|  |  |
| --- | --- |
| COMMUNITY S2B | THE TRAINING PROGRAMME FOR MEMBER  COURSE C/C++, CODE : C001 |
| PROGRAMMING PTIT CLUB |

**PROGRAMMING C/C++**

Họ và Tên :Trần Phú Quang

Lớp : D13CN3

MS-TV :B13DCCN161

# DESCIPTION

Đây là bản tài liệu dùng để lưu lại tất cả những kiến thức mà bạn đã học được trong các chương trình đào tạo mà S2B đưa ra với môn C/C++ cho các bạn.

Tất cả mọi kiến thức và kinh nghiệm bạn thu nhặt được sẽ được lưu giữ ở đây. Điều này sẽ giúp cho bạn có một kiến thức vững chắc hơn và đồng thời giúp S2B, cũng như câu lạc bộ trực thuộc S2B mà bạn tham gia có thể theo sát quá trình học tập của bạn.

Tài liệu quan trọng của bạn.

## S2B Note

Những gì bạn biết chỉ là một giọt nước, những gì bạn chưa biết là cả một đại dương.

Mong bạn sẽ cố gắng trong quá trình học tập đồng thời cũng rèn luyện cho mình những đức tính sau :

CHĂM CHỈ : Những thành tựu vĩ đại không được gặt hái bằng sức mạnh mà bằng sự chăm chỉ và kiên trì.

ĐÚNG GIỜ : Đúng giờ là việc thể hiện phép lịch sự , tôn trọng chính mình và tôn trọng người khác.

TRÁCH NHIỆM : Bạn phải có trách nhiệm với những gì bạn nói, những gì bạn làm. Điều đó thể hiện giá trị của con người bạn.

TRUNG THỰC : Hãy thành thật với chính mình và với những người khác, đó là cách tốt nhất để nuôi dưỡng tâm hồn bạn.

KHIÊM  TỐN : Không ai giỏi hết tất cả mọi thứ và “Trí tuệ con người trưởng thành trong tĩnh lặng và tính cách trưởng thành trong bão táp".

NHIỆT TÌNH : Bạn là tuổi trẻ, bạn có nhiệt huyết vậy học hết mình và chơi hết mình, luôn cháy trong tất cả các hoạt động. Hãy để cho tuổi trẻ của bạn là những trang sách thật đẹp.

CHIA SẺ : Cho đi là còn mãi, hãy chia sẻ những điều bạn biết bạn sẽ nhận được rất nhiều điều thú vị. Và nhớ là đừng bao giờ đi ăn một mình.

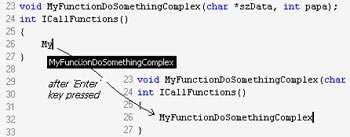
BIẾT ƠN : Uống nước nhớ nguồn, vì những gì mà S2B đã và đang mang lại cho bạn thì mong bạn hãy cùng chung tay xây dựng S2B, để S2B ngày càng phát triển và giúp được nhiều sinh viên hơn nữa.

|  |
| --- |
| Thay mặt cộng đồng S2B, chủ tịch :  CHẾ ĐÌNH SƠN |

## Your target – don’t forget

# SETUP ENVIRONMENT

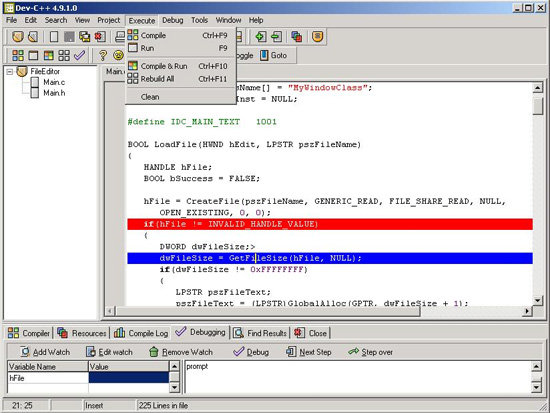
## C-Free

Ngôn ngữ C được xem là một trong số ngôn ngữ lập trình cơ bản mà hầu như sinh viên khoa CNTT của các trường cao đẳng, đại học đều gặp phải. Thế nhưng, việc viết code trong môi trường Borland C quá đơn điệu, đôi khi bạn gặp phải những lỗi hết sức cơ bản (viết sai tên hàm chẳng hạn) và…rối mắt với những câu báo lỗi “nhanh nhạy” của Borland C. Với C-Free bạn sẽ khắc phục được tình trạng này.   
  
C-Free - đây là một trình ứng dụng được thiết kế dành riêng cho ngôn ngữ lập trình “họ nhà C”. Với C-Free, việc viết code sẽ trở nên dễ dàng hơn rất nhiều, bạn chỉ cần gõ từ đầu tiên của từ khóa, chương trình sẽ tự động điền các từ khóa thích hợp vào cho bạn. Không chỉ thế, C-Free còn “học thuộc” cả những tên hàm mà bạn đã khai báo và sẽ tự động gọi tên hàm tương tự như cách gọi các từ khóa (đây là tính năng rất mới có tên gọi là Intelligent Input của C-Free).   
  
Cách thức sử dụng chương trình rất đơn giản:Bạn bấm biểu tượng New (hoặc dùng tổ hợp phím Ctrl+N) để tạo một tập tin mới, tập tin này sẽ có định dạng là .cpp. Trong khung soạn thảo của chương trình, bạn chỉ việc gõ những dòng code, nếu là từ khóa, chương trình sẽ tự động hiển thị giúp các phần còn lại.   
  
Không chỉ có thế, C-Free còn có rất nhiều tính năng rất hay khác, chẳng hạn:   
  
+ Bạn có thể dễ dàng tra cứu công dụng của một lệnh nào đó bằng cách tô đen dòng lệnh và nhấn phím F1.   
+ Khi biên dịch chương trình (phím F9), nếu code bị lỗi, bạn chỉ việc nhấp vào dòng thông báo, chương trình sẽ báo vị trí phát sinh lỗi.   
+ Dễ dàng biên dịch thông qua biểu tượng hoặc phím F5.   
+ Dễ dàng tạo và đưa file vào Project thông qua menu Project...   
  
C-Free còn rất nhiều tính năng hữu dụng khác, bạn có thể tự tìm hiểu thêm trong mục Help của chương trình. Download bản dùng thử (C-Free 3.5.2 - dung lượng 7248 KB) hoặc bản miễn phí (C-Free 2.0 – dung lượng 6529

Link download: http://www.programarts.com/cfree\_en/download.htm

## Dev-C

Bloodshed Dev-C++ là IDE dành cho C/C++. C/C++ là một ngôn ngữ lập trình cực mạnh, có khả năng tương tác cao và thích ứng với các hệ điều hành khác nhau. Hiện nay, có rất nhiều IDE hỗ trợ lập trình C/C++ như Turbo C, Visual C++, … Và Bloodshed Dev-C++ là IDE Portable đầu tiên.  
  
Chương trình hỗ trợ đầy đủ các chức năng mà một IDE chuyên nghiệp, từ việc soạn thảo, trợ giúp đến debug và hỗ trợ thư viện cho các lập trình viên C/C++ (kể cả viết chương trình cho DOS và chương trình cho Windows). Với IDE này, bạn dễ dàng tạo ra một chương trình C/C++ một cách chuyên nghiệp, nhanh chóng và đơn giản.

[](http://im.dowload.vn/Data/image/2013/Thang01/19/bloodshed-dev-c.jpg)

Link download: <http://sourceforge.net/projects/dev-cpp/files/Binaries/Dev-C%2B%2B%204.9.9.2/devcpp-4.9.9.2_setup.exe/download>

# TỰA ĐỀ

Thay mặt cho những người biên soạn tập tài liệu này,Tôi có đôi lời muốn nói với các bạn.

Trước hết, những người soạn tài liệu này KHÔNG phải là GIÁO SƯ, TIẾN SĨ…. Hay có một trình độ học vấn uyên bác nào đó. Nhưng tất cả những kiến thức trong này đều được tích cóp từ những nguồn đánh tin cậy từ “bố đẻ google”., đó là những phần kiến thức lý thuyết mà các THẦY GIÁO/ GIÁO SƯ/ TIẾN SĨ đã từng giảng dạy từ các trường đại học, nhưng đã được biến chuyển, hay chuyển thể sang cách học “ Bình dân” nhất để những người mới bắt đầu không thấy lạ lẫm hay quá khó khăn để bắt đầu với một ngôn ngữ lập trình.

Những người biên soạn chúng tôi, đã từng là những người chưa biết gì về lập trình, và chúng tôi đã phải học , phải sai để có thể ngồi đây biên soạn ra tập tài liệu này. Là một người đi trước, nên chúng tôi sẽ biết được bạn sẽ sai đâu, sẽ vướng mắc ở đâu, vậy nên trong bộ tài liệu này, chúng tôi đã giảm tải những gì quá cao siêu hoặc uyên bác mà các nhà lập trình giỏi họ nói, để hạ mức của nó xuống, phổ thông giúp các bạn có thể dễ dàng tiếp cận hơn. Vì vậy tôi hi vọng đây là tập tài liệu hữu ích với các bạn.

Hơn thế nữa, chúng tôi cũng là Sinh viên như chính các bạn, vì vậy tầm hiểu biết không thể quá sâu rộng có thể giúp bạn giải đáp mọi thắc mắc được.Cũng vì vậy tập tài liệu này sẽcó dôi chỗ còn thiếu sót, mong các bạn sẽ cùng đóng góp, nghiên cứu trao đổi để chúng ta có một bộ tài liệu Đầy đủ nhất có thể. Chúng tôi cần sự hợp tác của các bạn.

Điều cuối cùng, dù sau này bạn là một coder giỏi hay chăng nữa, bạn chắc chắn không thể bỏ qua điều này. Đó là **code invention – Phong cách lập trình C/C++.**

Code đẹp không chỉ để cho nó đẹp,khi các bạn cần làm project lớn chúng ta đều phải tuân thủ theo một nguyên tắc nhất định. VD như cả thế giới viết số một là ‘ 1 ‘ thì ai cũng hiều, nhưng nếu bạn tự tạo cho mình một chuẩn mực khác, một phong cách khác số một la ‘ 2’. Thì chỉ mình bạn hiểu bạn, như vậy bạn không thể giao tiếp với mọi người. Vì vậy mình rất đề cao về vấn đề **CODE INVENTION.**

Dưới đây là link tài liệu về **CODE INVENTION**, mình mong các bạn có thể đọc, nhớ dần dần theo những chhuẩn chung đó, để tạo cho mình một phong cách lập trình “đẹp” nhất.

<http://www.mediafire.com/download/amjvuoh9dtb1q3g/Phong+cach+lap+trinh-C%2B%2B.pdf>

Chúc các bạn thành công!

Thân ái.

(Nguyễn Linh)

# BASIC

## Module 1: làm quen với ngôn ngữ lập trình C

### Lý thuyết

#### Ngôn ngữ C:

Vào đầu những năm 70 tại phòng thí nghiệm Bell, Dennis Ritchie đã phát triển ngôn ngữ C. C được sử dụng lần đầu trên một hệ thống cài đặt hệ điều hành UNIX. C có nguồn gốc từ ngôn ngữ BCPL do Martin Richards phát triển. BCPL sau đó đã được Ken Thompson phát triển thành ngôn ngữ B, đây là người khởi thủy ra C.

Trong khi BCPL và B không hỗ trợ kiểu dữ liệu, thì C đã có nhiều kiểu dữ liệu khác nhau. Những kiểu dữ liệu chính gồm : kiểu ký tự (character), kiểu số nguyên (interger) và kiểu số thực (float).

C liên kết chặt chẽ với hệ thống UNIX nhưng không bị trói buộc vào bất cứ một máy tính hay hệ điều hành nào. C rất hiệu quả để viết các chương trình thuộc nhiều những lĩnh vực khác nhau.

C cũng được dùng để lập trình hệ thống. Một chương trình hệ thống có ý nghĩa liên quan đến hệ điều hành của máy tính hay những tiện ích hỗ trợ nó. Hệ điều hành (OS), trình thông dịch (Interpreters), trình soạn thảo (Editors), chương trình Hợp Ngữ (Assembly) là các chương trình hệ thống. Hệ điều hành UNIX được phát triển dựa vào C. C đang được sử dụng rộng rãi bởi vì tính hiệu quả và linh hoạt. Trình biên dịch (compiler) C có sẵn cho hầu hết các máy tính. Mã lệnh viết bằng C trên máy này có thể được biên dịch và chạy trên máy khác chỉ cần thay đổi rất ít hoặc không thay đổi gì cả. Trình biên dịch C dịch nhanh và cho ra mã đối tượng không lỗi.

C khi thực thi cũng rất nhanh như hợp ngữ (Assembly). Lập trình viên có thể tạo ra và bảo trì thư viện hàm mà chúng sẽ được tái sử dụng cho chương trình khác. Do đó, những dự án lớn có thể được quản lý dễ dàng mà tốn rất ít công sức.

C được hiểu là ngôn ngữ bậc trung bởi vì nó kết hợp những yếu tố của những ngôn ngữ cấp cao và những chức năng của hợp ngữ (ngôn ngữ cấp thấp). C cho phép thao tác trên những thành phần cơ bản của máy tính như bits, bytes, địa chỉ…. Hơn nữa, mã C rất dễ di chuyển nghĩa là phần mềm viết cho loại máy tính này có thể chạy trên một loại máy tính khác. Mặc dù C có năm kiểu dữ liệu cơ bản, nhưng nó không được xem ngang hàng với ngôn ngữ cao cấp về mặt kiểu dữ liệu. C cho phép chuyển kiểu dữ liệu. Nó cho phép thao tác trực tiếp trên bits, bytes, word và con trỏ (pointer). Vì vậy, nó được dùng cho lập trình mức hệ thống.

Thuật ngữ ngôn ngữ cấu trúc khối (block-structured language) không áp dụng với C. Ngôn ngữ cấu trúc khối cho phép thủ tục (procedures) hay hàm (functions) được khai báo bên trong các thủ tục và hàm khác. C không cho phép việc tạo hàm trong hàm nên nó không phải là ngôn ngữ cấu trúc khối. Tuy nhiên, nó được xem là ngôn ngữ cấu trúc vì nó có nhiều điểm giống với ngôn ngữ cấu trúc ALGOL, Pascal và một số ngôn ngữ tương tự khác.

C cho phép có sự tổng hợp của mã lệnh và dữ liệu. Ðiều này là một đặc điểm riêng biệt của ngôn ngữ cấu trúc. Nó liên quan đến khả năng tập hợp cũng như ẩn dấu tất cả thông tin và các lệnh khỏi phần còn lại của chương trình để dùng cho những tác vụ riêng biệt. Ðiều này có thể thực hiện qua việc dùng các hàm hay các khối mã lệnh (Code Block). Các**hàm**được dùng để định nghĩa hay tách rời những tác vụ được yêu cầu trong chương trình. Ðiều này cho phép những chương trình hoạt động như trong một đơn vị thống nhất. **Khối mã lệnh** là một nhóm các câu lệnh chương trình được nối kết với nhau theo một trật tự logic nào đó và cũng được xem như một đơn vị thống nhất. Một khối mã lệnh được tạo bởi một tập hợp nhiều câu lệnh tuần tự giữa dấu ngoặc mở và đóng xoắn như dưới đây:

do

{

i = i + 1;

} while (i < 40);

Ngôn ngữ cấu trúc hỗ trợ nhiều cấu trúc dùng cho vòng lặp (loop) như là **while,do-while,** và **for.** Những cấu trúc lặp này giúp lập trình viên điều khiển hướng thực thi trong chương trình.

#### Cấu trúc chương trình C:

C có một số từ khóa, chính xác là 32. Những từ khóa này kết hợp với cú pháp của C hình thành ngôn ngữ C. Nhưng nhiều trình biên dịch cho C đã thêm vào những từ khóa dùng cho việc tổ chức bộ nhớ ở những giai đoạn tiền xử lý nhất định.

Vài quy tắc khi lập trình C như sau :

-Tất cả từ khóa là chữ thường (không in hoa)

-Ðoạn mã trong chương trình C có phân biệt chữ thường và chữ hoa. Ví dụ :**do while** thì khác với **DO WHILE**

-Từ khóa không thể dùng cho các mục đích khác như đặt tên biến (variable name) hoặc tên hàm (function name)

-Hàm main() luôn là hàm đầu tiên được gọi đến khi một chương trình bắt đầu chạy (chúng ta sẽ xem xét kỹ hơn ở phần sau)

**Ghi chú:** Những khía cạnh khác nhau của chương trình C được xem xét qua đoạn mã trên. Ðoạn mã này xem như là đoạn mã mẫu, nó sẽ được dùng lại trong suốt phần còn lại của giáo trình này.

#### Hàm :

Chương trình C được chia thành từng đơn vị gọi là hàm. Ðoạn mã mẫu chỉ có duy nhất một hàm main(). Hệ điều hành luôn trao quyền điều khiển cho hàm main() khi một chương trình C được thực thi. Tên hàm luôn được theo sau là cặp dấu ngoặc đơn (). Trong dấu ngoặc đơn có thể có hay không có những tham số (parameters).

#### Dấu phân cách:

Sau định nghĩa hàm sẽ là dấu ngoặc xoắn mở **{**. Nó thông báo điểm bắt đầu của hàm. Tương tự, dấu ngoặc xoắn đóng **}** sau câu lệnh cuối trong hàm chỉ ra điểm kết thúc của hàm. Dấu ngoặc xoắn mở đánh dấu điểm bắt đầu của một khối mã lệnh, dấu ngoặc xoắn đóng đánh dấu điểm kết thúc của khối mã lệnh đó. Trong đoạn mã mẫu có 2 câu lệnh giữa 2 dấu ngoặc xoắn.  
Hơn nữa, đối với hàm, dấu ngoặc xoắn cũng dùng để phân định những đoạn mã trong trường hợp dùng cho cấu trúc vòng lặp và lệnh điều kiện..

#### Kết thúc câu lệnh:

Dòng **int i = 0;** trong đoạn mã mẫu là một câu lệnh (statement). Một câu lệnh trong C thì được kết thúc bằng dấu chấm phẩy (**;)**. C không hiểu việc xuống dòng dùng phím Enter, khoảng trắng dùng phím spacebar hay một khoảng cách do dùng phím tab. Có thể có nhiều hơn một câu lệnh trên cùng một hàng nhưng mỗi câu lệnh phải được kết thúc bằng dấu chấm phẩy. Một câu lệnh không được kết thúc bằng dấu chấm phẩy được xem như một câu lệnh sai.

#### Comment:

Những chú thích thường được viết để mô tả công việc của một lệnh đặc biệt, một hàm hay toàn bộ chương trình. Trình biên dịch sẽ không dịch chúng. Trong C, chú thích bắt đầu bằng ký hiệu /\* và kết thúc bằng \*/. Trường hợp chú thích có nhiều dòng, ta phải chú ý ký hiệu kết thúc (\*/), nếu thiếu ký hiệu này, toàn bộ chương trình sẽ bị coi như là một chú thích. Trong đoạn mã mẫu dòng chữ "This is a sample program" là dòng chú thích. Trong trường hợp chú thích chỉ trên một dòng ta có thể dùng //. Ví dụ:

int a = 0; // Biến ‘a’ đã được khai báo như là một kiểu số nguyên (interger)

#### Thư viện:

Tất cả trình biên dịch C chứa một thư viện hàm chuẩn dùng cho những tác vụ chung. Một vài bộ cài đặt C đặt thư viện trong một tập tin (file) lớn trong khi đa số còn lại chứa nó trong nhiều tập tin nhỏ. Khi lập trình, những hàm được chứa trong thư viện có thể được dùng cho nhiều loại tác vụ khác nhau. Một hàm (được viết bởi một lập trình viên) có thể được đặt trong thư viện và được dùng bởi nhiều chương trình khi được yêu cầu. Vài trình biên dịch cho phép hàm được thêm vào thư viện chuẩn trong khi số khác lại yêu cầu tạo một thư viện riêng.

**<assert.h>**: Bao gồm [macro](http://vi.wikipedia.org/w/index.php?title=Macro&action=edit&redlink=1) [assert](http://vi.wikipedia.org/w/index.php?title=Assert&action=edit&redlink=1) dùng để hỗ trợ trong việc phát hiện các lỗi lô-gíc và các kiểu lỗi khác trong các phiên bản dùng để tìm lỗi của một chương trình.

**<complex.h>**: Một tập hợp các hàm dùng để điều chỉnh các [số phức](http://vi.wikipedia.org/wiki/S%E1%BB%91_ph%E1%BB%A9c) (mới có trong chuẩn **C99**).

**<**[**ctype.h**](http://vi.wikipedia.org/w/index.php?title=Ctype.h&action=edit&redlink=1)**>**: bao gồm các hàm dùng để phân lớp các kí tự bởi các kiểu hay dùng để chuyển đổi giữa chữ viết hoa và viết thường.

**<errno.h>**: Dùng để thử (hay hiển thị) các lỗi được báo cáo từ các hàm thư viện.

**<fenv.h>**: Dùng để kiểm soát môi trường [chấm động](http://vi.wikipedia.org/w/index.php?title=Ch%E1%BA%A5m_%C4%91%E1%BB%99ng&action=edit&redlink=1) (mới có trong **C99**).

**<float.h>**: Bao gồm định nghĩa các hằng mà nêu ra các đặc tính xây dựng của thư viện [chấm động](http://vi.wikipedia.org/w/index.php?title=Ch%E1%BA%A5m_%C4%91%E1%BB%99ng&action=edit&redlink=1), như là sự khác nhau nhỏ nhất của hai số chấm động (có thể có) qua \_EPSILON, số lớn nhất của các chữ số của độ chính xác qua \_DIG và khoảng cách của các số mà có thể biểu thị được qua \_MIN và \_MAX.

**<inttypes.h>**: Dùng cho việc chuyển đổi kiểu chính xác giữa các kiểu nguyên (mới có trong **C99**).

**<iso646.h>**: Để lập trình trong [ISO 646](http://vi.wikipedia.org/w/index.php?title=ISO_646&action=edit&redlink=1) cho các bộ kí tự khác nhau (mới có trong [**NA1**](http://vi.wikipedia.org/wiki/Th%C6%B0_vi%E1%BB%87n_chu%E1%BA%A9n_C#NA1)).

**<limits.h>**: Chứa định nghĩa các hằng có đặc tính đặc biệt của các kiểu nguyên, như là khoảng cách của các số mà có thể biểu thị quan \_MIN, \_MAX.

Contains defined constants specifying the implementation-specific properties of the integer types, such as the range of numbers which can be represented (\_MIN, \_MAX).

**<locale.h>**: Dùng cho setlocale() và các hằng có liên quan. Việc này đuợc dùng để lựa chọn [địa phương](http://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BB%8Ba_ph%C6%B0%C6%A1ng&action=edit&redlink=1) cần thiết.

***<math.h>****: Cho việc tính các hàm số thông dụng.*

**<setjmp.h>**: Khai báo [setjump/longjump](http://vi.wikipedia.org/w/index.php?title=Setjump/longjump&action=edit&redlink=1) đưuợc dùng trong việc thoát ra của nơi không có tính địa phương.

**<signal.h>**: Để kiểm soát các điều kiện ngoại lệ.

**<stdarg.h>**: Để truy cập số lượng khác nhau của các đối số được chuyển vào hàm.

**<stdbool.h>**: Dùng cho khiểu dữ liệu [Bool](http://vi.wikipedia.org/w/index.php?title=Bool&action=edit&redlink=1) (mới có trong **C99**).

**<stdint.h>** : Dùng trong việc định nghĩa các kiểu nguyên khác nhau (mới có trong **C99**).

**<stddef.h>**: Cung cấp nhiều kiểu và macro hữu dụng.

***<stdio.h>****: Cung cấp cốt lõi của những khả năng nhập trong C. Tập tin này bao gồm họ hàm*[*printf*](http://vi.wikipedia.org/w/index.php?title=Printf&action=edit&redlink=1).

***<stdlib.h>****: Dùng để xúc tiến nhiều phép toán, bao gồm sự chuyển đổi, các số giả ngầu nhiên, cấp phát vùng nhớ, kiểm soát quá trình, môi trường, tín hiệu, tìm kiếm, và xếp thứ tự.*

***<string.h>****: Để điều chỉnh nhiều loại dãy kí tự.*

**<tgmath.h>**: Dùng cho các hàm toán kiểu thông dụng (mới có trong **C99**).

: Để chuyển đổi giữa các định dạng khác nhau về thì giờ và ngày tháng.

**<wchar.h>**: Để điều chỉnh độ rộng của các dòng (dữ liệu) và nhiều loại dãy kí tự sử dụng nhiều (loại) kí tự có độ lớn (Unicode chẳng hạn). -- Đây là chìa khóa để hỗ trợ nhiều ngôn ngữ (mới có trong **NA1**).

**<wctype.h>**: Để phân lớp các kí tự có độ lớn (mới có trong **NA1**).

#### Các nguyên tắc cho việc đặt tên

Các quy tắc đặt tên biến khác nhau tuỳ ngôn ngữ lập trình. Tuy nhiên, vài quy ước chuẩn được tuân theo như :

Tên biến phải bắt đầu bằng một ký tự chữ cái.

Các ký tự theo sau ký tự đầu bằng một chuỗi các chữ cái hoặc con số và cũng có thể bao gồm ký tự đặc biệt như dấu gạch dưới.

Tránh dùng ký tự O tại những vị trí mà có thể gây lầm lẫn với số không (0) và tương tự chữ cái l (chữ thường của chữ hoa L) có thể lầm lẫn với số 1.

Tên riêng nên tránh đặt tên cho biến.

Theo tiêu chuẩn C các chữ cái thường và hoa thì xem như khác nhau ví dụ. biến ADD, add và Add là khác nhau.

Việc phân biệt chữ hoa và chữ thường khác nhau tuỳ theo ngôn ngữ lập trình. Do đó, tốt nhất nên đặt tên cho biến theo cách thức chuẩn.

Tên một biến nên có ý nghĩa, gợi tả và mô tả rõ kiểu dữ liệu của nó. Ví dụ, nếu tìm tổng của hai số thì tên biến lưu trữ tổng nên đặt là sum (tổng)*.* Nếu đặt tên là *s* hay *ab12* thì không hay lắm.

#### Các toán tử số học và chức năng

Trong C, các toán tử hai ngôi có chức năng giống như trong các ngôn ngữ khác. Những toán tử như +, -, \* và / có thể đư­ợc áp dụng cho hầu hết kiểu dữ liệu có sẵn trong C. Khi toán tử / được áp dụng cho một số nguyên hoặc ký tự, bất kỳ phần dư­ nào sẽ đư­ợc cắt bỏ. Ví dụ, 5/2 sẽ bằng 2 trong phép chia số nguyên. Toán tử % sẽ cho ra kết quả là số dư của phép chia số nguyên. Ví dụ: 5%2 sẽ có kết quả là 1. Tuy nhiên, % không thể đư­ợc sử dụng với những kiểu có dấu chấm động.

Các toán tử Tăng và Giảm

C bao chứa hai toán tử hữu ích mà ta không tìm thấy được trong những ngôn ngữ máy tính khác. Chúng là ++ và --. Toán tử ++ thêm vào toán hạng của nó một đơn vị, trong khi toán tử -- giảm đi toán hạng của nó một đơn vị.

Sự khác nhau giữa việc xử lý trước hay sau trong toán tử một ngôi thật sự có ích khi nó đư­ợc dùng trong một biểu thức. Khi toán tử đứng trư­ớc toán hạng, C thực hiện việc tăng hoặc giảm giá trị tr­ước khi sử dụng giá trị của toán hạng. Ðây là tiền xử lý (pre-fixing). Nếu toán tử đi sau toán hạng, thì giá trị của toán hạng đ­ược sử dụng trư­ớc khi tăng hoặc giảm giá trị của nó. Ðây là hậu xử lý (post-fixing). Xem xét ví dụ sau :

a = 10;

b = 5;

c = a \* b++;

Trong biểu thức trên, giá trị hiện thời của b đư­ợc sử dụng cho tính toán và sau đó giá trị của b sẽ tăng sau. Tức là, c đư­ợc gán 50 và sau đó giá trị của b được tăng lên thành 6.

Tuy nhiên, nếu biểu thức trên là:

c = a \* ++b;

thì giá trị của c sẽ là 60, và b sẽ là 6 bởi vì b được tăng 1 trước khi thực hiện phép nhân với a, sau đó giá trị được gán vào c.

Trong trường hợp mà tác động của việc tăng hay giảm là riêng lẻ thì toán tử có thể đứng trước hoặc sau toán hạng đều được.

Hầu hết trình biên dịch C sinh mã rất nhanh và hiệu quả đối với việc tăng và giảm giá trị. Mã này sẽ tốt hơn so với khi ta dùng toán tử gán. Vì vậy, các toán tử tăng và giảm nên được dùng bất cứ khi nào có thể.

#### Bảng các kiểu dữ liệu

|  |  |  |
| --- | --- | --- |
| **Kiểu** | **Dung lượng xấp xỉ (đơn vị là bit)** | **Phạm vi** |
| char | 8 | -128 tới 127 |
| unsigned | 8 | 0 tới 255 |
| signed char | 8 | -128 tới 127 |
| int | 16 | -32,768 tới 32,767 |
| unsigned int | 16 | 0 tới 65,535 |
| signed int | 16 | Giống như kiểu int |
| short int | 16 | -128 tới 127 |
| unsigned short int | 16 | 0 tới 65, 535 |
| signed short int | 16 | Giống như kiểu short int |
| long int | 32 | -2,147,483,648 tới 2,147,483,647 |
| signed long int | 32 | Giống như kiểu long int |
| unsigned long int | 32 | 0 tới 4,294,967,295 |
| float | 32 | 6 con số thập phân |
| double | 64 | 10 con số thập phân |
| long double | 128 | 10 con số thập phân |

### Exercise

#### Exercise 1

Viết chương trình in ra màn hình chữ helloworld

#### Exercise 2

Viết chương trình nhập vào 2 số a và b, in ra màn hình kết quả a+b, a-b, a\*b

#### Exercise 3

Viết chương trình nhập vào 2 số a và b, in ra màn hình kết quả a/b (lấy sau dấu “,” 2 chữ số)

#### Exercise 4

Viết chương trình nhập vào 2 số a và b (a,b<10^9) in ra màn hình kết quả a+b, a-b, a\*b

## Module 2: cấu trúc rẽ nhánh

### ****Lý thyết****

Câu lệnh if cho phép lựa chọn một trong hai nhánh tùy thuộc vào giá trị của biểu thức logic là đúng (true) hay sai (false)

#### ****if****

**Cú pháp:**

**if (bt\_logic)**

**<khối lệnh>**

Nếu bt\_logic có giá trị đúng thì thực hiện khối lệnh và thoát khỏi if, ngược lại không làm gì cả và thoát khỏi if.

#### ****if else****

**Cú pháp:**

**if (bt\_logic)**

**<khối lệnh 1>;**

**else**

**<Khối lệnh 2>;**

Nếu **bt\_logic**có giá trị đúng thì thực hiện khối lệnh 1 và thoát khỏi if, ngược lại thực hiện khối lệnh 2 và thoát khỏi if.

##### ***Cấu trúc if …else if***

**Cú pháp:**

**if (bt\_logic1)**

**khối lệnh 1;**

**else if (bt\_logic 2)**

**khối lệnh 2;**

**…**

**else if (bt\_logic n-1)**

**khối lệnh n-1;**

**else**

**khối lệnh n;**

Nếu **bt\_logic 1 có giá trị đúng**thì thực hiện khối lệnh 1 và thoát khỏi cấu trúc if. Ngược lại, nếu **bt\_logic 2 có giá trị đúng**thì thực hiện khối lệnh 2 và thoát khỏi cấu trúc if.

Ngược lại, nếu **bt\_logic n-1**đúngthì thực hiện khối lệnh n-1 và thoát khỏi cấu trúc if, ngược lại thì thực hiện khối lệnh n.

#### ****switch****

**Cú pháp:**

**switch (biểu thức)**

**{**

**case giá trị 1 : khối lệnh 1;**

**break;**

**case giá trị 2 : khối lệnh 2;**

**break;**

**…**

**case giá trị n : khối lệnh n;**

**break;**

**default : khối lệnh;**

**[break;]**

**}**

Khi giá trị của **biểu thức** bằng **giá trị i** thì lệnh i sẽ được thực hiện. Nếu sau **lệnh i** không có lệnh **break** thì tiếp tục thực hiện các lệnh sau nó. Ngược lại thoát khỏi  cấu trúc switch. Nếu giá trị biểu thức không trùng với bất kỳ **giá trị i** nào thì lệnh tương ứng với từ khóa default sẽ được thực hiện.

**Lưu ý:**

- Không đặt dấu chấm phẩy sau câu lệnh **switch**.

**- Biểu thức**phải là có kết quả là**giá trị nguyên (char, int, long,…)**

**-** **Lệnh 1, 2…n**có thể gồm nhiều lệnh, nhưng không cần đặt trong cặp dấu { }

### Exercise

#### Exercise 1

Viết chương trình nhập vào số nguyên dương, in ra thông báo số vừa nhập là số chẵn hay lẻ.

#### Exercise 2

Viết chương trình nhập vào 2 số nguyên. Tìm và in ra số lớn nhất.

#### ****Exercise 3****

Viết chương trình giải phương trình bậc nhất: ax +b =0

#### Exercise 4

Viết chương trình giải phương trình bậc 2: ax2 + bx + c = 0, với a, b, c nhập vào từ bàn phím.

#### Exercise 5

Viết chương trình nhập vào giờ phút giây (hh:mm:ss). Cộng thêm số giây nhập vào và in ra kết quả dưới dạng hh:mm:ss.

#### Exercise 6

Viết chương trình nhập vào tháng, in ra tháng đó có bao nhiêu ngày.

#### Exercise 7

Viết chương trìn nhập vào một số, kiểm tra xem số đó chia hết cho 7 ko?

#### Exercise 8

Viết chương trình nhập vào 2 số x, y và 1 trong 4 toán tử +, -, \*, /. Nếu là + thì in ra kết quả x + y, nếu là – thì in ra x – y, nếu là \* thì in ra x \* y, nếu là / thì in ra x / y (nếu y = 0 thì thông báo không chia được).

#### Exercise 9

Viết chương trình nhập vào điểm của một học sinh. In ra xếp loại học tập của học sinh đó. (Cách xếp loại. Nếu điểm >= 9, Xuất sắc. Nếu điểm từ 8 đến cận 9, Giỏi. Nếu điểm từ 7 đến cận 8, Khá. Nếu điểm từ 6 đến cận 7, TBKhá. Nếu điểm từ 5 đến cận 6, TBình. Còn lại là Yếu).

#### Exercise 10

Viết chương trình trình bày khả năng của máy tính của bạn. Người dùng nhập và một ký tự trong bảng chữ cái và chương trình hiển thị ngôn ngữ lập trình tương ứng. Một vài ví dụ nhập và xuất như sau:  
  
Nhập Xuất  
A hoặc a Ada  
B hoặc b Basic  
C hoặc c COBOL  
D hoặc d dBASE III  
F hoặc f Fortran  
P hoặc p Pascal  
V hoặc v Visual C++  
  
Sử dụng lệnh ‘switch’ để chọn và hiển thị thông điệp thích hợp. Sử dụng nhãn default để hiển thị thông điệp nếu ký tự nhập không nằm trong danh sách liệt kê trên.

#### Exercise 11

Viết chương trình nhập vào hai số. Tính hiệu của hai số này. Nếu hiệu số này bằng với một trong hai số đã nhập thì hiển thị thông tin:  
  
Hiệu bằng giá tri <giá trị của số đã nhập vào>  
  
Nếu hiệu không bằng với một trong hai giá trị đã nhập, hiển thị thông tin:  
  
Hiệu không bằng bất kỳ giá trị nào đã được nhập

#### Exercise 12

Công ty A đưa ra các mức trợ cấp cho nhân viên ứng với từng loại nhân viên như sau:  
  
Loại nhân viên Mức trợ cấp  
A 3000  
B 2000  
Những loại khác 1000  
  
Tính lương cuối tháng của nhân viên (Mức lương và loại nhân viên được nhập từ người dùng).

#### Exercise 13

Viết chương trình kiểm tra một số nguyên dương bất kỳ (2 chữ số trở lên, không quá 9

chữ số) có chữ số bắt đầu và kết thúc bằng nhau hay không.

#### Exercise 14

Một tam giác là tam giác cân khi a=b !=c   
tam giác đều khi a=b=c

Tam giác vuông nếu a2+b2=c2

Viết chương trinh nhập vào 3 cạnh a,b,c xuất ra màn hình là tam giác gì? (lưu ý các trường hợp đặc biệt)

#### Exercise 15

Viết chương trình nhập vào số kw điện, tính và xuất ra số tiền phải trả (T) theo công thức sau:

          - Nếu kw<=100 thì Tính 2000 đ cho 1kw

          - Nếu 100<kw<=200 thì những kw vượt 100 tính 2500 đ cho 1 kw

#### Exercise 16

Viết chương trình nhập vào 4 số nguyên. Tìm và in ra số lớn nhất và bé nhất.

#### Exercise 17

Lập trình giải hpt: ax + by = c  
a’x + b’y = c’

#### Exercise 18

Nhập vào 4 số a,b,c,d ( không sử dụng mảng dữ liệu ):

In ra 2 số KHÔNG phải lớn nhất và nhỏ nhất .

#### Exercise 19

Tính tiền cước Taxi. Biết rằng: 1km đầu tiên là 13000đ, mỗi km tiếp theo là 12000đ, nếu lớn hơn 30km thì mỗi km thêm sẽ là 11000đ.

## Module 3: cấu trúc lặp

### Lý thuyết

#### Cấu trúc lặp do...while

Cú pháp:

 do

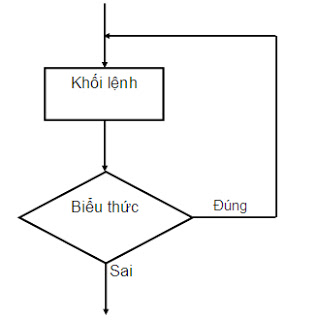
Khối lệnh;

while (biểu thức);

Bước 1: thực hiện khối lệnh sau do.

Bước 2: kiểm tra giá trị biểu thức <biểu thức> sau while, nếu có giá trị ‘đúng’ ( khác 0) thì lặp lại bước 1, nếu ‘sai’ (=0) thì kết thúc vòng lặp.

 Biểu diễn bằng lưu đồ:

[](http://4.bp.blogspot.com/-T1XhDnpgUZU/TgYdCIm76xI/AAAAAAAAABU/QTJmQLCb6Ko/s1600/1.bmp)

#### Cấu trúc lặp "while"

Cú pháp:

            while(biểu thức)

            Khối lệnh;

Sự hoạt động của vòng lặp while:

Bước 1: tính giá trị của (biểu thức) sau while

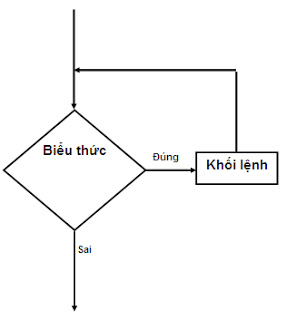
Bước 2: nếu giá trị tính được của (biểu thức) là ‘sai’ (==0) thì kết thúc  vòng lặp while.

Bước 3: nếu giá trị của (biểu thức) là ‘đúng’ (!=0) thì thực hiện khối lệnh sau while.

Bước 4: quay lại bước 1

Chú ý: Thân while có thể được thực hiện một lần hoặc nhiều lần và cũng có thể không được thực hiện lần nào nếu ngay từ đầu biểu thức sau while đã sai.

Biểu diễn bằng lưu đồ:



#### Cấu trúc lặp for

Cú pháp:

            for (biểu thức 1; biểu thức 2; biểu thức 3)

            Khối lệnh;

Trong đó:

    Biểu thức 1: biểu thức khởi đầu.

    Biểu thức 2: biểu thức điều kiện - điều kiện lặp.

    Biểu thức 3: bước nhảy - thường dùng với ý nghĩa là thay đổi bước nhảy.

Cả 3 biểu thức này đều là tuỳ chọn, chúng có thể vắng mặt trong câu lệnh cụ thể nhưng các dấu chấm phẩy vẫn phải có.

Sự hoạt động của vòng lặp for:

Bước 1: Thực hiện biểu thức khởi đầu – Biểu thức 1.

Bước 2: Tính giá trị biểu thức 2 để xác định điều kiện lặp.

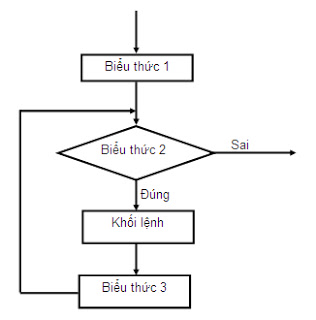
  Nếu biểu thức 2 có giá trị ‘sai’ (==0) thì ra khỏi vòng lặp.

  Ngược lại, nếu biểu thức có giá trị ‘đúng’ ( khác 0) thì chuyển tới bước 3.

Bước 3: Thực hiện khối lệnh sau for ( thân của for ), chuyển tới bước 4.

Bước 4: Thực hiện biểu thức 3, rồi quay về bước 2.

Biểu diễn bằng lưu đồ:

[](http://1.bp.blogspot.com/-ognNzsULg8E/TgYeKlQenLI/AAAAAAAAABY/AVGrzmQ9B0M/s1600/1.bmp)

### Exercise

#### Exercise 1

Nhập số tự nhiên n rồi tính tổng (lưu ý phép chia các số nguyên):

S =1+++...+

#### Exercise 2

Nhập số tự nhiên n rồi tính các tổng sau:

S=tổng các số tự nhiên không lớn hơn n, S1=tổng các số tự nhiên lẻ không lớn hơn n, S2=tổng các số tự nhiên chẵn không lớn hơn n.

#### Exercise 3

Nhập số tự nhiên n rồi tính tổng các số tự nhiên nhỏ hơn n và chia hết cho 7.

#### Exercise 4

Nhập 3 số thực a,b,c rồi giải và biện luận phương trình bậc 2 ax2 + bx + c = 0 .

#### Exercise 5

Viết chương trình C giải bài toán sau: trăm trâu trăm cỏ, trâu đứng ăn 5, trâu nằm ăn 3, lụ khụ trâu già, ba con ăn 1. Hỏi có bao nhiêu trâu đứng, bao nhiêu trâu nằm và bao nhiêu trâu già?

#### Exercise 6

Nhập số tự nhiên n rồi liệt kê các ước số của nó. Có bao nhiêu ước số?

#### Exercise 7

Hãy viết chương trình tính tổng các chữ số của một số nguyên bất kỳ. Ví dụ: Số 8545604 có tổng các chữ số là: 8+5+4+5+6+0+4= 32.

#### Exercise 8

Viết chương trình phân tích một số nguyên thành các thừa số nguyên tố

Ví dụ: Số 28 được phân tích thành 2 x 2 x 7

#### Exercise9

Viết chương trình liệt kê tất cả các số nguyên tố nhỏ hơn n cho trước.

#### Exercise 10

Viết chương trình liệt kê n số nguyên tố đầu tiên.

#### Exercise 11

Dãy số Fibonacci được định nghĩa như sau: F0 =1, F1 = 1; Fn = Fn-1 + Fn-2 với n>=2. Hãy viết chương trình tìm số Fibonacci thứ n.

#### Exercise 12

Một số được gọi là số thuận nghịch độc nếu ta đọc từ trái sang phải hay từ phải sang trái số đó ta vẫn nhận được một số giống nhau. Hãy liệt kê tất cả các số thuận nghịch độc có sáu chữ số (Ví dụ số: 558855).

#### Exercise 13

Nhập 2 số tự nhiên m,n rồi kiểm tra xem chúng có nguyên tố cùng nhau không. (Hai số nguyên tố cùng nhau là 2 số có USCLN là 1)

#### Exercise 14

Nhập vào 2 số tự nhiên m và n, sao cho m<n. Hãy liệt kê các số chính phương trong khoảng [m,n]. Có bao nhiêu số chính phương?

(Số tự nhiên p được gọi là chính phương nếu p = k2 với số tự nhiên k nào đó)

#### Exercise 15

Nhập một số c>0 (ví dụ c = 0.0001) rồi tính số π theo công thức:

π = 4\* (1-+-+...+(-1)n)

tổng được tính với n đủ lớn sao cho bất đẳng thức ≤ c thỏa mãn.

#### Exercise 16

Nhập một số c>0 (ví dụ c = 0.0001) và một số thực x rồi tính

ex = 1+  +  + ... +

tổng được tính với n đủ lớn sao cho bất đẳng thức || ≤ c thỏa mãn.

#### Exercise 17

Nhập một số c>0 (ví dụ c = 0.0001) và một số thực x rồi tính

sin x = -  +  -... +(-1)n

tổng được tính với n đủ lớn sao cho bất đẳng thức || ≤ c thỏa mãn.

So sánh kết quả trên đây với giá trị hàm chuẩn sin(x) có sẵn trong C.

## Module 4: hàm

### Lý thuyết

#### ****Khai báo và định nghĩa hàm****

Xây dựng một hàm bao gồm: khai báo kiểu hàm, đặt tên hàm, khai báo các đối và đưa ra câu lệnh cần thiết để thực hiện yêu cầu đề ra cho hàm. Một hàm được viết theo mẫu sau:

**<Kiểu\_trả\_về>  <tên\_hàm> ( [khai báo các tham số hình thức])**

{

            [Khai báo các biến cục bộ]

            [Các câu lệnh]

            [return[biểu thức];]

}

-<Kiểu\_trả\_về>: giá trị kiểu dữ liệu của dữ liệu sẽ trả về cho hàm

- <tên\_hàm>: tên của hàm mà bạn muốn định nghĩa, được đặt theo qui tắc đặt tên của C

- [khai báo các tham số hình thức]: các tham số hình thức và kiểu của chúng

-  [Khai báo các biến cục bộ]: khai báo các biến cục bộ, các biến này chỉ có tác dụng trong nội bộ hàm

-  [return]: là lệnh thực hiện gán giá trị trả về cho hàm

- [biểu thức]: là giá trị trả về cho hàm, có thể là biến, hằng, biểu thức nhưng phải có giá trị xác định và có kiểu dữ liệu là kiểu đã khai báo cho hàm

- Hàm có thể có giá trị trả về hoặc không, giá trị trả về phải cùng kiểu với kiểu trả về đã khai báo hàm. Nếu hàm không có giá trị trả về thì đặt từ khóa **void**trước tên hàm để báo hiệu là hàm không cần giá trị trả về cho hàm.

- Khi hàm khai báo không có kiểu ở trước nó thì nó được mặc định là kiểu int.

- Không nhất thiết phải khai báo nguyên mẫu hàm. Nhưng nói chung nên có vì nó cho phép chương trình biên dịch phát hiện lỗi khi gọi.

- Nguyên mẫu của hàm thực chất là dòng đầu tiên của hàm thêm vào dấu;. Tuy nhiên, trong nguyên mẫu có thể bỏ tên các tham số hình thức

#### ****Lời gọi hàm****

**Cú pháp:**

**tên hàm ([Danh sách các tham số thực])**

Danh sách các tham số thực phải bằng số tham số hình thức và lần lượt chúng có kiểu tương ứng với nhau.

#### ****Tham số hình thức, tham số thực và biến cục bộ****

Các tham số dùng khi khai báo hàm được gọi là tham số hình thức. Các tham số được cung cấp cho hàm khi gọi hàm là tham số thực. Tham số thực có thể là một biểu thức, trong khi tham số hình thức thì không thể là 1 biểu thức. Dãy các tham số thực phải tương ứng về kiểu với tham số hình thức.

Có những hàm không cần có tham số. Vì vậy, khi khai báo ta có thể dùng từ khóa void để báo rằng hàm không cần tham số.

Biến cục bộ là biến chỉ có phạm vi hoạt động trọng nội bộ hàm, được khia báo bên trong hàm. Do tham số thực và biến cục bộ đều có phạm vi hoạt động trong cùng một hàm nên tham số thực và biến cục bộ cần có tên khác nhau.

Tham số hình thức và biến cục bộ có thể trùng tên với các đại lượng ngoài hàm mà không gây ra nhầm lẫn nào.

Khi một hàm được gọi tới, việc đầu tiên là giá trị của các tham số thực được gán cho các tham số hình thức. Như vậy các tham số hình thức chính là các bản sao của các tham số thực.  Hàm chỉ làm việc trên các tham số hình thức.

Các tham số hình thức có thể bị biến đổi trong thân hàm, còn các tham số thực thì không bị thay đổi.

#### ****Quy tắc hoạt động của hàm****

Khi gặp một lời gọi hàm thì nó sẽ bắt đầu được thực hiện. Nói cách khác, khi máy gặp lời gọi hàm ở một vị trí nào đó trong chương trình, máy sẽ tạm dời chỗ đó và chuyển đến hàm tương ứng. Quá trình đó diễn ra theo trình tự sau:

-   Cấp phát bộ nhớ cho các biến cục bộ.

-   Gán giá trị của các tham số thực cho các tham số hình thức tương ứng.

-   Thực hiện các câu lệnh trong thân hàm.

-   Khi gặp câu lệnh return hoặc dấu } cuối cùng của thân hàm thì máy sẽ xoá các tham số hình thức, biến cục bộ và ra khỏi hàm.

Nếu trở về từ một câu lệnh return có chứa biểu thức thì giá trị của biểu thức được gán cho hàm. Giá trị của hàm sẽ được sử dụng trong các biểu thức chứa nó.

### Exercise

Làm lại tối thiểu 20 bài tập ở phần cấu trúc lặp và cấu trúc rẽ nhánh có sử dụng hàm.

## Module 5: mảng, ma trận

### Lý thuyết

#### ****Mảng 1 chiều****

Mảng 1 chiều là tập hợp các phần tử có cùng kiểu dữ liệu. Giả sử bạn muốn lưu n số nguyên để tính trung bình, bạn không thể khai báo n biến để lưu n giá trị rồi sau đó tính trung bình.

**int ia[10];**với **int**là kiểu của mảng, **ia** là tên mảng, 10 là số phần tử của mảng, tức mảng ia có tối đa 10 phần tử

Khai báo một mảng số nguyên gồm 10 phần tử, mỗi phần tử có kiểu int.

Các phần tử của mảng ia được mô tả như sau:

Từ ví dụ 2 ta có cú pháp khai báo mảng như sau:

**Tên\_kiểu tên\_biến[spt];**

- Tên\_kiểu là tên  của kiểu dữ liệu mà bạn muốn khai báo cho mảng

- Tên\_biến là tên của mảng mà bạn muốn khai báo

- Spt: là số phần tử tối đa của mảng mà bạn muốn khai báo hay còn gọi là kích thức của mảng

#### ****Tham chiếu đến từng phần tử mảng****

Sau khi mảng được khai báo, mỗi phần tử trong mảng đều có chỉ số để tham chiếu. Chỉ số bắt đầu từ 0 đến n-1 (với n là kích thước mảng). Trong ví dụ 2, ta khai báo mảng 10 phần tử thì chỉ số bắt đầu từ 0 đến 9.

Như vậy, để truy xuất đến phần tử thứ i trong mảng ia ta viết ia[i], trong đó i chỉ được phép nhận một trong các giá trị từ 0 đến 9 vì mảng ia được khia báo chỉ có 10 phần tử

#### ****Mảng 2 chiều****

Việc đọc dữ liệu của mảng chúng ta cần chỉ rõ là cần đọc dữ liệu của phần tử thứ mấy trong mảng

Ngoài kiểu int, bạn có thể khai báo mảng kiểu char, float, double…

**Khai báo mảng:**

Khai báo mảng 2 chiều **int ia[5][10];**với **int**là kiểu mảng, ia là tên mảng, số phần tử mảng là 5 x 10.

Khai báo một mảng 2 chiều số nguyên gồm 50 phần tử, mỗi phần tử có kiểu int.

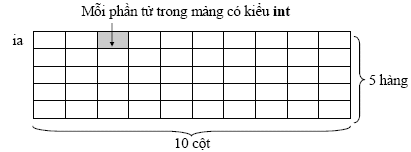
Từ ví dụ 1, ta có cú pháp khai báo mảng đa chiều như sau:

tên\_kiểu tên\_biến[spt1][spt2]…[sptn];

tên\_kiểu: kiểu dữ liệu muốn khai báo cho mảng

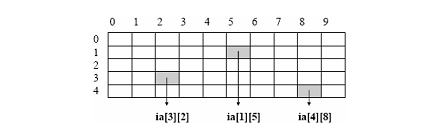
tên\_biến: Tên của biến mảng, tên được đặt theo qui tắc đặt tên của C

spt1, spt2,…, sptn: số phần tương ứng của các chiều của mảng n chiều



#### ****Tham chiếu đến từng phần tử mảng 2 chiều****

Sau khi mảng được khai báo, mỗi phần tử trong mảng 2 chiều đều có 2 chỉ số để tham chiếu, chỉ số hàng và chỉ số cột. Chỉ số hàng bắt đầu từ 0 đến số hàng – 1 và chỉ số cột bắt đầu từ 0 đến số cột – 1. Tham chiếu đến một phần tử trong mảng 2 chiều ia: ia[chỉ số hàng][chỉ số cột]



**Đọc dữ liệu từ mảng 2 chiều**

 Để đọc dữ liệu từ mảng hai chiều cần xác định là đọc tại phần tử thứ mấy trong mảng, tức cần biết chỉ số dòng và chỉ số cột của phần tử đó

### Exercise

#### Exercise 1

Nhập số n và dãy các số nguyên a[0], a[1],..., a[n-1]. Sau đó tìm số lớn nhất trong dãy này.

#### Exercise 2

Nhập số n và dãy các số nguyên a[0], a[1],..., a[n-1] rồi sắp xếp dãy trên theo thứ tự tăng dần

#### Exercise 3

Nhập số liệu cho dãy số nguyên a0 , a1 ,..., an-1 . Đếm xem có bao nhiêu cặp 2 phần tử liên tiếp bằng nhau trong dãy trên

#### Exercise 4

Nhập số liệu cho dãy số nguyên a0 , a1 ,..., an-1 và một số x bất kỳ. Đếm số lần xuất hiện của số x

trong dãy trên.

#### Exercise 5

Nhập số liệu cho dãy số nguyên a0 , a1 ,..., an-1 và 2 số nguyên b, c (b<c). Tính trung bình cộng các phần tử của dãy nằm trong khoảng [b, c].

#### Exercise 6

Viết chương trình nhập số nguyên N (0 < N ≤ 10 000) và mảng a gồm N số nguyên.

Tính và in ra trung bình cộng của các số âm, số dương trong a.

In ra tất cả các số nguyên tố của a.

#### Exercise 7

Nhập mảng (a, N) gồm các số nguyên dương.

a có phải là mảng đối xứng hay không (ví dụ: 15 2 1 2 15 là mảng đối xứng).

In phần tử có số lần xuất hiện nhiều nhất trong a.

Nhập số X. Xác định vị trí của số nguyên tố trên a có giá trị gần với X nhất.

#### Exercise 8

Nhập 2 mảng (a, N) và (b, M) và số nguyên p (0≤p<N). Hãy chèn mảng b vào vị trí p của a.

#### Exercise 9

Nhập số liệu cho ma trận A kiểu mxn có các phần tử là các số nguyên. Tìm các giá trị cực đại và

cực tiểu của các phần tử và chỉ rõ vị trí của chúng trong bảng

#### Exercise 10

Nhập số liệu cho dãy số nguyên a0 , a1 ,..., an-1 . Kiểm tra xem dãy đã được sắp xếp theo thứ tự tăng dần hay không. Nếu không hãy chỉ ra vị trí phần tử đầu tiên làm mất tính chất được sắp của dãy

#### Exercise 11

. Nhập dãy n số (n ≤ 1000). Xác định đường chạy dài nhất, xuất lên màn hình vị trí phần tử đầu tiên và độ dài của đường chạy đó. Đường chạy là một dãy liên tiếp các phần tử không giảm của dãy ban đầu.

#### Exercise 12

Viết chương trình nhập số liệu cho ma trận các số nguyên A cấp mxn trong đó m, n là các số tự

nhiên. Sau đó tìm ma trận chuyển vị B = (bij) cấp nxm, bij = aji i = 1,2,...,n;j =1,2,...,m

#### Exercise 13

Viết chương trình tính tích 2 ma trận các số nguyên A cấp mxn và B cấp nxk

#### Exercise 14

Nhập số liệu cho ma trận A kiểu mxn có các phần tử là các số nguyên. Tìm phần tử bé nhất của ma trận và đếm xem có bao nhiêu phần tử bằng phần tử bé nhất.

#### Exercise 15

Nhập số liệu cho ma trận A kiểu mxn có các phần tử là các số nguyên. Hãy liệt kê trên màn hình tất cả các phần tử của ma trận nhưng theo thứ tự tăng dần

#### Exercise 16

Nhập ma trận chữ nhật các số nguyên có cấp mxn. Tìm phần tử lớn nhất của mỗi hàng. In mỗi phần tử tìm được trên một dòng

#### Exercise 17

Viết chương trình nhập vào vào mảng A có n phần tử, các phần tử là những số nguyên lớn hơn 0 và nhỏ hơn 100 được nhập vào từ bàn phím. Thực hiện các chức năng sau:

a) Tìm phần tử lớn nhất và lớn thứ 2 trong mảng cùng chỉ số của các số đó.

b) Sắp xếp mảng theo thứ tự giảm dần .

c) Nhập một số nguyên x và chèn x vào mảng A sao cho vẫn đảm bảo tính sắp xếp giảm dần.

#### Exercise 18

Viết chương trình nhập vào vào ma trận A có n dòng, m cột, các phần tử là những số nguyên lớn hơn 0 và nhỏ hơn 100 được nhập vào từ bàn phím. Thực hiện các chức năng sau:

a) Tìm phần tử lớn nhất của ma trận cùng chỉ số của số đó.

b) Tìm và in ra các phần tử là số nguyên tố của ma trận (các phần tử không nguyên tố thì thay bằng số 0).

c) Tìm hàng trong ma trận có nhiều số nguyên tố nhất.

#### Exercise 19

Nhập số liệu cho ma trận A kiểu mxn có các phần tử là các số nguyên và một số nguyên x. Hãy

đếm xem số nguyên x xuất hiện bao nhiêu lần trong ma trận và tại các vị trí nào.

#### Exercise 20

Nhập ma trận chữ nhật các số nguyên có cấp mxn.Tìm hàng của ma trận sao cho tổng các phần tử tính theo hàng đó là lớn nhất so với các hàng còn lại

## Module 6: Xâu

### Lý thuyết :

#### Khai báo

##### Khai báo theo mảng

Cú pháp: char <Biến> [Chiều dài tối đa]

Ví dụ: Trong chương trình, ta có khai báo:

char Ten[12];

Trong khai báo này, bộ nhớ sẽ cung cấp 12-1 bytes để lưu trữ nội dung của chuỗi ký tự Ten; byte cuối cùng lưu trữ ký tự ‘\0’ để chấm dứt chuỗi.

Ghi chú:

* Chiều dài tối đa không nên khai báo thừa để tránh lãng phí bộ nhớ, nhưng cũng không nên khai báo thiếu.

##### Khai báo theo con trỏ

Cú pháp: char \*<Biến>

Ví dụ: Trong chương trình, ta có khai báo:

char\*Ten;

Trong khai báo này, bộ nhớ sẽ dành 2 byte để lưu trữ địa chỉ của biến con trỏ Ten đang chỉ đến, chưa cung cấp nơi để lưu trữ dữ liệu. Muốn có chỗ để lưu trữ dữ liệu, ta phải gọi đến hàm malloc() hoặc calloc() có trong “alloc.h”, sau đó mới gán dữ liệu cho biến.

##### Vừa khai báo vừa gán giá trị

Ví dụ:

1. #include<stdio.h>
2. #include<conio.h>
3. int main()
4. {
5. charChuoi[]="Mau nang hay la mau mat em";
6. printf("Vua khai bao vua gan tri : %s",Chuoi);
7. getch();
8. return0;
9. }

**Ghi chú**: Chuỗi được khai báo là một mảng các ký tự nên các thao tác trên mảng có thể áp dụng đối với chuỗi ký tự

#### Một số hàm xử lý chuỗi (trong string.h)

##### **Cộng chuỗi - Hàm strcat()**

Cú pháp: char \*strcat(char \*des, const char \*source)

Hàm này có tác dụng ghép chuỗi nguồn vào chuỗi đích.

Ví dụ: Nhập vào họ lót và tên của một người, sau đó in cả họ và tên của họ lên màn hình.

1. #include<conio.h>
2. #include<stdio.h>
3. #include<string.h>
4. int main()
5. {
6. char HoLot[30], Ten[12];
7. printf("Nhap Ho Lot: ");gets(HoLot);
8. printf("Nhap Ten: ");gets(Ten);
9. strcat(HoLot,Ten);*/\* Ghep Ten vao HoLot\*/*
10. printf("Ho ten la: ");puts(HoLot);
11. getch();
12. return0;
13. }

##### **Xác định độ dài chuỗi - Hàm strlen()**

Cú pháp: int strlen(const char\* s)

Ví dụ: Sử dụng hàm strlen xác định độ dài một chuỗi nhập từ bàn phím.

1. #include<conio.h>
2. #include<stdio.h>
3. #include<string.h>
4. int main(){
5. char Chuoi[255];
6. int Dodai;
7. printf("Nhap chuoi: ");gets(Chuoi);
8. Dodai =strlen(Chuoi);
9. printf("Chuoi vua nhap: ");puts(Chuoi);
10. printf("Co do dai %d",Dodai);
11. getch();
12. return0;
13. }

##### **Đổi một ký tự thường thành ký tự hoa - Hàm toupper()**

Hàm toupper() (trong ctype.h) được dùng để chuyển đổi một ký tự thường thành ký tự hoa.

Cú pháp: char toupper(char c)

##### **Đổi chuỗi chữ thường thành chuỗi chữ hoa, hàm strupr()**

Hàm strupr() được dùng để chuyển đổi chuỗi chữ thường thành chuỗi chữ hoa, kết quả trả về của hàm là một con trỏ chỉ đến địa chỉ chuỗi được chuyển đổi.

Cú pháp: char\* strupr(char \*s)

Ví dụ: Viết chương trình nhập vào một chuỗi ký tự từ bàn phím. Sau đó sử dụng hàm strupr() để chuyển đổi chúng thành chuỗi chữ hoa.

1. #include<conio.h>
2. #include<stdio.h>
3. #include<string.h>
4. int main()
5. {
6. char Chuoi[255],\*s;
7. printf("Nhap chuoi: ");gets(Chuoi);
8. s=strupr(Chuoi);
9. printf(“Chuoi chu hoa: ”);puts(s);
10. getch();
11. return0;
12. }

##### **Đổi chuỗi chữ hoa thành chuỗi chữ thường, hàm strlwr()**

Muốn chuyển đổi chuỗi chữ hoa thành chuỗi toàn chữ thường, ta sử dụng hàm strlwr(), các tham số của hàm tương tự như hàm strupr()

Cú pháp: char \*strlwr(char \*s)

##### **Sao chép chuỗi, hàm strcpy()**

Hàm này được dùng để sao chép toàn bộ nội dung của chuỗi nguồn vào chuỗi đích.

Cú pháp: char\* strcpy(char \*Des, const char \*Source)

Ví dụ: Viết chương trình cho phép chép toàn bộ chuỗi nguồn vào chuỗi đích.

1. #include<conio.h>
2. #include<stdio.h>
3. #include<string.h>
4. int main()
5. {
6. char Chuoi[255],s[255];
7. printf("Nhap chuoi: ");gets(Chuoi);
8. strcpy(s,Chuoi);
9. printf(“Chuoi dich: ”);puts(s);
10. getch();
11. return0;
12. }

##### **Sao chép một phần chuỗi, hàm strncpy()**

Hàm này cho phép chép n ký tự đầu tiên của chuỗi nguồn sang chuỗi đích.

Cú pháp: char \*strncpy(char \*Des, const char \*Source, size\_t n)

##### **Trích một phần chuỗi, hàm strchr()**

Để trích một chuỗi con của một chuỗi ký tự bắt đầu từ một ký tự được chỉ định trong chuỗi cho đến hết chuỗi, ta sử dụng hàm strchr().

Cú pháp: char \*strchr(const char \*str, int c)

Ghi chú:

* Nếu ký tự đã chỉ định không có trong chuỗi, kết quả trả về là NULL.
* Kết quả trả về của hàm là một con trỏ, con trỏ này chỉ đến ký tự c được tìm thấy đầu tiên trong chuỗi str.

##### **Tìm kiếm nội dung chuỗi, hàm strstr()**

Hàm strstr() được sử dụng để tìm kiếm sự xuất hiện đầu tiên của chuỗi s2 trong chuỗi s1.

Cú pháp: char\* strstr(const char \*s1, const char \*s2)

Kết quả trả về của hàm là một con trỏ chỉ đến phần tử đầu tiên của chuỗi s1 có chứa chuỗi s2 hoặc giá trị NULL nếu chuỗi s2 không có trong chuỗi s1.

Ví dụ: Viết chương trình sử dụng hàm strstr() để lấy ra một phần của chuỗi gốc bắt đầu từ chuỗi "hoc".

1. #include<conio.h>
2. #include<stdio.h>
3. #include<string.h>
4. int main()
5. {
6. char Chuoi[255],\*s;
7. printf("Nhap chuoi: ");gets(Chuoi);
8. s=strstr(Chuoi,"hoc");
9. printf("Chuoi trich ra: ");puts(s);
10. getch();
11. return0;
12. }

##### **So sánh chuỗi, hàm strcmp()**

Để so sánh hai chuỗi theo từng ký tự trong bảng mã Ascii, ta có thể sử dụng hàm strcmp().

Cú pháp: int strcmp(const char \*s1, const char \*s2)

Hai chuỗi s1 và s2 được so sánh với nhau, kết quả trả về là một số nguyên (số này có được bằng cách lấy ký tự của s1 trừ ký tự của s2 tại vị trí đầu tiên xảy ra sự khác nhau).

* Nếu kết quả là số âm, chuỗi s1 nhỏ hơn chuỗi s2.
* Nếu kết quả là 0, hai chuỗi bằng nhau.
* Nếu kết quả là số dương, chuỗi s1 lớn hơn chuỗi s2.

##### **So sánh chuỗi, hàm stricmp()**

Hàm này thực hiện việc so sánh trong n ký tự đầu tiên của 2 chuỗi s1 và s2, giữa chữ thường và chữ hoa không phân biệt.

Cú pháp: int stricmp(const char\*s1, const char \*s2)

Kết quả trả về tương tự như kết quả trả về của hàm strcmp()

##### **Khởi tạo chuỗi, hàm memset()**

Hàm này được sử dụng để đặt n ký tự đầu tiên của chuỗi là ký tự c.

Cú pháp: memset(char \*Des, int c, size\_t n)

##### **Đổi từ chuỗi ra số, hàm atoi(), atof(), atol() (trong stdlib.h)**

Để chuyển đổi chuỗi ra số, ta sử dụng các hàm trên.

Cú pháp:

* int atoi(const char \*s): chuyển chuỗi thành số nguyên
* long atol(const char \*s): chuyển chuỗi thành số nguyên dài
* float atof(const char \*s): chuyển chuỗi thành số thực

Nếu chuyển đổi không thành công, kết quả trả về của các hàm là 0.

Ngoài ra, thư viện string.h còn hỗ trợ các hàm xử lý chuỗi khác, ta có thể đọc thêm trong phần trợ giúp..

### Exercise

BÀI TẬP VỀ NHÀ – PHẦN XÂU KÝ TỰ

Mỗi sinh viên hoàn thành ít nhất 15 bài.

Hạn nộp bài: 27/4.

Bài 1. Viết chương trình chuyển đổi một số tự nhiên ở hệ cơ số 10 thành số ở hệ cơ số b

bất kì (1< b≤ 36).

Bài 2. Nhập một xâu ký tự. Đếm số từ của xâu ký tự đó. Thí dụ " Trường học " có 2

từ.

Bài 3: Sử dụng xâu ký tự để viết hàm kiểm tra số thuận nghịch. Áp dụng liệt kê các số

thuận nghịch có 6 chữ số.

Bài 4: Xây dựng tập thao tác với các số nguyên lớn (nhiều hơn 20 chữ số)

a) Tổng, hiệu hai số nguyên lớn

b) Tích hai số nguyên lớn

c) Thương 2 số nguyên lớn (\*)

Bài 5. Cho một chuỗi ký tự có độ dài n, hãy đếm số lần xuất hiện của các ký tự

‘A’,’B’,’C’ theo cách:

a. Có phân biệt chữ hoa chữ thường.

b. Không phân biệt chữ hoa chữ thường.

Bài 6. Viết chương trình thực hiện chuẩn hoá một xâu ký tự nhập từ bàn phím (loại bỏ

các dấu cách thừa, chuyển ký tự đầu mỗi từ thành chữ hoa, các ký tự khác thành chữ

thường)

Bài 7. Viết chương trình thực hiện nhập một xâu ký tự và tìm từ dài nhất trong xâu đó.

Từ đó xuất hiện ở vị trí nào? (Chú ý. nếu có nhiều từ có độ dài giống nhau thì chọn

từ đầu tiên tìm thấy).

Bài 8.Viết chương trình thực hiện nhập một xâu họ tên theo cấu trúc: họ...đệm...tên;

chuyển xâu đó sang biểu diễn theo cấu trúc tên…họ…đệm.

Bài 9. Nhập một chuỗi. Hãy xác định vị trí của từ có độ dài lớn nhất trong chuỗi.

Bài 10: Nhập một chuỗi, in ra cho biết có bao nhiêu ký tự là nguyên âm, phụ âm, ký tự số

và các ký tự khác.

Bài 11. Lập trình tính giá trị của một số viết dưới dạng LA MÃ.

Ví dụ: MDCLXVI = 1666. M:1000 ; D:500 ; C:100; L:50; X :10 ; V:5 ; I :1

Bài 12: Viết chương trình in ra các số điện thoại đẹp (với các đầu số hiện nay tại Việt

Nam). Giả sử các số điện thoại được coi là đẹp khi 6 số cuối thỏa mãn một trong các tính

chất sau:

 Tổng các chữ số chia cho 10 dư 8 hoặc 9

 Có dạng thuận nghịch

 Kết thúc bằng số 6 hoặc số 8

Bài 13: Viết chương trình nhập xâu s và từ s1. Sau đó nhập vị trí cần chèn và chèn s1 vào

s đúng vị trí đó.

Bài 14: Viết chương đảo ngược vị trí các từ trong một xâu ký tự.

Ví dụ: nhập “bo an co” in ra “co an bo”

Bài 15. Nhập một câu không quá 20 từ, mỗi từ không quá 10 ký tự. Viết chương trình

tách các từ trong câu và in các từ theo đúng thứ tự Alphabet.

Bài 16. Nhập 3 chuỗi ký tự s, s1, s2. Tìm tất cả những lần xuất hiện của s1 trong s và

thay thế bằng s2. Xuất s ra màn hình

Ví dụ: s = “TIM KIEM VA THAY THE”, s1=”TH”, s2=”123”. Kết quả chuỗi s sẽ là:

“TIM KIEM VA 123AY 123E”

Bài 17. Nhập hai xâu s1 và s2. Tìm xâu s2 trong s1. Nếu có hãy loại bỏ s2 trong s1. Chú

ý: phải loại bỏ cho đến khi không tìm được s2 trong s1 nữa.

Bài 18. Nhập liên tiếp các từ (các đoạn ký tự không chứa dấu cách) cho đến khi gặp từ

“ketthuc”. Hãy đếm xem có bao nhiêu từ khác nhau, mỗi từ xuất hiện bao nhiêu lần.

Bài 19. Nhập một danh sách n xâu ký tự và một xâu S. Tìm xem có thể ghép các xâu nào

trong danh sách để trở thành S được không.

Bài 20. Viết chương trình nhập 2 xâu ký tự. Tách ra các tập hợp từ của từng xâu. In ra

màn hình hợp, giao, hiệu của hai tập hợp đó theo thứ tự Alphabet.

## Module 7: con trỏ(tự học)

- Bạn muốn con trỏ trỏ tới một biến (thì con trỏ sẽ lưu trữ địa chỉ của biến đó, một số hexa mà bộ nhớ cấp phát )

- Bạn muốn con trỏ trỏ tới một mảng thì nó lưu trữ địa chỉ của phần tử đầu tiên của mảng đó.

- Bạn muốn con trỏ trỏ tới một hàm thì con trỏ sẽ lưu trữ địa chỉ của hàm đó (lát mình chỉ cách dùng)

- Bạn muốn con trỏ trỏ tới một struct(hình như khi học tới con trỏ thì chưa học tới cái này thì phải – lát mình chỉ)

- Con trỏ có thể trỏ tới con trỏ ….

và kiểu của con trỏ bạn có thể dùng là (int, char, array, structure, function, hay con trỏ).

## Module 8: structure

### Lý thuyết

Trong lập trình, đôi khi các kiểu dữ liệu cơ sở chưa đủ mềm dẻo để cho cái bài toán thực tế. Các sự vật, hiện tượng trong thực tế rất phức tạp và có nhiều tình nhất khác nhau. Người lập trình có thể định nghĩa các kiểu dữ liệu mới có cấu trúc phù hợp với từng bài toán cụ thể

Trong khuôn khổ của bài này, ta sẽ tìm hiểu cách khai báo và sử dụng Struct (1 loại kiểu dữ liệu có cấu trúc) trong ngôn ngữ lập trình C.

Khai báo kiểu dữ liệu

|  |  |
| --- | --- |
|  | struct TênCấuTrúc  {      KiểuDữLiệu1  ThànhPhần1;      KiểuDữLiệu2  ThànhPhần2;      ...      KiểuDữLiệun  ThànhPhầnn;  }; |

Khai báo biến có KDL là Struct

|  |  |
| --- | --- |
|  | struct TênCấuTrúc TênBiến; |

Để tiện cho việc khai báo các biến có KDL là 1 struct, người ta thường khai báo thêm :

|  |  |
| --- | --- |
|  | typedef struct TênCấuTrúc TênCấuTrúcMới; |

Khi đó để khai báo 1 biến có KDL cấu trúc, ta chỉ cần khai báo : TênCấuTrúcMới TênBiến;

Truy xuất 1 thành phần của biến cấu trúc

Biến thường : TênBiến.TênThànhPhần

Biến con trỏ : TênConTrỏ->TênThànhPhần

### Exercise

#### Exercise 1

Xây dựng các thao tác sau cho hai phân số:

1. Tạo lập 2 phân số.

2. Rút gọn phân số

3. Quy đồng hai phân số

4. Tổng, hiệu, tích, thương 2 phân số

#### Exercise 2

Xây dựng các thao tác sau cho hai số phức:

1. Tạo lập 2 số phức.

2. Tổng, hiệu, thương 2 số phức

#### Exercise 3

Xây dựng chương trình quản lý sinh viên, sử dụng cấu trúc sau:

struct SV {char ten[25]; float toan, ly, tb;};

Nhập danh sách cho n sinh viên (n là số tự nhiên được nhập vào), chỉ nhập ten và toan, ly.

Tính giá trị trường tb =(toan+ly)/2, sau đó sắp xếp lại danh sách theo tên và điểm trung

bình tăng dần (trong những người cùng tên thì người có điểm trung bình thấp hơn sẽ đứng

trước). Cho hiện kết quả trước và sau khi sắp xếp

#### Exercise 4

Xây dựng chương trình quản lý sinh viên, sử dụng cấu trúc sau:

struct SV {char ten[25]; float toan, ly, tb; char XepLoai[20]};

Nhập danh sách cho n sinh viên (n là số tự nhiên được nhập vào), chỉ nhập ten và toan, ly.

Tính giá trị trường tb =(toan+ly)/2, sau đó tính giá trị trường xếp loại theo cách sau:

XepLoai=”kem” nếu tb<5, = “Trung binh” nếu 5tb<7, = “Kha” nếu 7tb<8 và = “Gioi”

nếu 8tb. Sắp xếp lại danh sách theo trường XepLoai.

#### Exercise 5

Xây dựng chương trình quản lý học sinh. Mỗi học sinh quản lý các thông tin sau:

Họ tên, Năm sinh, Điểm trung bình.

Với các chức những sau:

- Nhập số liệu, mỗi lần có thể nhập m học sinh, m>0

- Xem danh sách: trên màn hình.

- Tìm kiếm :theo tên, theo năm sinh, theo tên và năm sinh.

- Sắp xếp: theo tên, theo điểm trung bình.

- Xóa khi biết tên

#### Exercise 6

Viết chương trình quản lý danh sách của một lớp học có không quá 500 sinh viên.

Thông tin của mỗi sinh viên gồm:

Mã sinh viên: đúng 8 ký tự, không có khoảng trắng.

Họ và tên: tối đa 30 ký tự (tên: 10, họ và chữ lót: 20)

Ngày tháng năm sinh

Giới tính: Nam/Nữ

Điểm các môn học: dãy gồm đúng 10 điểm số của 10 môn học.

## Module 9: tệp tin

Đọc file: Là cách thức bạn dùng ngôn ngữ lập trình duyệt và lấy được nội dung của file đó ra, còn phần còn lại xử lý nội dung đó như thế nào là tùy bạn. hi

Ghi file: Là công việc bạn ghi một nội dung nào đó lưu vào file.

Nhiều khi học bài này xong tui cảm giác nó không có tác dụng gì cả, hii đó là suy nghĩ của các bạn mới học lập trình. Công việc mình lập trình cốt lõi là tương tác lấy dữ liệu, thu thập thông tin … lưu trữ trong cơ sở dữ liệu. Có đôi khi không cần thiết phải làm công việc đó, chỉ lưu tạm để xử lý cho qua công đoạn nào đó thì chỉ cần lưu vào file để sử dụng rồi hủy sẻ nhẹ nhàn hơn rất nhìều.

Ghi file trong C

Sau khi nhập thì mảng a[] đã có giá trị lúc này chúng ta sẻ ghi vào trong file, giả sử là file bcdonline.txt.

|  |  |
| --- | --- |
|  | <!--  void ghifile(int a[],int n)  {      FILE \*f;      f=fopen("bcdonline.text","wt");      fprintf(f,"%d",n);      for(int i=1;i<=n;i++)          fprintf(f,"%3d",a[i]);      fclose(f);  }  --> |

Trước tiên để làm việc với file bạn cần khai báo biến con trỏ FILE bạn nhớ là phải viết hoa hoàn toàn từ này.

Sau đó bạn mở bằng lệnh bằng fopen(), trong đó tham số “wt” cho phép bạn ghi file.

fprintf(): sẽ giúp bạn ghi vào file như bạn khai báo, bạn hình dung thay vì hiện trên màn hình bằng printf() thì ở đây ta hiện lên file bằng cách fprintf().

Giả sử mình nhập n là 3, mảng là 9, 8, 7. Bạn vào thư mục gốc của file lưu sẻ thấy kết quả này khi mở file bcdonline.txt bằng notepad:

Đọc file trong C

Sau khi đã có file rồi thì chúng ta tiến hành đọc file lên và xử lý. Công việc bạn code như sau:

|  |  |
| --- | --- |
|  | <!--  void docfile(int a[],int &n)  {      FILE \*f;      f=fopen("bcdonline.text","rt");      fscanf(f,"%d",&n);      for(int i=1;i<=n;i++)          fscanf(f,"%d",&a[i]);      fclose(f);  }  --> |

Cũng như nói ở trên có các phần xử lý đặt trưng khi tương tác với FILE.

Tham số “rt” giúp bạn đọc dữ liệu từ file

Đọc file với lệnh fscanf();

Sau đó đóng file fclose();

Để kiểm tra bạn đã đọc được file hay chưa chúng ta tiến hành xử lý dữ liệu đã đọc được bằng cách xuất nó ra màn hình.

|  |  |
| --- | --- |
|  | <!--  void xuat(int a[],int n)  {      printf("\nKet Qua Doc File:\n\n");      for(int i=1;i<=n;i++)          printf("%3d",a[i]);    }  --> |

### Exercise

Làm lại ít nhất 20 bài ở module 3, 5

Yêu cầu:

File vào có dạng : baiX.INP

File ra có dạng :baiX.OUT

Viết rõ tên đề bài ở trong mỗi bài code

# ADVANCED

## Module 1: đệ quy

### Lý thuyết

Giải thuật đệ quy:

Giải thuật đệ quy là giải thuật có chứa thao tác gọi đến chính nó. Giải thuật đệ quy cho phép mô tả một dãy lớn các thao tác bằng một số ít các thao tác trong đó có chứa thao tác gọi lại giải thuật (gọi đệ quy).

Giải thuật giải bài toán bằng đệ quy thường rất đẹp, gọn gàng, dễ hiểu, dễ sửa đổi. Tuy nhiên, việc xử lý giải thuật đệ quy lại thường gây khó khăn cho máy tính (tốn không gian nhớ và thời gian xử lý), hơn nữa không phải mọi ngôn ngữ lập trình đều cho phép mã hóa giải thuật đệ quy (ví dụ: FORTRAN) .

Chương trình con đệ quy:

Chương trình con đệ quy là một chương trình con mà trong thân của nó có ít nhất một câu lệnh là lời gọi đến chính nó.

Chương trình con đệ quy phải có hai thành phần:

-        Thành phần không chứa đệ qui, đó là điều kiện để kết thúc quá trình đệ qui.

-        Thành phần có chứa đệ quy, sau mỗi bước, phạm vi của thành phần này phải thay đổi cho đến khi gặp điều kiện kết thúc.

@Lưu ý: Muốn giải một bài toán bằng giải thuật đệ qui việc đầu tiên ta phải đưa bài toán về một dạng tổng quát. Từ đây ta phải đi xác định cho được điều kiện suy biến của bài toán (tức điều kiện để kết thúc giải thuật đệ qui) và điều kiện gọi đệ qui.

Ví dụ bài toán tính n!

Ta có

n=0, 0!=1,

n=1, 1!=1x1 <=>0!x1

n=2, 2!=1x1x2<=>1!x2

n=3, 3!=1x1x2x3 <=>2!x3

=>n!=1x1x2x3x...x n<=>(n-1)! x n

Như vậy:

- Điều kiện suy biến  khi n=0, 0!=1

- Điều kiện gọi đệ qui n>0, n!=n x (n-1)!

Vậy, khi có được 0! =>1! =>2!=>3! ...=>n!

Giải thuật tính n!

#include <stdio.h>

long int gthua(int n);

void main(void)

{

int n;

scanf(“%d”,&n);

printf(“Giai thừa của%d là: %d”,n,gthua(n));

}

int long gthua(int n)

{

if(n==0)

          return 1;

      elsse

         return(n\*gthua(n-1));

}

-      Khi thực hiện lời gọi *gthua(3)* sẽ phát sinh lời gọi *gthua(2)*, đồng thời phải lưu giữ thông tin về trạng thái xử lý chưa hoàn thành *(return(3 \* gthua(2)))* vào Stack.

-      Gặp lời gọi *gthua(2),*tiếp tục làm phát sinh lời gọi *gthua(1)*, đồng thời vẩn phải lưu trử thông tin về trạng thái xử lý còn dang dở *(return( 2\*gthua(1)))*vào Stack.

-      Cứ như vậy cho tới khi gặp lời gọi của trường hợp suy biến *(return*(*1)))*.

-      Khi gặp trường hợp suy biến, những thông tin được lưu tạm trong Stack sẽ được lấy ra xử lý (thông tin lấy ra theo kiểu lưu trữ của Stack, thông tin vào sau sẽ được lấy ra trước). Và như vậy, dùng kết quả của gthua(0) để tính gthua(1), dùng kết quả của gthua(1) để tính gthua(2), dùng kết quả của gthua(2) để tính gthua(3). Cuối cùng được kết quả của phép tính giai thừa.

Cụ thể thực hiện lấy và tính toán trong Stack như sau:

-      Lấy return(1\*gthua(0)) để thực hiện gthua(1)=1\*gthua(0)=1\*1=1

-      Lấy return(2\*gthua(1)) để thực hiện gthua(2)=2\*gthua(1)=2\*1=3

-      Lấy return(3\*gthua(2)) để thực hiện gthua(3)=3\*gthua(2)=3\*2=6

### Exercise

#### Exercise 1

Sử dụng đệ qui để viết hàm tìm ước số chung lớn nhất của 2 số

#### Exercise 2

Sử dụng đệ qui để viết hàm tính tổng S = 1+2+….+n.

## Module 2: liệt kê bằng pp sinh và pp đệ quy

### Exercise 1 : binary (pp sinh)

|  |
| --- |
| #include<stdio.h>  #include<math.h>  main()  {  int a[100], i, n;  scanf("%d", &n);  for (i=0; i<n; i++)  a[i]=0;  for (int k=1; k<=pow(2, n); k++)  {  for (i=0; i<n; i++)  printf("%d", a[i]);  printf("\n");  for (i=n-1; i>=0; i--)  if (a[i]==0)  {  a[i]=1;  for (int j=i+1; j<n; j++)  a[j]=0;  break;  }  }  } |

### Exercise 2 :permutation(pp sinh)

|  |
| --- |
| #include "conio.h"  #include "stdio.h"  int a[9],n,s;  void inp()  {  s=0;  scanf("%d", &n);  for(int i=0;i<n;i++)  scanf("%d", &a[i]);  }  void swap(int &a,int &b)  {  int c;c=a;a=b;b=c;  }  void out()  {  for(int i=0;i<n;i++)  printf ("%d", a[i]);  printf("\n");  }  void permutation ()  {  int i=n-2;  while(i>=0&&a[i]>a[i+1]) i--;  if(i<0) s=1;  else  {  int k=n-1;  while(a[k]<a[i]) k--;  swap(a[k],a[i]);  int e=n-1,b=i+1;  while(e>b)  {  swap(a[e],a[b]);  b++;e--;  }  }  }  void enumerate()  {  inp();  while(!s)  {  permutation ();  out();  }  }  main()  {  enumerate();  } |

### Exercise 3 : combination(pp đệ quy)

|  |
| --- |
| #include<stdio.h>  int n,k,a[100],dem =0;  void inp()  {  scanf("%d", &n);  scanf("%d", &k);  for(int i=1;i<=n;i++)  a[i]=i;  }  void print()  {  for( int i=1;i<=k;i++)  printf("%d ", a[i]);  printf("\n");  }  void combination()  {  print();  int i=k;  while(a[i]>=(n-k+i))  i--;  if(i>0)  {  a[i]+=1;  for(int j =i+1;j<=k;j++)  a[j]=a[i]+j-i;  combination();  }  else return;  }  main()  {  inp();  combination();  } |

### Exercise 4 : binary (pp đệ quy)

|  |
| --- |
| #include "conio.h"  #include "stdio.h"  int x[30],k,i;  int n;  void print() // ham in  {  for(int j=0; j<n; j++)  printf("%d",x[j]);  printf("\n");  }  void bin(int i) // ham sinh nhi phan  {  int k;  for ( k=0; k<=1; k++)  {  x[i] = k;  if (i==n-1)  print();  else  bin(i+1);  }  }  main()  {  scanf("%d",&n);  bin(0);  getch();  } |

### Exercise 5 : permultation(đệ quy)

|  |
| --- |
| #include "conio.h"  #include "stdio.h"  int a[9],n,;  void inp()  {  scanf("%d", &n);  for(int i=0;i<n;i++)  scanf("%d", &a[i]);  }  void swap(int &a,int &b)  {  int c;c=a;a=b;b=c;  }  void out()  {  for(int i=0;i<n;i++)  printf ("%d", a[i]);  printf("\n");  }  void permutation(int n)  {  if(n==1)  out();  else  {  for ( int i = n-1; i>=0;i--)  {  swap(a[n-1],a[i]);  permutation(n-1);  swap(a[i],a[n-1]);  }  }  }  void enumerate()  {  inp();  permutation(n);  }  main()  {  enumerate();  } |

### Exercise 6 : bài toán trộn sơn

Anh A vừa mới xây nhà, anh mua được 4 loại sơn khác màu(xanh, đỏ, vàng, trắng), anh muốn mỗi phòng của mình có nhiều màu sơn nên có ý tưởng trộn các màu sơn lại với nhau.

Viết chương trình đưa ra tất cả các màu sơn có thể trộn được, biết trộn được tối đa 2 màu sơn.

|  |
| --- |
| #include<stdio.h>  int n,k,a[100],dem =0;  void inp()  {  n=4;  k=2;  for(int i=1;i<=n;i++)  a[i]=i;  }  void print()  {  for( int i=1;i<=k;i++)  {  if (a[i]==1) printf("xanh ");  if (a[i]==2) printf("do ");  if (a[i]==3) printf("vang ");  if (a[i]==4) printf("trang ");  }  printf("\n");  }  void combination()  {  print();  int i=k;  while(a[i]>=(n-k+i))  i--;  if(i>0)  {  a[i]+=1;  for(int j =i+1;j<=k;j++)  a[j]=a[i]+j-i;  combination();  }  }  main()  {  inp();  combination();  } |

## Module 3: 4 thuật toán sắp xếp cơ bản

### Exercise 1 : bubble sort

|  |
| --- |
| #include<stdio.h>  int a[100], n;  void  bubble sort(int n)  {  for (int i=0; i<n; i++)  for (int j=i+1; j<n; j++)  {  if (a[i]>a[j])  {  int temp=a[i];  a[i]=a[j];  a[j]=temp;  }  }  }  main()  {  scanf("%d", &n);  for (int i=0; i<n; i++)  scanf("%d", &a[i]);   bubble sort(n);  for (int i=0; i<n; i++)  printf("\n%d", a[i]);  } |

### Exercise 2 : select sort

|  |
| --- |
| #include<stdio.h>  int a[100], n;  void selectsort()  {  int i,imin,j,temp;  for (i=0; i<=n-2; i++)  {  imin = i;  for (j=i+1; j<=n-1; j++)  if (a[j] < a[imin])  {  imin = j;  }  temp = a[i];  a[i] = a[imin];  a[imin] = temp;  }  }  main()  {  scanf("%d", &n);  for (int i=0; i<n; i++)  scanf("%d", &a[i]);  selectsort();  for (int i=0; i<n; i++)  printf("\n%d", a[i]);  } |

### Exercise 3 : insertion sort

|  |
| --- |
| #include<stdio.h>  int a[100], n;  void insertion\_sort(int a[], int n)  {  int i,tg,x;  for (i=1;i<n;i++)  {  tg=a[i];  x=i-1;  while((x>=0)&&(a[x]>tg))  {  a[x+1] = a[x];  x--;  }  a[x+1] =tg;  }  }  main()  {  scanf("%d", &n);  for (int i=0; i<n; i++)  scanf("%d", &a[i]);  insertion\_sort(a, n);  for (int i=0; i<n; i++)  printf("\n%d", a[i]);  } |

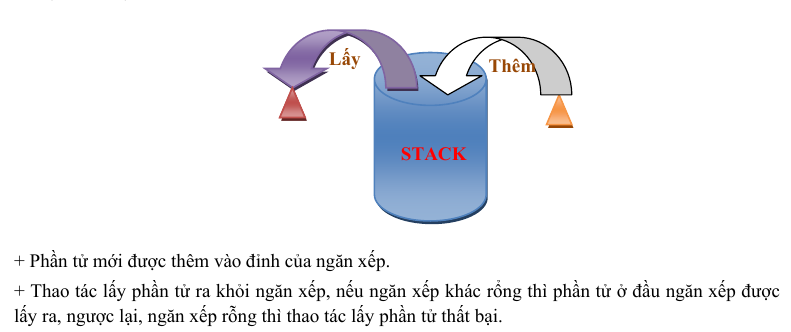
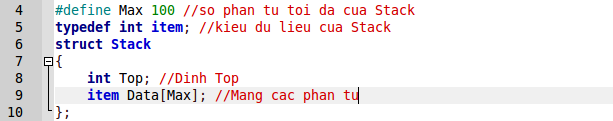
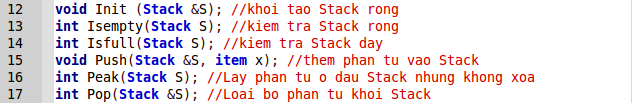
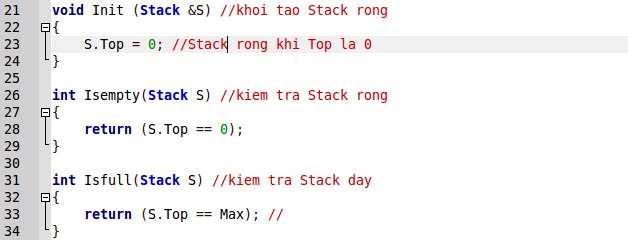
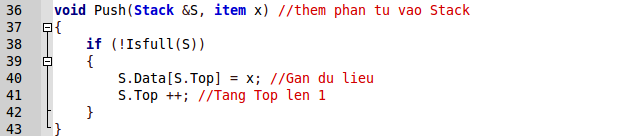
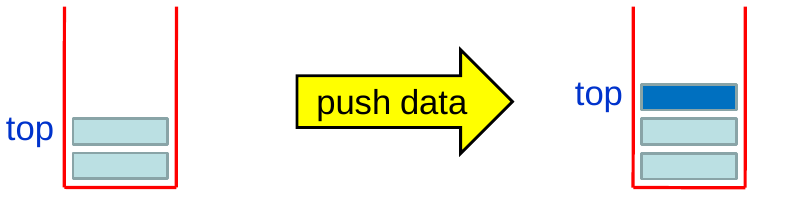
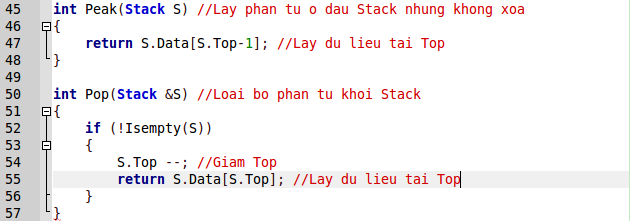
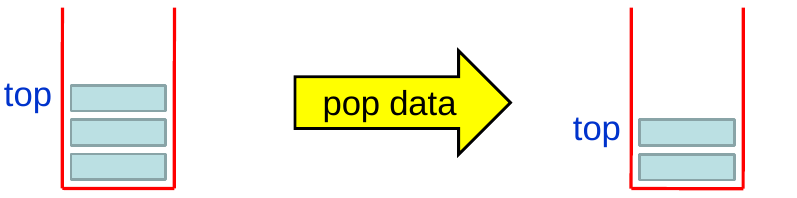
### Exercise 4 : merge sort

|  |
| --- |
| #include<stdio.h>  int a[100], n;  void merge\_sort(int a[], int n)  {  int i,j,k,low1,up1,low2,up2,size;  int dstam[100];size=1;  while(size<n)  {  low1=0;k=0;  while(low1 +size <n)  {  low2=low1+size;  up1=low2-1;  if (low2+size-1< n)  up2=low2+size-1;  else  up2=n-1;  for(i=low1, j=low2; i<=up1 && j<=up2; k++)  {  if(a[i]<=a[j])  dstam[k]=a[i++];  else  dstam[k] =a[j++];  }  for(;i<=up1;k++)  dstam[k]=a[i++];  for(;j<=up2;k++)  dstam[k]=a[j++];  low1=up2+1;  }  for (i=low1; k<n;i++)  dstam[k++]=a[i];  for(i=0;i<n;i++)  a[i]=dstam[i];  size\*=2;  }  }  main()  {  scanf("%d", &n);  for (int i=0; i<n; i++)  scanf("%d", &a[i]);  merge\_sort(a, n);  for (int i=0; i<n; i++)  printf("\n%d", a[i]);  } |

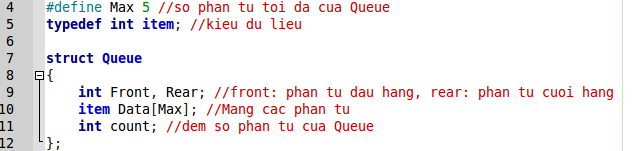
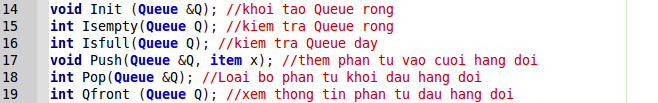
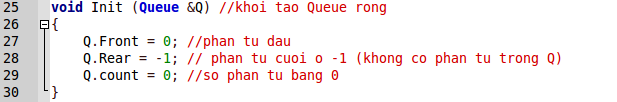
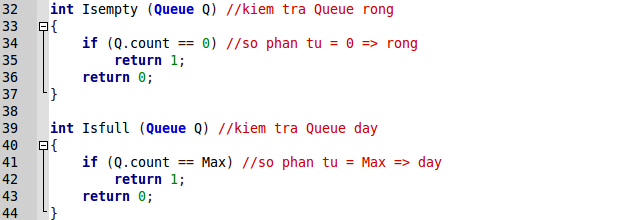
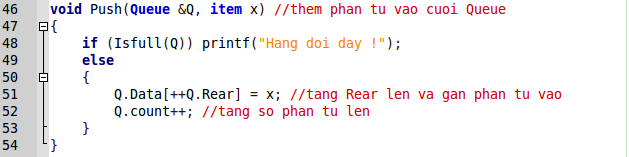
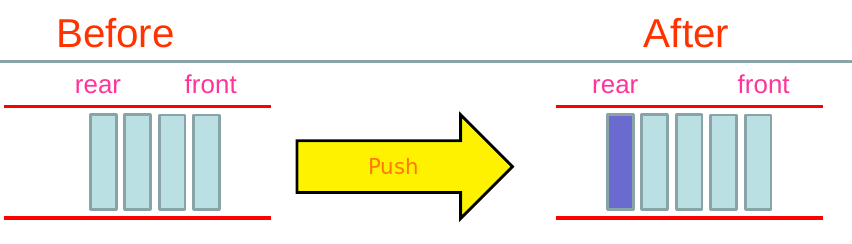
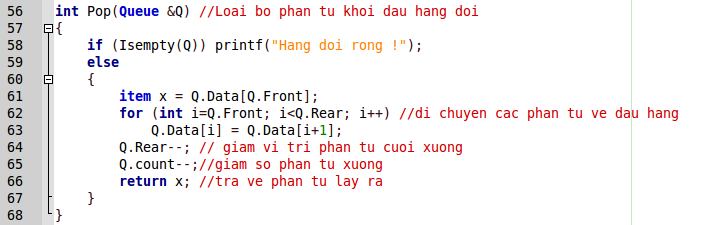
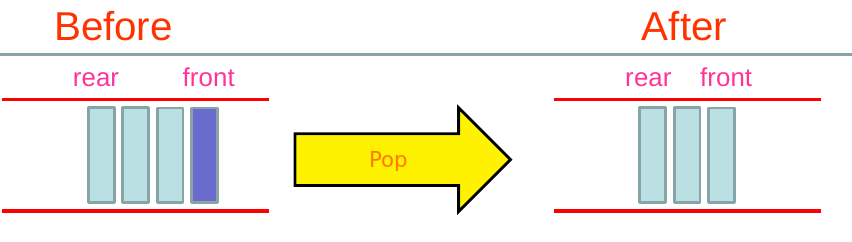
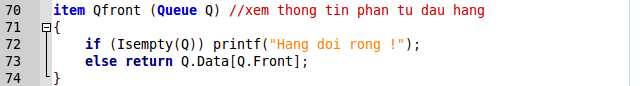
## Module 4 stack and queue

### Lý thuyết

#### 1.Stack: Ngăn xếp;

Stack là một danh sách có thứ tự mà phép chèn và xóa được thực hiện tại đầu cuối của danh sách và người ta gọi đầu cuối này là đỉnh (top) của stack.  
  
[](http://i.imgur.com/2atNFkw.png)  
1. Stack cài đặt trên mảng  
[](http://i.imgur.com/lcryIDO.png)  
  
  
Các thao tác chính trên Stack:  
[](http://i.imgur.com/cuEdROH.png)  
  
  
1.1 Khởi tạo danh sách rỗng, kiểm tra danh sách rỗng, đầy (cái này chắc không cần nói nhiều)  
[](http://i.imgur.com/eJZ4Opo.png)  
  
  
1.2 Push (Thêm phần tử vào Stack)  
Để chèn thêm phần tử vào Stack ta chèn vào vị trí Top, và tang Top lên 1 đơn vị  
[](http://i.imgur.com/Zjzg81F.png)  
  
  
  
1.3 Peak (lấy dữ liệu tại Top nhưng không xóa), Pop (Xóa và lấy dữ liệu tại Top)  
[](http://i.imgur.com/Ssk213r.png)  
[](http://i.imgur.com/y4oRFZC.png)

#### 2.Queue: Hàng Chờ;

àng đợi (tiếng Anh: queue) là một cấu trúc dữ liệu dùng để chứa các đối tượng làm việc theo cơ chế FIFO (viết tắt từ tiếng Anh: First In First Out), nghĩa là "vào trước ra trước" Trong hàng đợi, các đối tượng có thể được thêm vào hàng đợi bất kỳ lúc nào, nhưng chỉ có đối tượng thêm vào đầu tiên mới được phép lấy ra khỏi hàng đợi. Việc thêm một đối tượng luôn diễn ra ở cuối hàng đợi và một phần tử luôn được lấy ra từ đầu hàng đợi.  
[](http://i.imgur.com/MxuqLD6.png)  
  
  
**1. Queue cài đặt trên mảng**  
[](http://i.imgur.com/g0iYtPw.png)  
  
  
Các phép toán trên Queue  
[](http://i.imgur.com/X36hOt8.png)  
  
  
**1.1 Khởi tạo Queue rỗng.**  
Để khởi tạo Queue rỗng ta cần đưa vị trí Front về 0, Rear về -1, cout về 0.  
[](http://i.imgur.com/tHcYeXZ.png)  
  
  
**1.2 Kiểm tra Queue rỗng, đầy**  
Kiểm tra rỗng đầy chỉ cần kiểm tra count so với 0 và max  
[](http://i.imgur.com/hBNFmbk.png)  
  
  
**1.3 Thêm phần tử vào cuối Queue (Push)**  
Tăng vị trí của Rear lên 1 và đưa data vào vị trí đó  
[](http://i.imgur.com/oHOgcvF.png)  
[](http://i.imgur.com/wYi1e8i.png)  
  
  
**1.4 Xóa phần tử đầu Queue (Pop)**  
trước tiên phải kiểm tra Queue rỗng không, nếu không rỗng ta thực hiện di chuyển các phần tử trong hàng về đầu hàng bằng vòng for (giống như xếp hàng khi mua hàng) sau đó giảm Rear và count xuống  
[](http://i.imgur.com/Jp9pUst.png)  
[](http://i.imgur.com/xXgtsSu.png)  
  
  
**1.5 Xem thông tin phần tử đầu Queue**  
[](http://i.imgur.com/WAkog4v.png)

### Exercise

#### 1.Exercise 1 :Hunger Game

|  |
| --- |
| #include"conio.h"  #include"stdio.h"  int dem=0;  int hunger(int n, int a[])  {  int k=0;  int t=0;  int b[10000];  int c[10000];  int i=n-1;  while(i>0){  while(a[i]<a[i-1]){  i--;  }  if(i!=n-1){  b[t]=a[i];  t++;}  if(i==n-1) return dem;  i--;  }  for(int j=t;j>=0;j--)  {  c[k]=b[j];  k++;  }  dem++;  if(t>1) hunger(t,c);  if(t==1 || t==0) return dem;    }  main()  {  int n,a[10000];  int k=0;  printf("Nhap n"); scanf("%d",&n);  for(int i=0;i<n;i++)  {  printf("Nhap so thu %d: ",i); scanf("%d",&a[i]);  }  printf("%d",hunger(n,a));  } |

#### 2.Exercise 2 :Đổi cơ số (áp dụng stack)

|  |
| --- |
| #include"stdio.h"  #include"conio.h"  #define MAX 100  struct stack  {  int top;  int data[MAX];  };  stack S;  void khoitao(stack &S) // khoi tao ngan xep  {  S.top=0;  }  int empty(stack S) //ktra ngan xep rong  {  if(S.top==0) return 1;  return 0;  }  int full(stack S) //ktra ngan xep full  {  if(S.top==MAX) return 1;  return 0;  }  void push(stack &S, int n) //dua gia tri vao  {  if(!full(S))  {  S.data[S.top]=n;  S.top++;  }  }  int peak(stack S) // lay thong tin gia tri cuoi  {  return S.data[S.top-1];  }  int pop(stack &S) // lay gia tri cuoi  {  if(!empty(S))  {  S.top--;  return S.data[S.top];  }  }  int coso(int n, stack &S) // ham doi sang co so 2  {  int loop;  while(n>0)  {  push(S,n%2);  n=n/2;  }  }  main()  {  int n;  khoitao(S);  printf("Nhap n: "); scanf("%d",&n);  coso(n,S);  while(!empty(S))  {  printf("%d",pop(S));  }    } |

#### 3.Exercise 3 :Đảo ngược 1 dãy số (áp dụng stack)

|  |
| --- |
| #include"stdio.h"  #include"conio.h"  #define MAX 100  struct stack  {  int top;  int data[MAX];  };  stack S;  void khoitao(stack &S) //khoi tao ngan xep  {  S.top=0;  }  int empty(stack S) //ktra ngan xep co rong k  {  if(S.top==0) return 1;  return 0;  }  int full(stack S) //ktra ngan xep da day chua  {  if(S.top==MAX) return 1;  return 0;  }  void push(stack &S, int n) //dua gia tri vao  {  if(!full(S))  {  S.data[S.top]=n;  S.top++;  }  }  int peak(stack S) //lay thong tin gia tri cuoi cung  {  return S.data[S.top-1];  }  int pop(stack &S) //lay gia tri cuoi cung ra  {  if(!empty(S))  {  S.top--;  return S.data[S.top];  }  }  int dao(int n, stack &S, int a[])  {  for(int i=0;i<n;i++)  {  push(S,a[i]);  }  }  main()  {  int n, a[1000];  khoitao(S);  printf("Nhap n: "); scanf("%d",&n);  for(int i=0;i<n;i++)  {  printf("Nhap so thu %d: ",i+1); scanf("%d",&a[i]);  }  dao(n,S,a);  while(!empty(S))  {  printf("%d",pop(S));  }    } |

#### 4.Exercise 4 :Liệt kê dãy fibonacci (áp dụng queue)

|  |
| --- |
| #include"stdio.h"  #include"conio.h"  #define MAX 3  struct queue  {  int firt, last;  int data[MAX];  int dem;  };  queue Q;  void khoitao(queue &Q) //khoi tao ban dau cua hang cho  {  Q.firt=0;  Q.last=-1;  Q.dem=0;  }  int empty(queue Q) // ham kiem tra rong  {  if(Q.dem==0) return 1;  return 0;  }  int full(queue Q) // ham kiem tra full  {  if(Q.dem==MAX) return 1;  return 0;  }  int push(queue &Q, int n) //dua vao hang cho 1 gia tri  {  if(!full(Q))  {  Q.data[++Q.last]=n;  Q.dem++;  }  }  int pop(queue &Q) //lay ra gia tri dau tien  {  if(!empty(Q)){  int x=Q.data[Q.firt];  for(int i=Q.firt;i<=Q.last;i++)  {  Q.data[i]=Q.data[i+1];  }  Q.dem--;  Q.last--;  return x;  }  }  int peak(queue Q) //lay thong tin gia tri dau tien  {  return Q.data[Q.firt];  }  main()  {  khoitao(Q);  int n;  printf("Nhap n: ");scanf("%d",&n);  push(Q,0);  push(Q,1);  push(Q,1);  while(n--) //liet ke day fibonacci  {  printf("%d ",pop(Q));  int k=Q.data[Q.firt]+Q.data[Q.firt+1];  push(Q,k);  }    } |

#### 5.Exercise 5 :Đảo ngược 1 số (áp dụng queue)

|  |
| --- |
| #include"stdio.h"  #include"conio.h"  #define MAX 100  struct queue  {  int firt, last;  int data[MAX];  int dem;  };  queue Q;  void khoitao(queue &Q) //khoi tao ban dau cua hang cho  {  Q.firt=0;  Q.last=-1;  Q.dem=0;  }  int empty(queue Q) // ham kiem tra rong  {  if(Q.dem==0) return 1;  return 0;  }  int full(queue Q) // ham kiem tra full  {  if(Q.dem==MAX) return 1;  return 0;  }  int push(queue &Q, int n) //dua vao hang cho 1 gia tri  {  if(!full(Q))  {  Q.data[++Q.last]=n;  Q.dem++;  }  }  int pop(queue &Q) //lay ra gia tri dau tien  {  if(!empty(Q)){  int x=Q.data[Q.firt];  for(int i=Q.firt;i<=Q.last;i++)  {  Q.data[i]=Q.data[i+1];  }  Q.dem--;  Q.last--;  return x;  }  }  int peak(queue Q) //lay thong tin gia tri dau tien  {  return Q.data[Q.firt];  }  main()  {  khoitao(Q);  int n;  printf("Nhap n: ");scanf("%d",&n);  while(n>0)  {  push(Q,n%10);  n=n/10;  }  while(Q.last>-1)  {  printf("%d",pop(Q));  }    } |

## Module 5: DFS and BFS

### Lý thuyết

#### BFS (tìm kiếm theo chiều rộng)

là một [thuật toán tìm kiếm trong đồ thị](http://vi.wikipedia.org/w/index.php?title=Thu%E1%BA%ADt_to%C3%A1n_t%C3%ACm_ki%E1%BA%BFm_trong_%C4%91%E1%BB%93_th%E1%BB%8B&action=edit&redlink=1) trong đó việc tìm kiếm chỉ bao gồm 2 thao tác: (a) thăm một đỉnh của đồ thị; (b) thêm các đỉnh kề với đỉnh vừa thăm vào danh sách có thể thăm trong tương lai. Có thể sử dụng thuật toán tìm kiếm theo chiều rộng cho hai mục đích: tìm kiếm đường đi từ một đỉnh gốc cho trước tới một đỉnh đích, và tìm kiếm đường đi từ đỉnh gốc tới tất cả các đỉnh khác. Trong đồ thị không có trọng số, thuật toán tìm kiếm theo chiều rộng luôn tìm ra đường đi ngắn nhất có thể. Thuật toán BFS bắt đầu từ đỉnh gốc và lần lượt thăm các đỉnh kề với đỉnh gốc. Sau đó, với mỗi đỉnh trong số đó, thuật toán lại lần lượt thăm các đỉnh kề với nó mà chưa được thăm trước đó và lặp lại.

#### DFS (tìm kiếm theo chiều sâu)

là một [thuật toán](http://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n) duyệt hoặc tìm kiếm trên một [cây](http://vi.wikipedia.org/wiki/C%C3%A2y_(l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B)) hoặc một [đồ thị](http://vi.wikipedia.org/wiki/%C4%90%E1%BB%93_th%E1%BB%8B). Thuật toán khởi đầu tại gốc (hoặc chọn một đỉnh nào đó coi như gốc) và phát triển xa nhất có thể theo mỗi nhánh.

Thông thường, DFS là một dạng [tìm kiếm thông tin không đầy đủ](http://vi.wikipedia.org/w/index.php?title=T%C3%ACm_ki%E1%BA%BFm_th%C3%B4ng_tin_kh%C3%B4ng_%C4%91%E1%BA%A7y_%C4%91%E1%BB%A7&action=edit&redlink=1) mà quá trình tìm kiếm được phát triển tới đỉnh con đầu tiên của nút đang tìm kiếm cho tới khi gặp được đỉnh cần tìm hoặc tới một nút không có con. Khi đó giải thuật [quay lui](http://vi.wikipedia.org/wiki/Quay_lui_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)) về đỉnh vừa mới tìm kiếm ở bước trước.

### Exercise

#### 1.Exercise 1 : BFS

|  |
| --- |
| #include"stdio.h"  #include"conio.h"  #define MAX 100  int a[100][100]={0};  int dinh;  struct queue  {  int firt, last;  int data[MAX];  int dem;  };  queue Q;  void khoitao(queue &Q) //khoi tao ban dau cua hang cho  {  Q.firt=0;  Q.last=-1;  Q.dem=0;  }  int empty(queue Q) // ham kiem tra rong  {  if(Q.dem==0) return 1;  return 0;  }  int full(queue Q) // ham kiem tra full  {  if(Q.dem==MAX) return 1;  return 0;  }  int push(queue &Q, int n) //dua vao hang cho 1 gia tri  {  if(!full(Q))  {  Q.data[++Q.last]=n;  Q.dem++;  }  }  int pop(queue &Q) //lay ra gia tri dau tien  {  if(!empty(Q)){  int x=Q.data[Q.firt];  for(int i=Q.firt;i<=Q.last;i++)  {  Q.data[i]=Q.data[i+1];  }  Q.dem--;  Q.last--;  return x;  }  }  int peak(queue Q) //lay thong tin gia tri dau tien  {  return Q.data[Q.firt];  }  void nhapdothi(int a[100][100], int dinh) // Nhap vao do thi  {  int u=1;  while(dinh--)  {  int lk;  printf("Nhap so phan tu lien ket voi dinh -%d: ",u);// thong tin cac dinh lien ket voi dinh thu k  scanf("%d",&lk);  if(lk!=0) // neu co lien ket voi cac dinh khac  {  for(int i=0;i<lk;i++)  {  int v;  printf("Gia tri thu %d lien ket voi -%d la: ",i+1,u);  scanf("%d",&v);  a[u][v]=1; // danh dau 2 dinh u,v co lien ket vs nhau  }  }  u++;  }  }  void timduong(int u, int v, int dinhcha[]) //ham tim duong  {  if(u==v) // neu gia bat dau va ket thuc = nhau, tra ve  printf("%d ",v);  else  {  if(dinhcha[v]==-1) printf("Khong co duong di"); // neu gia tri ket thuc khong co dinh cha => khong co lien ket  else  {  timduong(u,dinhcha[v],dinhcha); //tim duong dua vao dinh cha cua dinh ket thuc  printf("%d ",v);  }  }  }  void BFS()  {  int start,finish; //nhap dinh bat dau  printf("Nhap dinh bat dau tim duong: ");scanf("%d",&start);  printf("Nhap dinh ket thuc tim duong: ");scanf("%d",&finish);  int danhdau[MAX],dinhcha[MAX];  for(int i=1;i<=dinh;i++)  {  danhdau[i]=0;  //danhdau[i]=0 chua di qua lan nao  //danhdau =1 dang xet  //danhdau =2 cac dinh ke voi i da duoc danh dau  dinhcha[i]=-1; //mang luu cac dinh cha cua dinh i  }  danhdau[start]=1;  push(Q,start); //Dua start vao hang cho  while(!empty(Q))  {  int u=pop(Q);//lay phan tu dau tien ra  for(int v=1;v<=dinh;v++) //xet cac dinh lan can  {  if(a[u][v]!=0 && danhdau[v]==0)  {  danhdau[u]=1; //danh dau dinh dang xet  dinhcha[v]=u; //xac dinh dinh cha cho cac dinh lan can  push(Q,v); //dua cac dinh lan can vao hang cho  }  }  danhdau[u]=2;  }  timduong(start,finish,dinhcha);    }  main()  {  khoitao(Q);  int n;  printf("Nhap so dinh: ");scanf("%d",&dinh);  nhapdothi(a,dinh);  BFS();    } |

#### 2.Exercise 2 : số lượng thành phần liên thông BFS

|  |
| --- |
| #include"stdio.h"  #include"conio.h"  #define MAX 100  int a[100][100]={0};  int dinh,x;  int danhdau[MAX],dinhcha[MAX];  struct queue  {  int firt, last;  int data[MAX];  int dem;  };  queue Q;  void khoitao(queue &Q) //khoi tao ban dau cua hang cho  {  Q.firt=0;  Q.last=-1;  Q.dem=0;  }  int empty(queue Q) // ham kiem tra rong  {  if(Q.dem==0) return 1;  return 0;  }  int full(queue Q) // ham kiem tra full  {  if(Q.dem==MAX) return 1;  return 0;  }  int push(queue &Q, int n) //dua vao hang cho 1 gia tri  {  if(!full(Q))  {  Q.data[++Q.last]=n;  Q.dem++;  }  }  int pop(queue &Q) //lay ra gia tri dau tien  {  if(!empty(Q)){  int x=Q.data[Q.firt];  for(int i=Q.firt;i<=Q.last;i++)  {  Q.data[i]=Q.data[i+1];  }  Q.dem--;  Q.last--;  return x;  }  }  int peak(queue Q) //lay thong tin gia tri dau tien  {  return Q.data[Q.firt];  }  void nhapdothi(int a[100][100], int dinh) // Nhap vao do thi  {  int u=1;  while(dinh--)  {  int lk;  printf("Nhap so phan tu lien ket voi dinh %d: ",u);// thong tin cac dinh lien ket voi dinh thu k  scanf("%d",&lk);  if(lk!=0) // neu co lien ket voi cac dinh khac  {  for(int i=0;i<lk;i++)  {  int v;  printf("Gia tri thu %d lien ket voi %d la: ",i+1,u);  scanf("%d",&v);  a[u][v]=1; // danh dau 2 dinh u,v co lien ket vs nhau  }  }  u++;  }  }  void BFS()  {  int start,finish; //nhap dinh bat dau  for(int i=1;i<=dinh;i++)  {  danhdau[i]=0;  //danhdau[i]=0 chua di qua lan nao  //danhdau =1 dang xet  //danhdau =2 cac dinh ke voi i da duoc danh dau  dinhcha[i]=-1; //mang luu cac dinh cha cua dinh i  }  danhdau[1]=1;  push(Q,1);  while(!empty(Q))  {  int u=pop(Q);//lay phan tu dau tien ra  for(int v=1;v<=dinh;v++) //xet cac dinh lan can  {  if(a[u][v]!=0 && danhdau[v]==0)  {  danhdau[u]=1; //danh dau dinh dang xet  dinhcha[v]=u; //xac dinh dinh cha cho cac dinh lan can  push(Q,v); //dua cac dinh lan can vao hang cho  }  }  danhdau[u]=2;  if(empty(Q) && u!=dinh) push(Q,u+1);  }    }  int tplt()  {  int dem=0;  for(int v=1;v<=dinh;v++)  {  if(dinhcha[v]==-1) dem++;  }  return dem;  }  main()  {  khoitao(Q);  int n;  printf("Nhap so dinh: ");scanf("%d",&dinh);  nhapdothi(a,dinh);  BFS();  printf("so thanh phan lien thong la %d",tplt());    } |

#### 3.Exercise 3 : DFS

|  |
| --- |
| #include "stdlib.h"  #include "conio.h"  #include"stdio.h"  #include <iostream>  #include <stack>  #define MaxV 20  int A[MaxV][MaxV]; // Ma tran ke  int V=0; // So dinh cua do thi  bool \*visited = new bool[MaxV+1];  int DocMTKe(char \*fileName, int A[][MaxV], int &V)  {  FILE \*f = fopen(fileName,"rt");  if(f==NULL)  {  printf("Doc file loi");  return 0;  }  fscanf(f,"%d",&V);  for(int i=0;i<V;i++)  {  for(int j=0;j<V;j++)  {  fscanf(f,"%d",&(A[i][j]));  }  }  return 1;  }  void XuatMTKe(int A[][MaxV],int V)  {  printf("\nMa tran ke:\n");  for(int i=0; i<V;i++)  {  for(int j=0;j<V;j++)  {  printf("%3d",A[i][j]);  }  printf("\n");  }  }  void init\_visited()  {  for(int i=0; i<=MaxV; i++)  {  visited[i]=false; // khoi tao false  }  }  void DFS\_DEQUY(int v)  {  printf("%d ",v);  visited[v]=true;  for(int u=0;u<V;u++)  {  if((A[v][u]!=0)&& !visited[u]) // Xac dinh cac dinh ke  {  DFS\_DEQUY(u);  }  }  }  main()  {  DocMTKe("C:\\Dothi\_1.txt",A,V);  XuatMTKe(A,V);  printf("\n\nDFS: ");  init\_visited();  DFS\_DEQUY(0);    } |

#### 4.Exercise 4 : Tính chất liên thông của đồ thị (DFS)

|  |
| --- |
| #include "stdlib.h"  #include "conio.h"  #include"stdio.h"  #include <iostream>  #include <stack>  #define MaxV 20  int A[MaxV][MaxV]; // Ma tran ke  int V=0; // So dinh cua do thi  bool \*visited = new bool[MaxV+1];  int DocMTKe(char \*fileName, int A[][MaxV], int &V)  {  FILE \*f = fopen(fileName,"rt");  if(f==NULL)  {  printf("Doc file loi");  return 0;  }  fscanf(f,"%d",&V);  for(int i=0;i<V;i++)  {  for(int j=0;j<V;j++)  {  fscanf(f,"%d",&(A[i][j]));  }  }  return 1;  }  void XuatMTKe(int A[][MaxV],int V)  {  printf("\nMa tran ke:\n");  for(int i=0; i<V;i++)  {  for(int j=0;j<V;j++)  {  printf("%3d",A[i][j]);  }  printf("\n");  }  }  void init\_visited()  {  for(int i=0; i<=MaxV; i++)  {  visited[i]=false; // khoi tao false  }  }  void DFS\_DEQUY(int v)  {  printf("%d ",v);  visited[v]=true;  for(int u=0;u<V;u++)  {  if((A[v][u]!=0)&& !visited[u]) // Xac dinh cac dinh ke  {  DFS\_DEQUY(u);  }  }  }  int LienThong(){  init\_visited();  DFS\_DEQUY(0);  for (int i=0; i<V; i++)  if (!visited[i])  return 0;  return 1;  }  main()  {  DocMTKe("C:\\Dothi\_1.txt",A,V);  XuatMTKe(A,V);  printf("\n\nDFS: ");  if(LienThong()) printf("\n Do thi lien thong");  else printf("\n Do thi khong lien thong");    } |

## Module 6: hamington

## Module 7: prim

## Module 8: krushkal

## Module 9: A\*

## Module 10: quy hoạch động

# PROJECT

## Project 1 : project\_name

## Project 2 : Project\_name

# FAULT END EXPRIENCE