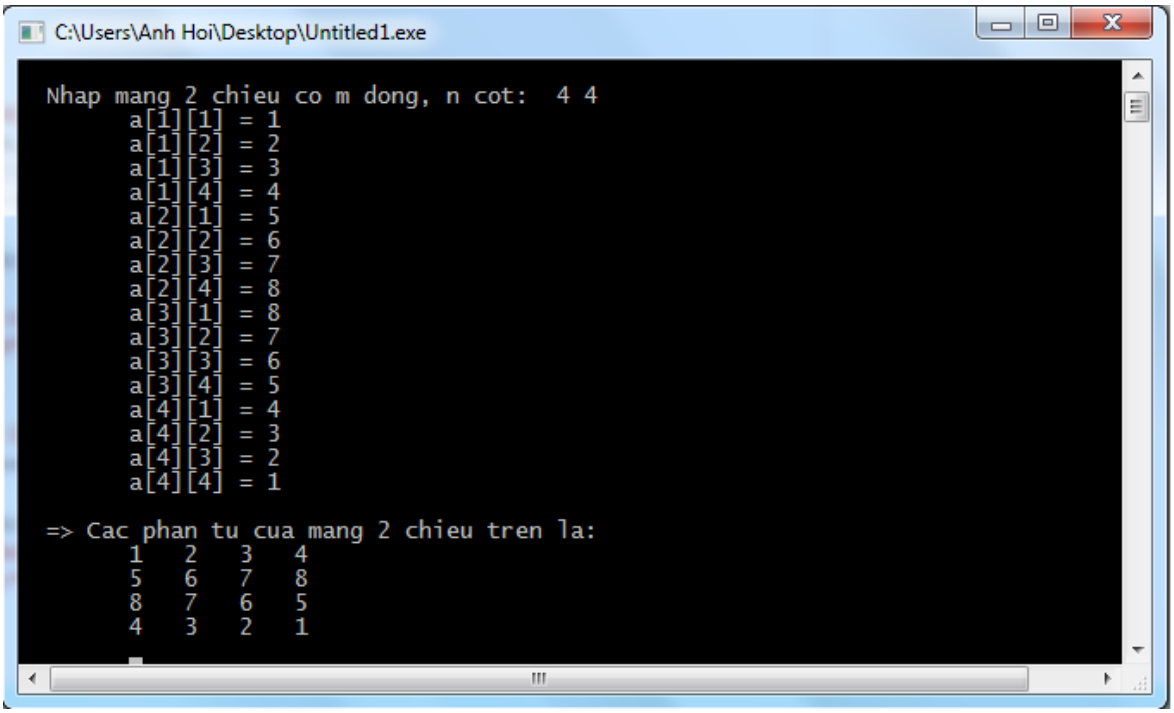
```
        for(j=1;j<=n;j++)
          {
           printf("\ta[%d][%d] = ",i,j);
           scanf("%d",&a[i][j]);
          }
       }
      printf("\n  => Cac phan tu cua mang 2 chieu tren la:\n\t");
      for(i=1;i<=m;i++)
       {
        for(j=1;j<=n;j++)
           printf("%d   ",a[i][j]);
        printf("\n\t");
       }
      getch();
      return main();
}
```

* *Giải thích:*

- Khai báo a[100][100] để định trước kích thước tối đa cho mảng a gồm 100 dòng và 100 cột.

- Mảng hai chiều gồm chỉ số dòng và chỉ số cột nên phải chọn chỉ số dòng là biến i và chỉ số cột là biến j. Cho cả i và j chạy trong 2 vòng lặp for lồng nhau để nhập các phần tử cũng như in các phần tử ra màn hình.

* *Màn hình kết quả như sau:*



<u>Bài 80:</u> Cho một mảng hai chiều gồm các số thực. Hãy tìm phần tử nhỏ nhất trên dòng thứ k (với k được nhập vào từ bàn phím)?

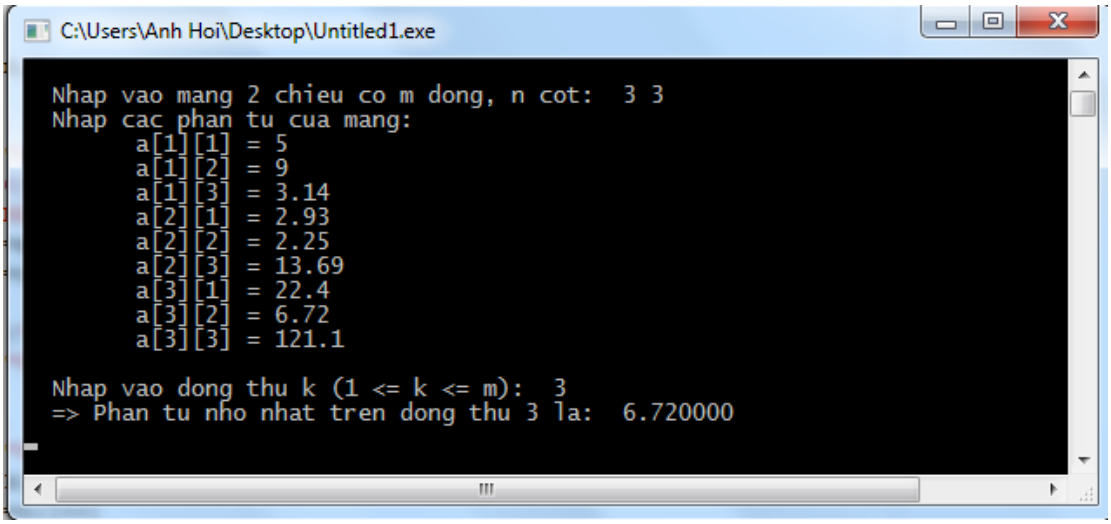
```
#include <stdio.h>
#include <conio.h>
main()
{
    int i,j,m,n,k;
    float min;
    printf("\n  Nhap vao mang 2 chieu co m dong, n cot:  ");
       scanf("%d%d",&m,&n);
    float a[100][100];
    printf("  Nhap cac phan tu cua mang:  \n");
    for(i=1;i<=m;i++)
    for(j=1;j<=n;j++)
    {
       printf("\ta[%d][%d] = ",i,j);
       scanf("%f",&a[i][j]);
    }
```

```
        printf("\n  Nhap vao dong thu k (1 <= k <= m):  ");
           scanf("%d",&k);
           min=a[k][1];
        for(j=1;j<=n;j++)
           if(min>=a[k][j])min=a[k][j];
        printf("  => Phan tu nho nhat tren dong thu %d la:  %f\n",k,min);
        getch();
        return main();
    }
```

* *Giải thích:*

*- Tìm số nhỏ nhất trên dòng thứ k: Ban đầu gán giá trị nhỏ nhất **min** cho phần tử đầu tiên của dòng thứ k là: a[k][1]. Sau đó, so sánh giá trị min này với các giá trị còn lại trên dòng k. Nếu min lớn hơn giá trị nào thì min lại được cập nhật mới bằng giá trị đó. Cứ thế so sánh min đến phần tử cuối cùng của dòng k.*

* *Màn hình kết quả như sau:*



Bài 81: Nhập vào hai ma trận có cùng kích thước mxn. Tính ma trận tổng và đưa ra màn hình ma trận đó? (X)

(Bài này thuộc lớp cô Huyền dạy lý thuyết vào chiều thứ 2 và thực hành sáng thứ 4)

```
#include <stdio.h>
#include <conio.h>
void Tong2MaTran(int a[][100],int b[][100],int c[][100],int m,int n);
main()
```

```
{
    int a[100][100],b[100][100],c[100][100],i,j,m,n;
    printf("\n  Nhap vao hai so m va n:  ");
    scanf("%d%d",&m,&n);
    printf("  Nhap tung phan tu cua ma tran a co %d hang va %d cot:\n\t",m,n);
    for(i=1;i<=m;i++)
      {
        for(j=1;j<=n;j++)
          scanf("%d",&a[i][j]);
        printf("\t");
      }
    printf("\r  Nhap tung phan tu cua ma tran b co %d hang va %d
cot:\n\t",m,n);
    for(i=1;i<=m;i++)
       {
         for(j=1;j<=n;j++)
            scanf("%d",&b[i][j]);
        printf("\t");
       }
    Tong2MaTran(a,b,c,m,n);
    printf("\r  Ma tran c la tong cua 2 ma tran a va b.\n");
    printf("  => Ma tran c la:\n\t");
    for(i=1;i<=m;i++)
      {
        for(j=1;j<=n;j++)
          printf("%5d",c[i][j]);
        printf("\n\t");
      }
    getch();
    return main();
}
void Tong2MaTran(int a[][100],int b[][100],int c[][100],int m,int n)
```

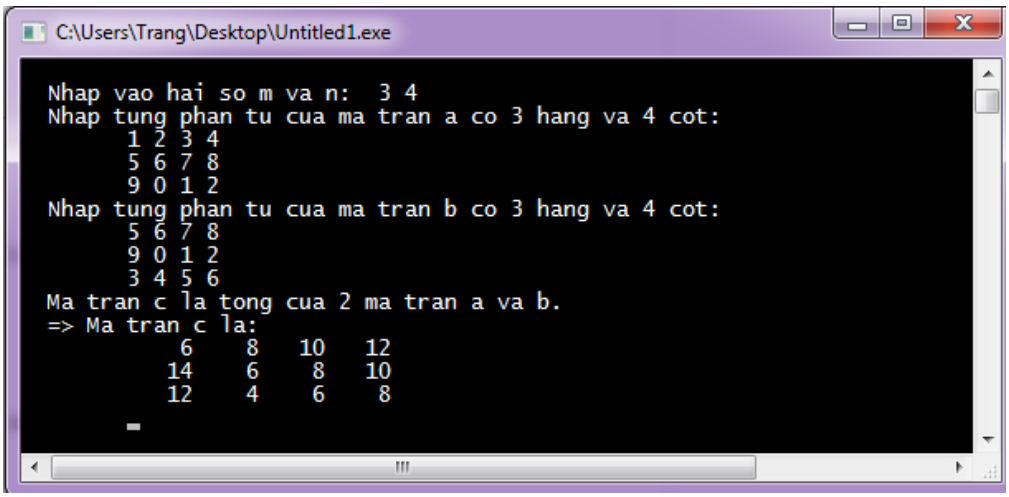
```
{
    int i,j,k;
    for(i=1;i<=m;i++)
      for(j=1;j<=n;j++)
        c[i][j]=a[i][j]+b[i][j];
}
```

* *Giải thích:*

- Đối với phép cộng hai ma trận thì chỉ cần cộng các phần tử ở vị trí tương ứng của hai ma trận đó với nhau là được. Giả sử ma trận c là tổng của hai ma trận a và b thì: c[i][j] = a[i][j] + b[i][j] với i đại diện cho hàng và j đại diện cho cột của ma trận. Phép cộng hai ma trận chỉ xảy ra đối với hai ma trận cùng kích cỡ mxn. Phép cộng này là đơn giản cho nên không cần định nghĩa thêm hàm, tuy nhiên mình vẫn định nghĩa hàm ***Tong2MaTran()*** *để nhìn cho dễ.*

- Muốn nhập vào ma trận có dạng khối thì khi nhập xong n phần tử ta nhấn phím ***Enter*** *để xuống dòng (n là số cột của ma trận). Tương tự khi xuất ra ma trận tổng có dạng khối thì phải dùng lệnh printf("\n") khi vòng lặp chạy xong một giá trị i. Ví dụ: khi nhập vào một ma trận có 3 hàng 4 cột thì ta nhập vào 4 phần tử rồi nhấn Enter và sau đó lại nhập 4 phần tử tiếp rồi lại nhấn Enter,...cho đến hàng cuối cùng.*

* *Màn hình kết quả như sau:*



<u>Bài 82:</u> Viết chương trình:

- Nhập ma trận a có kích cỡ m hàng n cột, nhập ma trận b có kích cỡ n hàng m cột.

- Tính ma trận c là tích của hai ma trận a và b (c = a . b)

- In ra màn hình ma trận a, b, c? (X)

```
#include <stdio.h>
```

```
#include <conio.h>
main()
{
    int a[100][100],b[100][100],c[100][100],i,j,k,m,n;
    printf("\n  Nhap vao hai so m va n:  ");
    scanf("%d%d",&m,&n);
    printf("  Nhap tung phan tu cua ma tran a co %d hang va %d cot:\n\t",m,n);
    for(i=1;i<=m;i++)
      for(j=1;j<=n;j++)
        scanf("%d",&a[i][j]);
    printf("  Nhap tung phan tu cua ma tran b co %d hang va %d cot:\n\t",n,m);
    for(i=1;i<=n;i++)
       for(j=1;j<=m;j++)
         scanf("%d",&b[i][j]);
    printf("\n  => Ma tran a la:\n\t");
    for(i=1;i<=m;i++)
      {
        for(j=1;j<=n;j++)
          printf("%5d",a[i][j]);
        printf("\n\t");
      }
    printf("\r  => Ma tran b la:\n\t");
    for(i=1;i<=n;i++)
      {
        for(j=1;j<=m;j++)
          printf("%5d",b[i][j]);
        printf("\n\t");
      }
    printf("\r  Ma tran c la tich cua 2 ma tran a va b.\n");
    for(i=1;i<=m;i++)
      for(j=1;j<=m;j++)
```

```
            {
              c[i][j]=0;
              for(k=1;k<=n;k++)
                c[i][j]+=a[i][k]*b[k][j];
            }
        printf("  => Ma tran c la:\n\t");
        for(i=1;i<=m;i++)
          {
            for(j=1;j<=m;j++)
              printf("%6d",c[i][j]);
            printf("\n\t");
          }
        getch();
        return main();
    }
```

* *Giải thích:*

a) Nhập 2 số m và n sau đó nhập vào từng phần tử của ma trận a và ma trận b. Cả hai ma trận đều có cùng số phần tử là m.n nhưng do ma trận a có m hàng n cột và ma trận b có n hàng m cột nên có sự khác nhau về điều kiện của vòng lặp for. Khi lặp để nhập giá trị cho ma trận a thì i chạy từ 1 đến m, j chạy từ 1 đến n còn khi lặp để nhập giá trị cho ma trận b thì i chạy từ 1 đến n, j chạy từ 1 đến m.

b) Xuất ra màn hình ma trận a: dùng 2 vòng for lồng nhau, vòng ngoài cùng thì biến i chạy từ 1 tới m và vòng for bên trong thì biến j chạy từ 1 đến n. Khi chạy hết tất cả các biến j của vòng for bên trong (tương ứng với hết một giá trị của biến i) thì gọi lệnh xuống dòng để tạo ra một khối có dạng ma trận. Tương tự đối với việc xuất ra màn hình ma trận b, chỉ khác là đảo vị trí m và n cho nhau.

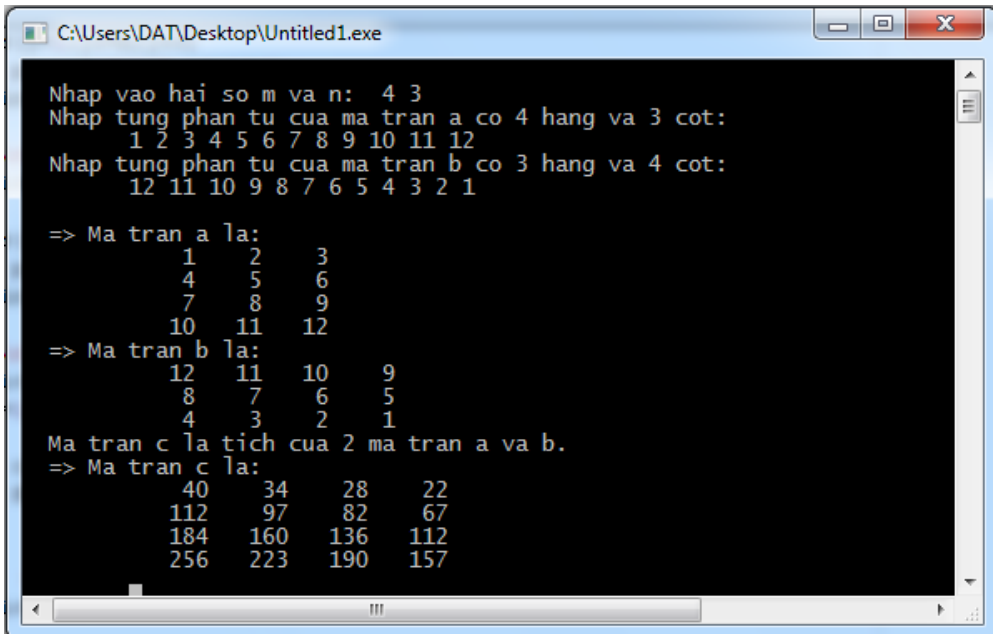
c) Tính ma trận c là tích của 2 ma trận a và b với điều kiện là số hàng của ma trận a phải bằng số cột của ma trận b (điều kiện này đề bài đã cho sẵn). Kết quả là một ma trận c có kích cỡ m.m (vì a có cỡ m.n và b có cỡ n.m). Chúng ta phải xác định được giá trị của phần tử c[i][j] của mảng c. Phần tử này được xác định theo công thức sau:

$$c[i][j] = \sum_{k=1}^{n} a_{ik} b_{kj} \quad (1)$$

*Phần tử c_{ij} bằng tổng của tích các cặp phần tử tương ứng giữa hàng i của ma trận a và cột j của ma trận b (lấy hàng thứ i của ma trận a nhân với cột thứ j của ma trận b). Phần định nghĩa tích hai ma trận thì mọi người xem thêm ở giáo trình "**Toán học cao cấp tập 1** – Đại số và hình học giải tích" của Nguyễn Đình Trí (chủ biên), Tạ Văn Đĩnh, Nguyễn Hồ Quỳnh, trang 97 để hiểu rõ hơn. Lưu ý là không có tính chất giao hoán của tích hai ma trận (a.b ≠ b.a)*

*Cách tính phần tử c[i][j]: cho 2 vòng for lồng nhau để xác định i, j còn muốn tính giá trị của c[i][j] thì bên trong vòng lặp for(j=1;j<=m;j++) ta làm những công việc sau: ban đầu gán c[i][j] = 0 và cho k chạy từ 1 đến n, giá trị của biến c[i][j] được cộng dồn lại và thêm một lượng a[i][k]*b[k][j] để thỏa mãn công thức (1). Sau cùng in ra màn hình ma trận c tương tự như xuất ma trận a và b.*

* *Màn hình kết quả như sau:*



Bài 83: Viết chương trình nhập vào từ bàn phím ma trận vuông a có kích cỡ n.n

- Tính và in ra vết của ma trận a.

- Tính tổng bình phương các phần tử a_{ij} với điều kiện (i – j) chẵn và chỉ ra có bao nhiêu phần tử như vậy trong ma trận. (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a[100][100],i,j,n,Vet=0,Tong=0,SoPhanTu=0;
    printf("\n  Nhap vao kich co cua ma tran vuong:  ");
    scanf("%d",&n);
    printf("  Nhap tung phan tu cua ma tran vuong:\n");
    for(i=1;i<=n;i++)
      {
```

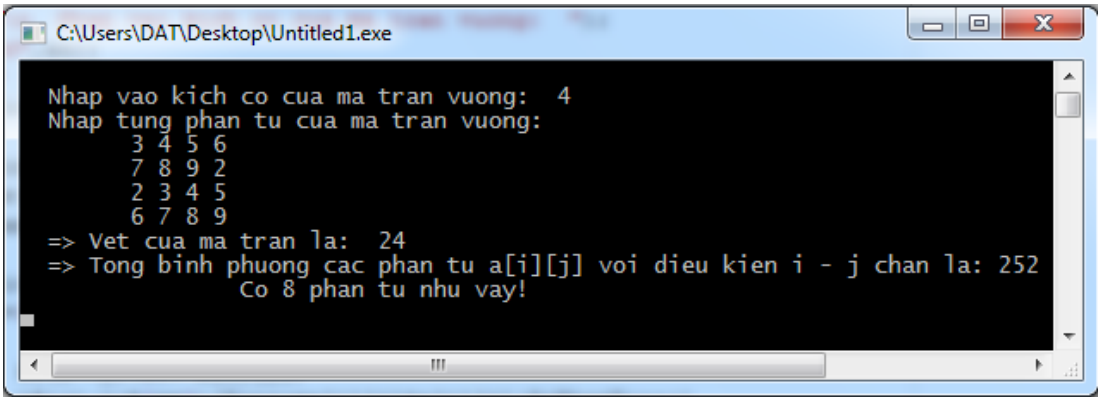
```
            printf("\t");
            for(j=1;j<=n;j++)
            scanf("%d",&a[i][j]);
          }
        for(i=1;i<=n;i++)
          for(j=1;j<=n;j++)
            {
              if(i==j)Vet+=a[i][j];
              if((i-j)%2==0){Tong+=a[i][j]*a[i][j];SoPhanTu++;}
            }
        printf("\r  => Vet cua ma tran la:  %d\n",Vet);
        printf("  => Tong binh phuong cac phan tu a[i][j] voi dieu kien i - j chan la:
    %d\n\t\tCo %d phan tu nhu vay!\n",Tong,SoPhanTu);
        getch();
        return main();
    }
```

* *Giải thích:*

*a) Vết của ma trận trận vuông là tổng của các phần tử trên đường chéo chính. Duyệt qua tất cả các phần tử a[i][j] nếu thấy i = j thì cộng dồn cho biến **Vet** thêm một lượng bằng giá trị của a[i][j] với khởi tạo bằng 0.*

*b) Tính tổng bình phương các phần tử a[i][j]: nếu thấy (i – j) chia hết cho 2 thì thực hiện cộng dồn cho biến **Tong** thêm một lượng bằng bình phương giá trị của a[i][j] với khởi tạo bằng 0 và tăng biến **SoPhanTu** lên 1 đơn vị. Đưa các kết quả tính được ra màn hình.*

* *Màn hình kết quả như sau:*



<u>Bài 84:</u> Viết chương trình nhập vào từ bàn phím ma trận a có kích cỡ m.n Chuyển đổi hàng của ma trận thành cột của ma trận. In ra kết quả trên màn hình ma trận trước và sau khi chuyển đổi? (X)

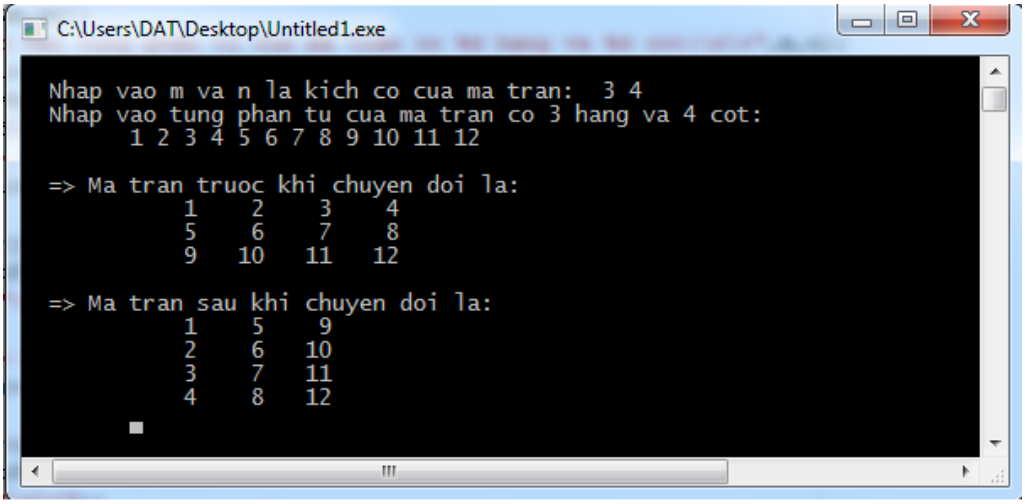
```
#include <stdio.h>
#include <conio.h>
main()
{
    int a[100][100],i,j,m,n;
    printf("\n  Nhap vao m va n la kich co cua ma tran:  ");
    scanf("%d%d",&m,&n);
    printf("  Nhap vao tung phan tu cua ma tran co %d hang va %d
cot:\n\t",m,n);
    for(i=1;i<=m;i++)
      for(j=1;j<=n;j++)
          scanf("%d",&a[i][j]);
    printf("\n  => Ma tran truoc khi chuyen doi la:\n\t");
    for(i=1;i<=m;i++)
      {
        for(j=1;j<=n;j++)
          printf("%5d",a[i][j]);
        printf("\n\t");
      }
    printf("\n  => Ma tran sau khi chuyen doi la:\n\t");
    for(j=1;j<=n;j++)
      {
        for(i=1;i<=m;i++)
          printf("%5d",a[i][j]);
        printf("\n\t");
      }
    getch();
    return main();
}
```

* *Giải thích:*

a) Nhập vào kích cỡ m, n của ma trận từ bàn phím và nhập từng phần tử của ma trận dùng 2 vòng for lồng nhau tương tự như những bài trên. Vòng for bên ngoài dùng biến i chạy từ 1 đến m còn ở vòng for bên trong dùng biến j chạy từ 1 đến n, ta có thể nhập cùng một lúc hết tất cả m.n phần tử của ma trận để lưu vào các địa chỉ &a[i][j] tương ứng.

b) Chuyển đổi hàng của ma trận thành cột của ma trận: bình thường thì vòng for bên ngoài dùng biến i với ý nghĩa là khi vòng for bên trong chạy chỉ liên quan tới biến j còn biến i đã nhận một giá trị xác định, tức là khi thực hiện nhập phần tử của ma trận thì cho hàng cố định còn cột thì chạy từ 1 đến n. Trong trường hợp đổi hàng thành cột thì phải đảo ngược lại, cho cột cố định còn hàng chạy từ 1 đến m. Ta lấy cột thứ j của ma trận ban đầu làm hàng thứ i của ma trận đã chuyển đổi và dòng đầu tiên của ma trận đã chuyển đổi gồm các phần tử: a_{11}, a_{21}, a_{31},.... Do đó ta cho vòng lặp for với biến j chạy từ 1 đến n ra bên ngoài và vòng for bên trong là biến i chạy từ 1 đến m. Lặp đến đâu thì xuất ra màn hình giá trị a[i][j] đến đó và sau khi hết một hàng thì dùng lệnh xuống dòng để cho các phần tử hiện ra có dạng khối.

* *Màn hình kết quả như sau:*



* *Chú ý:*

Với thuật toán như trên thì ta chỉ có thể thay đổi được thứ tự xuất hiện các phần tử của ma trận chứ thực tế thì giá trị của phần tử nằm ở hàng i và cột j của ma trận vẫn không thay đổi và ma trận vẫn có 3 hàng 4 cột, tức là bản chất của ma trận vẫn như thế. Ví dụ, sau khi chuyển đổi hàng thành cột thì phần tử a_{12} của ma trận vẫn là 2 chứ không phải 5

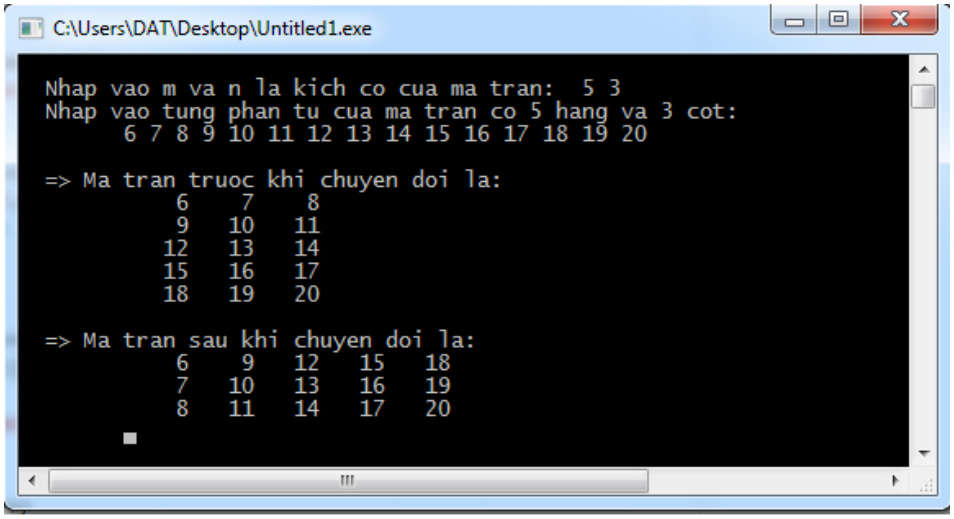
Muốn tạo một ma trận khác sao cho cột của ma trận khác này chính là hàng của ma trận ban đầu thì phải khai báo thêm ma trận b[100][100] và dùng thuật toán gán phần tử b[j][i] = a[i][j] và cho in ra màn hình các phần tử của ma trận b. Làm như vậy thì bản chất của ma trận đã chuyển đổi mới khác so với ma trận ban đầu (ma trận b là ma trận chuyển vị của ma trận a).

* *Thuật toán cho trường hợp này như sau:*

```
#include <stdio.h>
#include <conio.h>
```

```
main()
{
    int a[100][100],b[100][100],i,j,m,n;
    printf("\n  Nhap vao m va n la kich co cua ma tran:  ");
    scanf("%d%d",&m,&n);
    printf("  Nhap vao tung phan tu cua ma tran co %d hang va %d
cot:\n\t",m,n);
    for(i=1;i<=m;i++)
      for(j=1;j<=n;j++)
          scanf("%d",&a[i][j]);
    printf("\n  => Ma tran truoc khi chuyen doi la:\n\t");
    for(i=1;i<=m;i++)
      {
        for(j=1;j<=n;j++)
          printf("%5d",a[i][j]);
        printf("\n\t");
      }
    for(j=1;j<=n;j++)
      {
        for(i=1;i<=m;i++)
          b[j][i]=a[i][j];
      }
    printf("\n  => Ma tran sau khi chuyen doi la:\n\t");
    for(i=1;i<=n;i++)
      {
        for(j=1;j<=m;j++)
          printf("%5d",b[i][j]);
        printf("\n\t");
      }
    getch();
    return main();
}
```

* *Màn hình kết quả như sau:*



<u>Bài 85:</u> Viết chương trình:

- Lập hàm nhập một ma trận có kích cỡ m hàng n cột.

- Lập hàm xuất một ma trận có kích cỡ u hàng v cột.

- Gọi hai hàm vừa lập vào chương trình chính để nhập và xuất hai ma trận: ma trận a có 3 hàng 2 cột và ma trận b có 2 hàng 4 cột. (X)

```
#include <stdio.h>
#include <conio.h>
void NhapMaTran(int a[][100],int m,int n)
{
   int i,j;
   printf("\n  Nhap cac phan tu cua ma tran co %d hang %d cot:\n\t",m,n);
   for(i=1;i<=m;i++)
     for(j=1;j<=n;j++)
       scanf("%d",&a[i][j]);
}
void XuatMaTran(int a[][100],int u,int v)
{
   int i,j;
   printf("  => Ma tran vua nhap la:\n\t");
   for(i=1;i<=u;i++)
```

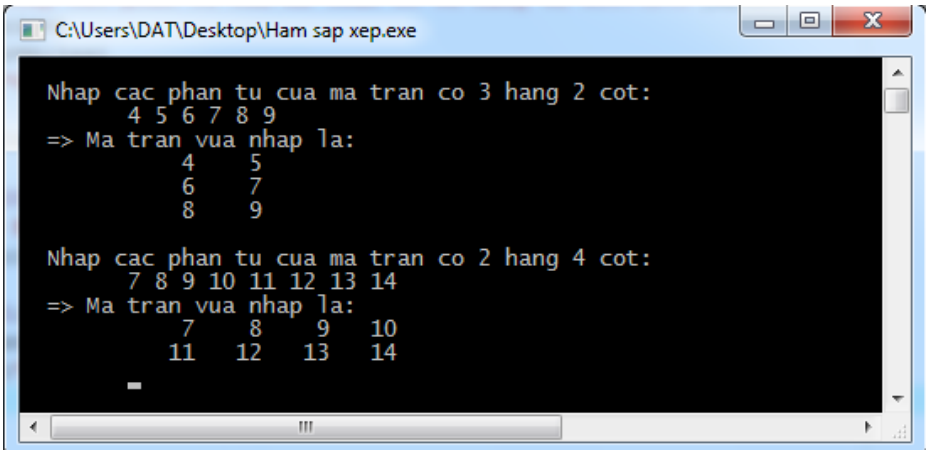
```
        {
          for(j=1;j<=v;j++)
            printf("%5d",a[i][j]);
          printf("\n\t");
        }
    }
    main()
    {
      int a[100][100],b[100][100];
      NhapMaTran(a,3,2);
      XuatMaTran(a,3,2);
      NhapMaTran(b,2,4);
      XuatMaTran(b,2,4);
      getch();
      return main();
    }
```

* *Giải thích:*

*- Hàm nhập và xuất các giá trị của ma trận ra màn hình thì không có kiểu trả về (kiểu **void**). Truyền biến vào cho hàm **NhapMaTran()** và hàm **XuatMaTran()** từ chương trình chính, còn các biến m, n, u, v ở chương trình con chỉ là những biến cục bộ, ta có thể chọn các biến này giống nhau ở hai hàm cho đỡ nhầm lẫn. Trong các chương trình con, phải truyền sẵn một mảng 2 chiều để khi nhập thì các phần tử mới có địa chỉ để lưu cũng như khi xuất mới tìm thấy địa chỉ để lấy ra giá trị. Cách khác ta cũng có thể sử dụng biến con trỏ và cấp phát động bộ nhớ để nhập và xuất các phần tử của chương trình con.*

- Trong chương trình chính thì thực hiện nhập và xuất xong ma trận a có 3 hàng 2 cột rồi tiếp đó nhập và xuất ma trận b có 2 hàng 4 cột.

* *Màn hình kết quả như sau:*



<u>Bài 86:</u> Viết chương trình:

- Nhập ma trận a có kích cỡ m hàng n cột, nhập ma trận b có kích cỡ n hàng m cột.

- Lập hàm tính tích của hai ma trận.

- Gọi hàm vừa lập vào chương trình chính tính tích của hai ma trận a, b. In ra ma trận tích vừa tìm được? (X)

```
#include <stdio.h>
#include <conio.h>
void Tich2MaTran(int a[][100],int b[][100],int c[][100],int m,int n);
main()
{
   int a[100][100],b[100][100],c[100][100],i,j,m,n;
   printf("\n  Nhap vao hai so m va n:  ");
   scanf("%d%d",&m,&n);
   printf("  Nhap tung phan tu cua ma tran a co %d hang va %d cot:\n\t",m,n);
   for(i=1;i<=m;i++)
     for(j=1;j<=n;j++)
       scanf("%d",&a[i][j]);
   printf("  Nhap tung phan tu cua ma tran b co %d hang va %d cot:\n\t",n,m);
   for(i=1;i<=n;i++)
      for(j=1;j<=m;j++)
        scanf("%d",&b[i][j]);
   Tich2MaTran(a,b,c,m,n);
```

```
        printf("\r  Ma tran c la tich cua 2 ma tran a va b.\n");
        printf("  => Ma tran c la:\n\t");
        for(i=1;i<=m;i++)
          {
            for(j=1;j<=m;j++)
              printf("%6d",c[i][j]);
            printf("\n\t");
          }
        getch();
        return main();
    }
    void Tich2MaTran(int a[][100],int b[][100],int c[][100],int m,int n)
    {
        int i,j,k;
        for(i=1;i<=m;i++)
          for(j=1;j<=m;j++)
            {
              c[i][j]=0;
              for(k=1;k<=n;k++)
                c[i][j]+=a[i][k]*b[k][j];
            }
    }
```

* *Giải thích:*

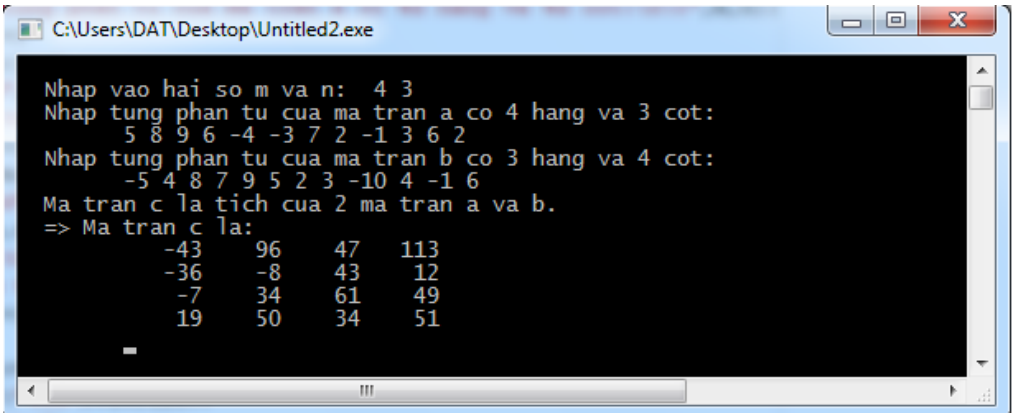
a) Bài này, sử dụng cách khai báo prototype (nguyên mẫu) trước rồi mới viết thân hàm ở sau hàm main(). Việc nhập và lưu địa chỉ của phần tử vào cho 2 ma trận a và b được thực hiện trong chương trình chính. Lưu ý là nhập vào các phần tử thì có thể để dạng khối hoặc dãy số đều được, chương trình sẽ tự động sắp xếp lại theo hàng và cột khi tính tích. Công việc tính tính của hai ma trận thì gọi hàm ***Tich2MaTran()*** *vào chương trình chính.*

b) Hàm tính tích của hai ma trận: Các tham số truyền vào cho hàm gồm ma trận a, ma trận b, ma trận c và các số m, n trong đó các ma trận đều là tham biến. Công việc của hàm là tính ma trận c hay nói cách khác là tính các phần tử c_{ij} với $i, j = \overline{1,m}$ (vì ma trận a có m dòng và ma trận b có m cột). Các phần tử c_{ij} phải thỏa mãn công thức:

$$c[i][j] = \sum_{k=1}^{n} a_{ik} b_{kj}$$ *(lấy tổng hàng i của a nhân với cột j của b)*

Sau đó, xuất ra màn hình ma trận c từ chương trình chính.

* *Màn hình kết quả như sau:*



<u>Bài 87:</u> Viết chương trình:

- Lập hàm nhập ma trận vuông có kích cỡ n.n

- Lập hàm tính tích của hai ma trận.

- Từ chương trình chính, gọi các hàm ở trên vào thực hiện: Nhập vào 3 ma trận vuông a, b, c. Tính tích của 3 ma trận, in ra màn hình ma trận tích vừa tìm được? (X)

```
#include <stdio.h>
#include <conio.h>
void NhapMaTran(int a[][100],int n);
void Tich2MaTran(int a[][100],int b[][100],int Tich[][100],int n);
main()
{
   int a[100][100],b[100][100],c[100][100],TrungGian[100][100],
d[100][100],n,i,j;
   printf("\n  Nhap vao kich co cua ma tran vuong:  ");
   scanf("%d",&n);
   printf("  Nhap vao ma tran a:\n\t");
   NhapMaTran(a,n);
   printf("  Nhap vao ma tran b:\n\t");
   NhapMaTran(b,n);
   printf("  Nhap vao ma tran c:\n\t");
```

```
   NhapMaTran(c,n);
   printf("  Ma tran d la tich cua 3 ma tran tren:\n");
   Tich2MaTran(a,b,TrungGian,n);
   Tich2MaTran(TrungGian,c,d,n);
   printf("  => Ma tran d la:\n\t");
   for(i=1;i<=n;i++)
     {
       for(j=1;j<=n;j++)
         printf("%7d",d[i][j]);
       printf("\n\t");
     }
   getch();
   return main();
}
void NhapMaTran(int a[][100],int n)
{
   int i,j;
   for(i=1;i<=n;i++)
     {
       for(j=1;j<=n;j++)
         scanf("%d",&a[i][j]);
       printf("\t");
     }
   printf("\r");
}
void Tich2MaTran(int a[][100],int b[][100],int Tich[][100],int n)
{
    int i,j,k;
    for(i=1;i<=n;i++)
      for(j=1;j<=n;j++)
        {
```

```
                Tich[i][j]=0;
                for(k=1;k<=n;k++)
                  Tich[i][j]+=a[i][k]*b[k][j];
              }
        }
```

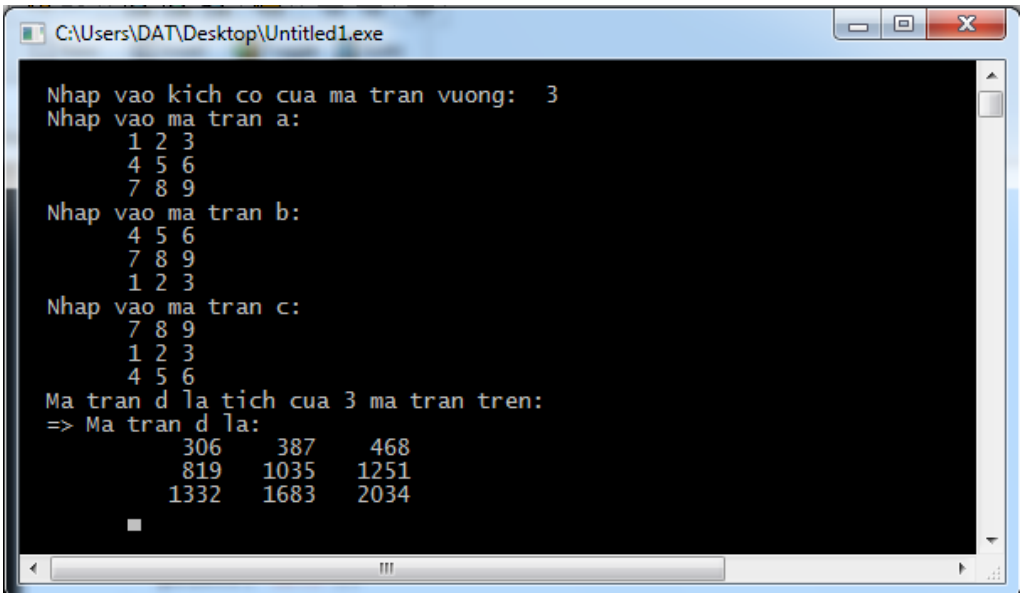
* *Giải thích:*

a) Hàm nhập ma trận vuông cỡ n: chỉ có 2 tham số gồm tham biến là ma trận tương ứng và tham trị là kích cỡ n của ma trận. Cứ nhập xong một dòng thì thực hiện lệnh xuống dòng và lùi vào 1 tab để ma trận nhập vào có dạng khối (có thể để ma trận nhập vào ở dạng dãy số).

*b) Tính tích của 3 ma trận a, b, c và trả về kết quả cho ma trận d: Đầu tiên ta tính tích của 2 ma trận a và b được ma trận **TrungGian[ ][ ]**, sau đó lấy tích của ma trận TrungGian và ma trận c ta được ma trận d. Sử dụng hàm tính tích của hai ma trận để thực hiện liên tiếp 2 công việc này. Có thể lấy tích của ma trận b và c được ma trận TrungGian sau đó lấy tích của ma trận a và ma trận TrungGian được kết quả là ma trận d.*

Tính chất của ma trận: a.b.c = (a.b).c = a.(b.c)

* *Màn hình kết quả như sau:*



<u>Bài 88:</u> Lập hàm với 2 đối số là n và x để tính giá trị của đa thức Legendre theo phương pháp đệ quy hàm. Biết rằng:

$$P_0(x)=1; \qquad P_1(x)=x; \qquad P_n(x)=\frac{2n-1}{n}.x.P_{n-1}(x) - \frac{n-1}{n}.P_{n-2}(x) \qquad \textbf{(X)}$$

```
#include <stdio.h>
#include <conio.h>
```

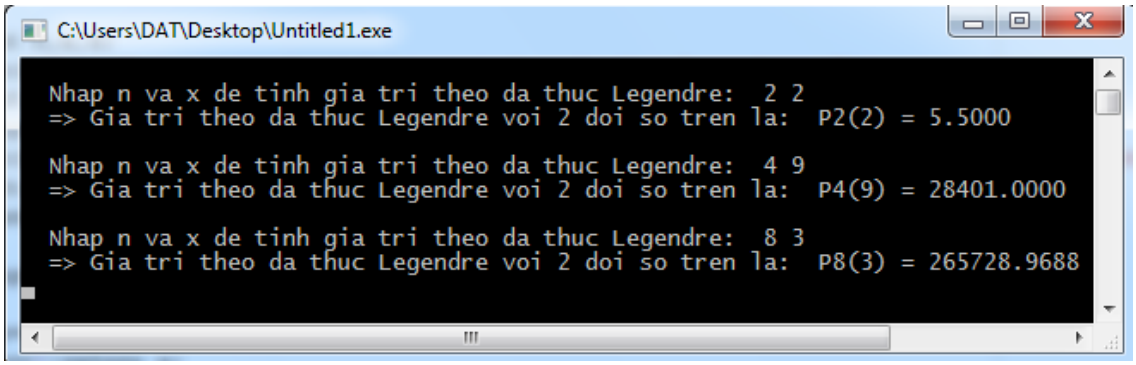
```
float Legendre (float n,float x);
main()
{
   float n,x,k;
   printf("\n  Nhap n va x de tinh gia tri theo da thuc Legendre:  ");
   scanf("%f%f",&n,&x);
   k=Legendre(n,x);
   printf("  => Gia tri theo da thuc Legendre voi 2 doi so tren la:
P%0.0f(%0.0f) = %0.4f\n",n,x,k);
   getch();
   return main();
}
float Legendre (float n,float x)
{
   if(n==0)return 1;
   if(n==1)return x;
   if(n>1) return (2*n-1)/n*x*Legendre(n-1,x)-(n-1)/n*Legendre(n-2,x);
}
```

* *Giải thích:*

*- Hàm **Legendre(n, x)** là hàm tính giá trị theo đa thức Legendre với 2 đối số n và x. Vì có hai trường hợp suy biến là* $P_0(x)=1$ *và* $P_1(x)=x$ *nên ta sử dụng phương pháp đệ quy hàm. Hàm Legendre sẽ gọi lại chính nó khi còn thỏa mãn điều kiện n > 1, và sau khi kết thúc sẽ cho kết quả của* $P_n(x)$ *(n và x là 2 số mà ta nhập vào từ bàn phím). Các tham số truyền vào cho hàm nên để ở dạng số thực và kết quả trả về cho hàm thì phải ở dạng số thực.*

- Gán giá trị của biến k bằng giá trị của hàm Legendre() sau đó đưa k ra màn hình vì $k=P_n(x)$*. Dấu "%0.0f" là định dạng số thực nhưng không lấy chữ số nào sau dấu phẩy ngăn cách phần nguyên và phần thập phân.*

* *Màn hình kết quả như sau:*



CHƯƠNG V. CON TRỎ

Bài 89: Sử dụng con trỏ để giải bài toán sắp xếp một dãy số nguyên theo chiều tăng (hoặc giảm) dần?

* *Đối với trường hợp sắp xếp tăng dần:*

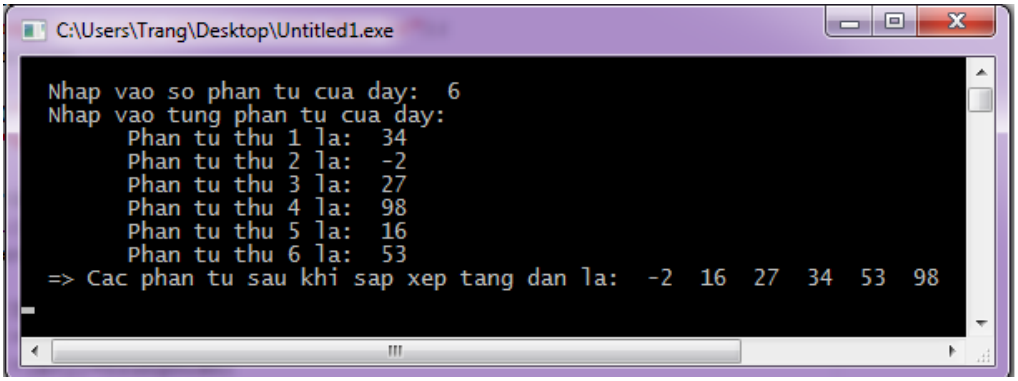
```
#include <stdio.h>
#include <conio.h>
main()
{
    int a[100],i,j,n,TrungGian;
    printf("\n  Nhap vao so phan tu cua day:  ");
    scanf("%d",&n);
    printf("  Nhap vao tung phan tu cua day:\n");
    for(i=1;i<=n;i++)
      {
       printf("\tPhan tu thu %d la:  ",i);
       scanf("%d",a+i);
      }
    for(i=1;i<n;i++)
      for(j=i+1;j<=n;j++)
        if(*(a+i)>*(a+j))
          {
            TrungGian=*(a+i);
            *(a+i)=*(a+j);
```

```
            *(a+j)=TrungGian;
        }
    printf("  => Cac phan tu sau khi sap xep tang dan la:  ");
    for(i=1;i<=n;i++)
        printf("%d  ",*(a+i));
    printf("\n");
    getch();
    return main();
}
```

* *Giải thích:*

*- Tên của mảng là một con trỏ trỏ tới vị trí của phần tử a[1]. (a+i) chính là địa chỉ của phần tử thứ i (&a[i]) còn *(a+i) chính là giá trị của phần tử thứ i (a[i]). Đối với trường hợp sắp xếp giảm dần thì đảo ngược điều kiện lại. Mọi người tự chỉnh và chạy thử nhé!*

* *Màn hình kết quả như sau*:



Bài 90: Nhập vào dãy n số nguyên (n ≥ 10), có bao nhiêu số nguyên tố và tính tổng các số nguyên tố đó bằng cách sử dụng con trỏ?

```
#include <stdio.h>
#include <conio.h>
int NguyenTo(int *p);
main()
{
    int a[100],n,i,j,Tong=0,Dem=0;
    printf("\n  Nhap vao so phan tu cua day:  ");
    scanf("%d",&n);
```

```
    if(n>=10)
      {
        printf("  Nhap vao cac phan tu cua day:\n");
        for(i=1;i<=n;i++)
          {
            printf("\tPhan tu thu %d la:  ",i);
            scanf("%d",a+i);
          }
        for(i=1;i<=n;i++)
        if(NguyenTo(a+i)==1)
          {Dem++;Tong=Tong+*(a+i);}
        printf("  => Co %d so nguyen to trong day vua nhap va tong cua chung
la %d\n",Dem,Tong);
        getch();
      }
    return main();
}
int NguyenTo(int *p)
{
    int i,d=0;
    if(*p>=2)
      {
       for(i=2;*p%i!=0;i++);
       if(i==*p)d=1;
      }
    return (d);
}
```

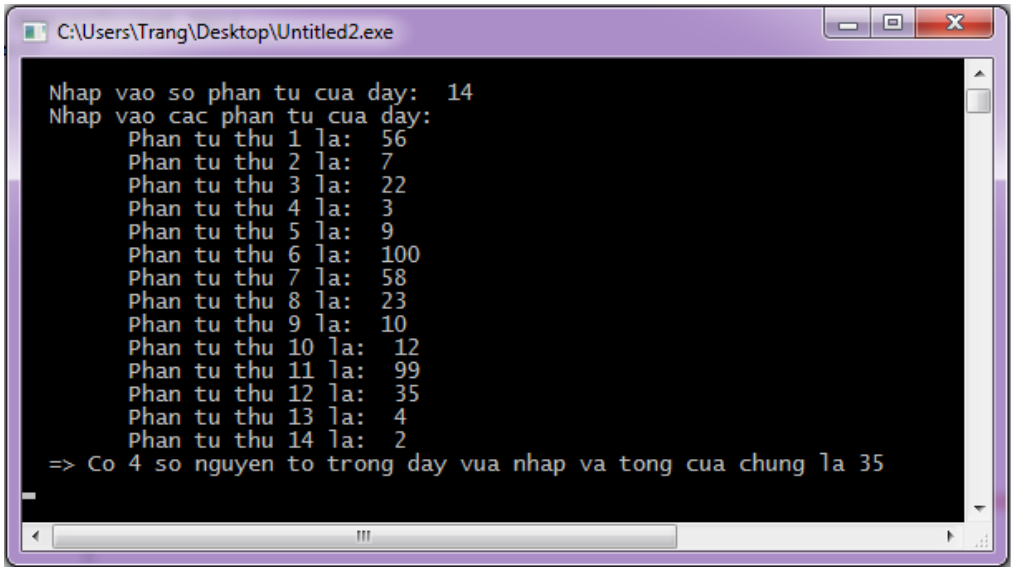
* *Giải thích:*

a) Nếu số n nhập vào mà ≥ 10 thì mới thực hiện công việc kiểm tra số nguyên tố cũng như đếm và tính tổng các số nguyên tố đó. Nếu số n nhập vào mà < 10 thì quay lại nhập số khác cho tới khi ≥ 10 mới thôi, dùng lệnh ***return main()****.*

b) Sau khi kiểm tra một phần tử mà thỏa mãn là số nguyên tố thì tăng biến ***Dem*** *lên 1 đơn vị và giá trị của biến* ***Tong*** *được cộng dồn.* ***Hàm NguyenTo(a+i)*** *là hàm kiểm tra xem giá trị* ****(a+i)***

có phải là số nguyên tố hay không, nếu là số nguyên tố thì trả về giá trị 1, ngược lại trả về giá trị 0. Bước kiểm tra xem phần tử nào đó có phải là số nguyên tố hay không, ta dùng phương pháp truyền con trỏ cho hàm. Truyền vào một địa chỉ cho hàm và thao tác trên giá trị của địa chỉ đó.

* *Màn hình kết quả như sau:*



<u>Bài 91:</u> Sử dụng con trỏ nhập vào mảng 2 chiều có kích thước m.n (m, n < 100)

a) Tìm phần tử chẵn lớn nhất trong mảng và đưa ra vị trí của nó (nếu có nhiều phần tử cùng lớn nhất thì chỉ cần đưa ra 1 vị trí)?

b) Nhập k (với 1 ≤ k ≤ m), sắp xếp dòng thứ k theo chiều tăng dần?

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i,j,k,l=1,m,n,TrungGian,*p,Dem=0,*Max;
    p=&i;
    printf("\n  Nhap vao m dong va n cot:  ");
    batdau:scanf("%d%d",&m,&n);
    if(m<100&&n<100)
      {
        printf("  Nhap vao cac phan tu cua mang:\n");
        for(i=1;i<=m;i++)
```

```
            for(j=1;j<=n;j++)
              {
                printf("\tPhan tu tren dong %d cot %d la:  ",i,j);
                 scanf("%d",p+l);
                 l++;
              }
        }
      else {printf("  Nhap lai m, n:  ");goto batdau;}
      for(i=1;i<=m*n;i++)
        if(*(p+i)%2==0){Dem++;Max=p+i;}
      if(Dem==0)printf("  => Khong co so chan nao!\n");
      else
        {
          for(i=1;i<=m*n;i++)
            if((*(p+i)%2==0)&&(*(p+i)>*Max))Max=p+i;
          printf("  => So chan lon nhat la %d va vi tri cua no tren RAM la
%p\n",*Max,Max);
        }
      printf("  Nhap vao dong thu k (1<=k<=m):  ");
      nhap:scanf("%d",&k);
      if(k>=1&&k<=m)
        {
          printf("  => Sap xep tang dan tren dong thu %d la:  ",k);
            for(i=n*k-n+1;i<n*k;i++)
              for(j=i+1;j<=n*k;j++)
                if(*(p+i)>*(p+j))
                  {
                    TrungGian=*(p+i);
                    *(p+i)=*(p+j);
                    *(p+j)=TrungGian;
                  }
          for(i=n*k-n+1;i<=n*k;i++)
```

```
            printf("%d  ",*(p+i));
        }
    else
        {
            printf("   Nhap lai k:  ");
            goto nhap;
        }
    printf("\n");
    getch();
    return main();
}
```

* *Giải thích:*

*a) Khai báo hai con trỏ là ***p** dùng để truy cập tới địa chỉ của phần tử thứ i và ***Max** truy cập tới địa chỉ của phần tử lớn nhất. Gán giá trị ban đầu của con trỏ p bằng địa chỉ của biến i.*

b) Khi nhập vào các phần tử của mảng 2 chiều có m dòng và n cột thì bản chất chính là nhập vào các phần tử của mảng 1 chiều có m.n phần tử (vì các phần tử của mảng 2 chiều được ghi lần lượt vào thanh RAM và mảng 2 chiều chính là mảng một chiều gồm m phần tử trong đó mỗi phần tử chính là một mảng một chiều gồm n phần tử). Vì vậy, để ghi vào ô nhớ cho phần tử a[i][j] thì ta dùng địa chỉ p+l trong đó l chạy từ 1 tới m.n

RAM là một loại bộ nhớ trong (bộ nhớ trong gồm có RAM và ROM). Khi chương trình làm việc thì dữ liệu được ghi trực tiếp vào thanh RAM và được xử lý dưới sự điều khiển của bộ xử lý trung tâm – CPU, tốc độ xử lý rất trên RAM là rất nhanh. Khi nguồn điện không được duy trì thì dữ liệu trên RAM sẽ biến mất. Vậy có thể cho rằng, RAM gần như một bộ nhớ tạm có chức năng ghi nhớ dữ liệu khi chương trình làm việc.

Tại sao lại cần địa chỉ của dữ liệu khi chạy trình dịch C? Vì các máy tính hiện nay đều làm việc theo nguyên lý Von Neumann, đó là: *"hoạt động theo chương trình và truy cập theo địa chỉ!"*

*c) Tìm phần tử chẵn lớn nhất: Trước tiên, tìm xem có số chẵn nào trong mảng hay không, nếu giá trị mà con trỏ (p+i) trỏ tới là số chẵn thì gán con trỏ **Max** bằng địa chỉ của phần tử đó và tăng biến **Dem** lên một đơn vị với giá trị khởi tạo bằng 0. Sau đó, nếu thấy Dem = 0 thì đưa ra màn hình dòng chữ "Không có số chẵn nào!", trường hợp Dem ≠ 0 thì thực hiện tìm số chẵn lớn nhất trong các số chẵn mà các con trỏ tương ứng trỏ tới và đưa ra màn hình giá trị mà con trỏ Max trỏ tới cũng như địa chỉ mà con trỏ Max trỏ tới.*

d) Thực hiện sắp xếp tăng dần trên dòng thứ k (với k nhập vào phải thỏa mãn $1 \le k \le m$). Phần tử đầu tiên trên dòng thứ k sẽ có thứ tự là $n.k - n + 1$ và phần tử cuối cùng trên dòng thứ k sẽ có thứ tự là n.k

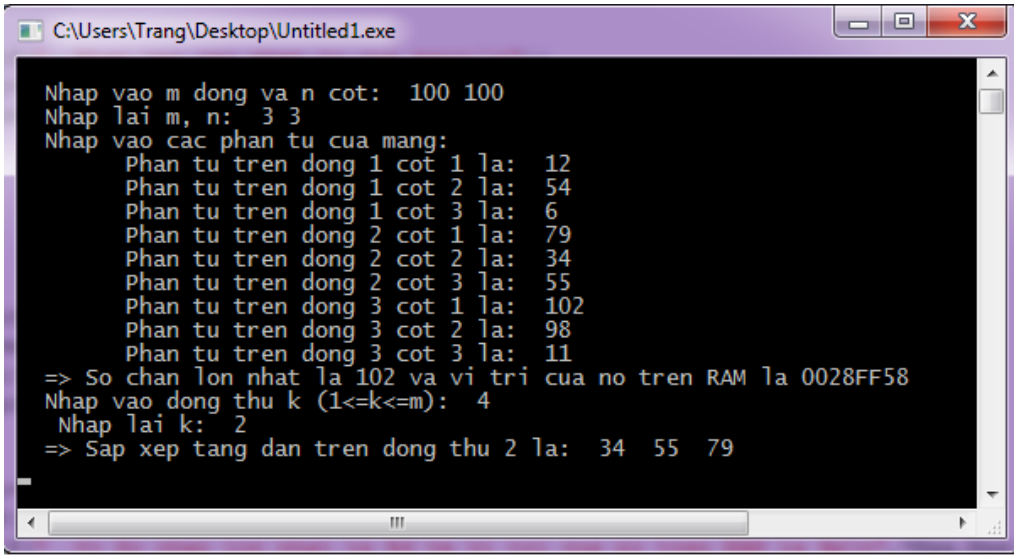
Ví dụ: nhập vào 5 dòng và 4 cột thì phần tử đầu tiên trên dòng thứ 3 có thứ tự là

$4.3 - 4 + 1 = 9$ và phần tử cuối cùng trên dòng thứ 3 có thứ tự là $4.3 = 12$

*Sử dụng biến i chạy từ phần tử đầu tiên đến phần tử có thứ tự n.k – 1 và biến j chạy từ i+1 đến phần tử cuối cùng trên dòng thứ k, nếu *(p+i) > *(p+j) thì thực hiện đảo vị trí. Cuối cùng in ra các giá trị đã sắp xếp tăng dần mà các con trỏ tương ứng trỏ tới.*

*e) Các giá trị của m, n, k nhập vào mà không thỏa mãn điều kiện của đề bài thì dùng lệnh nhảy vô điều kiện **goto** để di chuyển đến vị trí cần thiết và nhập lại giá trị của chúng.*

* *Màn hình kết quả như sau:*



<u>Bài 92:</u> Viết chương trình nhập vào số nguyên n, cấp phát bộ nhớ cho mảng a có n phần tử. Thực hiện:

- Nhập phần tử của mảng a?

- In ra địa chỉ của các phần tử trong mảng?

- Sắp xếp mảng a theo chiều giảm dần, in ra các phần tử sau khi đã được sắp xếp? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
    int *a,n,i,j,TrungGian;
    printf("\n  Nhap vao so phan tu cua mang a:  ");
    scanf("%d",&n);
    a=(int*)malloc(n*sizeof(int));
    printf("  Nhap vao cac phan tu cua mang a:\n");
```

```
    for(i=1;i<=n;i++)
      {
        printf("\tPhan tu thu %d la:  ",i);
        scanf("%d",a+i);
      }
    printf("  => Dia chi cua cac phan tu la:\n");
      for(i=1;i<=n;i++)
      printf("\tPhan tu thu %d:  %p\n",i,a+i);
    printf("  => Cac phan tu sau khi sap xep giam dan la:  ");
    for(i=1;i<n;i++)
      for(j=i+1;j<=n;j++)
        if(*(a+i)<*(a+j))
          {
            TrungGian=*(a+i);
            *(a+i)=*(a+j);
            *(a+j)=TrungGian;
          }
    for(i=1;i<=n;i++)printf("%d  ",*(a+i));
    printf("\n");
    getch();
    return main();
}
```

* *Giải thích:*

a) Khai báo một con trỏ p, lệnh sizeof(int) có tác dụng kiểm tra dung lượng bộ nhớ mà kiểu int chiếm (trong trình dịch Boland C chúng ta thường dùng thì kiểu int chiếm 4 byte). Hàm malloc là hàm cấp phát bộ nhớ cho dãy có n số nguyên, vì mỗi số nguyên chiếm 4 byte nên dãy n số nguyên sẽ chiếm 4n byte.

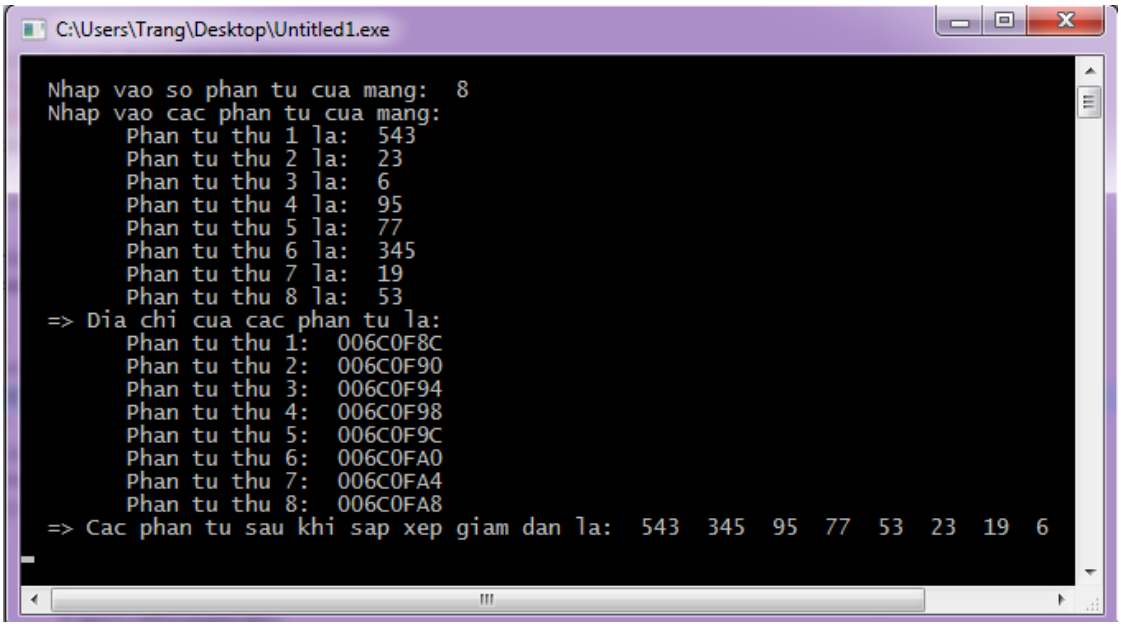
b) Nhập vào các phần tử của mảng: Mỗi phần tử sau khi nhập được lưu vào địa chỉ p+i (địa chỉ p+i tương đương với &a[i]).

c) In ra địa chỉ của các phần tử tương ứng chính là in ra các con trỏ p+ i (i chạy từ 1 đến n), thực chất là in ra chỉ số trên RAM mà các giá trị được lưu vào đó. Các địa chỉ của dữ liệu chính là các giá trị của con trỏ.

d) Sắp xếp các giá trị của mảng theo chiều giảm dần: So sánh các giá trị mà các con trỏ trỏ tới, nếu giá trị thứ i mà nhỏ hơn giá trị thứ i + 1 thì thực hiện đổi vị trí của chúng cho nhau. Giá

trị đầu tiên trong dãy đã sắp xếp là giá trị nhỏ nhất trong tất cả các giá trị, giá trị thứ hai trong dãy đã sắp xếp là giá trị nhỏ nhất trong các giá trị từ giá trị thứ hai đến giá trị thứ n,...,giá trị thứ n – 1 là giá trị nhỏ nhất trong hai giá trị thứ n – 1 và thứ n.

* *Màn hình kết quả như sau:*



<u>Bài 93:</u> Viết chương trình nhập vào số nguyên n, cấp phát bộ nhớ cho mảng c có n phần tử. Thực hiện:

- Nhập phần tử của mảng c.

- Tìm phần tử lớn nhất trong mảng, in ra giá trị và vị trí tương ứng của chúng trong mảng. (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
    int *c,n,i,*Max;
    printf("\n  Nhap vao so phan tu cua mang c:  ");
    scanf("%d",&n);
    c=(int*)malloc(n*sizeof(int));
    printf("  Nhap vao cac phan tu cua mang c:\n");
    for(i=1;i<=n;i++)
      {
```

```
        printf("\tPhan tu thu %d la:  ",i);
        scanf("%d",c+i);
      }
    Max=c+1;
    for(i=1;i<=n;i++)
      if(*Max<*(c+i))Max=c+i;
    printf("  => Phan tu lon nhat trong mang la:  %d\n",*Max);
    printf("  => Dia chi cua chung la:\n");
    for(i=1;i<=n;i++)
      if(*Max==*(c+i))printf("\t%p\n",(c+i));
    getch();
    return main();
}
```

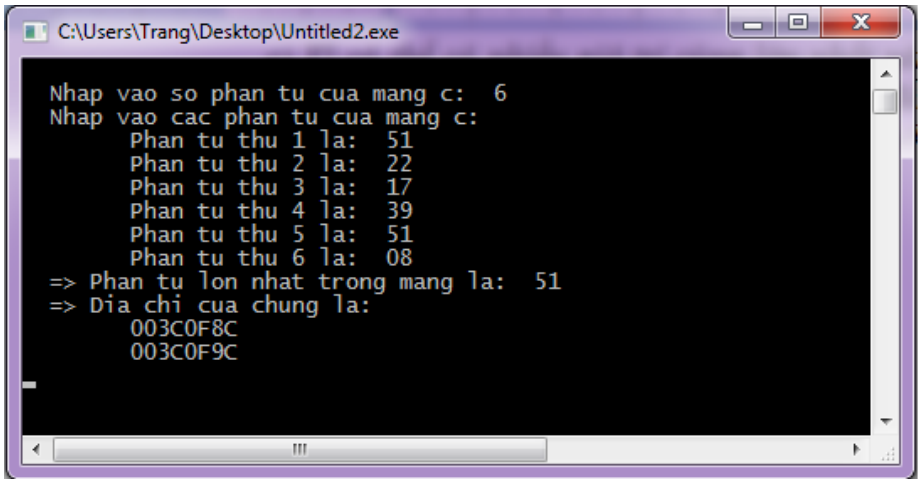
* *Giải thích:*

a) Khai báo các con trỏ cần thiết: con trỏ c để trỏ tới các giá trị của mảng, con trỏ Max để trỏ tới giá trị lớn nhất trong mảng. Dùng hàm malloc để cấp phát bộ nhớ cho các phần tử của mảng.

*b) Tìm phần tử lớn nhất trong mảng: Gán con trỏ Max bằng địa chỉ của phần tử đầu tiên, so sánh giá trị mà Max trỏ tới với các giá trị còn lại, nếu *Max nhỏ hơn *(c+i) thì gán con trỏ Max bằng con trỏ (c+i). Sau đó in ra giá trị mà Max trỏ tới.*

*c) Vì có thể có nhiều giá trị cùng lớn nhất nên phải in ra tất cả địa chỉ của những giá trị lớn nhất đó. Vòng lặp for chạy từ phần tử đầu tiên đến phần tử cuối cùng, nếu giá trị *(c+i) mà bằng giá trị con trỏ Max trỏ tới (*Max) thì in ra địa chỉ (c+i)*

* *Màn hình kết quả như sau:*



<u>Bài 94:</u> Dãy Fibonacci là dãy vô hạn các số tự nhiên bắt đầu bằng hai phần tử 0 và 1 được định nghĩa như sau:

$$F[n] = \begin{cases} 0 & khi\ n = 0 \\ 1 & khi\ n = 1 \\ F[n-1] + F[n-2] & khi\ n > 1 \end{cases}$$

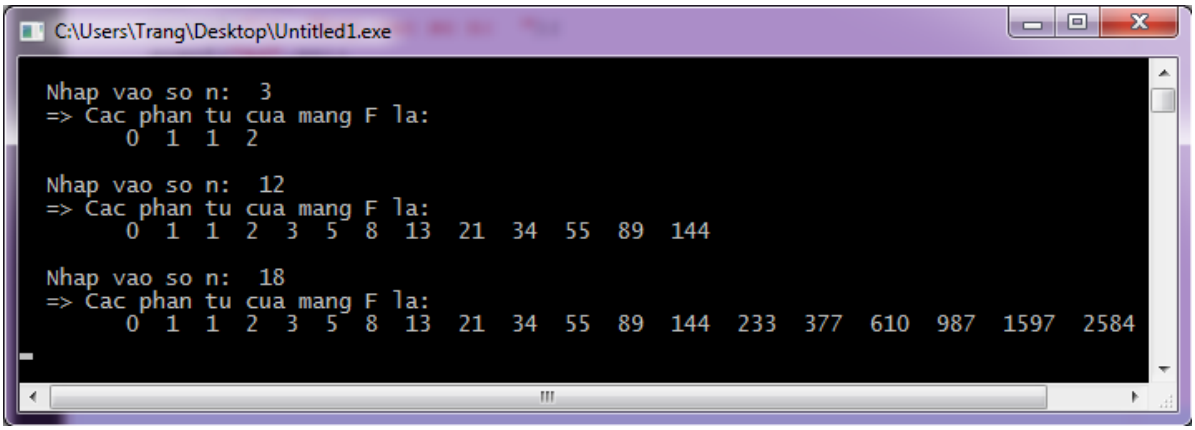
Viết chương trình nhập vào số nguyên dương n, cấp phát cho mảng F có n + 1 phần tử. Tính và in ra các phần tử của mảng? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
    int *F,n,i;
    printf("\n  Nhap vao so n:  ");
    scanf("%d",&n);
    F=(int*)malloc((n+1)*sizeof(int));
    printf("  => Cac phan tu cua mang F la:\n\t");
    *(F+0)=0;
    *(F+1)=1;
    for(i=2;i<=n;i++)
      *(F+i)=*(F+i-1)+*(F+i-2);
    for(i=0;i<=n;i++)
      printf("%d  ",*(F+i));
    printf("\n");
    getch();
    return main();
}
```

* *<u>Giải thích:</u>*

*- Vì có các phần tử từ F[0] đến F[n] nên phải cấp phát bộ nhớ cho n +1 phần tử. Con trỏ F sẽ trỏ tới phần tử F[0], ta gán giá trị cho F[0] = 0; F[1] = 1. Thực hiện tính các giá trị *(F+i) mà con trỏ (F+i) trỏ tới và đưa ra màn hình *(F+i)*

* *<u>Màn hình kết quả như sau:</u>*



<u>Bài 95:</u> Viết chương trình nhập vào số nguyên n, cấp phát bộ nhớ cho mảng b có n phần tử. Thực hiện:

a) Nhập phần tử của mảng b. Tính trung bình cộng của các phần tử dương.

b) Tìm các phần tử âm trong mảng. In ra giá trị và vị trí tương ứng của chúng trong mảng? (X)

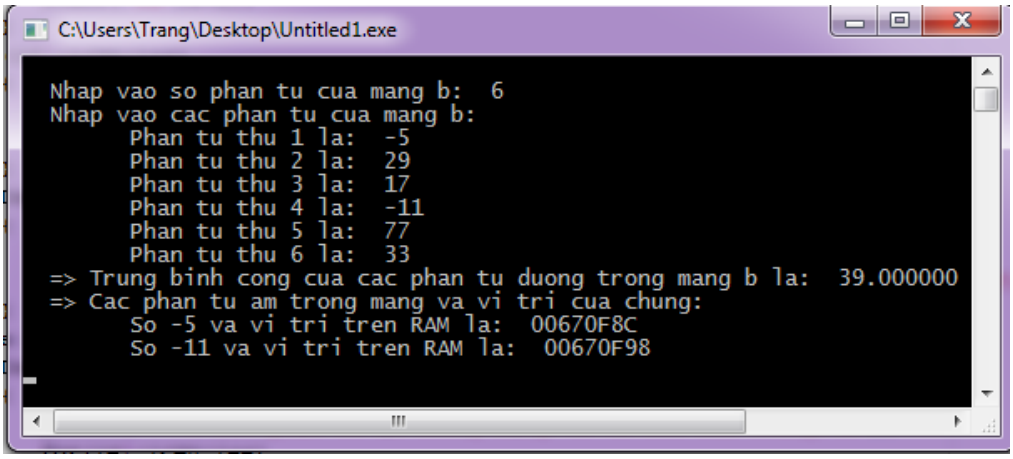
```
#include <stdio.h>
#include <conio.h>
main()
{
    int *b,n,i,DemSoDuong=0,DemSoAm=0,S=0;
    float TB;
    printf("\n  Nhap vao so phan tu cua mang b:  ");
    scanf("%d",&n);
    b=(int*)malloc(n*sizeof(int));
    printf("  Nhap vao cac phan tu cua mang b:\n");
    for(i=1;i<=n;i++)
      {
        printf("\tPhan tu thu %d la:  ",i);
        scanf("%d",b+i);
      }
    for(i=1;i<=n;i++)
      {
        if(*(b+i)>0){S=S+*(b+i);DemSoDuong++;}
```

```
            if(*(b+i)<0)DemSoAm++;
          }
        if(DemSoDuong!=0)
          {
            TB=(float)S/DemSoDuong;
             printf("  => Trung binh cong cua cac phan tu duong trong mang b la: %f\n",TB);
          }
        else printf("  => Khong co phan tu duong nao!\n");
        if(DemSoAm!=0)
          {
            printf("  => Cac phan tu am trong mang va vi tri cua chung:\n");
            for(i=1;i<=n;i++)
              if(*(b+i)<0)
                 printf("\tSo %d va vi tri tren RAM la:  %p\n",*(b+i),(b+i));
          }
        else printf("  => Khong co phan tu am nao!\n");
        getch();
        return main();
      }
```

* *Giải thích:*

- Kiểm tra các giá trị mà con trỏ trỏ tới nếu thỏa mãn là số dương thì tính trung bình cộng của các số đó. Tương tự đối với việc tìm ra những số âm và vị trí của chúng trên thanh RAM. Trường hợp không thấy xuất hiện số dương hoặc số âm nào thì không làm các công việc còn lại mà chỉ ra rằng dãy nhập vào không có số dương hoặc số âm.

* *Màn hình kết quả như sau:*



<u>Bài 96:</u> Viết chương trình nhập vào 2 mảng: a có n phần tử, b có m phần tử.

- Sắp xếp hai mảng theo thứ tự tăng dần.

- Xây dựng mảng c từ 2 mảng a và b. In ra các phần tử của mảng c.

- Sắp xếp c theo chiều giảm dần. In ra các phần tử của mảng mới. (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a[100],b[100],c[200],i,j,n,m,TrungGian;
    printf("\n  Nhap vao so phan tu cua mang a:  ");
    scanf("%d",&n);
    printf("  Nhap vao so phan tu cua mang b:  ");
    scanf("%d",&m);
    printf("  Nhap vao tung phan tu cua mang a:\n");
    for(i=1;i<=n;i++)
      {
        printf("\tPhan tu thu %d cua mang a:  ",i);
        scanf("%d",a+i);
      }
    printf("  Nhap vao tung phan tu cua mang b:\n");
    for(i=1;i<=m;i++)
      {
```

```
      printf("\tPhan tu thu %d cua mang b:  ",i);
      scanf("%d",b+i);
    }
printf("  => Sap xep tang dan cua mang a la:  ");
for(i=1;i<n;i++)
   for(j=i+1;j<=n;j++)
      if(*(a+i)>*(a+j))
         {
           TrungGian=*(a+i);
           *(a+i)=*(a+j);
           *(a+j)=TrungGian;
         }
 for(i=1;i<=n;i++)
    printf("%d  ",*(a+i));
printf("\n  => Sap xep tang dan cua mang b la:  ");
for(i=1;i<m;i++)
   for(j=i+1;j<=m;j++)
      if(*(b+i)>*(b+j))
         {
           TrungGian=*(b+i);
           *(b+i)=*(b+j);
           *(b+j)=TrungGian;
         }
 for(i=1;i<=m;i++)
    printf("%d  ",*(b+i));
printf("\n  => Cac phan tu cua mang c la:  ");
for(i=1;i<=n;i++)
   *(c+i)=*(a+i);
for(j=1;j<=m;j++)
   *(c+n+j)=*(b+j);
for(i=1;i<=m+n;i++)
```

```
    printf("%d  ",*(c+i));
  printf("\n  => Sap xep giam dan cua mang c la:  ");
  for(i=1;i<m+n;i++)
    for(j=i+1;j<=m+n;j++)
      if(*(c+i)<*(c+j))
        {
          TrungGian=*(c+i);
          *(c+i)=*(c+j);
          *(c+j)=TrungGian;
        }
  for(i=1;i<=m+n;i++)
    printf("%d  ",*(c+i));
  printf("\n");
  getch();
  return main();
}
```

* *Giải thích:*

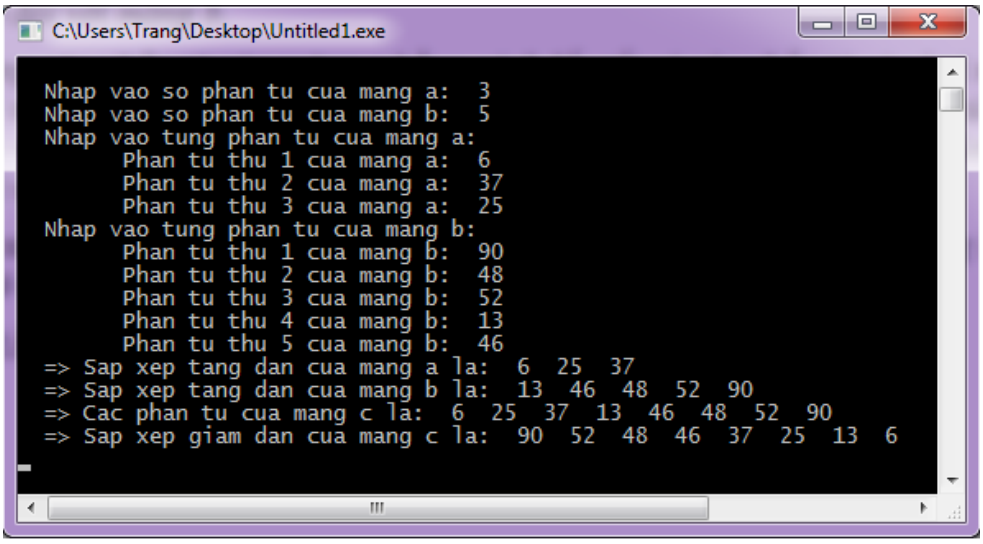
a) Khi khai báo mảng thì tên của mảng chính là một con trỏ trỏ tới phần tử đầu tiên của mảng: a là con trỏ trỏ tới a[1], b là con trỏ trỏ tới b[1] và c là con trỏ trỏ tới c[1].

b) Sắp xếp mảng theo chiều tăng dần sử dụng con trỏ: nếu giá trị mà con trỏ trỏ tới phần tử thứ i mà lớn hơn giá trị mà con trỏ trỏ tới phần tử thứ i+1 thì đổi giá trị cho nhau giữa các con trỏ. Tương tự đối với mảng b.

c) Xây dựng các phần tử của mảng c bằng cách lấy tất cả các phần tử của hai mảng a và b; mảng c sẽ có m + n phần tử; n phần tử của mảng a trùng với n phần tử đầu tiên của mảng c; từ phần tử thứ n + 1 của mảng c sẽ lấy trong m phần tử của mảng b (c[n+i] = b[i]). Sau đó sắp xếp giảm dần đối với m + n phần tử của mảng c vừa xây dựng.

Có thể xây dựng mảng c bằng nhiều cách khác nhau. Nếu mảng a và mảng b có cùng số phần tử thì có thể tạo mảng c bằng cách cộng hoặc nhân mảng a với mảng b,…

* *Màn hình kết quả như sau:*



<u>Bài 97:</u> Nhập vào một mảng 2 chiều mxn (m, n < 5) gồm các số thực:

- Nhập vào một số nguyên dương k (0 ≤ k < m; 0 ≤ k < n). Tính và đưa ra màn hình:

+ Tích các phần tử ở cột thứ k.

+ Số lượng các phần tử là số nguyên tố trên dòng thứ k.

- Tìm phần tử có giá trị nhỏ nhất trong mảng vừa nhập. Có bao nhiêu phần tử có giá trị bằng giá trị nhỏ nhất đó.

(Đề thi cuối kỳ lớp THCS 3, ĐHKHTN – ĐHQGHN, Thứ 6 ngày 21/12/2012, Kỳ I năm học 2012 – 2013, thời gian làm bài: 40 phút)

CHƯƠNG VI. XÂU KÍ TỰ

<u>Bài 97:</u> Viết chương trình nhập vào từ bàn phím một xâu kí tự. Thực hiện:

- Tính xem có bao nhiêu kí tự trong xâu đó.

- Đọc một kí tự từ bàn phím, tìm xem xâu đó có bao nhiêu kí tự giống với kí tự vừa đọc. Đưa kết quả ra màn hình? (X)

```
#include <stdio.h>
#include <conio.h>
main()
```

```
{
    char XauKiTu[1000],c;
    int i,Dem=0,SoKyTuGiong=0;
    printf("\n  Nhap vao mot xau ki tu:  ");
    gets(XauKiTu);
    while (XauKiTu[Dem]!='\0')Dem++;
    printf("  => So ki tu trong xau do la:  %d\n",Dem);
    printf("  Nhap vao mot ki tu tu ban phim:  ");
    scanf("%c",&c);
    for(i=0;i<Dem;i++)
      if(XauKiTu[i]==c)SoKyTuGiong++;
    printf("  => So ki tu cua xau giong voi ki tu %c vua nhap la:
%d\n",c,SoKyTuGiong);
    fflush(stdin);
    getch();
    return main();
}
```

* *Giải thích:*

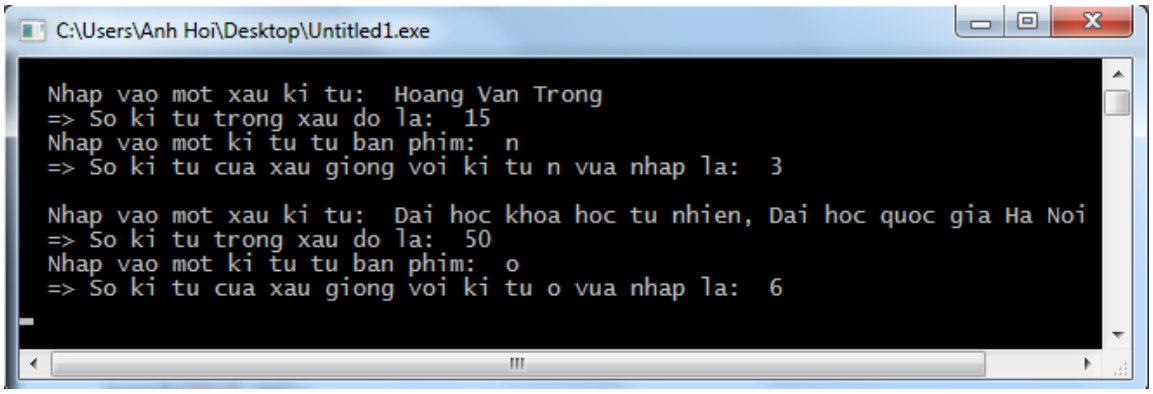
a) Xâu ký tự có n kí tự thực chất chỉ là một mảng 1 chiều gồm n phần tử. Khai báo kiểu kí tự cho xâu và giới hạn số kí tự tối đa của xâu là 1000. Ở đây phải dùng hàm đọc từ bàn phím ***gets()*** *chứ không được dùng hàm scanf(). Truy nhập tới một phần tử của xâu với tên biến và chỉ số đặt trong ngoặc vuông [ ] tương tự như truy nhập tới một phần tử của mảng.*

b) Tính xem có bao nhiêu kí tự trong xâu vừa nhập (hay xác định độ dài của xâu): đếm từ kí tự đầu tiên của xâu cho tới khi gặp kí tự '\0' thì dừng lại, dùng vòng lặp while.

c) Tìm xem trong xâu có bao nhiêu kí tự giống với kí tự được nhập vào từ bàn phím: dùng vòng for với biến i chạy từ 0 đến n – 1 (n là số kí tự của xâu). Nếu phần tử nào trong xâu giống với kí tự được nhập từ bàn phím thì tăng biến ***SoKiTuGiong*** *lên 1 đơn vị.*

d) Lệnh ***fflush(stdin)*** *có tác dụng xóa sạch bộ nhớ đệm bàn phím để đi vào lần nhập xâu kí tự khác. Nếu không dùng lệnh return main() thì cũng không cần dùng lệnh này.*

* *Màn hình kết quả như sau:*



<u>Bài 98:</u> Nhập vào từ bàn phím một xâu kí tự S. Đếm số lần xuất hiện của kí tự a trong S (a được nhập từ bàn phím) và kiểm tra xem S có phải là xâu đối xứng hay không?

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int DoiXung(char a[],int n);
main()
{
    char S[100],a;
    int i,GiongNhau=0;
    printf("\n  Nhap vao mot xau ki tu S:  ");
    gets(S);
    printf("  Nhap vao mot ki nao do tu ban phim:  ");
    scanf("%c",&a);
    for(i=0;i<strlen(S);i++)
      if(S[i]==a)GiongNhau++;
    printf("  => So lan xuat hien cua ki tu %c trong xau S la:
%d\n",a,GiongNhau);
    if(DoiXung(S,strlen(S))==1)printf("  => Xau vua nhap la mot xau doi
xung.\n");
    else printf("  => Xau vua nhap khong la mot xau doi xung.\n");
    fflush(stdin);
    getch();
```

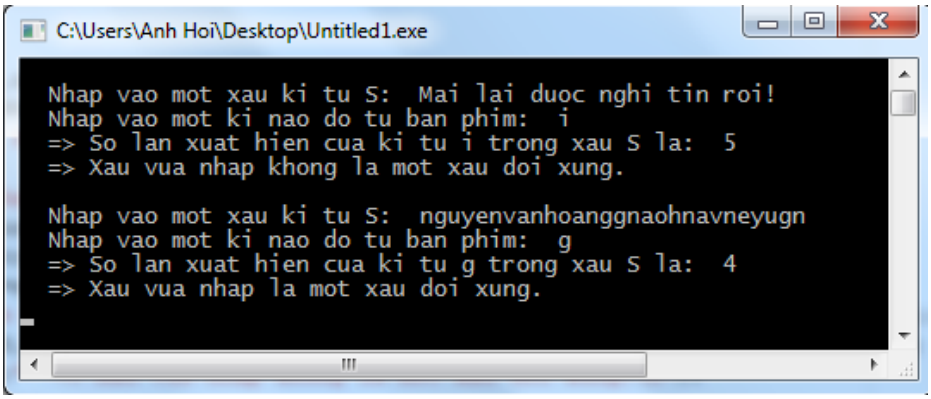
```
    return main();
}
int DoiXung(char a[],int n)
{
    int k=1,i;
    for(i=0;i<n;i++)
      if(a[i]!=a[n-i-1])k=0;
    return (k);
}
```

* *Giải thích:*

a) Đếm số lần xuất hiện của kí tự a trong xâu S: Dùng vòng lặp for duyệt qua từng kí tự của xâu S, nếu kí tự nào của xâu S mà giống với kí tự được nhập vào từ bàn phím thì tăng biến ***GiongNhau*** *lên 1 đơn vị với khởi tạo bằng 0. Giá trị của biến GiongNhau sau vòng lặp for cuối cùng chính là số lần xuất hiện của kí tự trong xâu và đưa ra màn hình giá trị của biến GiongNhau.*

b) Kiểm tra xem xâu S nhập vào có phải là xâu đối xứng hay không: Xâu đối xứng là xâu mà các kí tự cách đều 2 đầu mút của xâu thì giống hệt nhau (tương tự dãy đối xứng ở phần mảng giá trị một chiều). Ban đầu coi xâu S có tính đối xứng (gán k = 1). Vòng lặp for kiểm tra từng kí tự của xâu với biến i chạy từ 0 đến n – 1 (n là số kí tự của xâu S và được tính qua hàm strlen(S)), nếu S[i] ≠ S[n-i-1] thì lập tức kết luận xâu trên không có tính đối xứng (gán k=0). In ra màn hình những nội dung tương ứng với kết quả trả về của hàm DoiXung().

* *Màn hình kết quả như sau:*



Bài 99: Viết chương trình nhập vào hai xâu kí tự, ghép hai xâu này thành một xâu. Chuyển xâu kí tự đã ghép thành chữ hoa, in ra màn hình để kiểm tra kết quả chuyển đổi? (X)

```
#include <stdio.h>
```

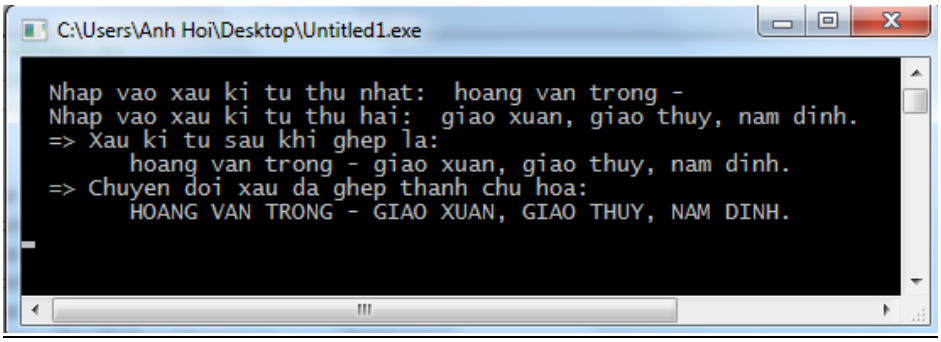
```
#include <conio.h>
#include <ctype.h>
main()
{
    char
XauKiTu1[1000],XauKiTu2[1000],XauKiTu3[3000],Dem1=0,Dem2=0,i,j;
    printf("\n  Nhap vao xau ki tu thu nhat:  ");
    gets(XauKiTu1);
    printf("  Nhap vao xau ki tu thu hai:  ");
    gets(XauKiTu2);
    while (XauKiTu1[Dem1]!='\0')Dem1++;
    while (XauKiTu2[Dem2]!='\0')Dem2++;
    printf("  => Xau ki tu sau khi ghep la:\n\t");
    for(i=0;i<Dem1;i++)
      XauKiTu3[i]=XauKiTu1[i];
    for(j=0;j<Dem2;j++)
      XauKiTu3[Dem1+j]=XauKiTu2[j];
    XauKiTu3[Dem1+Dem2]='\0';
    printf("%s\n",XauKiTu3);
    i=0;
    while(XauKiTu3[i]!='\0')
      {XauKiTu3[i]=toupper(XauKiTu3[i]);i++;}
    printf("  => Chuyen doi xau da ghep thanh chu hoa:\n\t%s\n",XauKiTu3);
    getch();
    return main();
}
```

* *Giải thích:*

*a) Ghép hai xâu đã cho thành một xâu thứ 3. Trước tiên, đếm xem xâu 1 và xâu 2 có bao nhiêu kí tự, sử dụng biến **Dem1** và biến **Dem2**. Do đó, xâu thứ 3 sẽ có Dem1 + Dem 2 kí tự. Các phần tử của xâu thứ 3 được lấy lần lượt của xâu 1 và xâu 2, lấy hết xâu 1 được Dem1 phần tử, sau đó lấy sang xâu 2 được Dem1 + Dem2 phần tử. Sử dụng vòng lặp for để gán từng phần tử của xâu 1 và xâu 2 cho xâu 3.*

b) Chuyển xâu kí tự thành chữ hoa: dùng vòng lặp while với i bắt đầu từ giá trị 0 (nếu dùng vòng for thì biến i chạy từ 0 đến Dem1 + Dem2 – 1) nếu XauKiTu3[i] khác kí tự kết thúc '\0' thì thực hiện chuyển đổi thành kí tự hoa với hàm toupper() được lấy trong thư viện ctype.h. Cuối cùng, in xâu đã chuyển đổi chữ hoa ra màn hình.

* *Màn hình kết quả như sau:*

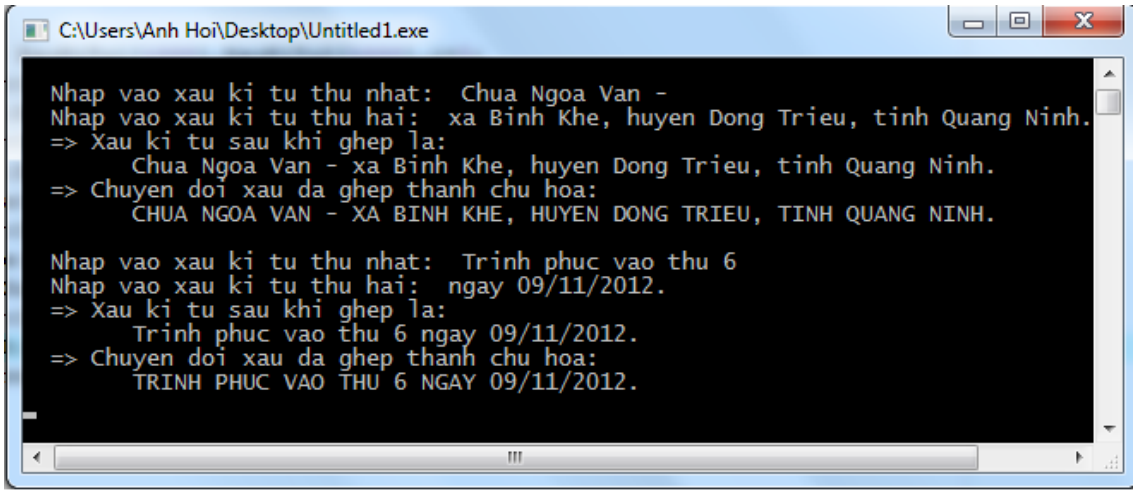


** Ta có thể ghép xâu 2 vào xâu 1 nhờ hàm strcat(xâu 1, xâu 2). Hàm này sẽ thêm vào sau xâu 1 các phần tử của xâu 2 (với điều kiện bộ nhớ dành cho xâu 1 đủ để chứa các phần tử của xâu 2). Sử dụng cách này sẽ làm cho số câu lệnh ít hơn rất nhiều. Mã cho trường hợp này như sau:*

```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
main()
{
    char XauKiTu1[1000],XauKiTu2[1000],i=0;
    printf("\n  Nhap vao xau ki tu thu nhat:  ");
    gets(XauKiTu1);
    printf("  Nhap vao xau ki tu thu hai:  ");
    gets(XauKiTu2);
    strcat(XauKiTu1,XauKiTu2);
    printf("  => Xau ki tu sau khi ghep la:\n\t%s\n",XauKiTu1);
    while(XauKiTu1[i]!='\0')
      {XauKiTu1[i]=toupper(XauKiTu1[i]);i++;}
    printf("  => Chuyen doi xau da ghep thanh chu hoa:\n\t%s\n",XauKiTu1);
    getch();
    return main();
```

```
}
```

* *Màn hình kết quả như sau:*



<u>Bài 100:</u> Nhập vào một xâu kí tự:

a) Xét xem trong xâu có k kí tự kề nhau và giống nhau hay không?

b) Thực hiện loại bỏ tất cả các kí tự kề nhau mà giống nhau, chỉ để lại một?

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
main()
{
    char Xau[100];
    int k,h=0,i,j,Dem;
    printf("\n  Nhap vao mot xau ki tu:  ");
    gets(Xau);
    printf("  Nhap so k de xem co k ki tu ke nhau va giong nhau hay khong:
");
    nhaplai:scanf("%d",&k);
    if(k>1)
      {
        for(i=0;i<strlen(Xau)-1;i++)
          {Dem=1;
```

```
                for(j=i+1;j<strlen(Xau);j++)
                  {
                    if(Xau[i]==Xau[j])Dem++;
                    else break;
                  }
                if(Dem>=k)h=1;
               }
             if(h==1)printf("  => Xau co %d ki tu ke nhau va giong nhau.\n",k);
             else printf("  => Xau khong co %d ki tu ke nhau va giong nhau.\n",k);
          }
        else {printf("  Nhap lai k cho anh mau:  ");goto nhaplai;}
        for(i=0;i<strlen(Xau)-1;i++)
          if(Xau[i]==Xau[i+1])
            { strcpy(Xau+i,Xau+i+1);
              i--;
            }
        printf("  => Xau ki tu sau khi chinh sua la:  %s\n",Xau);
        fflush(stdin);
        getch();
        return main();
    }
```

* *Giải thích:*

- Thực hiện nhập xâu kí tự bất kì và số k từ bàn phím. Nếu k > 1 thì mới làm tiếp bước kiểm tra trong xâu có k kí tự kề nhau và giống nhau hay không. Ngược lại, nếu k ≤ 1 thì tiến hành nhập lại giá trị của k bằng lệnh nhảy vô điều kiện ***goto*** *tới chỗ gán nhãn* ***nhaplai****. Mọi người tìm hiểu thêm lệnh nhảy goto ở giáo trình* ***Ngôn ngữ lập trình C*** *– Quách Tuấn Ngọc, trang 151; sử dụng lệnh này có rất nhiều tiện ích hay.*

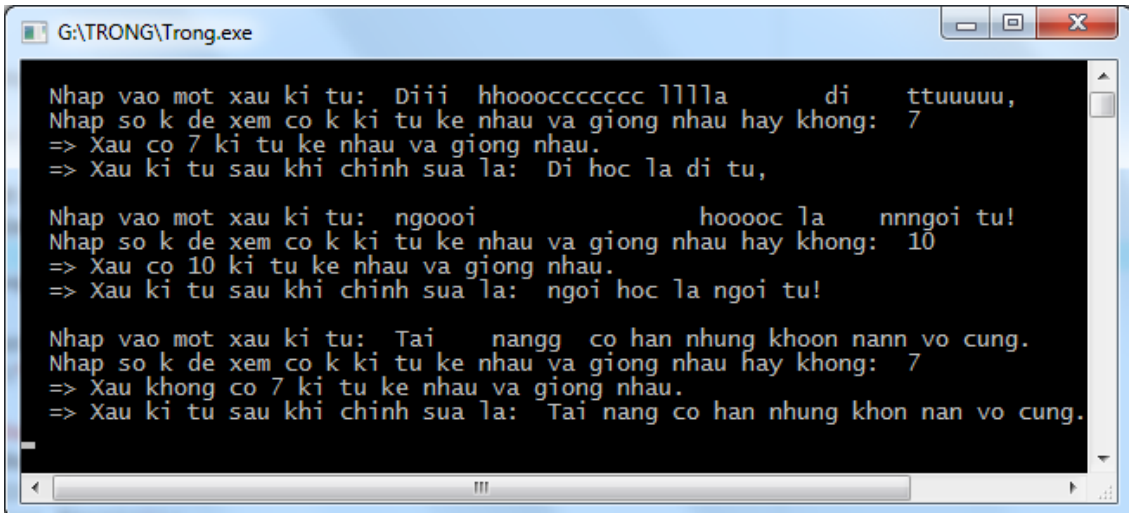
a) Thuật toán xét xem xâu có k kí tự kề nhau và giống nhau hay không: Trước tiên giả sử rằng xâu đã cho không có k kí tự kề nhau và giống nhau (khởi tạo h = 0). Thực hiện duyệt từng kí tự của xâu; với mỗi kí tự thứ i ta kiểm tra xem từ kí tự thứ i+1 đến kí tự cuối cùng có bao nhiêu kí tự kề ngay sau kí tự thứ i và giống với nó rồi trả kết quả về cho biến ***Dem****, trong trường hợp mà thấy kí tự thứ i và kí tự thứ j khác nhau (j chạy từ i+1 đến strlen(Xau) – 1) thì lập tức thoát khỏi vòng lặp for(j=i+1;j<strlen(Xau);j++) bằng lệnh nhảy vô điều kiện* ***break****. Khi ra khỏi vòng for(j=i+1;j<strlen(Xau);j++) thì thực hiện phép so sánh Dem với k, nếu Dem ≥ k thì có nghĩa là trong xâu đã xuất hiện nhiều hơn hoặc bằng k kí tự kề nhau và giống nhau (gán h = 1). Tiếp*

theo, dựa vào giá trị của biến h để kết luận xâu đã cho có k kí tự kề nhau và giống nhau hay không.

b) Nếu thấy có nhiều kí tự kề nhau và giống nhau thì chỉ giữ lại một: Dùng vòng for duyệt từng kí tự của xâu từ kí tự đầu tiên đến kí tự sát với kí tự cuối cùng (trong trường hợp này thì chỉ cần kiểm tra từ Xau[0] đến Xau[n-2] với n là tổng số kí tự). Nếu kí tự thứ i bằng kí tự thứ i+1 thì ghi đè địa chỉ của kí tự thứ i+1 lên địa chỉ của kí tự thứ i, lúc này kí tự thứ i+1 sẽ trở thành kí tự thứ i của xâu mới và tổng số kí tự của xâu mới bị giảm đi 1 vì vậy ta phải giảm i đi 1 đơn vị để tiếp tục duyệt các kí tự khác của vòng for thì mới cho kết quả chính xác.

Dùng hàm strcpy (Xau+i, Xau+i+1) để ghi đè địa chỉ của kí tự thứ i+1 lên địa chỉ của kí tự thứ i.

* *Màn hình kết quả như sau:*



<u>Bài 101:</u> Xâu kí tự chuẩn là xâu không có dấu cách ở đầu và cuối câu, đồng thời không có hai dấu cách liền kề nhau trong xâu. Viết chương trình:

- Nhập một xâu kí tự từ bàn phím.

- Đếm xem có bao nhiêu dấu cách trong xâu.

- Nếu xuất hiện 2 dấu cách liên tiếp nhau thì bỏ đi một. (X)

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
main()
{
   char Xau[1000];
   int i=0,SoKyTuTrang=0;
```

```
    printf("\n  Nhap vao mot xau ki tu:  ");
    gets(Xau);
    while(Xau[i]!='\0')
      {if(Xau[i]==' ')SoKyTuTrang++;i++;}
    printf("  => So ki tu trang trong xau la:  %d\n",SoKyTuTrang);
    i=0;
    while(Xau[i]!='\0')
     {
       if(Xau[i]==' '&&Xau[i]==Xau[i+1]){strcpy(Xau+i,Xau+i+1);i--;}
       i++;
     }
    printf("  => Xau da chinh sua dau cach la:  %s\n",Xau);
    getch();
    return main();
}
```

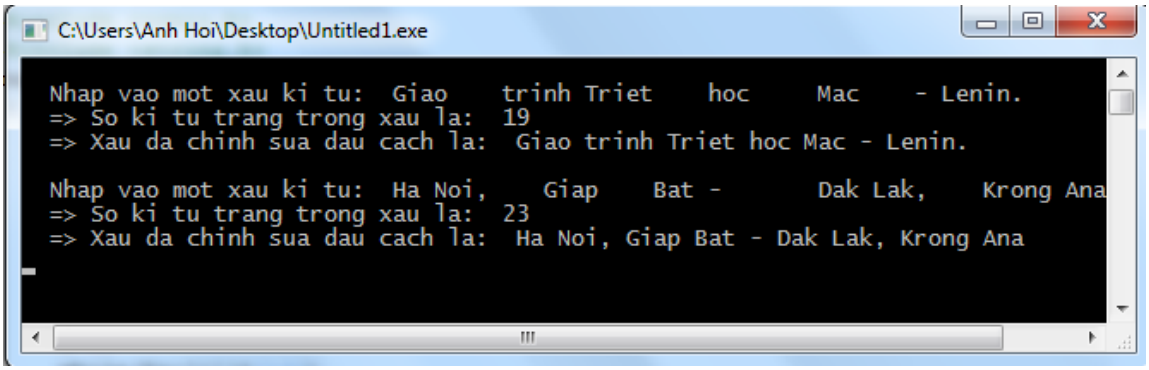
* <u>*Giải thích:*</u>

*- Sử dụng biến **SoKiTuTrang** để đếm số kí tự trắng với giá trị khởi tạo bằng 0. Dùng vòng lặp while duyệt qua từng kí tự của xâu, nếu kí tự nào bằng kí tự trắng thì tăng biến SoKiTuTrang lên 1 đơn vị.*

- Bỏ đi một dấu cách nếu có 2 dấu cách liên tiếp nhau: vòng lặp while chạy từ kí tự đầu tiên cho đến khi gặp kí tự '\0' thì dừng lại. Nếu kí tự thứ i bằng kí tự thứ i + 1 và bằng kí tự trắng thì copy đè địa chỉ của kí tự thứ i +1 lên địa chỉ của kí tự thứ i (lúc này xâu sẽ giảm đi một kí tự) bằng hàm strcpy() và khi copy đè như vậy thì kí tự thứ i +1 sẽ trở thành kí thự thứ i cho vòng lặp sau, chính vì vậy phải giảm giá trị của i xuống 1 đơn vị. Cuối cùng in ra xâu đã chỉnh sửa dấu cách.

* <u>*Lưu ý:*</u> *Nếu dãy ban đầu nhập vào có chứa nhiều dấu cách ở đầu câu hoặc cuối câu thì cuối cùng vẫn còn chứa 1 dấu cách ở đầu hoặc cuối câu. Vì đề bài không yêu cầu nên mình không viết lệnh xóa dấu cách ở đầu câu và cuối câu.*

* <u>*Màn hình kết quả như sau:*</u>



<u>Bài 102:</u> Viết chương trình nhập một xâu kí tự bất kì từ bàn phím. Chuẩn hóa xâu kí tự:

- Nếu có dấu cách ở đầu câu và cuối câu thì bỏ đi.

- Nếu trong xâu kí tự có hai dấu cách liên tiếp nhau thì bỏ đi một dấu cách.

- Đếm xem trong xâu kí tự vừa chuẩn hóa có bao nhiêu từ. (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
   char Xau[1000];
   int i=0,SoDauCach=0,SoKiTu;
   printf("\n  Nhap vao mot xau ki tu:  ");
   gets(Xau);
   while(Xau[i]==' ')
     strcpy(Xau+i,Xau+i+1);
   SoKiTu=strlen(Xau);
   i=SoKiTu-1;
   while(Xau[i]==' ')
     {Xau[i]='\0';i--;}
   i=0;
   while(Xau[i]!='\0')
    {
      if(Xau[i]==' '&&Xau[i]==Xau[i+1]){strcpy(Xau+i,Xau+i+1);i--;}
      i++;
```

```
    }
    printf("  => Xau da chuan hoa la:  %s\n",Xau);
    i=0;
    while(Xau[i]!='\0'){if(Xau[i]==' ')SoDauCach++;i++;}
    printf("  => So tu trong xau la:  %d\n",SoDauCach+1);
    getch();
    return main();
}
```

* *Giải thích:*

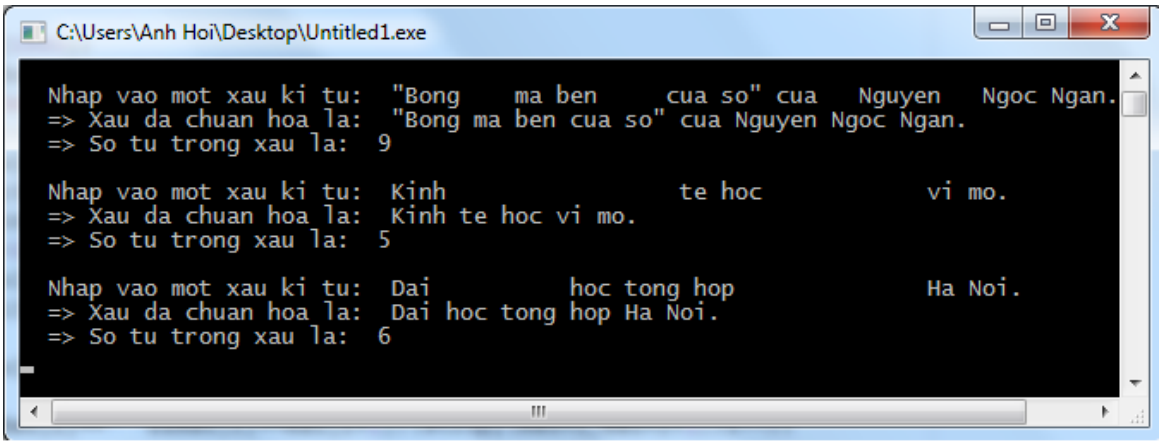
a) Bỏ đi các dấu cách ở đầu câu: Vòng lặp while duyệt phần tử đầu tiên mà nếu phần tử này bằng kí tự trắng thì copy đè địa chỉ của phần tử thứ hai lên phần tử thứ nhất và phần tử thứ hai này lại trở thành phần tử thứ nhất trong vòng lặp sau. Cứ thế cho tới khi gặp kí tự khác kí tự trắng thì dừng lại.

b) Bỏ đi các dấu cách ở cuối câu: Sau khi bỏ đi các kí tự trắng ở đầu câu thì xâu ban đầu trở thành xâu mới, số kí tự của xâu mới này được gán cho biến ***SoKiTu*** *và gán i = SoKiTu – 1. Vòng lặp while duyệt từ phần tử cuối cùng của xâu mới, nếu phần tử này bằng kí tự trắng thì gán nó bằng kí tự kết thúc '\0', sau đó giảm biến i xuống 1 đơn vị để đi vào vòng lặp sau.*

c) Sau khi bỏ đi các kí tự trắng ở cuối câu thì xâu lại trở thành xâu mới. Gán lại giá trị khởi tạo cho i = 0 và thực hiện bỏ đi một dấu cách nếu có hai dấu cách đứng cạnh nhau trong xâu. Nếu kí tự thứ i bằng kí tự thứ i + 1 và bằng kí tự trắng thì copy đè địa chỉ của phần tử thứ i +1 lên địa chỉ của phần tử thứ i. Hết bước này thì xâu ban đầu nhập vào đã được chuẩn hóa và ta chỉ cần viết lệnh in ra màn hình là ok.

d) Số từ trong xâu: Muốn biết trong xâu có bao nhiêu từ thì chỉ cần lấy số kí tự trắng cộng thêm 1 đơn vị, số kí tự trắng được xác định nhờ biến ***SoDauCach****. Nhưng số kí tự trắng ở đây phải được đếm sau khi xâu đã được chuẩn hóa. Nếu không số từ hiện ra sẽ không chính xác.*

* *Màn hình kết quả như sau:*



<u>Bài 103:</u> Viết chương trình nhập vào từ bàn phím một xâu kí tự. Tính xem có bao nhiêu loại chữ cái có trong xâu, mỗi loại chữ cái xuất hiện bao nhiêu lần, in kết quả thông báo ra màn hình? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
   char Xau[1000];
   int i,j,k,n,SoLoai,Dem=0,Giong=0;
   printf("\n  Nhap vao mot xau ki tu:  ");
   gets(Xau);
   SoLoai=strlen(Xau);
   n=strlen(Xau);
   for(i=0;i<n;i++)
     for(j=i-1;j>=0;j--)
       if(Xau[i]==Xau[j]){SoLoai--;break;}
   for(i=0;i<n;i++)
     if(Xau[i]==' '){SoLoai--;break;}
   printf("  => So loai chu cai co trong xau la:  %d\n",SoLoai);
   printf("  => So lan xuat hien cua cac chu cai co trong xau:\n");
   for(i=0;i<n;i++)
     {
       for(j=i-1;j>=0;j--)
         if(Xau[i]==Xau[j])Giong=1;
           if(Giong==0&&Xau[i]!=' ')
             {
              for(k=0;k<n;k++)
                if(Xau[i]==Xau[k])Dem++;
              printf("\tChu cai '%c' xuat hien %d lan.\n",Xau[i],Dem);
             }
           Dem=0;Giong=0;
     }
```

```
    getch();
    return main();
}
```

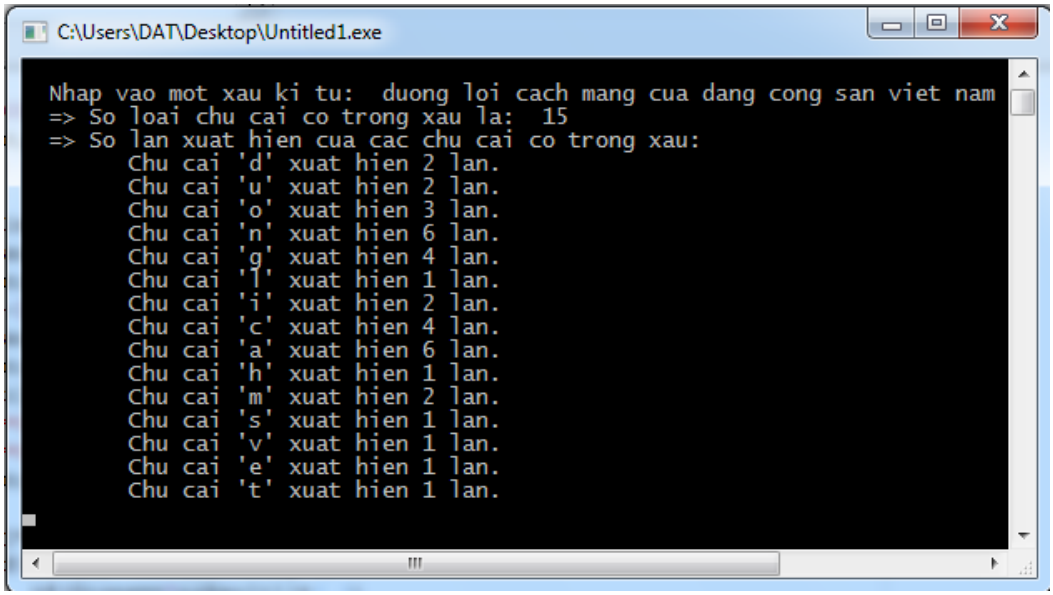
* *Giải thích:*

a) Tính xem có bao nhiêu loại chữ cái xuất hiện trong xâu: được xác định bằng biến ***SoLoai****. Ban đầu gán biến SoLoai bằng tổng số kí tự có trong xâu (kể cả kí tự trắng và kí tự đặc biệt như: ; , \ " " ? - _ ! @ # % ^ & * ....) và tổng số kí tự này được xác định bằng hàm* ***strlen(Xau)****, sau đó duyệt từng kí tự từ kí tự đầu tiên đến kí tự cuối cùng. Nếu kí tự nào trùng với một trong các kí tự đứng trước nó thì giảm SoLoai xuống 1 đơn vị. Cuối cùng, nếu thấy trong xâu có kí tự trắng thì lại giảm SoLoai xuống tiếp 1 đơn vị (giả sử coi như các kí tự đặc biệt cũng là các chữ cái).*

b) Đếm số lần xuất hiện của các chữ cái: Duyệt từng kí tự của xâu bằng vòng lặp for. Nếu kí tự nào không giống với ít nhất một kí tự trước nó và kí tự đó phải khác kí tự trắng thì thực hiện đếm các kí tự giống với nó trong xâu, khởi tạo biến ***Dem*** *bằng 0 và cũng duyệt qua tất cả kí tự của xâu nếu thấy trùng nhau thì tăng biến Dem lên 1 đơn vị. Sau cùng, in ra chữ cái và số lần xuất hiện tương ứng của chữ cái đó. Mọi người lưu ý là có sự kết hợp khá phức tạp giữa các vòng for lồng nhau ở đoạn chương trình trên.*

c) Nếu không cho các kí tự đặc biệt là chữ cái thì phải viết thêm 1 số câu lệnh nữa để không đếm các kí tự đặc biệt đó. Ở bài này mình chỉ cho kí tự trắng không phải chữ cái mà thôi! Một điều nữa là có sự phân biệt chữ thường và chữ hoa, chương trình sẽ đếm chữ hoa thành một loại chữ cái riêng và chữ thường thành một loại chữ cái riêng.

* *Màn hình kết quả như sau:*



<u>Bài 104:</u> Viết chương trình nhập vào từ bàn phím một xâu kí tự bất kì:

- Chuẩn hóa xâu kí tự vừa nhập.

- Chuyển đổi từ thứ k thành chữ hoa, in ra màn hình chuỗi kí tự để xem kết quả.

- Tách kí tự vừa chuyển thành chữ hoa đó ra khỏi xâu kí tự, in chuỗi kí tự này ra màn hình để xem kết quả. (X)

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>
main()
{
   char Xau[1000];
   int i=0,j=0,k,z,SoKiTu,SoDauCach=0,DoDaiTu=0,Dem=1;
   printf("\n  Nhap vao mot xau ki tu:  ");
   gets(Xau);
   while(Xau[i]==' ')
     strcpy(Xau+i,Xau+i+1);
   SoKiTu=strlen(Xau);
   i=SoKiTu-1;
   while(Xau[i]==' ')
     {Xau[i]='\0';i--;}
   i=0;
   while(Xau[i]!='\0')
    {
      if(Xau[i]==' '&&Xau[i]==Xau[i+1]){strcpy(Xau+i,Xau+i+1);i--;}
      i++;
    }
   printf("  => Xau da chuan hoa la:  %s\n",Xau);
   i=0;
   while(Xau[i]!='\0'){if(Xau[i]==' ')SoDauCach++;i++;}
   printf("  Nhap vao so k de chuyen tu thu k thanh chu hoa:  ");
```

```
   nhaplaidiku:scanf("%d",&k);
   if(k==1)
     {  z=0;
       for(i=0;i<strlen(Xau);i++)
         {
           if(Xau[i]!=' ')DoDaiTu++;
           if(Xau[i]==' ')break;
         }
       for(i=0;i<strlen(Xau);i++)
         {
           if(Xau[i]!=' ')Xau[i]=toupper(Xau[i]);
           if(Xau[i]==' ')break;
         }
       printf("  => Chuoi ki tu sau khi chuyen tu thu %d thanh chu hoa
la:\n\t%s\n",k,Xau);
     }
   else if(k>1&&k<=SoDauCach+1)
     {
       for(i=0;i<strlen(Xau);i++)
         {
           if(Xau[i]==' ')Dem++;
           if(Dem==k){j=i+1;break;}
         }
       z=j;
       for(i=j;i<strlen(Xau);i++)
         {
           if(Xau[i]!=' '){Xau[i]=toupper(Xau[i]);DoDaiTu++;}
           if(Xau[i]==' ')break;
         }
       printf("  => Chuoi ki tu sau khi chuyen tu thu %d thanh chu hoa
la:\n\t%s\n",k,Xau);
     }
```

```
        else
          {printf("  Nhap lai k di may:  ");goto nhaplaidiku;}
        strcpy(Xau+z,Xau+z+DoDaiTu);
        printf("  => Chuoi ki tu sau khi da tach chu hoa ra la:\n\t%s\n",Xau);
        fflush(stdin);
        getch();
        return main();
    }
```

* *Giải thích:*

Bài này mình viết code để xử lý tất cả các trường hợp đặc biệt có thể xảy ra nên phải sử dụng nhiều biến do đó nếu nhìn qua thì khá rắc rối, mọi người chịu khó nhé!

a) Phần chuẩn hóa xâu kí tự nhập vào: tức là xóa hết kí tự trắng ở đầu xâu và cuối xâu, nếu trong xâu mà có 2 kí tự trắng liên tiếp nhau thì bỏ đi một kí tự trắng. Thuật toán này đã có trong bài trước đó về chuẩn hóa xâu kí tự. Để xóa bỏ kí tự trắng ở đầu xâu thì duyệt các phần tử từ phần tử đầu tiên với điều kiện kí tự đó phải bằng kí tự trắng thì mới cho thực hiện vòng lặp, sau đó copy đè địa chỉ của phần tử thứ i +1 lên địa chỉ của phần tử thứ i và xâu đã bị giảm đi một kí tự, cứ như thế cho tới khi gặp kí tự khác kí tự trắng thì thôi. Còn xóa kí tự trắng ở cuối xâu thì lại duyệt ngược lại bắt đầu từ kí tự cuối cùng, nếu nó bằng kí tự trắng thì gán nó bằng kí tự kết thúc '\0'. Xóa 1 kí tự trắng nếu thấy xuất hiện 2 kí tự trắng đứng cạnh nhau thì làm tương tự, nếu thấy phần tử thứ i bằng phần tử thứ i +1 và bằng kí tự trắng thì copy đè địa chỉ của kí tự thứ i +1 lên địa chỉ của phần tử thứ i.

*b) Chuyển đổi từ thứ k thành chữ hoa với k được nhập vào từ bàn phím: Xâu sau khi đã được chuẩn hóa thì số dấu cách của xâu mới này được xác định bằng biến **SoDauCach** và số từ trong xâu sẽ là SoDauCach +1. Chính vì vậy, muốn nhập số k vào để chuyển đổi từ thứ k thành chữ hoa thì k phải nằm trong đoạn [1, SoDauCach + 1], nếu k không thỏa mãn điều kiện này thì tiến hành nhập lại có sử dụng tới lệnh nhảy vô điều kiện **goto**.*

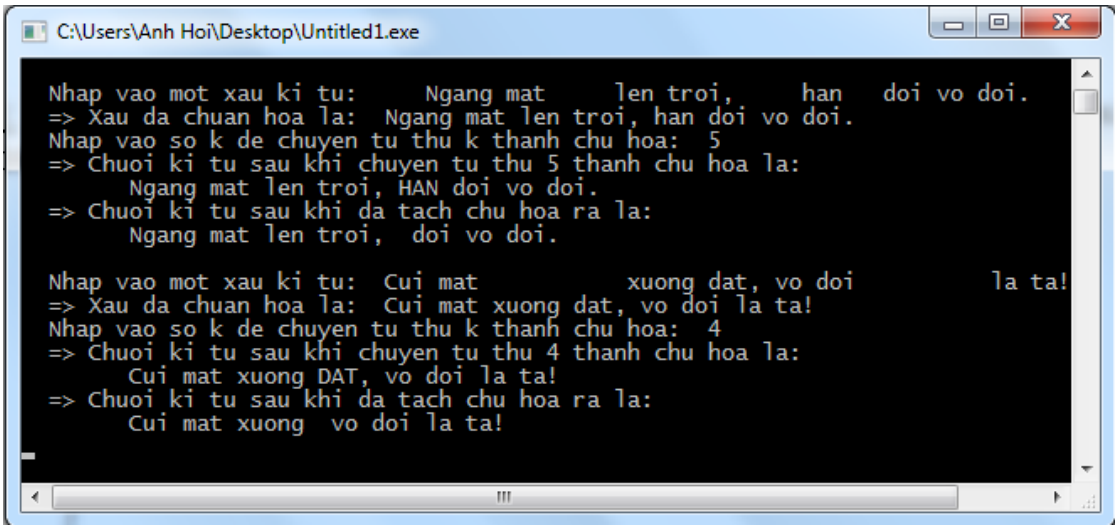
*Nếu k = 1 thì ta chỉ cần chuyển từ thứ nhất thành chữ hoa bắt đầu từ kí tự Xau[0] đến khi gặp kí tự trắng thì dừng lại và thoát khỏi vòng lặp for. Biến **z** để xác định vị trí bắt đầu của từ cần chuyển thành chữ hoa (trường hợp này z = 0), dùng biến này để thuận tiện cho việc cắt từ ở những câu lệnh sau đó, biến **DoDaiTu** để xác định độ dài của từ cần chuyển thành chữ hoa. Tiếp theo, hiện ra xâu kí tự đã chuyển từ thứ k thành chữ hoa.*

*Nếu $1 < k \leq SoDauCach + 1$ thì công việc lại phức tạp hơn, ta phải xác định được vị trí bắt đầu của từ thứ k. Biến **Dem** được dùng để đếm số dấu cách cho tới khi Dem = k (với Dem được khởi tạo bằng 1) thì vị trí bắt đầu của từ thứ k chính là j = i +1 và thoát khỏi vòng lặp for. Khi đã xác định được vị trí bắt đầu của từ thứ k rồi thì dùng hàm toupper(Xau[i]) để chuyển thành chữ hoa từ kí tự i = j đến khi gặp kí tự trắng thì thoát khỏi vòng lặp. Trong trường hợp này của k cũng sử dụng biến z để xác định vị trí bắt đầu của từ thứ k (chính là z = j) và biến DoDaiTu để xác định độ dài của từ thứ k. Tiếp theo, hiện ra xâu kí tự đã chuyển từ thứ k thành chữ hoa.*

c) Tách từ vừa in thành chữ hoa ra khỏi chuỗi: Ở bước trên ta đã xác định được vị trí bắt đầu và độ dài của từ thứ k thì đến phần này ta chỉ cần copy đè địa chỉ của phần tử thứ z + DoDaiTu lên địa chỉ của phần tử thứ z thì lập tức xâu sẽ mất một từ thứ k. Chỗ mà có từ bị cắt này sẽ xuất

hiện 2 kí tự trắng (mình cố tình để như thế cho dễ nhìn thấy chỗ bị cắt), còn nếu muốn chỉ xuất hiện 1 kí tự trắng thì thay vì copy đè địa chỉ của phần tử thứ z + DoDaiTu thì ta copy đè địa chỉ của phần tử z + DoDaiTu + 1 lên địa chỉ của phần tử thứ z.

* *Màn hình kết quả như sau:*



<u>Bài 105:</u> Viết chương trình nhập vào từ bàn phím một xâu kí tự bất kì:

- Chuẩn hóa xâu kí tự vừa nhập và chuyển kí tự đầu tiên của xâu thành chữ hoa.

- Thay tất cả các kí tự của từ cuối cùng trong xâu thành các dấu *. In kết quả.

- In mỗi từ trong xâu ra một dòng. (X)

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>
main()
{
   char Xau[1000];
   int i=0,SoKiTu;
   printf("\n  Nhap vao mot xau ki tu:  ");
   gets(Xau);
   while(Xau[i]==' ')
     strcpy(Xau+i,Xau+i+1);
```

```
    SoKiTu=strlen(Xau);
    i=SoKiTu-1;
    while(Xau[i]==' ')
      {Xau[i]='\0';i--;}
    i=0;
    while(Xau[i]!='\0')
     {
       if(Xau[i]==' '&&Xau[i]==Xau[i+1]){strcpy(Xau+i,Xau+i+1);i--;}
       i++;
     }
    printf("  => Xau da chuan hoa la:  %s\n",Xau);
    Xau[0]=toupper(Xau[0]);
    printf("  => Xau ki tu sau khi bien doi ki tu dau tien thanh ki tu hoa
  la:\n\t%s\n",Xau);
    for(i=strlen(Xau)-1;i>=0;i--)
      {
        if(Xau[i]!=' ')Xau[i]='*';
        else break;
      }
    printf("  => Xau ki tu sau khi bien doi tu cuoi cung la:\n\t%s\n",Xau);
    printf("  => In moi tu trong xau ra mot dong:\n\t");
    for(i=0;i<strlen(Xau);i++)
      {
        if(Xau[i]!=' ')printf("%c",Xau[i]);
        else printf("\n\t");
      }
    printf("\n");
    getch();
    return main();
  }
```

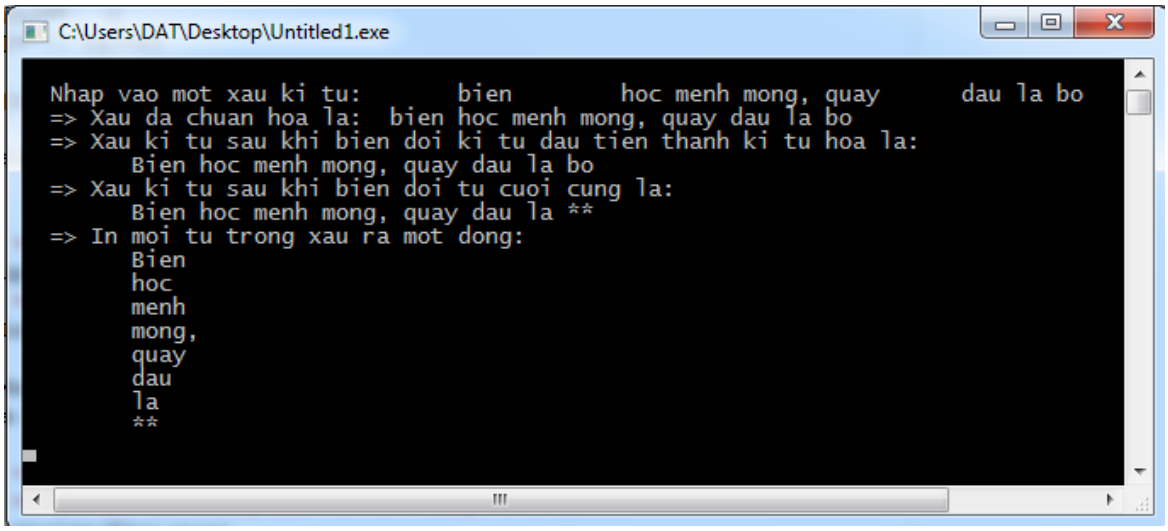
* *Giải thích:*

a) Chuẩn hóa xâu kí tự cũng giống như các bài ở trên, cũng phải làm các công việc để xóa bỏ kí tự trắng ở đầu và cuối xâu, nếu trong xâu có 2 kí tự trắng cạnh nhau thì bỏ đi một kí tự trắng. Sau khi chuẩn hóa xong thì vừa biến đổi kí tự đầu tiên thành kí tự hoa vừa gán kí tự hoa đó cho kí tự đầu tiên.

*b) Thay tất cả các kí tự của từ cuối cùng trong xâu thành kí tự * thì dùng vòng lặp for duyệt từ kí tự cuối cùng cho tới khi gặp kí tự trắng thì thoát vòng lặp. Duyệt đến đâu thì gán kí tự * cho phần tử thứ i của xâu.*

*c) Sau khi thay tất cả các kí tự của từ cuối cùng thành kí tự * thì tiến hành in ra các từ trong xâu, in mỗi từ trên một dòn. Vòng lặp for duyệt từ phần tử đầu tiên đến phần tử cuối cùng. Nếu phần tử thứ i khác kí tự trắng thì cho in ra màn hình, còn nếu phần tử thứ i bằng kí tự trắng thì thực hiện lệnh xuống dòng và lùi vào một khoảng cách bằng tab.*

* *Màn hình kết quả như sau:*



<u>Bài 106:</u> Viết chương trình nhập nhiều tên người vào từ bàn phím. Hãy sắp xếp lại theo thứ tự alphabet (thứ tự abc…z) và in ra màn hình kết quả đã sắp xếp theo thứ tự đó?

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void SapXep(char Xau[][100],int n);
main()
{
    char Xau[100][100],n,i;
    printf("\n  Nhap vao so luong ten nguoi:  ");
    scanf("%d",&n);
```

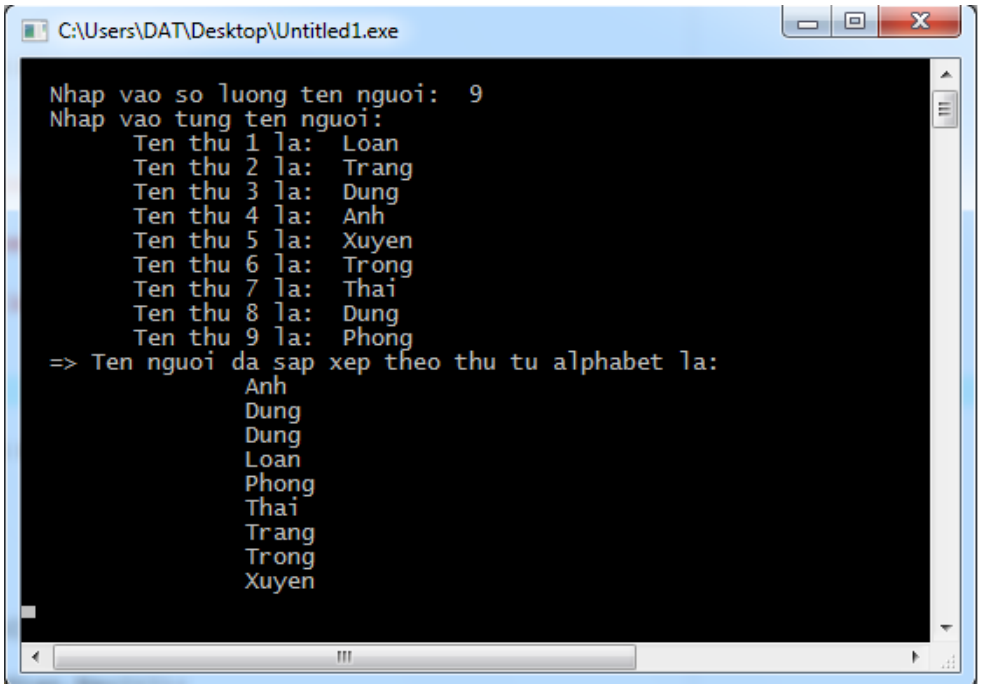
```
    printf("  Nhap vao tung ten nguoi:\n");
    for(i=1;i<=n;i++)
      {printf("\tTen thu %d la:  ",i);scanf("%s",Xau[i]);}
    SapXep(Xau,n);
   printf("  => Ten nguoi da sap xep theo thu tu alphabet la:\n");
   for(i=1;i<=n;i++)
     printf("\t\t%s\n",Xau[i]);
    getch();
    return main();
}
void SapXep(char Xau[][100],int n)
{
   int i,j;
   char XauTrungGian[100];
   for(i=1;i<n;i++)
     for(j=i+1;j<=n;j++)
       if(strcmp(Xau[i],Xau[j])>0)
         {
           strcpy(XauTrungGian,Xau[i]);
           strcpy(Xau[i],Xau[j]);
           strcpy(Xau[j],XauTrungGian);
         }
}
```

* *Giải thích:*

a) Khai báo một mảng các xâu kí tự và coi mỗi xâu kí tự là một phần tử của mảng, cung cấp sẵn một bộ nhớ chứa tối đa 100 xâu và mỗi xâu chứa tối đa 100 kí tự. Sau đó nhập vào số lượng xâu cần thiết và nhập vào từng xâu.

*b) Định nghĩa thêm hàm **SapXep()** để sắp xếp các xâu theo thứ tự alphabet (thứ tự abc), tiếp theo truyền mảng các xâu vào cho hàm. Hàm **strcmp()** là hàm so sánh 2 xâu dựa theo chỉ số của từng kí tự trong bảng mã ASCII. Nếu xâu thứ nhất lớn hơn xâu thứ hai (đứng sau xâu 2 theo thứ tự alphabet) thì thực hiện đổi vị trí hai xâu này cho nhau bằng hàm **strcpy()** và có sử dụng **XauTrungGian**. Qua hàm **SapXep()** thì không cần trả về giá trị cho hàm, sau đó in ra các xâu theo thứ tự đã sắp xếp.*

* *Màn hình kết quả như sau:*



<u>Bài 107:</u> Viết chương trình nhập vào một xâu có tối đa 80 kí tự. Thống kê xâu đó theo các thông tin sau:

- Số lượng kí tự trong xâu.

- Số lượng kí tự là nguyên âm và tỷ lệ kí tự nguyên âm tính theo %.

- Số từ trong xâu biết các từ cách nhau bởi một hoặc một nhóm kí tự trắng.

- Tiến hành chuẩn hóa xâu kí tự (loại bỏ các kí tự trắng thừa). Viết hoa kí tự đầu tiên?

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>
main()
{
   char Xau[81];
   int i,SoKiTu,KiTuNguyenAm=0,SoTu,ToanTrang=1;
   printf("\n  Nhap vao mot xau ki tu:\n\t");
   gets(Xau);
   printf("  => So luong ki tu trong xau la:  %d\n",strlen(Xau));
```

```
    for(i=0;i<strlen(Xau);i++)
      if(Xau[i]=='a'||Xau[i]=='e'||Xau[i]=='i'||Xau[i]=='o'||Xau[i]=='u')
        KiTuNguyenAm++;
    printf("  => So luong ki tu nguyen am la  %d va chiem %0.2f (phan tram)\n",KiTuNguyenAm,(float)KiTuNguyenAm/strlen(Xau)*100);
    for(i=0;i<strlen(Xau)-1;i++)
      if(Xau[i]!=' ')ToanTrang=0;
    if(ToanTrang==1)SoTu=0;
      else if(ToanTrang==0&&Xau[0]!=' ')SoTu=1;
        else SoTu=0;
    for(i=0;i<strlen(Xau)-1;i++)
      if(Xau[i]==' '&&Xau[i+1]!=' ')SoTu++;
    printf("  => So tu co trong xau la:  %d\n",SoTu);
    i=0;
    while(Xau[i]==' ')
      strcpy(Xau+i,Xau+i+1);
    SoKiTu=strlen(Xau);
    i=SoKiTu-1;
    while(Xau[i]==' ')
      {Xau[i]='\0';i--;}
    i=0;
    while(Xau[i]!='\0')
     {
       if(Xau[i]==' '&&Xau[i]==Xau[i+1]){strcpy(Xau+i,Xau+i+1);i--;}
       i++;
     }
    Xau[0]=toupper(Xau[0]);
    printf("  => Xau da chuan hoa va viet hoa ki tu dau tien la:\n\t%s\n",Xau);
    getch();
    return main();
  }
```

* *Giải thích:*

a) Nhập vào xâu có tối đa 80 kí tự thì phải dành ra 81 ô nhớ vì còn một ô nhớ để chứa kí tự kết thúc '\0'.

*Tổng số lượng kí tự có trong xâu được tính bằng hàm **strlen** (strlen = string length = độ dài xâu kí tự). Số lượng kí tự là nguyên âm thì sử dụng biến **KiTuNguyenAm** để đếm với giá trị khởi tạo bằng 0, dùng vòng lặp for với i chạy từ 0 đến n – 1 (n là tổng số kí tự của xâu). Nếu kí tự thứ i bằng 1 trong 5 kí tự nguyên âm (a, e, i, o, u) thì tăng biến KiTuNguyenAm lên một đơn vị.*

Muốn tính tỷ lệ % của các kí tự nguyên âm thì lấy giá trị của biến KiTuNguyenAm chia cho tổng số kí tự của xâu rồi nhân với 100. Phải ép kiểu số thực (float) cho giá trị của tỷ lệ. Dấu %0.2f có ý nghĩa là lấy đến 2 chữ số sau dấu phẩy ngăn cách phần nguyên và phần thập phân.

*b) Tính số từ có trong xâu (trong trường hợp xâu chưa chuẩn hóa): sử dụng biến **SoTu** nhưng tùy theo từng trường hợp mà gán giá trị ban đầu cho biến SoTu:*

+ Nếu xâu nhập vào mà toàn kí tự trắng thì gán SoTu = 0.

+ Nếu xâu nhập vào mà chứa kí tự khác trắng và kí tự đầu tiên khác trắng thì SoTu = 1.

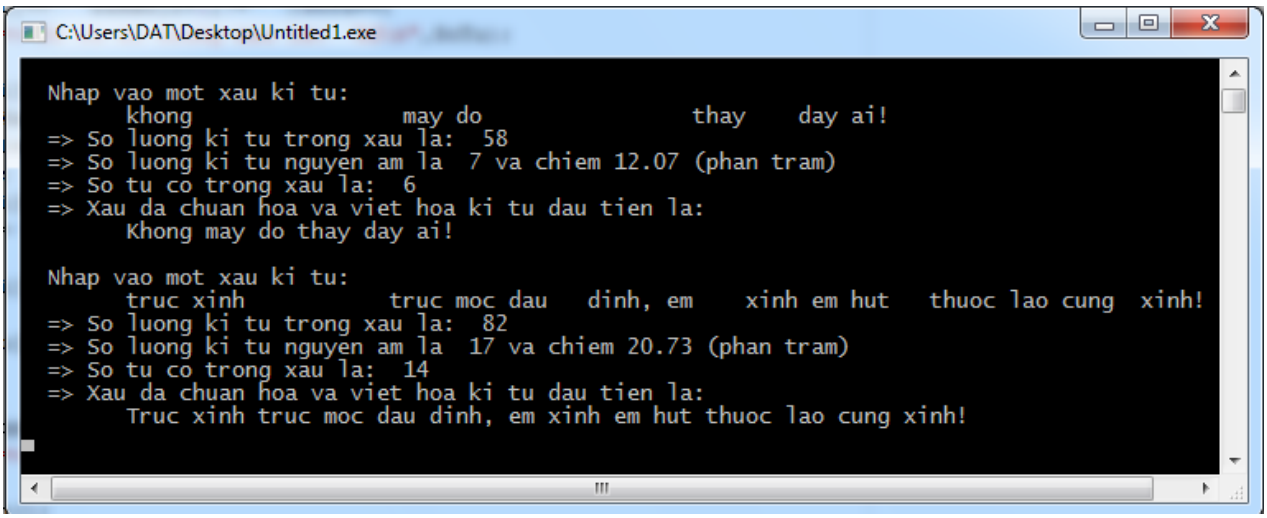
+ Nếu xâu nhập vào mà chứa kí tự khác trắng và kí tự đầu tiên bằng kí tự trắng thì gán SoTu = 0.

Khi đã biết giá trị khởi tạo ban đầu của biến SoTu thì thực hiện vòng lặp để duyệt qua các kí tự. Nếu kí tự thứ i bằng kí tự trắng và kí tự thứ i + 1 khác kí tự trắng thì tăng biến SoTu lên 1 đơn vị. Cuối cùng in ra màn hình số từ có trong xâu.

c) Tiến hành chuẩn hóa xâu kí tự: tức là cắt các kí tự trắng xuất hiện ở đầu xâu và cuối xâu, nếu bên trong xâu mà có 2 kí tự trắng cạnh nhau thì bỏ đi 1 kí tự trắng (phần này mọi người xem ở các bài tập trên, nội dung chuẩn hóa xâu kí tự nhé !). Nếu muốn kí tự đầu ưu tiên thành kí tự hoa thì vừa dùng hàm chuyển đổi chữ hoa vừa gán kí tự đã chuyển đổi cho phần tử đầu tiên của xâu. Đưa ra màn hình chuỗi đã chuẩn hóa và viết hoa kí tự đầu tiên.

Ta có thể xác định số từ của xâu bằng cách sau khi chuẩn hóa xong ta mới đếm số kí tự trắng. Lúc này, số từ của xâu bằng số kí tự trắng cộng thêm 1.

** Màn hình kết quả như sau:*



Bài 108: Nhập vào một xâu S chỉ gồm các chữ cái thường từ a -> z. Đưa ra độ dài xâu S và kiểm tra xem xâu S có phải là xâu đối xứng hay không. Hãy cho biết số lần xuất hiện của mỗi chữ cái có trong xâu?

(Đề thi cuối kỳ lớp THCS 3, ĐHKHTN – ĐHQGHN, Thứ 4 ngày 19/12/2012, Kỳ I năm học 2012 – 2013, thời gian làm bài: 30 phút)

Bài 108*: Nhập vào một xâu S có n phần tử (n ≤ 100) :

- Tính số lượng kí tự có trong xâu và đưa xâu S ra màn hình.

- Viết xâu S theo thứ tự ngược lại.

- Chuẩn hóa xâu S (Viết hoa kí tự đầu tiên và xóa kí tự trắng thừa).

(Đề thi cuối kỳ lớp THCS 3, ĐHKHTN – ĐHQGHN, Thứ 6 ngày 21/12/2012, Kỳ I năm học 2012 – 2013, thời gian làm bài: 40 phút)

CHƯƠNG VII. KIỂU DỮ LIỆU CẤU TRÚC

Bài 108: Nhập vào 2 phân số. In ra tổng, hiệu, tích, thương của 2 phân số đó?

```
#include <stdio.h>
#include <conio.h>
struct phanso{int TuSo;int MauSo;};
main()
{
    float Tong,Hieu,Tich,Thuong;
    struct phanso P1,P2;
    printf("\n  Nhap vao tu so va mau so cua phan so 1 (mau khac 0):  ");
      nhaplai1:scanf("%d%d",&P1.TuSo,&P1.MauSo);
      if(P1.MauSo==0){printf("  Nhap lai phan so 1:  ");goto nhaplai1;}
    printf("  Nhap vao tu so va mau so cua phan so 2 (mau khac 0):  ");
      nhaplai2:scanf("%d%d",&P2.TuSo,&P2.MauSo);
```

```
        if(P2.MauSo==0){printf("  Nhap lai phan so 2:  ");goto nhaplai2;}

    Tong=(float)(P1.TuSo*P2.MauSo+P2.TuSo*P1.MauSo)/(P1.MauSo*P2.MauSo);
      printf("  => Tong hai phan so la:  %f\n",Tong);
    Hieu=(float)(P1.TuSo*P2.MauSo-
P2.TuSo*P1.MauSo)/(P1.MauSo*P2.MauSo);
      printf("  => Hieu hai phan so la:  %f\n",Hieu);
    Tich=(float)(P1.TuSo*P2.TuSo)/(P1.MauSo*P2.MauSo);
      printf("  => Tich hai phan so la:  %f\n",Tich);
    if(P2.TuSo==0)printf("  => Khong thuc hien duoc phep chia!\n");
    else
      {
       Thuong=(float)(P1.TuSo*P2.MauSo)/(P1.MauSo*P2.TuSo);
       printf("  => Thuong hai phan so la:  %f\n",Thuong);
      }
    getch();
    return main();
}
```

* *Giải thích:*

*- Xây dựng một cấu trúc **phanso** gồm có 2 thành phần là **TuSo** và **MauSo** và cả hai thành phần này đều có kiểu số nguyên. Sau đó khai báo hai biến **P1**, **P2** có kiểu dữ liệu phanso, còn các biến **Tong**, **Hieu**, **Tích**, **Thuong** dùng để tính tổng, hiệu, tích, thương của hai phân số P1, P2. Phân số chỉ xác định khi mẫu số khác 0 nên nếu trường hợp nhập vào mà mẫu = 0 thì phải nhập lại, dùng lệnh nhảy **goto**.*

a) Tổng của hai phân số: Quy đồng 2 phân số P1, P2 sao cho cùng mẫu rồi cộng tử số với nhau và giữ nguyên mẫu số. Giả sử 2 phân số có dạng $\frac{a}{b}$ *và* $\frac{c}{d}$ *thì tổng của chúng là:* $\frac{ad+cb}{bd}$

Tương ứng với kiểu dữ liệu cấu trúc thì 2 phân số có dạng $\frac{P_1.TuSo}{P_1.MauSo}$ *và* $\frac{P_2.TuSo}{P_2.MauSo}$ *thì tổng của chúng sẽ là:* $\frac{P_1.TuSo*P_2.MauSo+P_2.TuSo*P_1.MauSo}{P_1.MauSo*P_2.MauSo}$*. Chú ý là giá trị trả về thường là số thực nên phải ép kiểu **(float)** cho nó.*

b) Tương tự câu a, hiệu của hai phân số sẽ là:

$$\frac{P_1.TuSo * P_2.MauSo - P_2.TuSo * P_1.MauSo}{P_1.MauSo * P_2.MauSo}$$

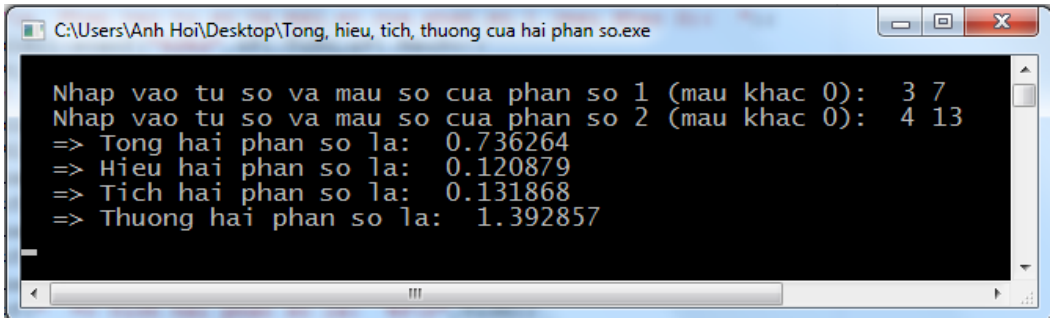
c) Tích của hai phấn số bằng:

$$\frac{P_1.TuSo * P_2.TuSo}{P_1.MauSo * P_2.MauSo}$$

d) Thương của hai phân số chỉ xác đinh khi phân số thứ hai khác 0. Hay nói cách khác là tử và mẫu của phân số thứ hai phải khác 0 thì mới thực hiện được phép chia phân số P_1 cho phân số P_2. Nếu phân số thứ hai khác 0 thì phép chia được thực hiện bằng cách lấy tích của phân số thứ nhất với nghịch đảo của phân số thứ 2. Biểu thức tính như sau:

$$\frac{P_1.TuSo * P_2.MauSo}{P_1.MauSo * P_2.TuSo}$$

* *Màn hình kết quả như sau:*



Bài 108: Hãy xây dựng một cấu trúc số phức? Nhập vào n số phức và in ra màn hình các số phức đó? (X)

```
#include <stdio.h>
#include <conio.h>
struct SoPhuc {float Thuc; float Ao;};
main()
{
    int n,i;
    struct SoPhuc a[100];
    printf("\n  Nhap vao so luong cac so phuc:  ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
      {
```

```
            printf("  Nhap vao so phuc thu %d:\n",i+1);
            printf("\tNhap phan thuc:  ");scanf("%f",&a[i].Thuc);
            printf("\tNhap phan ao:  ");scanf("%f",&a[i].Ao);
          }
        printf("\n  => Cac so phuc vua nhap la:\n");
        for(i=0;i<n;i++)
          printf("\tSo phuc thu %d:  %.0f + %.0f.i\n",i+1,a[i].Thuc, a[i].Ao);
        printf("\n\t    (Trong do i^2 = -1)\n");
        getch();
        return main();
    }
```

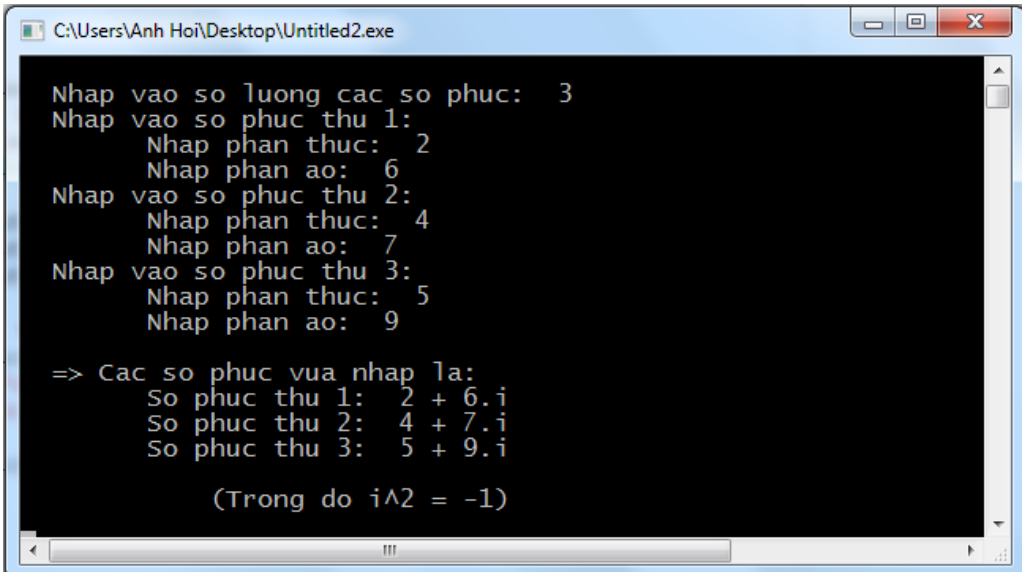
* *Giải thích:*

a) Xây dựng một cấu trúc số phức:

Tất cả các biểu thức có dạng a + bi trong đó a, b $\in R$ và $i^2 = -1$ được gọi là một số phức. Xuất phát từ định nghĩa trên, ta thấy số phức gồm 2 phần: a là phần thực và b là phần ảo. Chính vì vậy số phức có kiểu cấu trúc và gồm hai thành phần; thành phần thứ nhất đại diện cho phần thực, thành phần thứ hai đại diện cho phần ảo và cả 2 thành phần đều có kiểu số thực ***float****.*

b) Khai báo mảng a gồm tối đa 100 biến có kiểu số phức ***SoPhuc*** *sau đó nhập số lượng các số phức cần thiết và lưu vào cho biến n. Tiếp theo nhập từng thành phần cho từng số phức, dùng vòng lặp* ***for()****. Lưu ý cách truy cập đến từng thành phần của biến có kiểu cấu trúc bằng cách sử dụng dấu chấm ngăn cách tên biến với các thành phần con tương ứng. Sau khi nhập xong các số phức thì có thể thực hiện các phép toán cộng, trừ, nhân, chia số phức nếu đề bài yêu cầu.*

* *Màn hình kết quả như sau:*



<u>Bài 109:</u> Nhập vào 2 số phức z_1 và z_2 .Tính tổng, hiệu, tích, thương của 2 số phức đó?

```
#include <stdio.h>
#include <conio.h>
struct SoPhuc {float Thuc; float Ao;};
main()
{
   struct SoPhuc z1,z2,z3,z4,z5,z6;
   printf("\n  Nhap phan thuc va phan ao cua so phuc thu nhat:  ");
     scanf("%f%f",&z1.Thuc,&z1.Ao);
   printf("  Nhap phan thuc va phan ao cua so phuc thu hai:  ");
     scanf("%f%f",&z2.Thuc,&z2.Ao);
   printf("\n  So phuc z3 la tong cua hai so phuc z1 va z2:\n");
     z3.Thuc=z1.Thuc+z2.Thuc;
     z3.Ao=z1.Ao+z2.Ao;
     printf("  => So phuc z3 la:  z3 = %.0f + %.0f.i\n",z3.Thuc,z3.Ao);
   printf("\n  So phuc z4 la hieu cua hai so phuc z1 va z2:\n");
     z4.Thuc=z1.Thuc-z2.Thuc;
     z4.Ao=z1.Ao-z2.Ao;
     printf("  => So phuc z4 la:  z4 = %.0f + %.0f.i\n",z4.Thuc,z4.Ao);
   printf("\n  So phuc z5 la tich cua hai so phuc z1 va z2:\n");
     z5.Thuc=z1.Thuc*z2.Thuc-z1.Ao*z2.Ao;
     z5.Ao=z1.Thuc*z2.Ao+z1.Ao*z2.Thuc;
     printf("  => So phuc z5 la:  z5 = %.0f + %.0f.i\n",z5.Thuc,z5.Ao);
   printf("\n  So phuc z6 la thuong cua hai so phuc z1 va z2:\n");

z6.Thuc=(z1.Thuc*z2.Thuc+z1.Ao*z2.Ao)/(z2.Thuc*z2.Thuc+z2.Ao*z2.Ao);
     z6.Ao=(z1.Ao*z2.Thuc-
z1.Thuc*z2.Ao)/(z2.Thuc*z2.Thuc+z2.Ao*z2.Ao);
     printf("  => So phuc z6 la:  z6 = %f + %f i\n",z6.Thuc,z6.Ao);
   printf("\n\t\t(Trong do i^2 = -1)\n");
   getch();
```

```
    return main();
}
```

* <u>*Giải thích:*</u>

*- Xây dựng một cấu trúc số phức **SoPhuc** có 2 thành phần là **Thuc** và **Ao**, cả hai thành phần này đều có kiểu số thực. Sau đó, khai báo các biến z_1, z_2, z_3, z_4 ,z_5, z_6 có kiểu cấu trúc SoPhuc trong đó các biến z_3, z_4 ,z_5, z_6 là để tính tổng, hiệu, tích, thương của hai số phức z_1 và z_2. Thực hiện nhập từng thành phần của từng biến z_1, z_2 và lưu vào địa chỉ tương ứng.*

a) Muốn tìm số phức z_3 là tổng của 2 số phức z_1 và z_2 thì phần thực của số phức z_3 là tổng hai phần thực của số phức z_1 và z_2 và phần ảo của số phức z_3 cũng là tổng hai phần ảo của z_1 và z_2.

Giả sử $z_1 = a + b.i$ và $z_2 = c + d.i$ thì:

$$z_3 = z_1 + z_2 = (a + c) + (b + d).i$$

b) Muốn tìm số phức z_4 là hiệu của hai số phức z_1 và z_2 thì:

$$z_4 = z_1 - z_2 = (a - c) + (b - d).i$$

c) Muốn tìm số phức z_5 là tích của 2 số phức z_1 và z_2 thì phần thực của số phức z_5 là (ac – bd) và phần ảo của số phức z_5 là (ad + bc)

$$\begin{aligned} z_5 = z_1.z_2 &= (a + b.i)(c + d.i) \\ &= ac + ad.i + bc.i + bd.i^2 \\ &= ac + (ad + bc).i - bd \qquad (\text{vì } i^2 = -1) \\ &= (ac - bd) + (ad + bc).i \end{aligned}$$

Thực hiện tính từng biểu thức (ac – bd) và (ad + bc) và gán cho từng thành phần của số phức z_5.

d) Số phức z_6 là thương của số phức z_1 và z_2 thì z_6 được tính bằng cách nhân cả tử và mẫu với số phức liên hợp của số phức z_2:

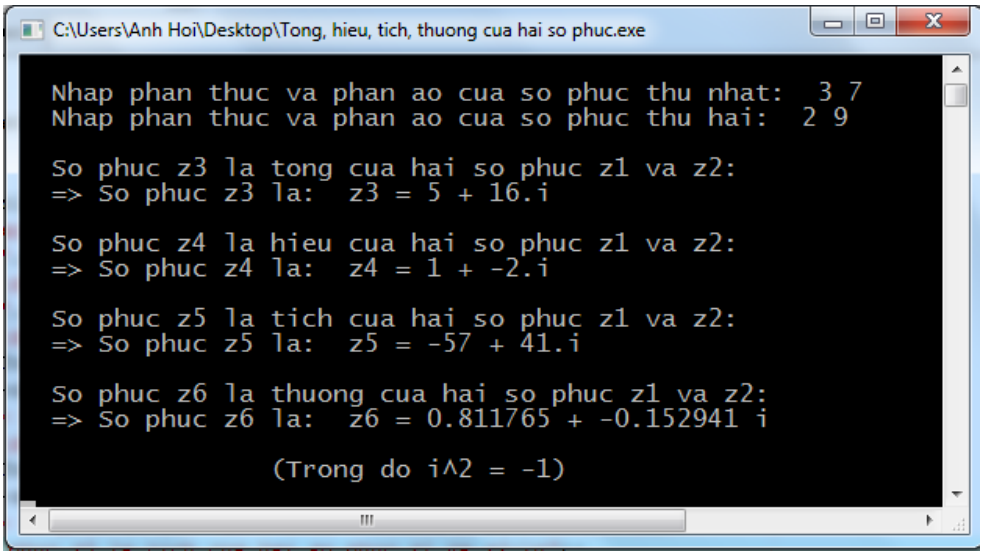
$$z_6 = \frac{a+bi}{c+di} = \frac{(a+bi)(c-di)}{(c+di)(c-di)} = \frac{ac-adi+bci-bdi^2}{c^2-d^2i^2} = \frac{(ac+bd)+(bc-ad).i}{c^2+d^2}$$

(vì $i^2 = -1$)

$$\Leftrightarrow z_6 = \frac{(ac+bd)}{c^2+d^2} + \frac{(bc-ad)}{c^2+d^2}.i$$

Sau đó, tính riêng phần thực và phần ảo của số phức z_6.

* <u>*Màn hình kết quả như sau:*</u>



<u>Bài 110:</u> Viết chương trình xây dựng một cấu trúc *ths* (thí sinh) gồm các thành phần sau: *Họ tên; năm sinh; số báo danh.* Nhập số liệu từ bàn phím cho các thành phần của một biến *sv* có kiểu cấu trúc *ths*. In ra màn hình thông tin của *sv* vừa nhập? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
    struct ths{char HoTen[100];int NamSinh;int SoBaoDanh;}sv;
    printf("\n  Nhap vao Ho va Ten:  ");
    gets(sv.HoTen);
    printf("  Nhap vao Nam sinh:  ");
    scanf("%d",&sv.NamSinh);
    printf("  Nhap vao So bao danh:  ");
    scanf("%d",&sv.SoBaoDanh);
    printf("  => Thong tin cua sinh vien vua nhap la:\n\tHo va Ten:\t\tNam
sinh:\tSo bao danh:\n");
    printf("\t%s\t\t%d\t\t%d\n",sv.HoTen,sv.NamSinh,sv.SoBaoDanh);
    fflush(stdin);
    getch();
    return main();
}
```

* *Giải thích:*

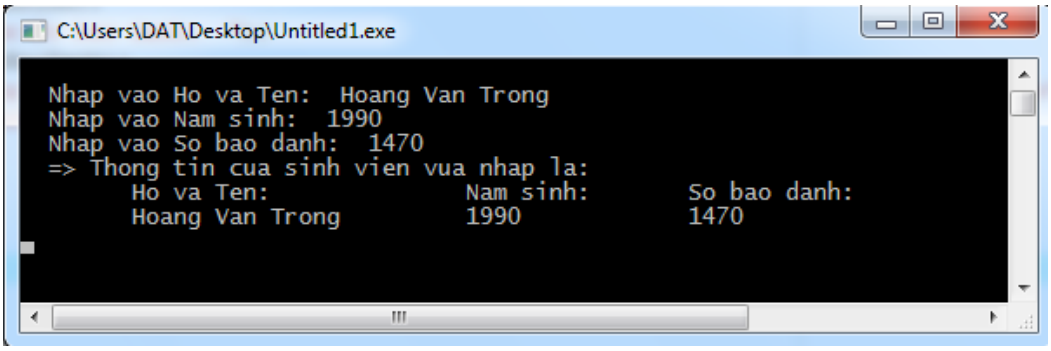
*a) Biến **sv** là một biến có kiểu cấu trúc **struct**, gồm 3 thành phần: **HoTen** có kiểu xâu kí tự, **NamSinh** có kiểu nguyên và **SoBaoDanh** có kiểu nguyên. Trong trường hợp số báo danh mà có cả phần chữ và phần số thì phải khai báo biến SoBaoDanh có kiểu struct và lúc này ta có dạng khai báo các kiểu cấu trúc lồng nhau.*

b) Truy nhập vào các phần tử hoặc các thành phần của biến sv có kiểu struct bao gồm tên biến sau đó đến dấu chấm và đến thành phần cấp nhỏ hơn. Ví dụ, truy nhập vào kí tự thứ 3 trong họ tên của biến sv là: sv.HoTen[2]

Từ đó, áp dụng để nhập và đọc các thành phần của biến. Phần định dạng kiểu khi xuất ra màn hình thì tương ứng với từng phần tử hoặc từng thành phần của biến.

*- Lệnh **fflush(stdin)** có tác dụng làm sạch bộ nhớ đệm bàn phím để không làm ảnh hưởng đến kết quả của lần nhập tiếp theo.*

* *Màn hình kết quả như sau:*



Bài 111: Xây dựng một mảng cấu trúc có kiểu *ths* bao gồm các thành phần: *Họ tên, năm sinh, số báo danh*; với số phần tử của mảng là 5. Nhập vào thông tin 5 phần tử của mảng. In ra thông tin của các phần tử trong mảng vừa nhập? (X)

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i;
    struct ths{char HoTen[100];int NamSinh;int SoBaoDanh;}sv[5];
    printf("\n");
    for(i=0;i<5;i++)
      {
        printf("  Nhap vao Ho ten, Nam sinh va So bao danh cua nguoi thu %d:\n\t",i+1);
```

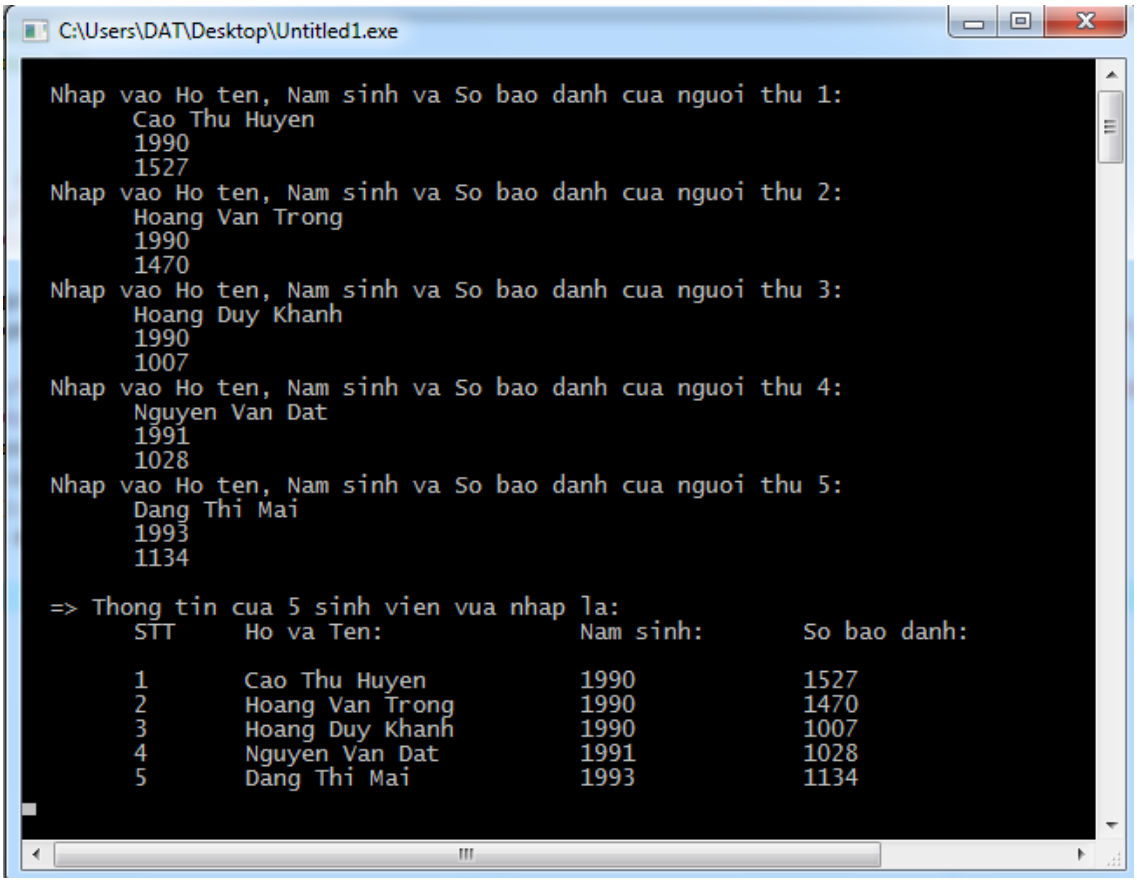
```
gets(sv[i].HoTen);printf("\t");scanf("%d",&sv[i].NamSinh);printf("\t");scanf("
%d",&sv[i].SoBaoDanh);fflush(stdin);
      }
    printf("\n  => Thong tin cua 5 sinh vien vua nhap la:\n\tSTT\tHo va
Ten:\t\tNam sinh:\tSo bao danh:\n\n");
    for(i=0;i<5;i++)

printf("\t%d\t%s\t\t%d\t\t%d\n",i+1,sv[i].HoTen,sv[i].NamSinh,sv[i].SoBaoDa
nh);
    fflush(stdin);
    getch();
    return main();
}
```

* *Giải thích:*

*a) Đầu tiên, khai báo một mảng chứa tối đa 5 biến có cùng kiểu cấu trúc **struct**. Vòng lặp **for** với i chạy từ 1 đến 5 và duyệt từ biến có kiểu cấu trúc thứ nhất đến biến có kiểu cấu trúc thứ 5, duyệt đến biến cấu trúc nào thì thực hiện nhập và đọc các thành phần tương ứng với biến cấu trúc đó. Khi nhập xong mỗi biến cấu trúc thì dùng lệnh **fflush(stdin)** xóa bộ nhớ đệm để vòng for chuyển sang biến cấu trúc tiếp theo (nhất thiết phải dùng lệnh này).*

b) In ra thông tin của 5 sinh viên nhập vào: Cũng dùng vòng for để duyệt từng biến trong 5 biến cấu trúc. Duyệt tới biến nào thì cho hiện ra màn hình các thành phần của biến đó. Mọi người lưu ý là truy cập tới một thành phần của mảng chứa các biến cấu trúc thì tương tự cách truy cập đến từng phần tử của mảng 1 chiều, chỉ khác là phần đuôi có thêm các cấp bậc thấp hơn.

* *Màn hình kết quả như sau:*



<u>Bài 112:</u> Xây dựng một mảng cấu trúc có n phần tử *hs* (học sinh) gồm các thành phần sau: *Họ tên, ntn, quê quán*. Trong đó, thành phần *ntn* là một cấu trúc gồm 3 thành phần: *Năm sinh, tháng sinh và ngày sinh.*

Lập hàm nhập thông tin cho các phần tử của mảng. Nhập vào năm sinh bất kì, tìm và in ra những thí sinh có trùng năm sinh với năm vừa nhập, nếu không có thì in ra dòng chữ: "*Không tìm thấy thông tin*". (X)

```
#include <stdio.h>
#include <conio.h>
struct hs{char HoTen[100];struct haha{int NamSinh;int ThangSinh;int
NgaySinh;}ntn;char QueQuan[100];};
void NhapThongTin(struct hs *HS, int i);
main()
{
    int i,n,y,Dem=0;
    struct hs HocSinh[100];
    printf("\n  Nhap vao tong so hoc sinh:  ");
```

```
    scanf("%d",&n);
    fflush(stdin);
    for(i=0;i<n;i++)
      NhapThongTin(&HocSinh[i],i);
    printf("  Nhap vao nam sinh bat ki:  ");
    scanf("%d",&y);
    printf("  => Nhung thi sinh co trung nam sinh voi nam vua nhap la:\n\t");
    for(i=0;i<n;i++)
      if(HocSinh[i].ntn.NamSinh==y)
        {
          printf("%s\n\t",HocSinh[i].HoTen);
          Dem++;
        }
    if(Dem==0)printf("\r\tKhong tim thay thong tin!\n");
    fflush(stdin);
    getch();
    return main();
}
void NhapThongTin(struct hs *HS, int i)
{
    printf("  Nhap vao thong tin cua sinh vien thu %d:\n",i+1);
    printf("\tHo Ten:  ");gets(HS->HoTen);fflush(stdin);
    printf("\tNam Sinh:  ");scanf("%d",&(*HS).ntn.NamSinh);fflush(stdin);
    printf("\tThang Sinh:  ");scanf("%d",&(*HS).ntn.ThangSinh);fflush(stdin);
    printf("\tNgay Sinh:  ");scanf("%d",&(*HS).ntn.NgaySinh);fflush(stdin);
    printf("\tQue Quan:  ");gets(HS->QueQuan);fflush(stdin);
}
```

* *Giải thích:*

*a) Khai báo sẵn một mảng gồm 100 học sinh trong đó mỗi một biến học sinh **HocSinh** là một kiểu cấu trúc **hs** gồm **HoTen**, **ntn** và **QueQuan**; trong đó thành phần **ntn** lại là một kiểu cấu trúc **haha**. Vì vậy ta phải định nghĩa các kiểu cấu trúc lồng nhau. Kiểu dữ liệu cấu trúc **hs** ta có thể khai báo prototype trước rồi sau đó sử dụng được cả trong hàm chính lẫn hàm được định nghĩa thêm.*

b) Hàm nhập thông tin cho biến HocSinh: Vì khi truyền biến cấu trúc vào cho hàm thì có nghĩa là ta truyền tham trị cho nó, hay nói cách khác là mọi thay đổi trong hàm được định nghĩa thêm thì khi ra khỏi hàm sẽ không còn tác dụng. Vì vậy để nhập và lưu được thông tin vào cho từng biến HocSinh ta phải truyền địa chỉ của từng biến vào cho hàm ***NhapThongTin()****. Tham số hình thức tương ứng trong hàm nhập thông tin phải khai báo dạng con trỏ.*

+ Nhập thông tin cho thành phần HoTen và QueQuan: vì các thành phần này là một xâu kí tự và có thể chứa những kí tự đặc biệt nên phải dùng lệnh đọc từ bàn phím là gets(). Dấu -> có ý nghĩa là con trỏ HS trỏ tới thành phần HoTen và thành phần QueQuan, tương tự như dùng dấu chấm khi không làm việc với con trỏ.

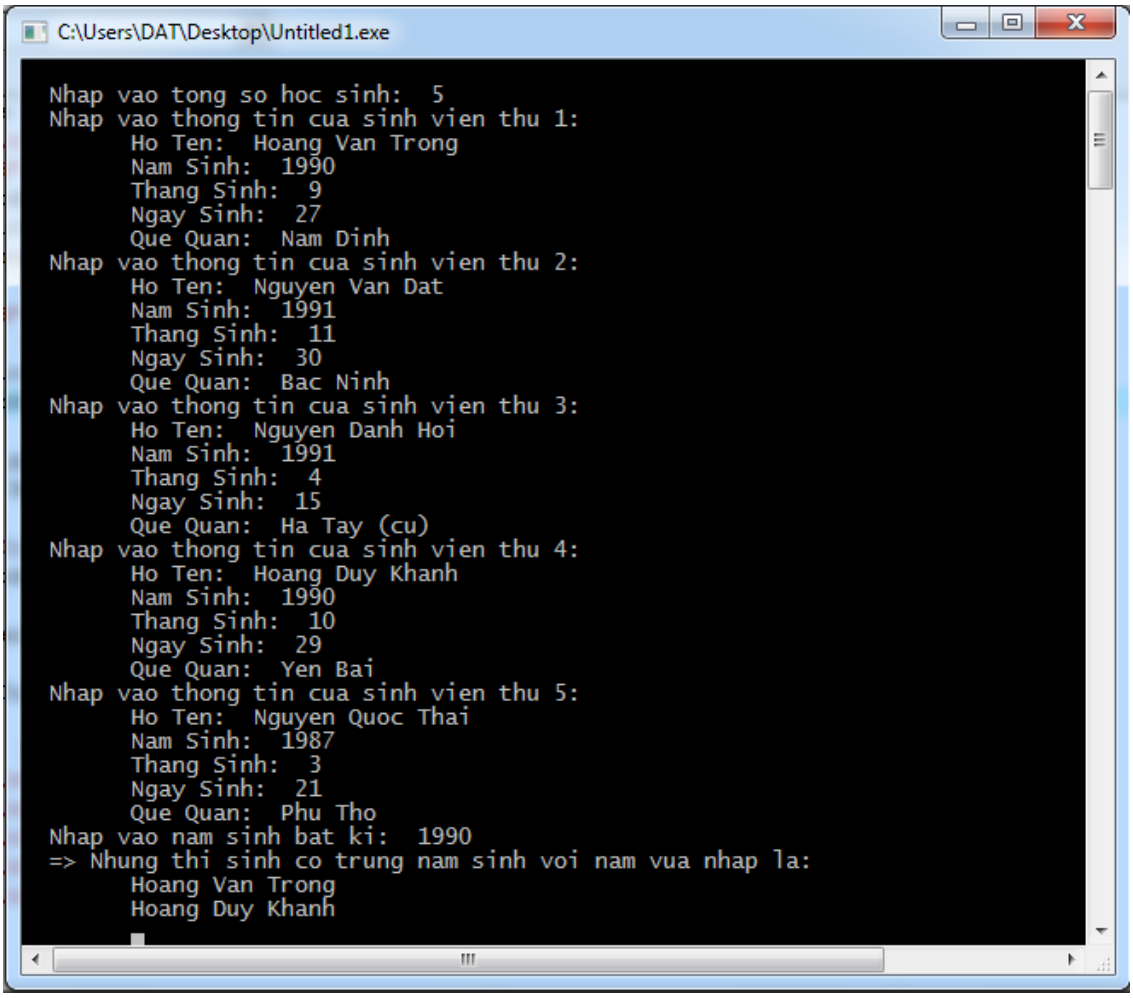
+ Nhập thông tin cho thành phần NamSinh, ThangSinh, NgaySinh: các thành phần này đều có kiểu số nguyên và là thành phần con của thành phần ntn. Trong trường hợp này dùng lệnh scanf() để đọc từ bàn phím và lưu vào địa chỉ của giá trị mà con trỏ trỏ tới.

c) Sau khi nhập xong thông tin của tất cả các biến HocSinh thì nhập tiếp vào một năm bất kì và lưu vào địa chỉ cho biến y. Duyệt từng biến HocSinh từ biến thứ nhất là HocSinh[0] đến biến cuối cùng là HocSinh[n-1], nếu thành phần NamSinh của biến nào mà bằng y thì in ra màn hình thành phần HoTen của biến đó cũng như tăng biến ***Dem*** *lên 1 đơn vị.*

Nếu thấy Dem = 0 thì xuất ra màn hình dòng chữ "Không tìm thấy thông tin"

d) Ở trong bài này mình sử dùng khá nhiều lệnh xóa bộ nhớ đệm bàn phím ***fflush(stdin).*** *Nếu không dùng lệnh này thì các kí tự đặc biệt sẽ lưu lại cho thông tin của biến HocSinh tiếp theo, như thế kết quả sẽ không chính xác cũng như việc nhập không như ta mong muốn.*

** <u>Màn hình kết quả như sau</u>:*



<u>Bài 113:</u> Bài toán quản lý sinh viên:

- Nhập vào một danh sách sinh viên bao gồm các thông tin: Họ tên, điểm Toán, điểm Tin học. Tính và thêm thông tin điểm trung bình vào danh sách sinh viên.

- Sắp xếp và in danh sách thông tin của sinh viên theo thứ tự điểm trung bình giảm dần? (X)

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
struct SinhVien {char HoTen[100];float DiemToan;float DiemTin; float
DiemTB;};
main()
{
   int n,i,j;
```

```
    struct SinhVien SV[100],TrungGian;
    printf("\n  Nhap vao so luong sinh vien:  ");
    scanf("%d",&n);
    fflush(stdin);
    printf("  Nhap vao thong tin cho tung sinh vien:\n");
    for(i=0;i<n;i++)
      {
        printf("  Nhap thong tin cho sinh vien thu %d:\n",i+1);
        printf("\tHo va Ten:  ");gets(SV[i].HoTen);fflush(stdin);
        printf("\tDiem Toan la:  ");scanf("%f",&SV[i].DiemToan);
        printf("\tDiem Tin la:  ");scanf("%f",&SV[i].DiemTin);
        SV[i].DiemTB=(SV[i].DiemToan+SV[i].DiemTin)/2;
        fflush(stdin);
      }
    for(i=0;i<n-1;i++)
      for(j=i+1;j<n;j++)
        if(SV[i].DiemTB<SV[j].DiemTB)
          {
            TrungGian=SV[i];
            SV[i]=SV[j];
            SV[j]=TrungGian;
          }
    printf("  => Danh sach sinh vien theo thu tu diem trung binh giam dan
la:\n\n");
    for(i=0;i<n;i++)
      printf("\t%s\t%.1f\n",SV[i].HoTen,SV[i].DiemTB);
    getch();
    return main();
}
```

***** *Giải thích:*

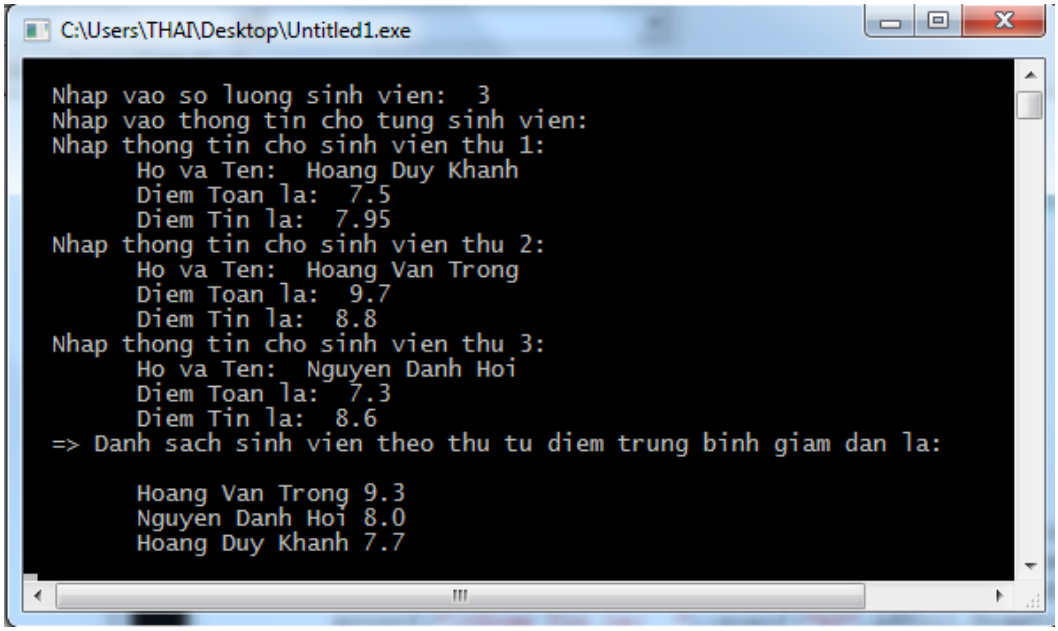
- Xây dựng một kiểu cấu trúc dạng ***SinhVien*** *gồm 4 thành phần:* ***HoTen*** *có kiểu xâu kí tự;* ***DiemToan****,* ***DiemTin*** *và* ***DiemTB*** *có kiểu số thực float. Trong chương trình chính, ta khai báo*

*một mảng gồm tối đa 100 biến **SV** có kiểu SinhVien. Thực hiện nhập vào số lượng biến SV cần thiết và gán giá trị cho n.*

a) Nhập vào các thành phần DiemToan và DiemTin cho từng biến SV[i] với giá trị i chạy từ 0 đến n – 1. Nhập xong thông tin của biến SV[i] nào thì tính luôn thành phần DiemTB của biến SV[i] đó bằng cách lấy trung bình cộng của 2 thành phần vừa nhập.

*b) Sắp xếp các biến SV[i] theo chiều có DiemTB giảm dần: Dùng 2 vòng for lồng nhau với biến i chạy từ 0 đến n – 2 và biến j chạy từ i + 1 đến n – 1. Nếu biến SV[i] mà có DiemTB nhỏ hơn DiemTB của biến SV[j] thì thực hiện đổi chỗ cho nhau nhờ biến **TrungGian** (thuật toán này tương tự như thuật toán sắp xếp dãy số theo chiều giảm dần). Cuối cùng in kết quả ra màn hình gồm 2 thành phần là HoTen và DiemTB của các biến SV[i].*

* *Màn hình kết quả như sau:*



<u>Bài 114:</u> Bài toán quản lý thư viện:

- Nhập vào thông tin các sách trong thư viện: Tên sách, tác giả, thể loại, năm xuất bản.

- In ra danh sách các sách thuộc thể loại văn học.

- Tìm kiếm sách của tác giả "Lê Lựu" xuất bản vào những năm 90's.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
struct ThongTin {char TenSach[100];char TacGia[100];char TheLoai[100];int NamXB;};
```

```
main()
{
    int i,n;
    struct ThongTin Sach[100];
    printf("\n  Nhap vao so luong sach:  ");
    scanf("%d",&n);
    printf("  Nhap vao thong tin cho tung quyen sach:\n");
    fflush(stdin);
    for(i=0;i<n;i++)
      {
        printf("  Nhap thong tin cho quyen sach thu %d:\n",i+1);
        printf("\tTen sach:  ");gets(Sach[i].TenSach);fflush(stdin);
        printf("\tTac gia:  ");gets(Sach[i].TacGia);fflush(stdin);
        printf("\tThe loai:  ");gets(Sach[i].TheLoai);fflush(stdin);
        printf("\tNam xuat ban: ");scanf("%d",&Sach[i].NamXB);
fflush(stdin);
      }
    printf("\n  => Cac sach thuoc the loai van hoc la:\n\t");
    for(i=0;i<n;i++)
      if(strcmp(Sach[i].TheLoai,"Van hoc")==0)
        printf("%s\n\t",Sach[i].TenSach);
    printf("\n  => Sach cua Le Luu xuat ban vao nhung nam 90's la:\n\t");
    for(i=0;i<n;i++)
      if(strcmp(Sach[i].TacGia,"Le
Luu")==0&&Sach[i].NamXB>=1990&&Sach[i].NamXB<=1999)
        printf("%s\n\t",Sach[i].TenSach);
    getch();
}
```

* *Giải thích:*

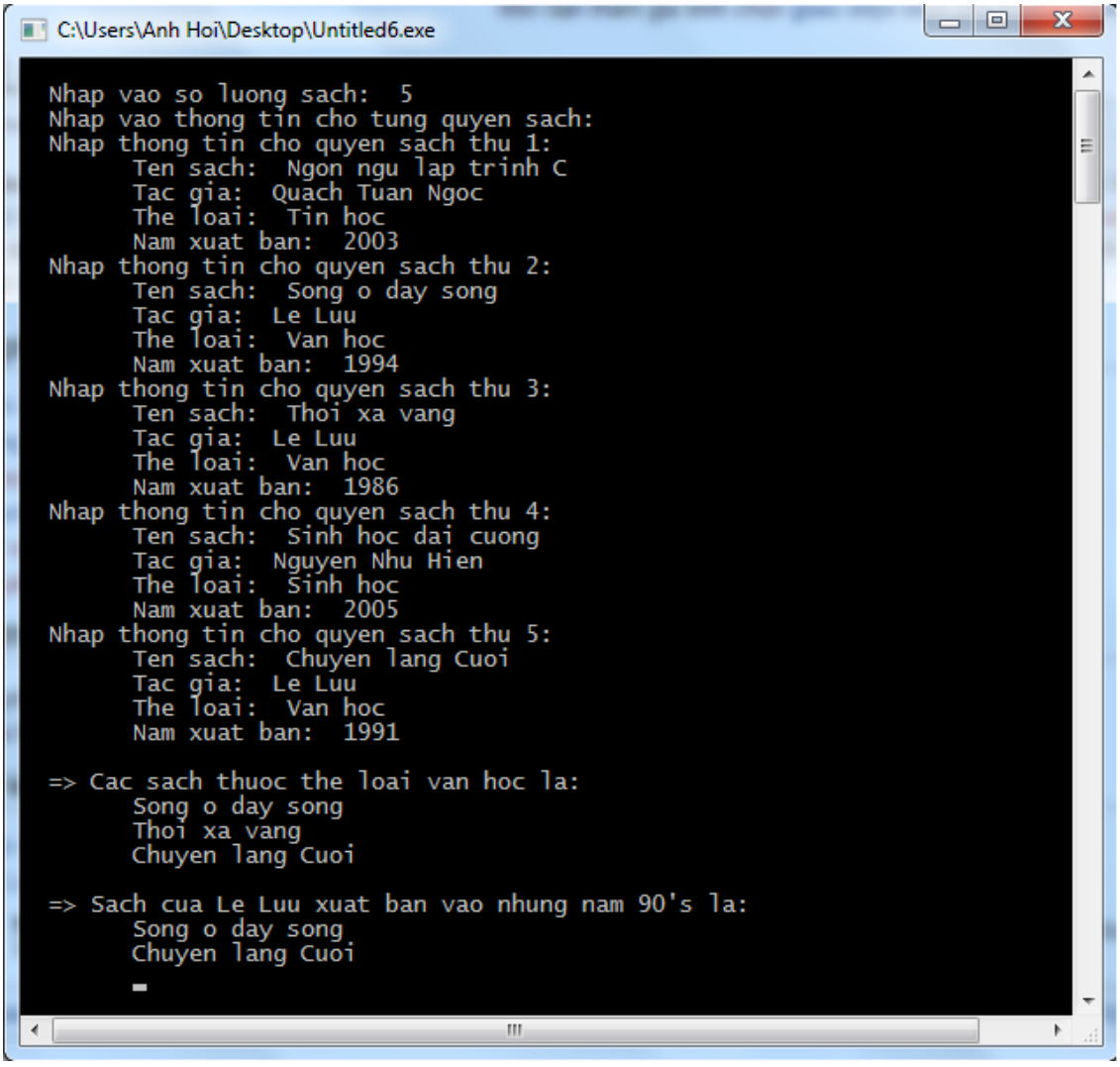
*a) Khai báo một kiểu cấu trúc dạng **ThongTin** gồm 4 thành phần: **TenSach**, **TacGia**, **TheLoai** có kiểu xâu kí tự và **NamXB** có kiểu số nguyên. Trong chương trình chính, ta khai báo một mảng gồm tối đa 100 biến **Sach** có kiểu ThongTin và tùy chọn nhập vào số lượng sách cần thiết theo ý muốn của mỗi người. Vòng lặp for với biến i chạy từ 0 đến n – 1 (n là số lượng sách).*

Biến i chạy đến đầu thì truy cập tới từng thành phần của Sach[i] và nhập thông tin cho các thành phần đó.

*b) In ra các sách thuộc thể loại văn học: dùng hàm **strcmp()** để so sánh chuỗi kí tự **"Van hoc"** với thành phần TheLoai của biến Sach[i], nếu thấy chúng bằng nhau (giá trị của hàm = 0) thì in ra màn hình TenSach của biến Sach[i].*

*c) In ra các sách của Lê Hựu được xuất bản vào những năm 90's (tức là những năm từ 1990 đến 1999): Nếu biến Sach[i] nào thỏa mãn 3 điều kiện là Sach[i].TacGia = **"Le Luu"** và Sach[i].NamXB ≥ 1990 và Sach[i].NamXB ≤ 1999 thì in ra màn hình TenSach.*

* *Màn hình kết quả như sau:*



CHƯƠNG VIII. KIỂU FILE

Bài 115: Tạo ra một tệp gồm tên và điểm của các sinh viên trong lớp, với tên và điểm được nhập từ bàn phím?

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i,n;
    FILE *f;
    struct hihihi {char Ten[100];float Diem;}SinhVien[100];
    printf("\n  Nhap vao so luong sinh vien:  ");
    scanf("%d",&n);
    f=fopen("hoangtronghus.doc","w");
    fflush(stdin);
    for(i=0;i<n;i++)
      {
        printf("  Nhap vao Ten va Diem cua sinh vien thu %d:\n\t",i+1);

gets(SinhVien[i].Ten);printf("\t");scanf("%f",&SinhVien[i].Diem);printf("\r");
        fprintf(f,"%s\t\t%0.2f\n",SinhVien[i].Ten,SinhVien[i].Diem);
        fflush(stdin);
      }
    fclose(f);
    printf("\n   => Ban hay vao thu muc luu file .c nay va mo file
hoangtronghus.doc len de xem ket qua.Thank!");
    getch();
}
```

* *Giải thích:*

- Đây là bài tập tương đối tổng hợp nhiều loại kiến thức: vòng lặp; mảng và con trỏ; kiểu dữ liệu cấu trúc; mở, đóng và ghi dữ liệu vào file,...

*a) Khai báo: n là tổng số sinh viên mà chúng ta sẽ nhập vào, i là biến chạy trong vòng lặp for để nhập và ghi thông tin của từng sinh viên vào file, con trỏ *f có kiểu FILE để thực hiện các thao tác với file dữ liệu.*

*b) Mỗi một biến **SinhVien** là một kiểu cấu trúc gồm 2 thành phần là **Ten** có kiểu kí tự và **Diem** có kiểu số thực. Nhưng lại có nhiều kiểu cấu trúc dựa theo số lượng các thành phần cũng như sự kết hợp giữa các thành phần ấy, cho nên mình lại định nghĩa các biến SinhVien có kiểu **hihihi**. Hay nói cách khác, kiểu hihihi cũng là kiểu cấu trúc nhưng chỉ là trường hợp riêng của kiểu cấu trúc. Khai báo sẵn một mảng gồm tối đa 100 biến SinhVien có kiểu hihihi (cấp phát tĩnh một vùng nhớ chứa tối đa 100 sinh viên có kiểu hihihi).*

c) Lệnh f=fopen("hoangtronghus.doc","w") là lệnh dùng con trỏ f để mở và ghi một file mới có tên là hoangtronghus và có định dạng là document (định dạng này dùng Microsoft Word để mở).

*Lệnh **fflush(stdin)** là lệnh xóa bộ nhớ đệm liên quan đến bàn phím. Vì hàm **gets()** có thể đọc được các kí tự đặc biệt như dấu cách, kí tự enter nên trước khi nhập thông tin cho sinh viên thì ta phải xóa những kí tự không cần thiết đã được tạo ra trước đó. Từ nội dung kiểu dữ liệu cấu trúc và dữ liệu file trở về sau thì lệnh fflush(stdin) rất hay được sử dụng.*

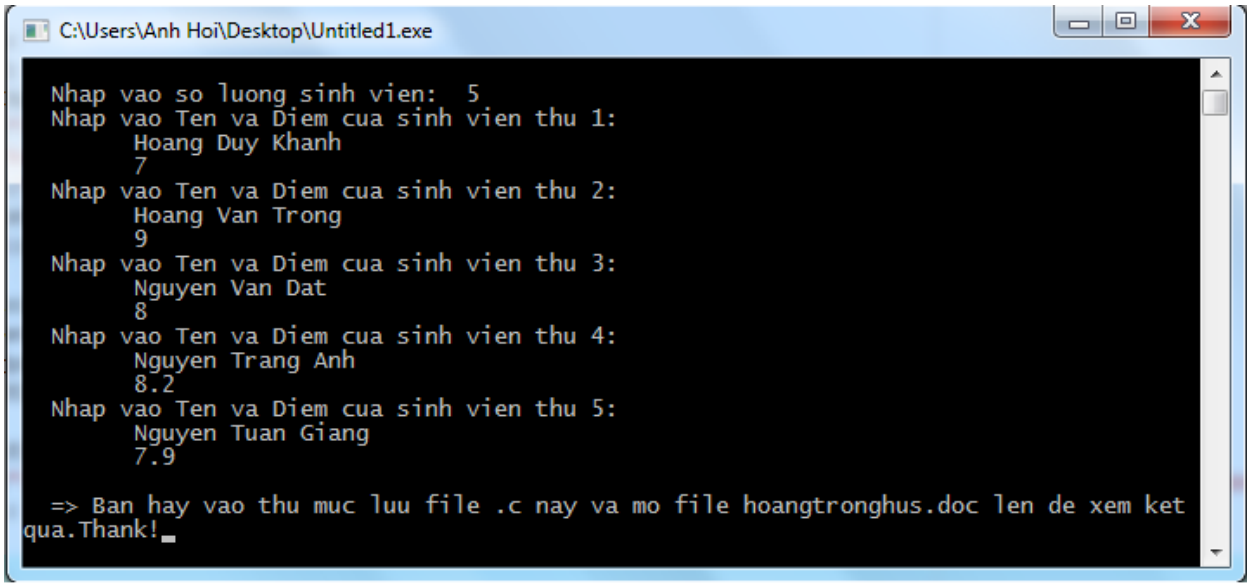
*d) Nhập thông tin cho từng sinh viên: Vì mỗi sinh viên có hai thành phần nên ta phải nhập từng thành phần đó cho từng sinh viên. Nhập thành phần nào thì đọc từ bàn phím thành phần đó và lưu vào địa chỉ tương ứng. Ví dụ: nhập tên sinh viên có dạng xâu thì dùng hàm gets() để đọc và lưu vào địa chỉ **Ten** là tên của xâu đó, xâu Ten[100] là một mảng một chiều nên bản thân tên xâu đã là địa chỉ; còn nhập điểm cho sinh viên thì phải có toán tử & để lưu vào địa chỉ cho biến **Diem**.*

Nhập hết thông tin của sinh viên nào thì ghi vào file đến đó. Sử dụng hàm fprintf() để ghi theo định dạng. Trong bài, mình sử dụng nhiều dấu như "\t\t\r\n" chỉ mang tính thẩm mỹ, mọi người có thể dùng hoặc không.

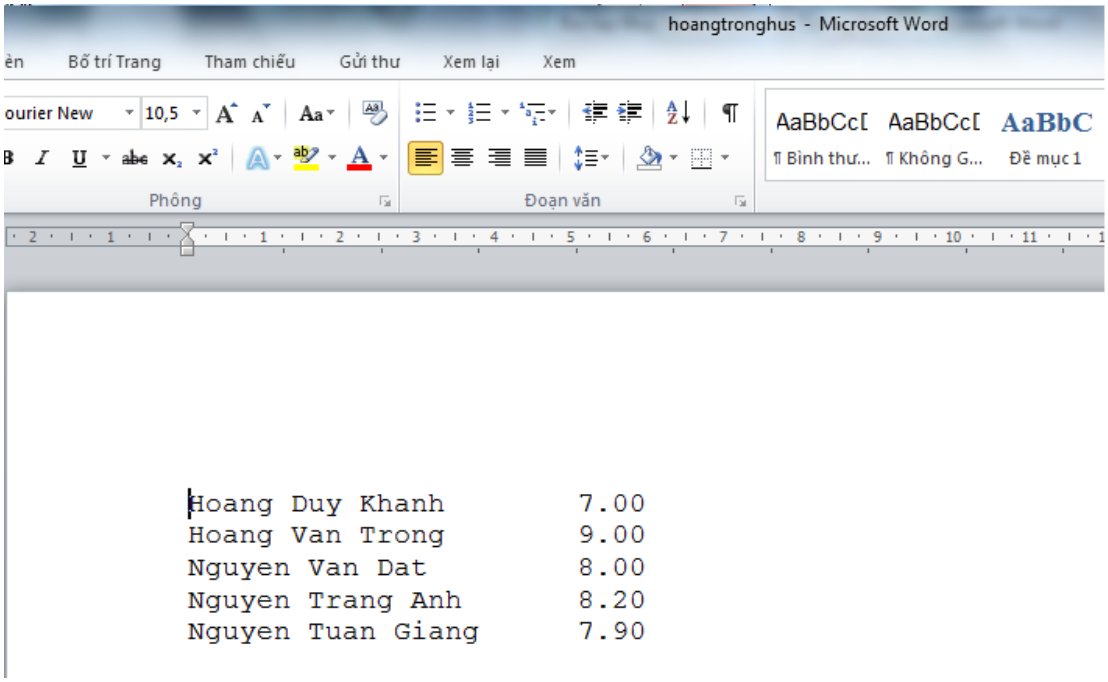
e) Khi thực hiện xong xuôi các công việc ở trên về khai báo, mở file, nhập và ghi thông tin vào file thì phải có bước đóng file để đảm bảo an toàn dữ liệu khi gọi file để đọc hay ghi ở một chương trình tiếp theo.

*Lưu ý là khi ta nhập thông tin xong cho tất cả sinh viên thì trên màn hình sẽ hiện ra dòng chữ hướng dẫn tìm tới thư mục của file kết quả để mở lên xem. Và vì mình không để đường dẫn tới địa chỉ lưu file kết quả nên máy sẽ mặc định lưu file mới tạo ra trong cùng thư mục với file chạy chương trình ***.c Kết quả thông tin của các sinh viên không được hiển thị trên màn hình do ta không viết lệnh in ra màn hình. Ah còn nữa, phần định dạng cho file kết quả thì mình thích định dạng đuôi gì cũng được miễn sao là định dạng file văn bản (doc, data, txt, cpp, c, pas, java,...)*

** <u>Màn hình kết quả như sau:</u>*



Như trong bài này mình để file chạy chương trình C ở ngoài desktop nên file mới được tạo ra là hoangtronghus.doc cũng nằm ngoài desktop, chúng ta chỉ cần click chuột vào để xem kết quả. Mình ra desktop mở file word lên và được kết quả như sau:



<u>Bài 116:</u> Cho hàm số y = sin(x) xác định trong khoảng $-\pi \le x \le \pi$. Viết chương trình tính các giá trị của y trong khoảng x vừa cho, với bước tăng liên tiếp giữa các điểm hoành độ x là $h = \frac{2\pi}{100}$ (có nghĩa là $x_{i+i} - x_i = \frac{2\pi}{100}$). Ghi giá trị của x và y thành 2 cột vào một file văn bản có tên là *data.txt*. Đọc giá trị của x và y từ file *data.txt* và in ra màn hình? (X)

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#define pi 3.14159265358979
struct ToaDo{float x;float y;};
main()
{
    float i;
    int j,Dem=0;
    struct ToaDo a[200];
    FILE *f;
    printf("\n  Ket qua duoc luu trong file data.txt tren o D.\n");
    f=fopen("D:/data.txt","w");
    for(i=-pi;i<=pi;i=i+2*pi/100)
      Dem++;
    fprintf(f,"\t%d\n\n",Dem);
    for(i=-pi;i<=pi;i=i+2*pi/100)
      fprintf(f,"\t%9.6f\t%9.6f\n",i,sin(i));
    fclose (f);
    printf("  Nhan phim bat ki de xem ket qua:  ");
    getch();
    f=fopen("D:/data.txt","r");
      fscanf(f,"%d",&Dem);
    for(j=0;j<Dem;j++)
      fscanf(f,"%f%f",&a[j].x,&a[j].y);
    printf("\n  => Day la ket qua sau khi doc file data.txt\n\n");
    printf("\t    x\t\t y=sin(x)\n\t---------\t---------\n\n");
```

```
        for(j=0;j<Dem;j++)
          printf("\t%9.6f\t%9.6f\n",a[j].x,a[j].y);
        getch();
    }
```

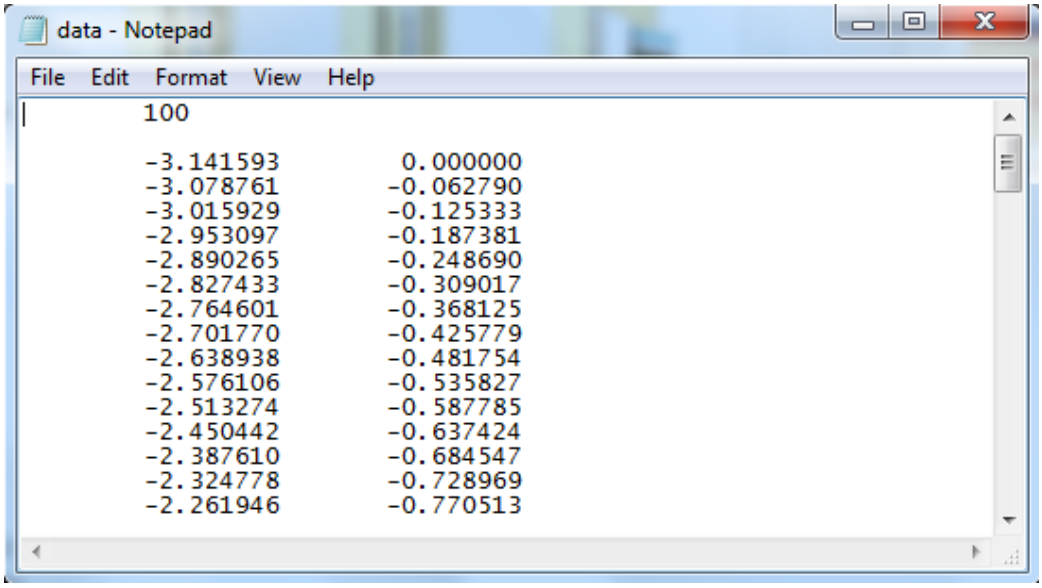
* <u>*Giải thích:*</u>

a) Mở một file mới để ghi dữ liệu vào đó, file này có tên ***data.txt*** *và lưu vào ổ D. Vì giá trị của i lấy trong đoạn [–π, π] và mỗi bước tăng i là 2π/100 nên sẽ chia đoạn trên thành 100 phần bằng nhau và có tất cả 101 giá trị của i. Tuy nhiên, so sự sai lệch nhất định nào đó về số π khai báo ban đầu nên điểm cuối cùng vượt ra khỏi giá trị π và ta chỉ có tất cả 100 điểm với 99 khoảng cách mà thôi. Vòng for đầu tiên đếm số lượng các giá trị của i và gán cho biến* ***Dem****. Vòng for thứ hai với biến i chạy từ –π đến π, chạy đến đâu thì ghi vào file data.txt hai giá trị là i và sin(i). Khi chạy xong vòng for thì dùng lệnh đóng file* ***fclose()*** *và đến đây ta đã tạo được file dữ liệu đầu ra là data.txt gồm 100 dòng trong đó mỗi dòng gồm 2 giá trị là i và sin(i).*

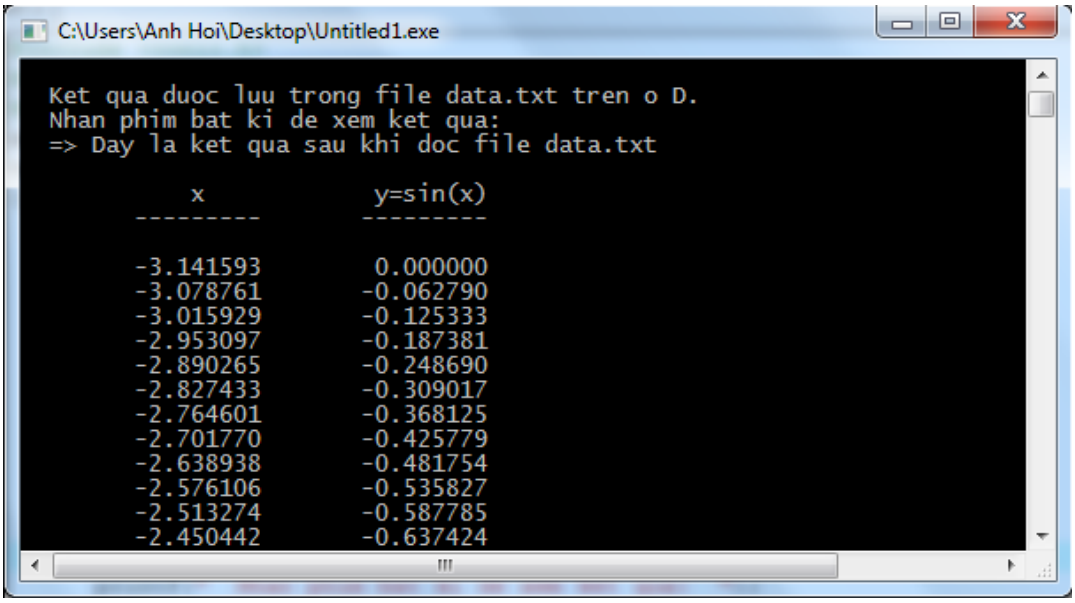
b) Tiếp theo ta dùng lệnh đọc file data.txt ***f=fopen("D:/data.txt","r");*** *giá trị đầu tiên tìm được sẽ gán cho biến Dem. Các giá trị tiếp theo được gán tương ứng cho các thành phần của biến a[i], mảng các biến a[i] được khai báo kiểu cấu trúc* ***ToaDo*** *với 2 thành phần là x và y và cả x và y đều có kiểu số thực.*

c) In ra màn hình các thông tin đọc được từ file data.txt: thực chất là in ra thông tin của các biến có kiểu cấu trúc mà ta đã ghi thông tin ở bước b. In ra thông tin về thành phần của từng biến a[i] sao cho mỗi thành phần cách nhau bằng 1 tab và xong mỗi biến a[i] thì xuống dòng để in thông tin cho biến khác. Dấu "%9.6f" có nghĩa rằng sẽ dành một vị trí là 9bit để in cho kết quả trong đó 6bit in cho các chữ số sau dấu phẩy thập phân, 1bit in cho dấu phẩy và 2bit in cho phần nguyên; làm như thế để cho kết quả hiện ra sẽ có dấu phẩy thẳng hàng nhau.

* <u>*Kết quả trên file đầu ra data.txt*</u> *(vì có tận 100 dòng kết quả nên mình chỉ lấy một phần làm đại diện):*



** Màn hình kết quả như sau (tương tự file đầu ra, mình chỉ lấy một phần kết quả để hiện ra trên màn hình):*



Bài 117: Viết chương trình cấp phát động cho mảng a có N phần tử, nhập vào giá trị các phần tử và ghi ra file văn bản *ma.txt* thành 2 cột. Cột thứ nhất chứa chỉ số i và cột thứ hai chứa giá trị của a[i]. Sắp xếp mảng theo thứ tự tăng dần rồi ghi vào file văn bản *mb.txt*. Đọc dữ liệu từ 1 trong 2 file và in ra màn hình? (X)

```
#include <stdio.h>
#include <conio.h>
struct ToaDo{int x;float y;};
main()
{
    int i,j,N;
    float *a,TrungGian;
    struct ToaDo b[100];
    FILE *fa, *fb;
    printf("\n  Nhap vao so nguyen duong N:  ");
    scanf("%d",&N);
    a=(float*)malloc(N*sizeof(float));
    fa=fopen("D:/ma.txt","w");
    fprintf(fa,"%d\n",N);
```

```
printf("  Nhap vao cac phan tu cua mang a:  ");
for(i=1;i<=N;i++)
  scanf("%f",a+i);
for(i=1;i<=N;i++)
  fprintf(fa,"%d\t%f\n",i,*(a+i));
fclose(fa);
printf("  Moi ban vao o D va mo file ma.txt len de xem!\n");
for(i=1;i<N;i++)
  for(j=i+1;j<=N;j++)
    if(*(a+i)>*(a+j))
      {
        TrungGian=*(a+i);
        *(a+i)=*(a+j);
        *(a+j)=TrungGian;
      }
fb=fopen("D:/mb.txt","w");
fprintf(fb,"%d\n",N);
for(i=1;i<=N;i++)
  fprintf(fb,"%f\n",*(a+i));
fclose(fb);
printf("  Moi ban vao o D va mo file mb.txt len de xem!\n\n");
printf("  => Doc du lieu tu file ma.txt la:\n");
fa=fopen("D:/ma.txt","r");
fscanf(fa,"%d",&N);
for(i=1;i<=N;i++)
  fscanf(fa,"%d%f",&b[i].x,&b[i].y);
for(i=1;i<=N;i++)
  printf("\t%d\t%f\n",b[i].x,b[i].y);
printf("\n  => Doc du lieu tu file mb.txt la:\n");
fb=fopen("D:/mb.txt","r");
fscanf(fb,"%d",&N);
```

```
    for(i=1;i<=N;i++)
      fscanf(fb,"%f",a+i);
    for(i=1;i<=N;i++)
      printf("\t%f\n",*(a+i));
    getch();
    return 0;
}
```

* *Giải thích:*

- Đề bài chỉ yêu cầu in ra màn hình một trong hai file vừa tạo ra nhưng mình viết chương trình để in ra màn hình cả 2 file đó.

*a) Cấp phát động cho mảng a có N phần tử: Thực hiện cấp phát bình thường và sử dụng con trỏ *a để cấp phát với cú pháp của hàm **malloc**. Giả sử các phần tử của mảng a đều có kiểu số thực nên phải khai báo con trỏ dưới dạng thực **float**. Cấu trúc cấp phát như sau:*

a=(float)malloc(N*sizeof(float));*

Chương trình sẽ dành ra 4N (byte) để chứa vùng dữ liệu cho mảng a (vì kiểu float có kích cỡ 4byte) và con trỏ a sẽ trỏ tới phần tử đầu tiên của mảng.

- Nhập vào N là số phần tử và nhập vào các phần tử của mảng.

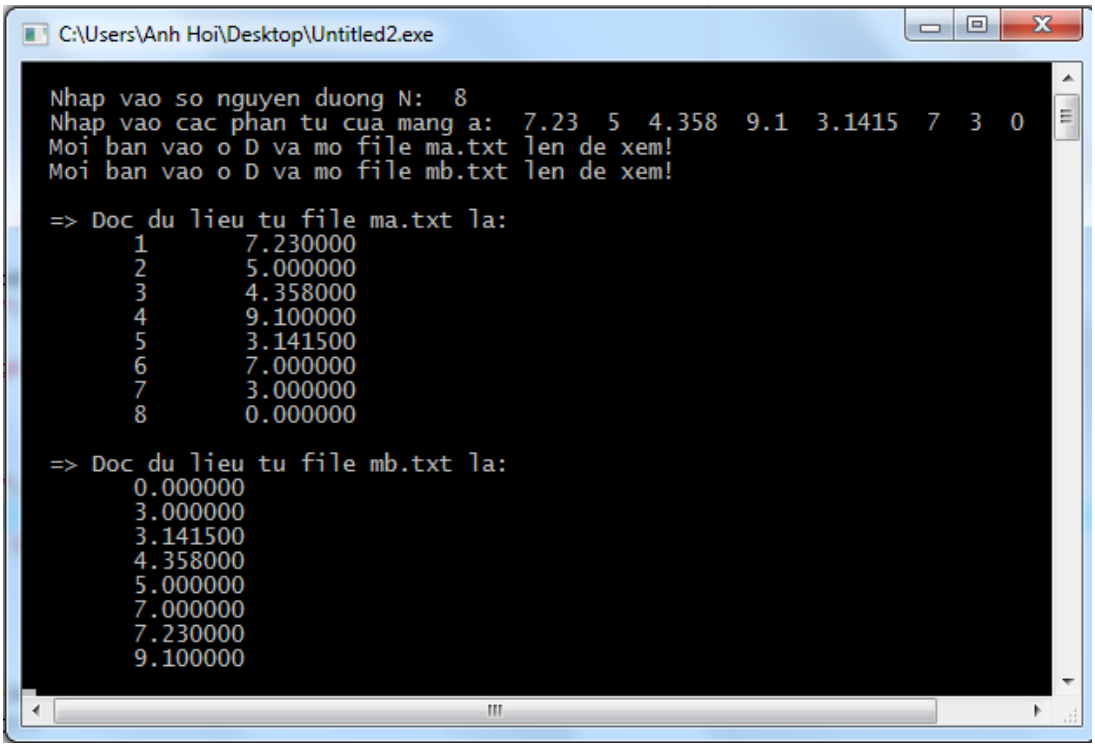
*b) Ghi thông tin ra file văn bản **ma.txt**: Đầu tiên mở một file và đặt tên là ma.txt bằng con trỏ ***fa** và in giá trị N lên dòng đầu tiên của file đó. Các giá trị *(a+i) được ghi lên file ma.txt nhờ lệnh **fprintf()**, mỗi dòng ghi 2 giá trị là biến chạy i có kiểu nguyên và giá trị của phần tử *(a + i) có kiểu thực.*

*c) Ghi thông tin ra file văn bản **mb.txt** gồm những phần tử được sắp xếp theo chiều tăng dần: Thực hiện sắp xếp tăng dần dãy các phần tử *(a + i) với i chạy từ 1 đến N. Tiếp theo, mở một file mới có tên mb.txt và tạo đường dẫn vào ổ D, dùng con trỏ ***fb**. Dòng đầu tiên trong file mb.txt chứa giá trị N và N dòng tiếp theo mỗi dòng chứa một giá trị *(a + i) sau khi đã được sắp xếp tăng dần.*

d) Đọc và in ra màn hình file ma.txt: Dùng lệnh mở file và đọc thông tin từ file ma.txt. Lưu giá trị ban đầu tìm được cho biến N. Các phần tử tiếp theo được đọc và lưu vào cho các thành phần của các biến cấu trúc b[i]. Muốn in ra màn hình thì chỉ cần truy cập đến từng thành phần của biến b[i] sau đó xuất ra màn hình có N + 1 dòng trong đó dòng đầu tiên chứa giá trị N và N dòng tiếp theo mỗi dòng chứa 2 giá trị tương ứng là 2 thành phần của biến cấu trúc.

*e) Đọc và in ra màn hình file mb.txt: Dùng lệnh mở file và đọc thông tin từ file mb.txt. Lưu giá trị đầu tiên tìm được cho biến N. Các phần tử tiếp theo được lưu cho biến ở dạng con trỏ (a + i) sau đó xuất ra màn hình giá trị mà con trỏ (a + i) trỏ tới, đó là *(a + i). Mỗi dòng chỉ in ra một phần tử *(a + i).*

* *Màn hình kết quả như sau:*



<u>Bài 118:</u> (Mai Siêu Phong)

Mai Siêu Phong có thú vui sưu tầm đầu lâu. Mỗi ngày chị Phong thu thập một cơ số đầu lâu rồi bỏ vào quan tài lưu trữ thành 5 buổi. Tuy nhiên do bị mắc bệnh hay quên nên chị ta không nhớ mình đã thu thập được tổng cộng bao nhiêu đầu lâu và số đầu lâu lớn nhất cũng như nhỏ nhất mình thu thập được trong một ngày là bao nhiêu. Hãy giúp chị Phong thống kê lại công việc này!

Dữ liệu đầu vào có dạng sau:

MSP.INP

- Dòng đầu chứa số nguyên dương M – là số ngày chị Phong đi thu thập đầu lâu.

- M dòng tiếp theo, mỗi dòng chứa 5 số thể hiện số đầu lâu thu thập được trong ngày.

Dữ liệu đầu ra có dạng sau:

MSP.OUT

- Dòng đầu là số đầu lâu lớn nhất mà chị Phong đã từng thu thập được .

- M dòng tiếp theo, mỗi dòng chứa 2 số thể hiện số đầu lâu bé nhất và lớn nhất thu thập được trong ngày.

<u>Ví dụ:</u>

MSP.INP	MSP.OUT

4	12
12 0 8 3 11	0 12
0 0 5 5 3	0 5
4 1 9 2 0	0 9
8 3 1 5 1	1 8

(Đề này bị thiếu một phần tử ở dòng cuối, từ dòng thứ 2 đến dòng thứ 5 thì mỗi dòng phải có 5 số trong khi dòng thứ 5 chỉ có 4 số. Mình tự ý thêm vào số 1 ở cuối. Nếu không thêm vào cho đủ thì không thể truy cập đến từng phần tử theo kiểu mảng 2 chiều được).

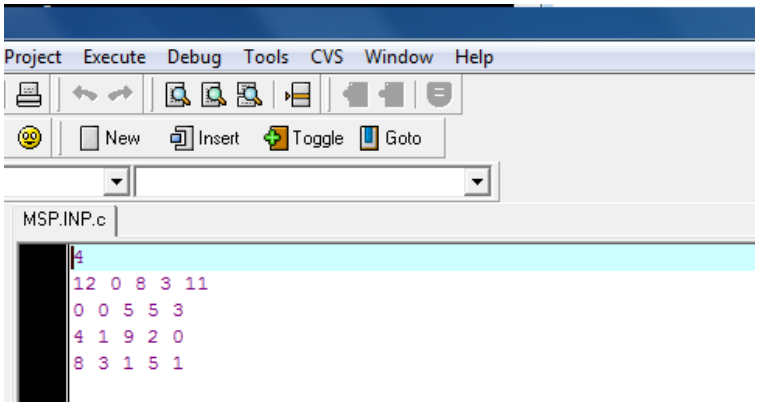
```
#include <stdio.h>
#include <conio.h>
main()
{
   int M,i,j,Max,Min,LonNhat,a[100][100];
   FILE *f;
   f=fopen("D:/MSP.INP.c","r");
   fscanf(f,"%d",&M);
   for(i=0;i<M;i++)
     for(j=0;j<5;j++)
        fscanf(f,"%d",&a[i][j]);
   f=fopen("D:/MSP.OUT.doc","w");
   LonNhat=a[0][0];
   for(i=0;i<M;i++)
     for(j=0;j<5;j++)
         if(a[i][j]>LonNhat)LonNhat=a[i][j];
   fprintf(f,"%d\n",LonNhat);
   for(i=0;i<M;i++)
     {
       Max=a[i][0];
       Min=a[i][0];
       for(j=0;j<5;j++)
          {
            if(a[i][j]>Max)Max=a[i][j];
```

```
                if(a[i][j]<Min)Min=a[i][j];
              }
            fprintf(f,"%d%6d\n",Min,Max);
          }
        fclose(f);
        printf("\n\n   => Moi ban vao o D sau do mo file MSP.OUT len de xem ket
    qua.Thank!\n");
        getch();
        return main();
    }
```

* *Giải thích:*

- Đề bài cho mỗi dòng có 5 số nhưng dòng thứ 5 trong MSP.INP chỉ có 4 số nên mình tự ý thêm vào số 1 (mình nghĩ thầy viết thiếu). Nếu không đủ 5 số thì không thể truy nhập tới phần tử theo cách của mảng 2 chiều được mà chỉ truy nhập theo dạng mảng 1 chiều, như thế sẽ không tính được phần tử lớn nhất và nhỏ nhất trên dòng thứ i.

a) Đầu tiên tạo ra tệp MSP.INP bằng bất kì trình soạn thảo văn bản nào và lưu vào vị trí nào đó. Ở đây, mình tạo ra file đầu vào trên chính trình soạn thảo của Dev – Cpp và lưu với đuôi .c sau đó copy file vừa tạo vào ổ D. Mình tạo file nguồn MSP.INP như hình dưới đây (giống với file nguồn trong ví dụ, các bạn có thể tạo file nguồn khác) và phần tử đầu tiên chính là số dòng:



*Trong chương trình chính, ta khai báo các biến cần sử dụng: **M** là tổng số dòng, **i** và **j** là biến để chạy trong vòng lặp for, **LonNhat** là biến để xác định số đầu lâu lớn nhất mà chị Phong đã từng thu thập được, biến **Max** để xác định số đầu lâu lớn nhất trong từng ngày và biến **Min** để xác định số đầu lâu nhỏ nhất trong từng ngày, con trỏ ***f** để trỏ tới file cần thực hiện – con trỏ này có dạng FILE. Ta thấy phần dữ liệu đầu vào có dạng ma trận 4 hàng 5 cột nên cần khai báo thêm mảng 2 chiều a[100][100] có tối đa 100 hàng và 100 cột.*

b) Các lệnh:

*- Lệnh mở file để đọc: **f=fopen("D:/MSP.INP.c","r");** chỉ rõ đường dẫn tới file cần mở cũng như tên file cần mở là MSP.INP.c và chế độ mở file là mở để đọc (r = read = đọc). Công việc mở file có thể thành công hay không thành công và được gán cho con trỏ f.*

*- Lệnh **fscanf(f,"%d",&M);** là lệnh đọc từ file nguồn rồi lưu giá trị đầu tiên mà nó tìm thấy cho biến M. Giá trị M chính là số dòng. Như vậy là ta đã xác định được dữ liệu đầu vào có M dòng và 5 cột.*

*- Các vòng lặp for lồng nhau với biến i chạy từ 0 đến M – 1 và biến j chạy từ 0 đến 4. Các biến i, j chạy tới đâu thì đọc từ file nguồn đến đó và lưu vào cho phần tử a[i][j]. Lệnh đọc từ file là **fscanf()** tương tự với lệnh đọc từ bàn phím là **scanf()**.*

*- Lệnh **f=fopen("D:/MSP.OUT.doc","w");** là lệnh mở một file mới cho mục đích ghi dữ liệu vào file đó (ghi đè), lệnh này được gán cho con trỏ f. Sau khi tạo ra file MSP.OUT.doc thì lưu vào ổ D.*

⇒ Đến đây ta đã có toàn bộ dữ liệu file nguồn MSP.INP.c được nạp vào máy và xác định được địa chỉ của dữ liệu đầu ra là file MSP.OUT.doc trong ổ D. Dữ liệu đầu ra có thể định dạng ở nhiều loại khác nhau, mình lấy định dạng document làm ví dụ (.doc – file word).

*c) Tìm số đầu lâu lớn nhất mà chị Phong đã từng thu thập được: sử dụng biến **LonNhat**, ban đầu gán biến LonNhat bằng phần tử đầu tiên LonNhat = a[0][0], sau đó duyệt từng phần tử của mảng 2 chiều nếu thấy a[i][j] > LonNhat thì gán lại giá trị LonNhat = a[i][j]. Sau khi tính xong giá trị LonNhat thì ghi nó lên dòng đầu tiên của file MSP.OUT.doc bằng lệnh **fprintf()**.*

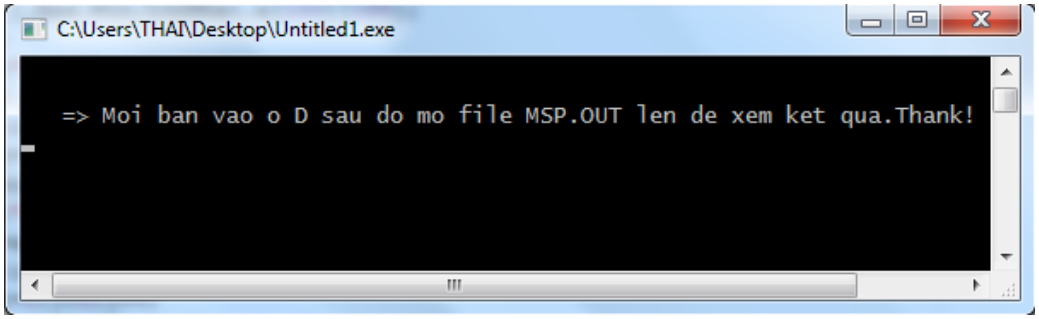
*d) Tìm số đầu lâu lớn nhất và số đầu lâu nhỏ nhất trong 1 ngày: thực chất là tìm giá trị lớn nhất và giá trị nhỏ nhất trên từng dòng của mảng 2 chiều, sử dụng biến Max và Min. Gán Max cũng như Min bằng phần tử đầu tiên của dòng đó: Max = Min = a[i][0], sau đó duyệt từng phần tử trên dòng i nếu thấy a[i][j] > Max thì gán lại Max = a[i][j] và nếu thấy a[i][j] < Min thì gán lại Min = a[i][j]. Sau khi tìm xong phần tử lớn nhất và nhỏ nhất trên dòng thứ i thì ghi 2 giá trị đó lên file MSP.OUT.doc bằng lệnh **fprintf()**, ghi xong thì xuống dòng.*

*e) Sau khi thực hiện xong các công việc trên thì phải có bước đóng file bằng lệnh **fclose(f);** thì dữ liệu mới có thể lưu lại được khi gọi file để sử dụng cho chương trình khác.*

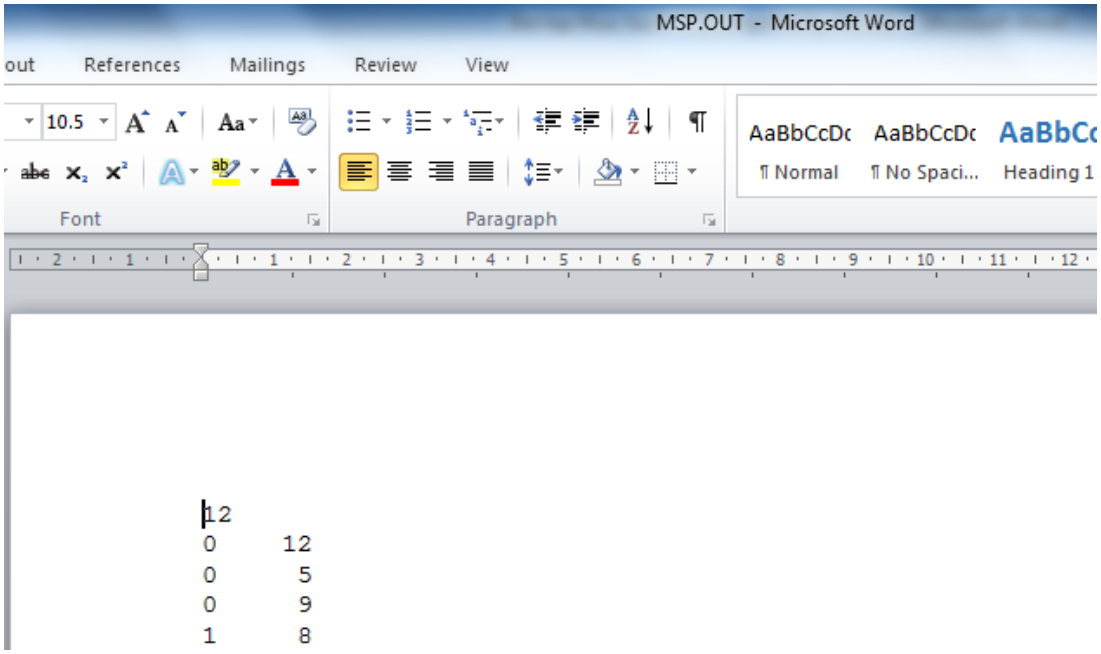
Tất cả các bước chúng ta làm ở trên chỉ sử dụng dữ liệu đầu vào có sẵn và lưu vào file khác mà ta xác định địa chỉ cho nó. Các kết quả này chúng ta không đưa ra màn hình nên khi chạy chương trình thì trên màn hình không có thông tin gì cả. Mình chỉ xuất ra màn hình dòng chữ hướng dẫn người lập trình vào ổ D và mở file MSP.OUT.doc lên để xem kết quả.

⇒ Những bài tập dạng VÀO/ RA file như bài này thì chỉ thêm các lệnh đọc file, ghi file, đóng file,…còn các công việc khác thì tương tự như thao tác với mảng 1 chiều, 2 chiều, xâu kí tự, con trỏ, dữ kiệu kiểu cấu trúc,…

** <u>Màn hình kết quả như sau:</u>*



Sau đó, vào ổ D và mở file word MSP.OUT lên xem thì được kết quả như sau:



<u>Bài 119:</u> (Ma trận trôn ốc)

Ma trận trôn ốc là ma trận có các phần tử được sắp xếp dạng xoáy trôn ốc (theo chiều kim đồng hồ) bắt đầu từ phần tử 1 ở góc trên bên trái cho đến phần tử NxN với N là số cho trước. Cho trước số N hãy in ra ma trận trôn ốc tương ứng.

Dữ liệu đầu vào có dạng sau:

MTTO.INP

- Chỉ có một dòng chứa số nguyên dương N.

Dữ liệu đầu ra có dạng sau:

MTTO.OUT

- Ma trận vuông cấp NxN với các phần tử dạng xoáy trôn ốc.

Ví dụ:

MTTO.INP	MTTO.OUT
4	1 2 3 4
	12 13 14 5
	11 16 15 6
	10 9 8 7

```
#include <stdio.h>
```

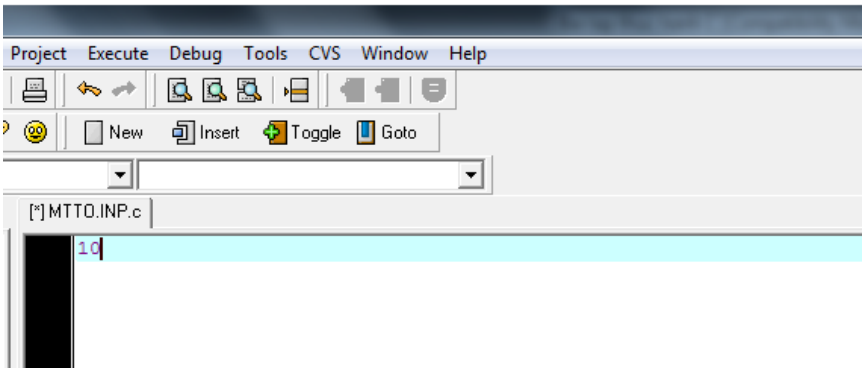
```
#include <conio.h>
main()
{
   int N,GiaTri=1,CotTrai=0,HangTren=0,CotPhai,HangDuoi;
   int i,j,k,h,a[100][100];
   FILE *f;
   f=fopen("D:/MTTO.INP.c","r");
   fscanf(f,"%d",&N);
   CotPhai=N-1;
   HangDuoi=N-1;
   while(GiaTri<=N*N)
     {
       for(i=CotTrai;(i<=CotPhai)&&(GiaTri<= N*N);i++,GiaTri++)
         a[HangTren][i]=GiaTri;
       HangTren++;
       for(j=HangTren;(j<=HangDuoi)&&(GiaTri<= N*N);j++,GiaTri++)
         a[j][CotPhai]=GiaTri;
       CotPhai--;
       for(k=CotPhai;(k>=CotTrai)&&(GiaTri<= N*N);k--,GiaTri++)
         a[HangDuoi][k]=GiaTri;
       HangDuoi--;
       for(h=HangDuoi;(h>=HangTren)&&(GiaTri<= N*N);h--,GiaTri++)
         a[h][CotTrai]=GiaTri;
       CotTrai++;
     }
   f=fopen("D:/MTTO.OUT.doc","w");
   for(i=0;i<N;i++)
     {
       for(j=0;j<N;j++)
         fprintf(f,"%5d",a[i][j]);
       fprintf(f,"\n");
```

```
    }
    fclose(f);
    printf("\n\n   => Moi ban vao o D sau do mo file word MTTO.OUT len de
xem ket qua.Thank!\n");
    getch();
    return main();
}
```

* *Giải thích:*

a) Trước tiên khai báo các biến cần sử dụng cho chương trình: Cách làm bài này là dùng 4 biến đại diện cho hàng trên, hàng dưới, cột trái, cột phải là ***HangTren***, ***HangDuoi***, ***CotTrai***, ***CotPhai*** *để đánh dấu. HangTren là hàng từ trên xuống với giá trị ban đầu bằng 0, HangDuoi là hàng từ dưới lên với giá trị ban đầu bằng N – 1 (N là kích cỡ của ma trận), CotTrai là cột từ trái sang phải với giá trị ban đầu bằng 0 và CotPhai là cột từ bên phải lùi lại trái với giá trị ban đầu bằng N – 1.*

b) Khai báo một con trỏ dạng FILE ****f*** *và thực hiên mở file MTTO.INP.c lên để đọc. File MTTO.INP.c được người lập trình tạo ra từ trước và copy vào ổ D. Đọc từ file nguồn và lưu giá trị tìm được cho biến N là kích cỡ của ma trận. Viết file nguồn thì chỉ cần viết duy nhất một phần tử dạng số trong trình soạn thảo của Dev – Cpp, màn hình file nguồn MTTO.INP như sau (nhập giá trị ví dụ bằng 10):*



c) Sử dụng vòng lặp ***while*** *để tìm phần tử ở chỉ số tương ứng. Khi điều kiện số phần tử còn nhỏ hơn hoặc bằng N^2 – là số phần tử của ma trận thì thực hiện vòng 4 vòng for bên trong. Vòng for thứ nhất để ghi các phần tử trên hàng đầu tiên theo chiều từ trái sang phải. Vòng for thứ hai để ghi các phần tử ở cột ngoài cùng bên phải theo chiều từ trên xuống dưới. Vòng for thứ ba để ghi các phần tử ở hàng dưới cùng theo chiều từ phải sang trái. Vòng for thứ tư để ghi các phần tử ở cột đầu tiên theo chiều từ dưới lên trên. Sau mỗi vòng lặp while thì CotTrai và HangTren đều tăng lên 1 đơn vị còn CotPhai và HangDuoi đều giảm xuống1 đơn vị.*

Vòng while cứ lặp đi lặp lại như thế cho tới khi số phần tử bằng N^2.

Để hiểu rõ hơn, mình giả sử lấy N = 4 và minh họa kết quả cho vòng lặp while đầu tiên như sau:

+ Trong vòng for thứ nhất, i chạy từ 0 đến 3 và biến HangTren = 0, sau mỗi một lượt thì i tăng lên 1 đơn vị và biến GiaTri cũng tăng lên 1 đơn vị. Biến GiaTri được khởi tạo ban đầu bằng 1. Như vậy sau vòng for thứ nhất ta có các kết quả:

a[0][0] = 1; a[0][1] = 2; a[0][2] = 3; a[0][3] = 4.

Tăng biến HangTren lên 1 đơn vị: HangTren = 1. Biến GiaTri lúc này đang nhận giá trị 4.

+ Trong vòng for thứ hai, j chạy từ HangTren đến 3 mà theo kết quả vòng for thứ nhất thì HangTren = 1 do đó j chạy từ 1 đến 3. Lúc này CotPhai đang nhận giá trị 3. Như vậy sau vòng for thứ hai ta có các kết quả:

a[1][3] = 5; a[2][3] = 6; a[3][3] = 7.

Giảm CotPhai xuống 1 đơn vị: CotPhai = 2. Biến GiaTri lúc này đang nhận giá trị 7.

+ Trong vòng for thứ ba, k chạy từ CotPhai đến CotTrai mà theo khởi tạo ban đầu CotTrai = 0 và theo vòng for thứ hai thì CotPhai = 2 do đó k chạy lùi từ 2 về 0. HangDuoi đang nhận giá trị khởi tạo ban đầu là 3. Như vậy sau vòng for thứ ba ta có các kết quả:

a[3][2] = 8; a[3][1] = 9; a[3][0] = 10.

Giảm HangDuoi xuống 1 đơn vị: HangDuoi = 2. Biến GiaTri lúc này đang nhận giá trị 10.

+ Trong vòng for thứ tư, h chạy từ HangDuoi đến HangTren mà theo vòng for thứ ba thì HangDuoi = 2 và theo vòng for thứ nhất thì HangTren = 1 do đó h chạy lùi từ 2 đến 1. CotTrai lúc này đang nhận giá trị khởi tạo bằng 0. Như vậy sau vòng for thứ 4 ta có các kết quả sau:

a[2][0] = 11; a[1][0] = 12.

Tăng CotTrai lên 1 đơn vị: CotTrai = 1. Biến GiaTri lúc này đang nhận giá trị 12.

Vậy, sau 4 vòng for đầu tiên (hay sau vòng while thứ nhất) thì chỉ có các phần tử màu đỏ của ma trận mới có giá trị (bảng dưới):

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]
a[3][0]	a[3][1]	a[3][2]	a[3][3]

Tương đương với kết quả nhận được là:

1	2	3	4
12	a[1][1]	a[1][2]	5
11	a[2][1]	a[2][2]	6
10	9	8	7

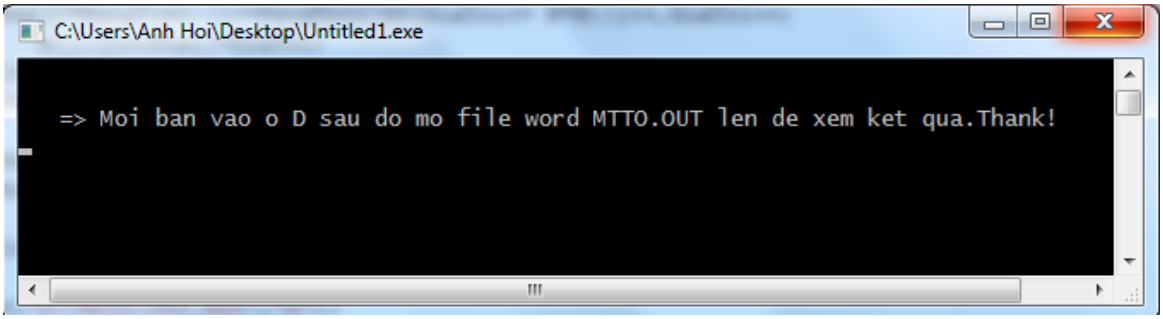
Tiếp tục sang vòng while tiếp theo khi điều kiện GiaTri ≤ N^2 còn thỏa mãn. Lưu ý là sang vòng while tiếp theo thì các biến HangTren, HangDuoi, CotTrai, CotPhai đã được thay đổi sang giá trị mới.

*d) Mở lên một file mới có dạng MTTO.OUT.doc để ghi kết quả vào file đó, dùng lệnh **fopen**. Mình để đường dẫn tới nơi lưu file là ổ D.*

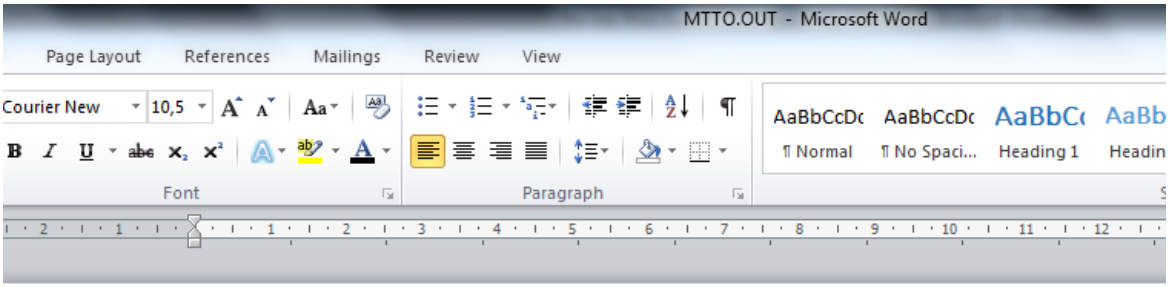
*Sau khi đã có tất cả các phần tử ma trận trôn ốc thì làm công việc ghi các phần tử đó lên file đầu ra MTTO.OUT.doc nhờ lệnh **fprintf()**. Ghi các phần tử vào ma trận trôn ốc thì ghi tuần tự với chỉ số i, j như bình thường.*

*Ghi xong dữ liệu vào file đầu ra thì phải dùng lệnh đóng file **fclose(f)**, cứ thao tác với việc ghi file thì phải có bước này và các bài liên quan đến ghi file về sau mình sẽ luôn dùng lệnh fclose() với ý nghĩa như thế. Tương tự bài trước thì chương trình chỉ in kết quả lên file đầu ra chứ màn hình thì không có thông tin gì cả, mình chỉ in ra dòng hướng dẫn tới nơi lưu file đầu ra mà thôi.*

* *Màn hình kết quả như sau:*



Sau khi vào ổ D và mở file word MTTO.OUT lên thì được kết quả như sau:



1	2	3	4	5	6	7	8	9	10
36	37	38	39	40	41	42	43	44	11
35	64	65	66	67	68	69	70	45	12
34	63	84	85	86	87	88	71	46	13
33	62	83	96	97	98	89	72	47	14
32	61	82	95	100	99	90	73	48	15
31	60	81	94	93	92	91	74	49	16
30	59	80	79	78	77	76	75	50	17
29	58	57	56	55	54	53	52	51	18
28	27	26	25	24	23	22	21	20	19

Bài 120: (Điểm yên ngựa)

Trong một ma trận MxN, phần tử a[i][j] được gọi là điểm yên ngựa của ma trận nếu như nó là phần tử nhỏ nhất trên hàng i và lớn nhất trên cột j của ma trận. Cho ma trận MxN, hãy tìm điểm yên ngựa của ma trận trên.

Dữ liệu đầu vào có dạng sau:

YENNGUA.INP

- Dòng đầu chứa hai số nguyên dương M, N

- M dòng tiếp theo, mỗi dòng chứa N số thực thể hiện là số phần tử của ma trận.

Dữ liệu đầu ra có dạng sau:

YENNGUA.OUT

- Giá trị điểm yên ngựa.

Ví dụ:

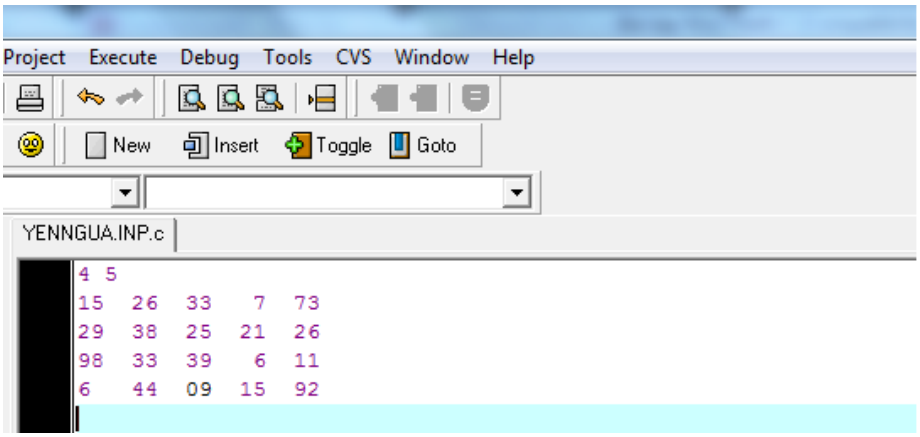
YENNGUA.INP	YENNGUA.OUT
3 2 15 10 30 20 40 14	20

```
#include <stdio.h>
#include <conio.h>
int Min(int a[][100],int b,int x,int N);
int Max(int a[][100],int b,int x,int M);
main()
{
   int a[100][100],M,N,i,j;
   FILE *f;
   f=fopen("D:/YENNGUA.INP.c","r");
   fscanf(f,"%d%d",&M,&N);
   for(i=0;i<M;i++)
     for(j=0;j<N;j++)
        fscanf(f,"%d",&a[i][j]);
```

```
    printf("\n\n  => Moi ban vao o D va mo file word YENNGUA.OUT de xem
ket qua!");
    f=fopen("D:/YENNGUA.OUT.doc","w");
    for(i=0;i<M;i++)
      for(j=0;j<N;j++)
        if(Min(a,a[i][j],i,N)==1&&Max(a,a[i][j],j,M)==1)
          fprintf(f,"%d",a[i][j]);
    fclose(f);
    getch();
    return 0;
}
int Min(int a[][100],int b,int x,int N)
{
    int k=1,j;
    for(j=0;j<N;j++)
      if(b>a[x][j])k=0;
    return (k);
}
int Max(int a[][100],int b,int x,int M)
{
    int k=1,i;
    for(i=0;i<M;i++)
      if(b<a[i][x])k=0;
    return (k);
}
```

* *Giải thích:*

a) Tạo thông tin và lưu cho file đầu vào tên là YENNGUA.INP sau đó copy vào ổ D. Ví dụ tạo file thông tin đầu vào như sau: (dùng luôn trình soạn thảo văn bản của Dev – Cpp để viết thông tin nguồn)



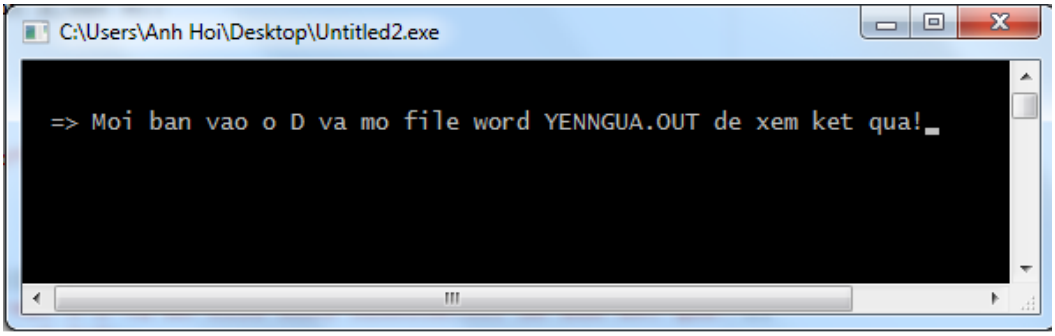
b) Đọc từ file YENNGUA.INP.c và lưu hai giá trị đầu tiên cho biến M và N là số hàng và số cột của ma trận. Các giá trị tiếp theo được lưu cho biến a[i][j] tương ứng (i chạy từ 0 đến M – 1 và j chạy từ 0 đến N – 1). Sau khi có giá trị a[i][j] của ma trận a thì tiến hành tìm điểm yên ngựa của ma trận đó, điểm yên ngựa chính là phần tử nhỏ nhất trên hàng i và lớn nhất trên cột j.

c) Để xác định một giá trị nào đó có phải là điểm yên ngựa hay không thì viết thêm hàm để kiểm tra. Hàm ***Min*** *để xác định xem a[i][j] có phải là nhỏ nhất trên hàng i hay không và hàm* ***Max*** *để xác định xem a[i][j] có phải là lớn nhất trên cột j hay không.*

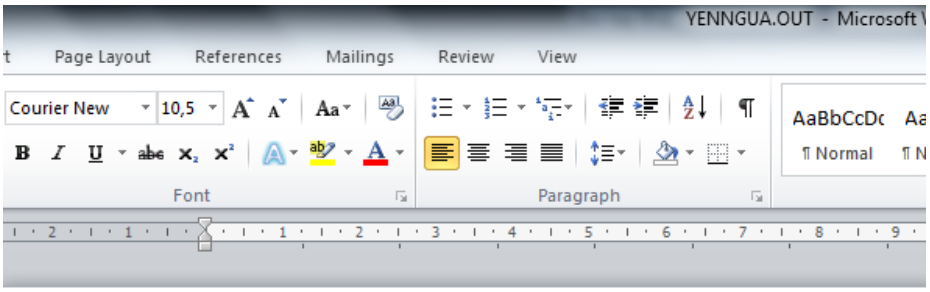
Trong hàm Min() thì các đối số truyền cho hàm gồm ma trận a, giá trị a[i][j], chỉ số hàng i và tổng số cột N. Ban đầu giả sử giá trị a[i][j] là nhỏ nhất trên hàng i (gán k = 1) sau đó so sánh giá trị a[i][j] với các giá trị từ a[i][0] đến a[i][N – 1] nếu thấy a[i][j] lớn hơn bất kì một giá trị nào thì chứng tỏ a[i][j] không phải là nhỏ nhất trên dòng thứ i (gán lại k = 0). Trả về giá trị k cho tên hàm.

Trong hàm Max() thì các đối số truyền cho hàm gồm ma trận a, giá trị a[i][j], chỉ số cột j và tổng số hàng M. Ban đầu giả sử giá trị a[i][j] là lớn nhất trên cột j (gán k = 1) sau đó so sánh giá trị a[i][j] với các giá trị từ a[0][j] đến a[M – 1][j] nếu thấy a[i][j] nhỏ hơn bất kì một giá trị nào thì chứng tỏ a[i][j] không phải là lớn nhất trên cột j (gán lại k = 0). Trả về giá trị k cho tên hàm.

* *<u>Màn hình kết quả như sau:</u>*



Sau khi vào ổ D và mở file word YENNGUA.OUT thì được kết quả là giá trị 21:



21

<u>Bài 121:</u> (Tam giác Pascal)

Tam giác Pascal là tam giác có dạng sau:

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
```

Dữ liệu đầu vào có dạng sau:

PASCAL.INP

- Chỉ có 1 dòng chứa số nguyên dương N là chiều cao của tam giác.

Dữ liệu đầu ra có dạng sau:

PASCAL.OUT

- Tam giác Pascal có chiều cao N giống với dạng như trên. (X)

<u>Ví dụ:</u>

PASCAL.INP	PASCAL.OUT
5	(Như hình trên)

```
#include <stdio.h>
#include <conio.h>
main()
{
```

```
    int i,j,N,a[100][100],Trang;
    FILE *f;
    f=fopen("D:/PASCAL.INP.c","r");
    fscanf(f,"%d",&N);
    for(i=1;i<=N;i++)
      {
        a[i][1]=1;
        a[i][i]=1;
      }
    for(i=3;i<=N;i++)
      for(j=2;j<i;j++)
        a[i][j]=a[i-1][j-1]+a[i-1][j];
    f=fopen("D:/PASCAL.OUT.doc","w");
    for(i=1;i<=N;i++)
      {
        Trang=25-2*i;
        for(j=1;j<=Trang;j++)
          fprintf(f," ");
        for(j=1;j<=i;j++)
          fprintf(f,"%4d",a[i][j]);
        fprintf(f,"\n");
      }
    fclose(f);
    printf("\n\n  => Moi ban vao o D va mo file word PASCAL.OUT de xem ket
  qua!  ");
    getch();
  }
```

* *Giải thích:*

- Tương tự như những bài tập trước, nếu đề bài yêu cầu dùng file đầu vào và xuất ra file đầu ra thì phải tạo trước thông tin cho file đầu vào cũng như sử dụng các lệnh mở file để đọc, ghi file và đóng file với việc khai báo một con trỏ dạng FILE. Với các bài sau mà cùng dạng này thì mình sẽ không giải thích lại các bước như trên nữa.

a) Các phần tử của tam giác Pascal có dạng khối nên dùng thêm một mảng 2 chiều để lưu các phần tử này. Vấn đề quan trọng nhất của bài toán này chính là xác định được giá trị của các phần tử tương ứng. Ta thấy rằng trong tam giác Pascal thì trên dòng thứ i sẽ có i phần tử và phần tử đầu tiên cũng như phần tử cuối cùng của dòng thứ i đều bằng 1. Chính vì vậy, ta sử dụng vòng for với biến i chạy từ 1 đến N, chạy tới dòng nào thì gán giá trị đầu tiên và giá trị cuối cùng của dòng đó bằng 1 (a[i][1] = a[i][i] = 1).

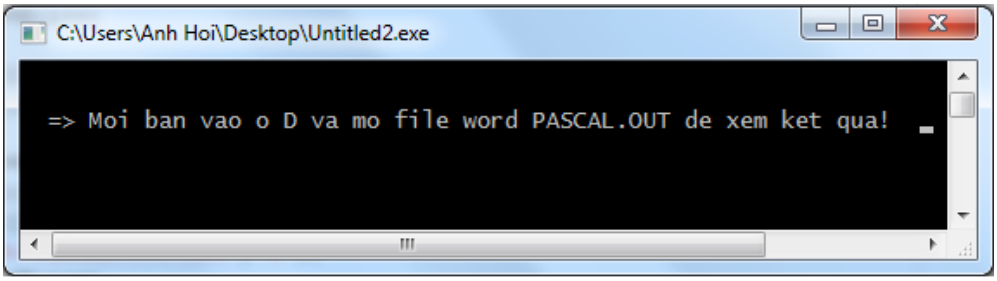
Tiếp đến là các giá trị bên trong của mỗi dòng i thì sử dụng công thức tổ hợp để tính:

$$C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$$

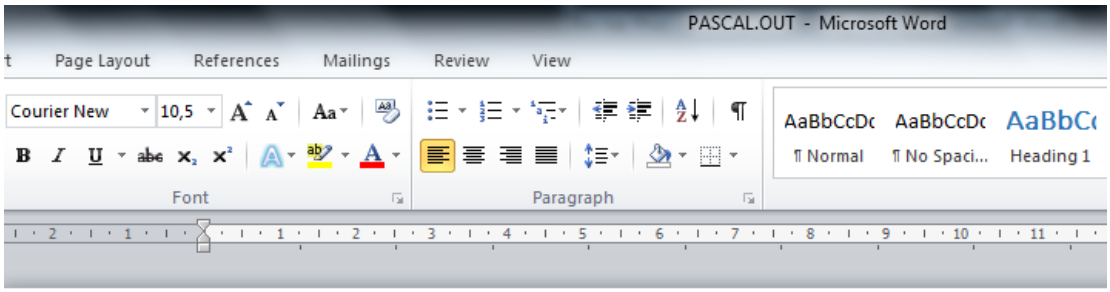
Hay: a[i][j] = a[i-1][j-1] + a[i-1][j] với biến j chạy từ 2 đến i – 1.

b) Các kết quả tính được ở câu a được ghi vào cho file PASCAL.OUT định dạng document (file word). Tuy nhiên, để in ra dưới dạng tam giác cân thì phải chèn thêm các kí tự trắng vào đằng trước các dòng. Dòng đầu chèn nhiều kí tự trắng nhất, dòng 2 chèn ít kí tự trắng hơn dòng 1 và dòng 3 chèn ít hơn dòng 2,...Biến ***Trang*** *là để xác định số kí tự trắng cần chèn vào cho mỗi dòng, mình khởi tạo bằng 25 (các bạn có thể khởi tạo bằng 30, 40 tùy ý miễn sao đảm bảo tính thẩm mỹ cho hình tam giác là được) và dòng sau được chèn ít hơn dòng trước nó 2 kí tự trắng.*

* *Màn hình kết quả như sau:*



Sau khi mở file word PASCAL.OUT trong ổ cứng D thì được kết quả như sau (với file đầu vào chứa giá trị bằng 8):




```
                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
      1   5  10  10   5   1
    1   6  15  20  15   6   1
  1   7  21  35  35  21   7   1
```

<u>Bài 122:</u> (Thay từ)

Cho file D1.INP chứa một đoạn văn bản và D2.INP chứa N dòng, mỗi dòng gồm 2 từ: nguồn và đích trong đó nguồn là các từ xuất hiện trong D1.INP cần thay thế bởi đích. Hãy thay thế các từ này và lưu kết quả vào file KQ.OUT

<u>Ví dụ:</u>

D1.INP	D2.INP
Xin chào lớp học Lập trình C khóa 2012. Chúc các bạn thi tốt!	bạn em
	! .
	khóa Khóa
KQ.OUT	
Xin chào lớp học Lập trình C Khóa 2012. Chúc các em thi tốt.	

<u>Bài 123:</u> (Tấm Cám)

Ngày xửa ngày xưa, nhà kia có hai chị em cùng cha khác mẹ, chị là Tấm, em là Cám. Mẹ Tấm mất sớm, ít năm sau thì cha Tấm cũng qua đời. Tấm ở với dì ghẻ là mẹ Cám. Tấm rất muốn đi dữ lễ hội nhưng Cám ghen ghét không muốn chị đi vì sợ

mình thua thiệt. Cám bầy mưu nhờ dì ghẻ bắt Tấm phải đếm đủ số bi mới cho đi. Với số lượng bi cho trước, Tấm phải chuyển sang dạng nhị phân từ phần tử 0 cho đến số bi đó. Tấm rất cô đơn và thất vọng. Hãy giúp Tấm giải quyết khó khăn này.

Dữ liệu đầu vào có dạng sau:

TAM.INP

- Một dòng chứa số nguyên dương N là số lượng bi.

Dữ liệu đầu ra có dạng sau:

TAM.OUT

N + 1 dòng, mỗi dòng là một dãy số nhị phân thể hiện một số từ 0 đến N.

Ví dụ:

TAM.INP	TAM.OUT
4	000
	001
	010
	011
	100

```
#include <stdio.h>
#include <conio.h>
void MaNhiPhan(int a,FILE *f);
main()
{
    int N,i;
    FILE *fa,*fb;
    fa=fopen("D:/TAM.INP.c","r");
      fscanf(fa,"%d",&N);
    fb=fopen("D:/TAM.OUT.doc","w");
    for(i=0;i<=N;i++)
      MaNhiPhan(i,fb);
    printf("\n  => Moi ban vao o cung D va mo file word TAM.OUT de xem ket qua!\n");
    getch();
}
```

```
void MaNhiPhan(int a,FILE *f)
{
    int b[100],i=0,j;
    do
      {
        b[i]=a%2;
        a=a/2;
        i++;
      }
    while (a>0);
    for(j=i-1;j>=0;j--)
      fprintf(f,"%d",b[j]);
    fprintf(f,"\n");
}
```

* Giải thích:

Bài 124: (Siêu nguyên tố)

Số siêu nguyên tố là số nguyên tố mà khi bỏ đi một số tùy ý các chữ số bên phải của nó thì phần còn lại vẫn tạo thành số nguyên tố. VD: 7333 là số siêu nguyên tố do 733, 73 và 7 đều là các số nguyên tố. Cho trước một số nguyên dương N, hãy tìm tất cả các số siêu nguyên tố có N chữ số.

Dữ liệu đầu vào có dạng sau:

SNT.INP

- Một dòng chứa số nguyên dương N

Dữ liệu đầu ra có dạng sau:

SNT.OUT

- Một số lượng dòng trong đó mỗi dòng là một số siêu nguyên tố có N chữ số.

Ví dụ:

SNT.INP	SNT.OUT
2	23
	29

	31
	37
	53
	59
	71
	73
	79

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
int NguyenTo(int a);
int SieuNguyenTo(int b,int n);
main()
{
    int N,i;
    FILE *fa,*fb;
    fa=fopen("D:/SNT.INP.c","r");
      fscanf(fa,"%d",&N);
    fb=fopen("D:/SNT.OUT.doc","w");
    for(i=pow(10,N-1);i<pow(10,N);i++)
      if(SieuNguyenTo(i,N)==1)
        fprintf(fb,"%d\n",i);
    printf("\n  => Moi ban vao o cung D va mo file word SNT.OUT de xem ket
qua!\n");
    getch();
    return 0;
}
int NguyenTo(int a)
{    int i,k=0;
    if(a>=2)

      {
```

```
            for(i=2;a%i!=0;i++);
            if(i==a)k=1;
           }
        return (k);
    }
    int SieuNguyenTo(int b,int n)
    {
        int k=1,i;
        for(i=1;i<n;i++)
          if(NguyenTo(b)==0||NguyenTo(b/pow(10,i))==0)k=0;
        return (k);
    }
```

* Giải thích:

CHƯƠNG IX. ĐỒ HỌA

CHƯƠNG X. MỘT SỐ ĐỀ THI ĐẶC BIỆT

CHƯƠNG XI. MỘT SỐ CHƯƠNG TRÌNH VIRUS ĐƠN GIẢN

Bài 149:

CHƯƠNG XII. ỨNG DỤNG LẬP TRÌNH C VÀO CÁC BÀI TOÁN THỰC TẾ VÀ CÁC BÀI TOÁN CHUYÊN NGÀNH

Do C là một ngôn ngữ bậc trung nên nó thích hợp với việc viết phần mềm hệ thống hơn là các phần mềm ứng dụng. Tuy nhiên, nếu sử dụng C để viết chương trình giải quyết các bài toán nhỏ, hay gặp trong thực tế cũng như trong các môn học của sinh viên thì cũng là một lựa chọn hợp lý. Với những ứng dụng nhỏ và không cần giao diện đẹp thì

hiệu quả làm việc trong C sẽ khiến bạn hài lòng. Đối với những dạng bài tập mà chúng ta học ở trường thì hầu hết đều có thể viết lại bằng ngôn ngữ C. Dưới đây mình có làm một số bài liên quan đến môn Giải tích, Xác suất – Thống kê, Địa Lý, Khí hậu,....Do điều kiện không cho phép nên mình không làm được nhiều bài tập, mình chỉ giới thiệu những bài có tính khái quát và có thể do đề bài yêu cầu nhiều nội dung nên chương trình khá dài. Mình sẽ cố gắng giải thích dễ hiểu nhất có thể, hy vọng các bạn có thể hiểu ý mình. Thank!

Bài 150: Có hai người cùng thực hiện một công việc nào đó. Xác suất thực hiện thành công của người thứ nhất là a và xác suất thực hiện thành công của người thứ hai là b (với 0 < a, b <1). Mỗi người được thực hiện công việc đó n lần (n nguyên dương). Gọi X và Y lần lượt là số lần thực hiện thành công của người thứ nhất và người thứ hai:

a) Lập bảng phân phối xác suất của X và Y. Tính EX, EY, DX, DY, σ_X, σ_Y

b) Tính xác suất để số lần thực hiện thành công của hai người là như nhau.

c) Tính xác suất để số lần thành công của người thứ nhất lớn hơn của người thứ hai.

d) Gọi Z là tổng số lần thực hiện thành công của cả hai người (Z = X + Y). Lập bảng phân phối xác suất của Z. Tính EZ, DZ, σ_Z

e) Z có phân phối nhị thức hay không.

g) Tính xác suất để đại lượng ngẫu nhiên G = 2X + 3Y là một số chẵn.

h) Nếu người thứ nhất thực hiện một lần không thành công thì bị phạt 3 cái tát còn người thứ hai thực hiện một lần không thành công thì bị phạt 2 cú đấm. Tính số đơn vị đau trung bình mà cả 2 người bị phạt, biết rằng 1 cái tát bằng 1,5 đơn vị đau và 1 cú đấm bằng 2 đơn vị đau.

Mỗi người thực hiện công việc là độc lập với nhau, các giá trị a, b, n là bất kỳ và được nhập từ bàn phím.

Lời giải:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
int GiaiThua (int x);
int ToHop (int y,int z);
float KyVong(float z[],int n);
float PhuongSai(float k[],int n);
```

```
void SapXep(float m[],int n);
main()
{
    int n,i,j,k,l,o;
    float
a,b,p,M,H,V,KV,X[100],Y[100],Z[100],G[100],P1[100],P2[100],Q1[100],Q2[
100],R[100],T[100],W[100],L[100];
    tieptuc:printf("\n  Nhap vao xac suat thanh cong tuong ung cua hai nguoi:
");
    scanf("%f%f",&a,&b);
    while(a<=0||a>=1||b<=0||b>=1)
      { printf("  Nhap lai cac xac suat:  ");
        scanf("%f%f",&a,&b);
      }
    printf("  Nhap vao so lan duoc thuc hien:  ");
    scanf("%d",&n);
    while (n<=0)
      { printf("  Nhap lai so lan thuc hien:  ");
        scanf("%d",&n);
      }

    printf("\n  a)\n  => Bang phan phoi xac suat cua X la:\n\n");
    for(i=0;i<=n;i++)
      X[i]=ToHop(i,n)*pow(a,i)*pow(1-a,n-i);
    printf("\tX       |");
    for(i=0;i<=n;i++)
      printf("%8d",i);
    printf("\n     ");
    for(i=1;i<=15+8*(n+1);i++)
      printf("-");
    printf("\n      P(X = xi) |    ");
    for(i=0;i<=n;i++)
```

```
    printf("%.4f  ",X[i]);
  printf("\n\n     Ky vong cua X la:  EX = %.4f",KyVong(X,n));
  printf("\n     Phuong sai cua X la:  DX = %.4f",PhuongSai(X,n));
  printf("\n     Do lech chuan cua X la: Delta(X) = %.4f\n",sqrt(PhuongSai(X,n)));

  printf("\n  => Bang phan phoi xac suat cua Y la:\n\n");
  for(i=0;i<=n;i++)
    Y[i]=ToHop(i,n)*pow(b,i)*pow(1-b,n-i);
  printf("\tY       |");
  for(i=0;i<=n;i++)
    printf("%8d",i);
  printf("\n     ");
  for(i=1;i<=15+8*(n+1);i++)
    printf("-");
  printf("\n      P(Y = yi) |    ");
  for(i=0;i<=n;i++)
    printf("%.4f  ",Y[i]);
  printf("\n\n     Ky vong cua Y la:  EY = %.4f",KyVong(Y,n));
  printf("\n     Phuong sai cua Y la:  DY = %.4f",PhuongSai(Y,n));
  printf("\n     Do lech chuan cua Y la: Delta(Y) = %.4f\n",sqrt(PhuongSai(Y,n)));

  printf("\n  b)\n  => Xac suat de so lan thuc hien thanh cong cua hai nguoi bang nhau la:  ");
  M=0;
  for(i=0;i<=n;i++)
    M+=X[i]*Y[i];
  printf("%.4f\n",M);

  printf("\n  c)\n  => Xac suat de so lan thanh cong cua nguoi thu nhat lon hon cua nguoi thu hai la:  ");
```

```
H=0;
for(i=0;i<=n;i++)
  for(j=0;j<i;j++)
    H+=X[i]*Y[j];
printf("%.4f\n",H);

printf("\n  d)\n     Z la tong so lan thuc hien thanh cong cua hai nguoi.\n");
printf("  => Bang phan phoi xac suat cua Z la:\n\n");
for(i=0;i<=2*n;i++)
  { Z[i]=0;
    for(j=0;j<=n;j++)
      for(k=0;k<=n;k++)
        if(i==j+k)
          Z[i]+=X[j]*Y[k];
  }
printf("\tZ       |");
for(i=0;i<=2*n;i++)
  printf("%8d",i);
printf("\n     ");
for(i=1;i<=15+8*(2*n+1);i++)
  printf("-");
printf("\n      P(Z = zi) |    ");
for(i=0;i<=2*n;i++)
  printf("%.4f  ",Z[i]);
printf("\n\n     Ky vong cua Z la:  EZ = %.4f",KyVong(Z,2*n));
printf("\n     Phuong sai cua Z la:  DZ = %.4f",PhuongSai(Z,2*n));
printf("\n                Do lech chuan cua Z la: Delta(Z) =
%.4f\n",sqrt(PhuongSai(Z,2*n)));

printf("\n  e)\n");
k=1;
p=1-pow(Z[0],(float)1/2/n);
```

```
    for(i=0;i<=2*n;i++)
      if(fabs(Z[i]-ToHop(i,2*n)*pow(p,i)*pow(1-p,2*n-i))>0.000001)
        k=0;
    if(k==0)printf("  => Z khong co phan phoi nhi thuc.\n");
    else printf("  => Z co phan phoi nhi thuc.\n");

    printf("\n  g)\n     Dai luong ngau nhien G = 2X + 3Y.\n");
    printf("  => Bang phan phoi xac suat cua G la:\n\n");
    for(i=0;i<=5*n;i++)
      { G[i]=0;
       for(j=0;j<=n;j++)
         for(k=0;k<=n;k++)
           if(i==2*j+3*k)
             G[i]+=X[j]*Y[k];
      }
    printf("\tG       |");
    for(i=0;i<=5*n;i++)
      printf("%8d",i);
    printf("\n     ");
    for(i=1;i<=15+8*(5*n+1);i++)
      printf("-");
    printf("\n      P(G = gi) |    ");
    for(i=0;i<=5*n;i++)
      printf("%.4f  ",G[i]);
    printf("\n\n  => Xac suat de G = 2X + 3Y la mot so chan bang:  ");
    V=0;
    for(i=0;i<=5*n;i++)
      if(i%2==0)V+=G[i];
    printf("%.4f\n",V);

    printf("\n  h)\n  => Bang phan bo xac suat so don vi dau cua nguoi thu
nhat:\n\n");
```

```
a=1-a;
b=1-b;
for(i=0;i<=n;i++)
  P1[i]=4.5*i;
for(i=0;i<=n;i++)
  P2[i]=ToHop(i,n)*pow(a,i)*pow(1-a,n-i);
printf("\tR       |");
for(i=0;i<=n;i++)
  printf("%8.1f",P1[i]);
printf("\n     ");
for(i=1;i<=15+8*(n+1);i++)
  printf("-");
printf("\n      P(R = ri) |    ");
for(i=0;i<=n;i++)
  printf("%.4f  ",P2[i]);

printf("\n\n          Bang phan bo xac suat so don vi dau cua nguoi thu hai:\n\n");
for(i=0;i<=n;i++)
  Q1[i]=4*i;
for(i=0;i<=n;i++)
  Q2[i]=ToHop(i,n)*pow(b,i)*pow(1-b,n-i);
printf("\tT       |");
for(i=0;i<=n;i++)
  printf("%8.1f",Q1[i]);
printf("\n     ");
for(i=1;i<=15+8*(n+1);i++)
  printf("-");
printf("\n      P(T = ti) |    ");
for(i=0;i<=n;i++)
  printf("%.4f  ",Q2[i]);
```

```
k=0;
for(i=0;i<=n;i++)
  for(j=0;j<=n;j++)
    { R[k]=P1[i]+Q1[j];
      k++;
    }
SapXep(R,k);
W[0]=R[0];
i=1;
for(j=1;j<k;j++)
  if(R[j]!=R[j-1])
    { W[i]=R[j];
      i++;
    }
l=i;
for(i=0;i<l;i++)
  { L[i]=0;
    for(j=0;j<=n;j++)
      for(k=0;k<=n;k++)
        if(W[i]==P1[j]+Q1[k])
          L[i]+=P2[j]*Q2[k];
  }

printf("\n\n     Bang phan bo xac suat cua tong so don vi dau la:\n\n");
n=l;
printf("\tK       |");
for(i=0;i<n;i++)
  printf("%7.1f",W[i]);
printf("\n     ");
for(i=1;i<=15+7*n;i++)
  printf("-");
```

```
    printf("\n      P(K = ki) |  ");
    for(i=0;i<n;i++)
      printf("%.4f ",L[i]);
    KV=0;
    for(i=0;i<n;i++)
      KV+=W[i]*L[i];
    printf("\n\n    => So don vi dau trung binh ma ca hai nguoi bi phat la: %.4f\n\n",KV);


    getch();
    printf("  Nhap vao phim 1 de tiep tuc, nhap phim khac de thoat:  ");
    scanf("%d",&o);
    if(o==1)
      {printf("\n\n");
       for(i=1;i<=148;i++)
          printf("_");printf("\n");
       goto tieptuc;
      }
    else
      printf("\n\n\n\t\t\t*The end*\n\t\t   Chuc cac ban may man!\n");
    for(i=1;i<=148;i++)
          printf("_");printf("\n");
    getch();
}


int GiaiThua (int x)
{
    if(x<=1)return 1;
    else return x*GiaiThua(x-1);
}
int ToHop (int y,int z)
{
```

```
        return GiaiThua(z)/GiaiThua(y)/GiaiThua(z-y);
    }
    float KyVong(float z[],int n)
    {
        int i;
        float EX=0;
        for(i=0;i<=n;i++)
          EX+=i*z[i];
        return (EX);
    }
    float PhuongSai(float k[],int n)
    {
        int i;
        float EX2=0;
        for(i=0;i<=n;i++)
          EX2+=i*i*k[i];
        return (EX2-pow(KyVong(k,n),2));
    }
    void SapXep(float m[],int n)
    {
        int i,j;
        float TrungGian;
        for(i=0;i<n-1;i++)
          for(j=i+1;j<n;j++)
            if(m[i]>m[j])
              { TrungGian=m[i];
                m[i]=m[j];
                m[j]=TrungGian;
              }
    }
```

* Giải thích:

Bài 200:

Hết

Mọi người có thể vào địa chỉ sau để download file pdf này cũng như phần mềm C/C++ và tài liệu hướng dẫn học C, các bài tập – bài thi của những lớp khác, trường khác,…

Mail: dhkhtnhn334@gmail.com

Mật khẩu: daihoctunhien

Sau đó vào thư mục: HỌC TẬP/ THCS 3 ở khung bên trái.

Mình sẽ cố gắng cập nhật liên tục các bài tập theo từng buổi học!

Lưu ý: Phần mềm C/C++ đã được cài đặt sẵn, chỉ cần copy cả folder vào ổ C và đưa biểu tượng ra desktop là chạy bình thường.

Có gì thắc mắc thì liên hệ với mình: 0974.971.149

Mail : hoangtronghus@yahoo.com.vn