[Tự học lập trình C - Bài 1: Một số khái niệm cơ bản](http://tuhocanninhmang.com/ngon-ngu-lap-trinh/c-cpp/bai-1-mot-so-khai-niem-co-ban-trong-c.htm)

**1.1. Bộ ký tự**

Mọi ngôn ngữ lập trình đều được xây dựng từ một bộ ký tự nào đó. Ngôn ngữ C được xây dựng trên bộ ký tự sau:

26 chữ cái hoa : A B C .. Z

26 chữ cái thường : a b c .. z

10 chữ số : 0 1 2 .. 9

Các ký hiệu toán học : + - \* / = ( )

Ký tự gạch nối : \_

Các ký tự khác : . , : ; [ ]  {} ! \ & % # $ ...

Dấu cách (space) dùng để tách các từ.

***@*Lưu ý**: Khi viết chương trình, ta không được sử dụng bất kỳ ký tự nào khác ngoài các ký tự trên.

**1.2 Từ khóa**

Từ khoá là những từ được sử dụng để khai báo các kiểu dữ liệu, để viết các toán tử và các câu lệnh. Bảng dưới đây liệt kê các từ khoá của TURBO C:

|  |  |  |  |
| --- | --- | --- | --- |
| Asm | break | case | cdecl |
| Char | const | continue | default |
| Do | double | else | enum |
| extern | far | float | for |
| Goto | huge | if | int |
| interrupt | long | near | pascal |
| register | return | short | signed |
| sizeof | static | struct | switch |
| tipedef | union | unsigned | void |
| volatile | while |  |  |

**@Lưu ý:**

- Không được dùng các từ khoá để đặt tên cho các hằng, biến, mảng, hàm,...

- Từ khoá phải được viết bằng chữ thường, ví dụ: viết từ khoá khai báo kiểu nguyên là int chứ không phải là INT.

**1.3 Tên**

Khái niệm tên rất quan trọng trong quá trình lập trình, nó không những thể hiện rõ ý nghĩa trong chương trình mà còn dùng để xác định các đại lượng khác nhau khi thực hiện chương trình. Tên thường được đặt cho hằng, biến, mảng, con trỏ, nhãn,… Chiều dài tối đa của tên là 32 ký tự.

Tên biến hợp lệ là một chuỗi ký tự liên tục gồm: Ký tự chữ, số và dấu gạch dưới. Ký tự đầu của tên phải là chữ hoặc dấu gạch dưới. Khi đặt tên không được đặt trùng với các từ khóa.

**Ví dụ 1.1:**

Các tên đúng: delta, a\_1, Num\_ODD, Case

Các tên sai:

3a\_1 (ký tự đầu là số)

num-odd (sử dụng dấu gạch ngang)

int (đặt tên trùng với từ khóa)

del ta (có khoảng trắng)

f(x) (có dấu ngoặc tròn)

Ví dụ 1.2: number khác Number

case khác Case (case là từ khóa, do đó bạn đặt tên là Case vẫn đúng)

***@*Lưu ý:**Trong C, tên phân biệt chữ hoa, chữ thường

**1.4 Kiểu dữ liệu**

Có 4 kiểu dữ liệu cơ bản trong C là: char, int, float, double.

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Kiểu dữ liệu** | **Kích thước** | **Miền giá trị** |
| 1  2  3  4  5  6  7  8  9  10  11 | unsigned char  char  enum  unsigned int  short int  int  unsigned long  long  float  double  long double | 1 byte  1 byte  2 bytes  2 bytes  2 bytes  2 bytes  4 bytes  4 bytes  4 bytes  8 bytes  10 bytes | 0 đến 255  – 128 đến 127  – 32,768 đến 32,767  0 đến 65,535  – 32,768 đến 32,767  – 32,768 đến 32,767  0 đến 4,294,967,295  – 2,147,483,648 đến 2,147,483,647  3.4 \* 10–38 đến 3.4 \* 1038  1.7 \* 10–308 đến 1.7 \* 10308  3.4 \* 10–4932 đến 1.1 \* 104932 |

**1.5 Lời chú thích**

Trong khi lập trình cần phải ghi chú để giải thích các biến, hằng, thao tác xử lý giúp cho chương trình rõ ràng dễ hiểu, dễ nhớ, dễ sửa chữa và để người khác đọc vào dễ hiểu. Trong C có các ghi chú sau: // hoặc /\* nội dung ghi chú \*/

Tóm lại, đối với ghi chú dạng // dùng để ghi chú một hàng và dạng /\* …. \*/ có thể ghi chú một hàng hoặc nhiều hàng.

[**Tự học lập trình C - Bài 2: Cấu trúc chương trình C**](http://tuhocanninhmang.com/ngon-ngu-lap-trinh/c-cpp/bai-2-cau-truc-cua-chuong-trinh-c.htm)

Một chương trình bao gồm một hoặc nhiều hàm, mỗi hàm được người lập trình tổ chức để giải quyết một hoặc một số công việc nào đó của bài toán cần giải quyết. Một chương trình C để có thể thực thi được luôn cần phải có hàm main().

Cấu trúc cơ bản của chương trình như sau:

•    Các #include ( dùng để khai báo sử dụng các hàm chuẩn)

•    Các #define  ( dùng để định nghĩa các hằng )

•    Khai báo các đối tượng dữ liệu ngoài ( biến, mảng, cấu trúc vv..).

•    Khai báo nguyên mẫu các hàm.

•    Hàm main().

•    Định nghĩa các hàm ( hàm main có thể đặt sau hoặc xen vào giữa các hàm khác).

**Ví dụ 1.3:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | #include <stdio.h>  //k.báo sử dụng thư viện xuất/nhập chuẩn của C  void main(void)      {          double x,y;                      //Khai báo 2 biến x,y kiểu số thực          printf("\n Nhap x va y");        //xuất dữ liệu ra màn hình          scanf("%lf%lf",&x,&y);           //nhập dữ liệu từ bàn phím và lưu vào vùng địa chỉ cấp cho 2 biến x,y  } |

***@*Lưu ý:** **Một số qui tắc cần nhớ khi viết chương trình:**

- Mỗi câu lệnh có thể viết trên một hay nhiều dòng nhưng phải kết thúc bằng dấu (;).

- Trong chương trình, khi ta sử dụng các hàm chuẩn, ở đầu chương trình ta phải khai báo sử dụng, ví dụ: #include "stdio.h".

[**Tự học lập trình C - Bài 3: Hằng - Biến - Toán tử - B.thức**](http://tuhocanninhmang.com/ngon-ngu-lap-trinh/c-cpp/tu-hoc-lap-trinh-c-bai-3.htm)

**3.1 Khai báo hằng**

Hằng là các đại lượng mà giá  trị của nó không thay đổi trong quá trình tính toán.   Nguyên tắc đặt tên hằng theo nguyên tắc đặt tên của C.

+ Khai báo

**Cú pháp:**

            #define   <tên hằng>   <giá trị>

Diễn giải:

#define: Từ khóa để định nghĩa một biến hằng

<tên hằng>: là tên của hằng mà ta cần định nghĩa

<giá trị>: Giá trị khởi gán cho hằng

**Ví dụ:**

            #define   MAX   1000

            Lúc này, tất cả các tên MAX trong chương trình xuất hiện sau này đều được thay bằng 1000. Vì vậy, ta thường gọi MAX là tên hằng, nó biểu diễn số 1000.

**3.2 Khai báo biến**

+ Khai báo

**Cú pháp:**

              <Kiểu dữ liệu> <Danh sách tên biến>;

Diễn giải:

- <Kiểu dữ liệu>: là kiểu dữ liệu muốn khai báo cho biến

- <Danh sách tên biến>: gồm các tên biến có cùng kiểu dữ liệu, mỗi tên biến cách nhau dấu phẩy (,), cuối cùng là dấu chấm phẩy (;).

**Ví dụ**

int iTuoi; //khai báo biến iTuoi có kiểu int

float fTrongLuong; //khai báo biến fTrongLuong có kiểu float

char cKyTu1, cKyTu2; //khai báo biến cKyTu1, cKyTu2 có kiểu char

+ Vừa khai báo vừa khởi gán giá trị cho biến

Có thể kết hợp việc khai báo với toán tử gán để biến nhận ngay giá trị lúc mới khai báo.

**Ví dụ**

*Khai báo trước, gán giá trị sau*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | void main(void)  {      int a, b, c;      a = 1;      b = 2;      c = 5;      …  } |

*Vừa khai báo vừa gán giá trị*:

|  |  |
| --- | --- |
| 1  2  3  4  5 | void main(void)  {      int a = 1, b = 2, c = 5;      …  } |

+ Phạm vi của biến

Khi lập trình, phải nắm rõ phạm vi của biến. Nếu khai báo và sử dụng không đúng, không rõ ràng sẽ dẫn đến sai sót khó kiểm soát được.

a. Khai báo biến ngoài (biến toàn cục):

Vị trí biến đặt bên ngoài tất cả các hàm, cấu trúc...Các biến toàn cục có ảnh hưởng đến toàn bộ chương trình. Chu trình sống của nó là bắt đầu từ khi chạy chương trình đến lúc kết thúc chương trình.

b. Khai báo biến trong (biến cục bộ):

Vị trí biến đặt bên trong hàm, cấu trúc…. Chỉ ảnh hưởng nội bộ bên trong hàm, cấu trúc đó…. Chu trình sống của nó bắt đầu từ lúc hàm, cấu trúc được gọi thực hiện đến lúc thực hiện xong.

**3.3 Biểu thức**

Biểu thức là sự phối hợp của những toán tử và toán hạng.

**Ví dụ:**

a + b

b = 1 + 5 \* 2/i

a = 6 % (7 + 1)

x++ \* 2/4 + 5 – power(i, 2)

Toán hạng sử dụng trong biểu thức có thể là hằng số, biến, hàm.

**3.4 Phép toán**

Trong C có 4 nhóm toán tử chính yếu sau đây:

+ Phép toán số học

+ : phép cộng

– : phép trừ

\* : phép nhân

/ : phép chia

% : phép chia lấy phần dư, được áp dụng trên các toán hạng có kiểu dữ liệu char, int, long

+ Phép toán quan hệ

> : lớn hơn

>= : lớn hơn hoặc bằng

<   : nhỏ hơn

<= : nhỏ hơn hoặc bằng

== : bằng

!= : khác

**@Lưu ý:**

- Kết quả của phép toán quan hệ là số nguyên kiểu int, bằng 1 nếu đúng, bằng 0 nếu sai.

- Phép toán quan hệ ngoài toán hạng được sử dụng là kiểu dữ liệu số hoặc kiểu dữ liệu char.

+ Phép toán logic

! : (phép toán Phủ định)

&&: (phép toán Và)

|| : (phép Hoặc)

+ Phép toán trên bit (bitwise)

& : và(AND)

| : hoặc (OR)

^ : hoặc loại trừ (XOR)

>> : dịch phải

<< : dịch trái

~ : đảo

+ Phép gán hợp

Biểu thức gán là biểu thức có dạng:

            v=e

Trong đó v là một biến (hay phần tử mảng ), e là một biểu thức. Giá trị của biểu thức gán là giá trị của e, kiểu của nó là kiểu của v. Nếu đặt dấu ; vào sau biểu thức gán ta sẽ thu được phép toán gán có dạng:

            v=e;

Biểu thức gán có thể sử dụng trong các phép toán và các câu lệnh như các biểu thức khác. Ví dụ như khi ta viết

            a=b=5;

thì điều đó có nghĩa là gán giá trị của biểu thức b=5 cho biến a. Kết qủa là b=5 và a=5.

Hoàn toàn tương tự như :

            a=b=c=d=6; gán 6 cho cả a, b, c và d

            z=(y=2)\*(x=6);            { ở đây \* là phép toán nhân }

gán 2 cho y, 6 cho x và nhân hai biểu thức lại cho ta z=12.

+ Phép toán tăng giảm

C đưa ra hai phép toán một ngôi để tăng và giảm các biến (nguyên và thực). Toán tử tăng là ++ sẽ cộng 1 vào toán hạng của nó, toán tử giảm là -- sẽ trừ toán hạng đi 1.

**Ví dụ**

            n=5

            m=4

            ++n      Cho ta   n=6

            --m       Cho ta   m=3

Ta có thể viết phép toán ++ và -- trước hoặc sau toán hạng: ++n, n++, --n, n--.

Sự khác nhau của ++n và n++ ở chỗ : trong phép n++ thì tăng giá trị của biến n lên 1 sau khi giá trị của nó đã được sử dụng, còn trong phép ++n thì n được tăng trước khi sử dụng. Sự khác nhau giữa n-- và --n cũng như vậy.

+ Toán tử điều kiện

**Cú pháp:**

<biểu thức đều kiện>?<biểu thức1>:<biểu thức2>

Diễn giải:Nếu biểu thức điều kiện có giá trị đúng thì kết quả là <biểu thức 1>, nếu sai kết quả là <biểu thức 2>

**Ví dụ:**

            a=2

            b=5

            max=(a>b)? a:b;

kết quả là max = 5

[**Tự học lập trình C - Bài 4: Nhập/xuất dữ liệu**](http://tuhocanninhmang.com/ngon-ngu-lap-trinh/c-cpp/tu-hoc-lap-trinh-c-bai-4-nhap-xuat-du-lieu.htm)

**4.1 Lệnh xuất**

***Cú pháp:***

**printf**("chuỗi định dạng"[, đối mục 1, đối mục 2,…]);

Chức năng: Đưa kết quả các <đối mục> ra màn hình

- <“chuỗi định dạng”>: dùng để định dạng cho dữ liệu xuất ra màn hình của các <đối mục>

- <Đối mục 1>, <Đối mục >…: là các mục dữ kiện cần in ra màn hình. Các <đối mục này> có thể là biến, hằng hoặc biểu thức phải được định trị trước khi in ra.

- Chuỗi định dạng: được đặt trong cặp nháy kép (" "), gồm 3 loại:

+ Đối với chuỗi kí tự ghi như thế nào in ra giống như thế ấy.

+ Đối với những kí tự chuyển đổi dạng thức cho phép kết xuất giá trị của các đối mục ra màn hình tạm gọi là mã định dạng.

Sau đây là các dấu mô tả định dạng:

%c : Ký tự đơn

%s : Chuỗi

%d : Số nguyên hệ thập phân

%f : Số chấm động (ký hiệu thập phân)

%e : Số chấm động (ký hiệu có số mũ)

%g : Dùng %e hoặc %f, tuỳ theo loại nào ngắn hơn, không in các số 0 vô nghĩa

%x : Hệ 16 không dấu

%u : Số thập phân không dấu

%o : Số nguyên bát phân không dấu

l : Tiền tố dùng kèm với %d, %u, %x, %o để chỉ số nguyên dài (ví dụ %ld)

**Ví dụ:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #include <stdio.h>  //cần khai báo tiền xử lý stdio.h, vì trong hàm main()  // có dùng hàm printf  void main (void)  {         printf(“Hello!”);  } |

//Cho ra màn hình kết quả: Hello

**Ví dụ:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | #include <stdio.h>  //cần khai báo tiền xử lý stdio.h, vì trong hàm main()  // có dùng hàm printf  void main (void)  {       int n=65;       printf(“Gia tri cua n la: %d”,n);  } |

//Cho ra màn hình kết quả: Gia tri cua n la:65

//%d dùng để định dạng cho giá trị của n

// nếu thay %d thành %c thì kết quả cho ra màn hình là: ***Gia tri cua n la A,***vì %c là mã định dạng cho ký tự, tương ứng với mã 65 là ký tự A

**+ Các ký tự điều khiển và ký tự đặc biệt**

\n : Nhảy xuống dòng kế tiếp canh về cột đầu tiên.

\t : Canh cột tab ngang.

\r : Nhảy về đầu hàng, không xuống hàng.

\a : Tiếng kêu bip.

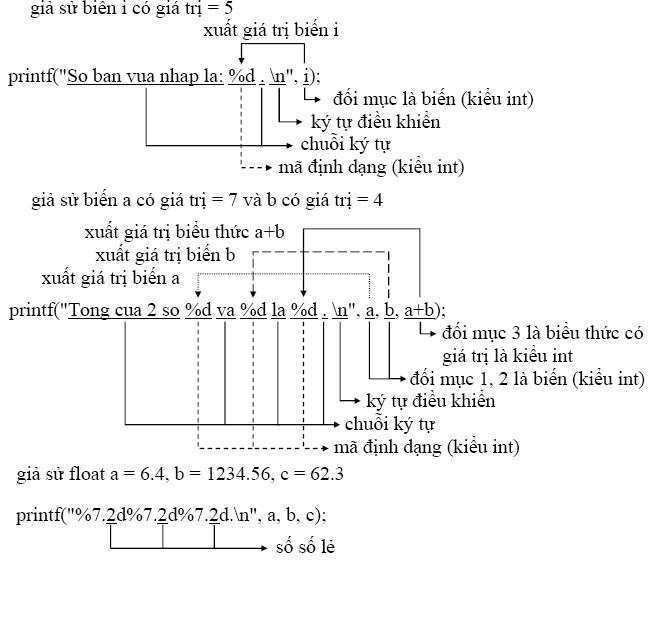
\\ : In ra dấu \

\" : In ra dấu "

\' : In ra dấu '

%%: In ra dấu %

**Vi dụ:**



**4.2 Lệnh nhập**

***Cú pháp:*scanf**("chuỗi định dạng"[, đối mục 1, đối mục 2,…]);

Chức năng: Đọc dữ liệu từ bàn phím

- Đạnh dạng tương tự như hàm printf

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | #include <stdio.h>  void main(void)  {         float a,b,c       printf("nhap vao he so a, b, c");       scanf("%f%f%f", &a,&b,&c);  } |

 -Nhập vào 3 số a, b,c phải cách nhau bằng **khoảng trắng**hoặc **enter**.

- Các ký tự định dạng là tương ứng từng cặp 1 của các biến

- Ký tự & là ký tự lấy địa chỉ của biến vì hàm scanf sẽ đọc dữ liệu từ bàn phím lưu vào vùng địa chỉ được xác định

**4.3. Bài tập**

1. Viết chương trình đọc vào 2 số nguyên và in ra kết quả của phép (+), phép trừ (-), phép nhân (\*), phép chia (/).

2. Viết chương trình đọc từ bàn phím 3 số nguyên biểu diễn ngày, tháng, năm và xuất ra màn hình dưới dạng "ngay/thang/nam".

[**Tự học lập trình C - Bài 5: Câu lệnh điều kiện**](http://tuhocanninhmang.com/ngon-ngu-lap-trinh/c-cpp/bai-5-cau-lenh-dieu-kien.htm)

**5.1 Lệnh if**

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)

**- Dạng 1**

**Cú pháp:**

**if (bt\_logic)**

**<khối lệnh>;**

**Diễn giải:**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.

**Ví dụ 1:**Viết chương trình nhập vào 2 số nguyên a, b. Tìm và in ra số lớn nhất.

a. Phác họa lời giải:

Trước tiên ta cho giá trị a là giá trị lớn nhất bằng cách gán a cho max (max là biến được khai báo cùng kiểu dữ liệu với a, b). Sau đó so sánh b với a, nếu b lớn hơn a ta gán b cho max và cuối cùng ta được kết quả max là giá trị lớn nhất.

b. Viết chương trình

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | #include <stdio.h>  void  main(void){                int ia, ib, imax;                printf("Nhap vao so a: ");                scanf("%d", &ia);                printf("Nhap vao so b: ");                scanf("%d", &ib);                imax = ia;                if (ib > ia)                            imax = ib;                printf("So lon nhat = %d.\n", imax);    } |

**- Dạng 2**

**Cú pháp:**

**if (bt\_logic)**

**<khối lệnh 1>;**

**else**

**<Khối lệnh 2>;**

**Diễn giải:** 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.

**Ví dụ 2:**Viết chương trình nhập vào 2 số nguyên a, b. In ra thông báo "a bằng b" nếu a = b, ngược lại in ra thông báo "a khác b".

a. Phác họa lời giải:

So sánh a với b, nếu a bằng b thì in ra câu thông báo "a bằng b", ngược lại in ra thông báo "a khác b".

b. Viết chương trình:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | #include <stdio.h>  void main(void)    {         int ia, ib;         printf("Nhap vao so a: ");         scanf("%d", &ia);         printf("Nhap vao so b: ");         scanf("%d", &ib);    if (ia == ib)         printf("a bang b\n");    else         printf("a khac b\n");    } |

**Lưu ý:**  Nếu <**khối lệnh>,**bao gồm từ 2 lệnh trở lên thì phải đặt trong cặp dấu {}

**5.2 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;**

**Diễn giải**: 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.

**Ví dụ 3:** Viết chương trình nhập vào 2 số nguyên a, b. In ra thông báo "a lớn hơn b" nếu a>b, in ra thông báo "a nhỏ hơn b" nếu a<b, in ra thông báo "a bằng b" nếu a=b.

a. Phác họa lời giải

Trước tiên so sánh a với b. Nếu a > b thì in ra thông báo "a lớn hơn b", ngược lại nếu a < b thì in ra thông báo "a nhỏ hơn b", ngược với 2 trường hợp trên thì in ra thông báo "a bằng b".

b. Viết chương trình:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | #include <Stdio.h>    void main(void)    {    int ia, ib;         printf("Nhap vao so a: ");scanf("%d", &ia);         printf("Nhap vao so b: ");scanf("%d", &ib);    if (ia>ib)         printf("a lon hon b.\n");    else if...} |

**5.3 Lệnh 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;]**

**}**

**Diễn giải**: 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 { }

**Ví dụ 4:** Nhập vào một tháng trong năm, cho biết tháng này thuộc quý nào trong năm

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | #include <stdio>    #include <stdio>    void main(void)    {    int ithang;    printf("Nhap vao thang: ");    scanf("%d", &ithang);    switch(ithang)    {        case 1: case 2: case 3 : printf("Quy 1.\n");break;        case 4: case 5: case 6: printf("Quy 2.\n");break;        case 7: case 8: case 9: printf("Quy 3.\n");break;        case 10: case 11: case 12: printf("Quy 4.\n");break;        default : printf("Phai nhap vao so trong khoang 1..12\n");    }    getch(); //Dừng màn hình để xem kết quả, gõ phím bất kỳ để tiếp tục      } |

**5.4 Bài thực hành**

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

**Hướng dẫn:** Nhập vào số nguyên dương x. Kiểm tra nếu x chia hết cho hai (tức x chia 2 dư 0) thì x là số chẵn, ngược lại là số lẻ.

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

**Hướng dẫn:**Ta có 4 số nguyên a, b, c, d. Tìm 2 số nguyên lớn nhất x, y của 2 cặp (a, b) và (c, d). Sau đó so sánh 2 số nguyên x, y để tìm ra số nguyên lớn nhất.

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

**Hướng dẫn:**Nhập vào 2 biến a, b

Nếu a=0 thì

            Nếu b=0 thì

                        Phương trình vô số nghiệm

            Ngược lại

                        Phương trình vô nghiệm

            Hết nếu

Ngược lại

            x= -b/a

Hết nếu

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

**Hướng dẫn:**Nhập vào 3 biến a, b, c.

Nếu a=0 thì

            Giải phương trình bậc nhất

Ngược lại

Tính Delta = b\*b - 4\*a\*c

Nếu Delta < 0 thì

Phương trình vô nghiệm

Ngược lại

Nếu Delta = 0 thì

x1 = x2 = - b/(2\*a)

Ngược lại

x1 = (- b - sqrt(Delta))/(2\*a)

x2 = (- b + sqrt(Delta))/(2\*a)

Hết Nếu

Hết Nếu

Hết nếu

*- Sqrt(Delta) là hàm lấy căn bậc hai của Delta*

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.

**Hướng dẫn:**Nhập giờ, phút, giây vào 3 biến gio, phut, giay và nhập và0 giây công thêm cho biến them:

phut = (phut + (giay + them)/60)%60

giay = (giay + them)%60

gio = (gio + phut/60)%24

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

**Hướng dẫn:**Nhập vào tháng

Nếu là tháng 1, 3, 5, 7, 8, 10, 12 thì có 30 ngày

Nếu là tháng 4, 6, 9, 11 thì có 31 ngày

Nếu là tháng 2 và là năm nhuận thì có 29 ngày ngược lại 28 ngày

(Năm nhuận là năm chia hết cho 4)

7. Viết chương trình xác định biến ký tự color rồi in ra thông báo

- RED, nếu color = 'R' hoặc color = 'r'

- GREEN, nếu color = 'G' hoặc color = 'g'

- BLUE, nếu color = 'B' hoặc color = 'b'

- BLACK, nếu color có giá trị khác.

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).

9. Viết chương trình nhập vào điểm 3 môn thi: Toán, Lý, Hóa của học sinh. Nếu tổng điểm >= 15 và không có môn nào dưới 4 thì in kết quả đậu. Nếu đậu mà các môn đều lớn hơn 5 thì in ra lời phê "Học đều các môn", ngược lại in ra "Học chưa đều các môn", các trường hợp khác là "Thi hỏng".

10. Viết chương trình nhập số giờ làm và lương giờ rồi tính số tiền lương tổng cộng. Nếu số giờ làm lớn hơn 40 thì những giờ làm dôi ra được tính 1,5 lần.

11. Viết chương trình tính tiền điện gồm các khoảng sau:

- Tiền thuê bao điện kế: 1000đ/tháng

- Định mức sử dụng điện cho mỗi hộ là: 50 KW với giá 230đ/KW

- Nếu phần vượt định mức <= 50KW thì tính giá 480đ/KW

- Nếu 50KW < phần vượt định mức < 100KW thì tính giá 700đ/KW

- Nếu phần vượt định mức <= 100KW thì tính giá 900đ/KW

Chỉ số mới và cũ được nhập vào từ bàn phím

- In ra màn hình chỉ số cũ, chỉ số mới, tiền trả định mức, tiền trả vượt định mức, tổng tiền phải trả.

[**Tự học lập trình C - Bài 6: Câu lệnh lặp (1)**](http://tuhocanninhmang.com/ngon-ngu-lap-trinh/c-cpp/tu-hoc-lap-trinh-c-bai-6-cau-lenh-lap-voi-so-lan-xac-dinh.htm)

**6.1 Lệnh for:**

**Cú pháp:**

**for ([biểu thức 1]; [biểu thức 2]; [biểu thức 3])**

**<khối lệnh>;**

**Ý nghĩa:**Là vòng lặp với số lần lặp được xác định trước, tức thực hiện <khối lệnh> n lần (n>=0), Quá trình lặp kết thúc khi [biểu thức 2]cho giá trị sai, hoặc khi thực hiện <khối lệnh> gặp lệnh nhảy ra khỏi vòng lặp

***Diễn giải***:

- Biểu thức 1: khởi tạo giá trị ban đầu cho biến điều khiển.

- Biểu thức 2: là quan hệ logic thể hiện điều kiện tiếp tục vòng lặp.

- Biểu thức 3: phép gán dùng thay đổi giá trị biến điều khiển.

***Nhận xét***:

- Biểu thức 1 bao giờ cũng chỉ được tính toán một lần khi gọi thực hiện for.

**@ *Lưu ý***:

- Biểu thức 1, 2, 3phải phân cách bằng dấu chấm phẩy (;)

- Nếu biểu thức 2 không có, vòng for được xem là luôn luôn **đúng**. Muốn thoát khỏi vòng lặp for phải dùng một trong 3 lệnh **break**, **goto**hoặc **return**.

- Với mỗi biểu thức có thể viết thành một dãy biểu thức con phân cách nhau bởi dấu phẩy. Khi đó các biểu thức con được xác định từ trái sang phải. Tính đúng sai của dãy biểu thức con trong biểu thức thứ 2 được xác định bởi biểu thức con cuối cùng.

- Trong thân for (<khối lệnh>) có thể chứa một hoặc nhiều cấu trúc điều khiển khác.

- Khi gặp lệnh **break**, cấu trúc lặp xâu nhất sẽ thoát ra.

- Trong thân for có thể dùng lệnh **goto**để thoát khỏi vòng lặp đến vị trí mong muốn.

- Trong thân for có thể sử dụng **return**để trở về một hàm nào đó.

- Trong thân for có thể sử dụng lệnh continue để chuyển đến đầu vòng lặp (bỏ qua các câu lệnh còn lại trong thân).

**Ví dụ 1:** Viết chương trình in ra màn hình n số tự nhiên đầu tiên.

#include <stdio.h>

#include <conio.h>

void main(void)

{

int i, n;

printf("Nhap vao gia tri n:");

scanf("%d", &n);

for(i = 1; i<=n; i++)

printf("%d :", i);

getch();

}

// trường hợp in theo chiều ngược lại

for(i = n; i>=1; i--)

printf("%d :", i);

**Ví dụ 2:**Viết chương trình nhập vào số nguyên n. Tính tổng các giá trị lẻ nhỏ hơn n.

#include <stdio.h>

#include <conio.h>

void main(void)

{

int i, in, is = 0;

printf("Nhap vao so n: ");

scanf("%d", &in);

is = 0;

for(i = 0; i<in; i++)

if (i % 2 != 0) //neu i la so le

is = is + i; //hoac is += i;

printf("Tong: %d", is);

getch();

}

**Ví dụ 3:** Đọc vào một loạt kí tự trên bàn phím, đếm số kí tự nhập vào. Kết thúc khi gặp dấu chấm '.'

#include <stdio.h>

#include <conio.h>

#define DAU\_CHAM '.'

void main(void)

{

char c;

int idem;

for(idem = 0; (c = getchar()) != DAU\_CHAM; )

idem++;

printf("So ki tu: %d.\n", idem);

getch();

}

Cách khác, dùng vòng lặp for khuyết các biểu thức

#include <stdio.h>

#include <conio.h>

#define DAU\_CHAM '.'

void main(void)

{

char c;

int idem = 0;

for(; ;)

{

c = getchar();

if (c == DAU\_CHAM) //nhap vao dau cham

break; //thoat vong lap

idem++;

}

printf("So ki tu: %d.\n", idem);

getch();

}

**Bài tập thực hành**

1. Viết chương trình in ra bảng mã ASCII

2. Viết chương trình tính tổng bậc 3 của N số nguyên đầu tiên.

3. Viết chương trình nhập vào một số nguyên rồi in ra tất cả các ước số của số đó.

4. Viết chương trình tính tổng nghịch đảo của N số nguyên đầu tiên theo công thức

                   S = 1 + 1/2 + 1/3 + … + 1/N

5. Viết chương trình tính tổng bình phương các số lẻ từ 1 đến N.

6. Viết chương trình nhập vào N số nguyên, tìm số lớn nhất, số nhỏ nhất.

7. Viết chương trình nhập vào N rồi tính giai thừa của N

(Còn tiếp)

[**Tự học lập trình C - Bài 7: Câu lệnh lặp (2)**](http://tuhocanninhmang.com/ngon-ngu-lap-trinh/c-cpp/tu-hoc-lap-trinh-c-bai-7-cau-lenh-lap-voi-so-lan-lap-chua-xac-dinh-truoc.htm)

**7.1 Lệnh while**

Vòng lặp thực hiện lặp lại khối lệnh trong khi biểu thức còn đúng

+ Cú pháp:

while (biểu thức)

<khối lệnh>;

**Diễn giải:**

- Biểu thức: có thể là một biểu thức hoặc nhiều biểu thức con. Nếu là nhiều biểu thức con thì cách nhau bởi dấu phẩy (,) và tính đúng sai của biểu thức được quyết định bởi biểu thức con cuối cùng.

- Trong thân while (<khối lệnh>) có thể chứa một hoặc nhiều cấu trúc điều khiển khác.

- Trong thân while có thể sử dụng lệnh continue để chuyển đến đầu vòng lặp (bỏ qua các câu lệnh còn lại trong thân).

- Muốn thoát khỏi vòng lặp while tùy ý có thể dùng các lệnh **break, goto, return**như trong lệnh **for.**

**Ý nghĩa:**Trước tiên **biểu thức**được kiểm tra, nếu **sai**thì kết thúc vòng lặp while (<khối lệnh> không được thi hành1 lần nào) nếu **đúng**thực hiện khối lệnh; lặp lại kiểm tra biểu thức

**Ví dụ 1:**Viết chương trình in ra n số tự nhiên đầu tiên.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | #include <studio>    #include <studio>    #include <studio>    void main(void)    {          int in,i=0;          printf(“nhap gia tri n”);          scanf(“%d”,&in)          while (i++ <=n)               printf("%d", i);          getch();    } |

**Ví dụ 2:**  Viết chương trình tính tổng các số nguyên từ 1 đến n, với n được nhập vào từ bàn phím.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | #include <studio>    #include <studio>    #include <studio>    void main(void)    {    int i = 0, in, is = 0;       printf("Nhap vao so n: ");    scanf("%d", &in);    while (i++ < in)         is = is + i; //hoac is += i;    printf("Tong: %d", is);    getch();    } |

**7.2 Lệnh do-while**

Vòng lặp thực hiện lặp lại khối lệnh cho đến khi biểu thức cho gia trị sai.

+ Cú pháp:

do

<khối lệnh>;

while (biểu thức);

**Diễn giải:**

- Biểu thức: có thể là một biểu thức hoặc nhiều biểu thức con. Nếu là nhiều biểu thức con thì cách nhau bởi dấu phẩy (,) và tính đúng sai của biểu thức được quyết định bởi biểu thức con cuối cùng.

- Trong thân do…while (<khối lệnh>) có thể chứa một hoặc nhiều cấu trúc điều khiển khác.

- Trong thân do…while có thể sử dụng lệnh continue để chuyển đến đầu vòng lặp (bỏ qua các câu lệnh còn lại trong thân).

- Muốn thoát khỏi vòng lặp do…while tùy ý có thể dùng các lệnh **break, goto, return**

**Ý nghĩa**

Thực hiện khối lệnh. Kiểm tra biểu thức, nếu **đúng**thì lặp lại thực hiện khối lệnh, nếu **sai**thì kết thúc vòng lặp.

**Ví dụ 3:**  Viết chương trình kiểm tra password.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | #include <studio>    #include <studio>    # define PASSWORD 12345    void main(void)    {    int in;    do    {        printf("Nhap vao password: ");        scanf("%d", &in);    }      while (in != PASSWORD)    } |

**7.3 Bài tập thực hành**

1. Viết chương trình tìm USCLN, BSCNN của 2 số.

2. Viết chương trình nhập vào một số và kiểm tra xem số đó có phải là số nguyên tố hay không?

3. Viết chương trình tính số hạng thứ n của dãy Fibonaci.

Dãy Fibonaci là dãy số gồm các số hạng p(n) với:

p(n) = p(n-1) + p(n-2) với n>2 và p(1) = p(2) = 1

Dãy Fibonaci sẽ là: 1 1 2 3 5 8 13 21 34 55 89 144…

4. Viết chương trình in ra bảng cửu chương

5. Viết chương trình xác định xem một tờ giấy có độ dày 0.1 mm. Phải gấp đôi tờ giấy bao nhiêu lần để nó có độ dày 1m.

6. Viết chương trình tìm các số nguyên tố từ 2 đến N, với N được nhập vào.

7. Viết chương trình tính biểu thức: 1-2+3-4+…+(2n-1)-2n

[**Tự học lập trình C - Bài 8: Chương trình con - Hàm**](http://tuhocanninhmang.com/ngon-ngu-lap-trinh/c-cpp/tu-hoc-lap-trinh-c-bai-8-chuong-trinh-con-ham.htm)

Một chương trình viết trong ngôn ngữ C là một dãy các hàm, trong đó có một hàm chính (hàm main()). Hàm chia các bài toán lớn thành các công việc nhỏ hơn, giúp thực hiện những công việc lặp lại nào đó một cách nhanh chóng mà không phải viết lại đoạn chương trình. Thứ tự các hàm trong chương trình là bất kỳ, song chương trình bao giờ cũng đi thực hiện từ hàm main(). Trong C chương trình con chỉ tồn tại dới dạng hàm chứ không có thủ tục.

Hàm có thể xem là một đơn vị độc lập của chương trình. Các hàm có vai trò ngang nhau, vì vậy không có phép xây dựng một hàm bên trong các hàm khác.

**8.1 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];]

}

Giải thích:

-     <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.

**Ví dụ 1:** Hàm tìm giá trị lớn nhất giữa hai giá trị

int tim\_max(int a, int b)

{

            if(a>=b)

                        return a;

            else

                        return b;

}

**@ Lưu ý:**

- 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.

**Ví dụ 2:**Hàm in ra dãy số từ 1 đến n

void In\_dayso(int n)

{

             for(int i=1;i<=n;i++)

                         Printf(“%d”,i);

}

Hàm này không cần có giá trị trả về nên ta khai báo từ khóa void trước tên hàm.

**8.2 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.

**Ví dụ 3**

#include <stdio.h>

#include <conio.h>

// khai bao prototype

int tim\_max(int a, int b);

void main(void)

{

int a=5, b=7;

printf(“Max là %d ”,tim\_max(a,b));

getch();

}

// ham so sanh a và b

int tim\_max(int a, int b)

{

            if(a>=b)

                        return a;

            else

                        return b;

}

**8.3 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.

**Ví dụ 4:** Hàm in ra bảng cửu chương 2

void in\_cuuchuong2(void)

{

             for(int i=1;i<=10;i++)

                         printf(“2 x %d = %d\n”, i, i\*2);

}

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.

**Ví dụ 5:**

#include <stdio.h>

#include <conio.h>

// khai bao prototype

int power(int, int);

void main(void)

{

printf("2 mu 2 = %d.\n", power(2, 2));

printf("2 mu 3 = %d.\n", power(2, 3));

getch();

}

// ham tinh so mu

int power(int ix, int in)

{

int i, ip = 1;

for(i = 1; i <= in; i++)

ip \*= ix; //tương đương với ip=ip\*ix

return ip; //giá trị trả về cho hàm

}

Giải thích chương trình:

Hàm **power**có hai tham số truyền vào là ix, in có kiểu int và kiểu trả về cũng có kiểu int.

Dòng lệnh: return ip, trả về giá trị sau khi tính toán

Hai tham số ix, in của hàm power là dạng truyền tham trị.

**Ví dụ 6:**

#include <stdio.h>

#include <conio.h>

// khai bao prototype

void time(int & , int &);//co the k can ghi tham so hinh thuc

// ham doi phut thanh gio:phut

void time(int &ig, int &ip)

{

ig = ip/60;

ip %= 60;

}

void main(void)

{

int igio, iphut;

printf("Nhap vao so phut: ");

scanf("%d", &iphut);

time(igio, iphut);

printf("%02d:%02d\n", igio, iphut);

getch();

}

Giải thích chương trình:

Hàm **time**có hai tham số hình thức là ig, ip có kiểu int. 2 tham số này có toán tử địa chỉ & đi trước cho biết 2 tham số này là dạng truyền tham biến.

**8.4 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.

**8.5 Bài tập thực hành**:

Viết lại các bài tập ở bài số 6 & 7 dưới dạng hàm.

[**Tự học lập trình C - Bài 9: Đệ qui**](http://tuhocanninhmang.com/ngon-ngu-lap-trinh/c-cpp/tu-hoc-lap-trinh-c-bai-9-de-qui.htm)

**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

**Bài tập thực hành**

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

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

[**Tự học lập trình C - Bài 10: Mảng một chiều**](http://tuhocanninhmang.com/ngon-ngu-lap-trinh/c-cpp/tu-hoc-lap-trinh-c-bai-10-mang-mot-chieu.htm)

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.

**Ví dụ 1**: bạn muốn tính trung bình 10 số nguyên nhập vào từ bàn phím, bạn sẽ khai báo 10 biến: a, b, c, d, e, f, g, h, i, j có kiểu int và lập thao tác nhập cho 10 biến này như sau:

printf("Nhap vao bien a: ");

scanf("%d", &a);

10 biến bạn sẽ thực hiện 2 lệnh trên 10 lần, sau đó tính trung bình:

(a + b + c + d + e + f + g + h + i + j)/10

Điều này chỉ phù hợp với n nhỏ, còn đối với n lớn thì khó có thể thực hiện được. Vì vậy, khái niệm mảng được sử dụng

**10.1 Khai báo**

**Ví dụ 2**: **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

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

**Tên\_kiểu tên\_biến[spt];**

**Diến giải:**

- 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

**10.2 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

**10.3 Nhập dữ liệu cho mảng**

Mảng là một dãy các phần tử có cùng kiểu dữ liệu, việc nhập dữ liệu cho từng phần tử của mảng cũng giống như nhập dữ liệu cho biến thông thường

Ví dụ 3

for (i = 0; i < 10; i++) //vòng for có giá trị i chạy từ 0 đến 9

{

printf("Nhap vao phan tu thu %d: ", i + 1);

scanf("%d", &ia[i]);

}

Ví dụ 4: Viết chương trình nhập vào n số nguyên. Tính và in ra trung bình cộng.

#include <stdio.h>

#include <conio.h>

void main(void)

{

int ia[50], i, in, isum = 0;

printf("Nhap vao gia tri n: ");

scanf("%d", &in);

//Nhap du lieu vao mang

for(i = 0; i < in; i++)

{

printf("Nhap vao phan tu thu %d: ", i + 1);

scanf("%d", &ia[i]); //Nhap gia tri cho phan tu thu i

}

//Tinh tong gia tri cac phan tu

for(i = 0; i < in; i++)

       isum += ia[i]; //cong don tung phan tu vao isum

printf("Trung binh cong: %.2f\n", (float) isum/in);

getch();

}

Ví dụ 5: Có 4 loại tiền 1, 5, 10, 25 và 50 đồng. Hãy viết chương trình nhập vào số tiền sau đó cho biết số số tiền trên gồm mấy loại tiền, mỗi loại bao nhiêu tờ

#include <stdio.h>

#include <conio.h>

#define MAX 5

void main(void)

{

int itien[MAX] = {50, 25, 10, 5, 1}; //Khai bao va khoi tao mang voi 5 phan tu

int i , isotien, ito;

printf("Nhap vao so tien: ");

scanf("%d", &isotien); //Nhap vao so tien

for (i = 0; i < MAX; i++)

{

ito = isotien/itien[i]; //Tim so to cua loai tien thu i

printf("%4d to %2d dong\n", ito, itien[i]);

isotien = isotien%itien[i]; //So tien con lai sau khi da loai tru cac loai tien da co

}

getch();

}

**10.4 Đọc dữ liệu từ mảng**

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

Ví dụ 5:

for(i = 0; i < 10; i++)

printf("%3d ", ia[i]);

**10.5 Sử dụng biến mảng**

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

Ví dụ 6: char cloai[20]; float ftemp[10];

Cách tham chiếu, nhập dữ liệu, đọc dữ liệu thực hiện như trên.

**Bài tập thực hành**.

Xây dựng các hàm để thực hiện các công việc sau

- Hàm Khoi\_tao cho phép nhập vào một mảng gồm n phần tử số nguyên

- Hàm In\_xuôi và hàm In\_nguoc cho phép in mảng ra màn hình theo thứ tự xuôi và ngược

- Hàm Tim\_max và hàm Tim\_min để tìm giá trị nhỏ nhất và lớn nhất của mảng

- Hàm Tinh\_tong cho phép Tính tổng các giá trị trong mảng

- Hàm Tinh\_tong\_duong và hàm Tinh\_tong\_am cho phép tính tổng các phần tử có giá trị dương và âm trong mảng

- Hàm Sap\_xep\_tang và hàm Sap\_xep\_giam để sắp xếp mảng theo thứ tự tăng dần và giảm dần

- Xây dựng chương trình chính gồm 1 menu như sau:

***----------------------------------------------------***

***MENU CHUONG TRINH***

***---------------------------------------------------***

*1: Khoi tao mang*

*2: In mang ra man hinh*

*3: In mang dao nguoc ra man hinh*

*4: Tim gia tri max cua mang*

*5: Tim gia tri min cua mang*

*6: Tinh tong*

*7: Tinh tong cac phan tu duong*

*8: Tinh tong cac phan tu am*

*9: Sap xep tang dan*

*10 Sap xep giam dan*

*11: Thoat*

*Moi ban chon so tuong ung:*

Khi người sử dụng chọn 1 số tương ứng của mục tương ứng của menu thì gọi đến hàm tương ứng đó để thực hiện

Gợi ý:

- Xây dựng tất cả các hàm tương ứng với yêu cầu của đề bài

- Xây dựng thêm hàm có tên gọi là inmenu để in ra menu như yêu cầu

- Viết hàm main, đầu tiên là gọi hàm inmenu, tiếp đến dùng lệnh switch theo mẫu

switch (ichon){

            case 1: gọi hàm khoi tạo mảng

                        break;

            case 2: gọi hàm in mảng ra màn hình

                        break;

            …

case 10: gọi hàm sắp xếp giảm dần

}

Lưu ý, chương trình chỉ đóng lại khi người sử dụng chọn số 11 (dùng vòng lặp bao bên ngoài lệnh switch, điều kiện dừng khi người sử dụng nhấn số 11).

[**Tự học lập trình C - Bài 11: Mảng nhiều chiều**](http://tuhocanninhmang.com/ngon-ngu-lap-trinh/c-cpp/tu-hoc-lap-trinh-c-bai-11-mang-nhieu-chieu.htm)

**Khai báo mảng:**

Ví dụ 1:

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.

**Ý nghĩa**: 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:

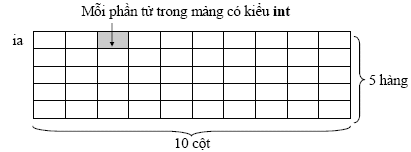
tên\_kiểu tên\_biến[spt1][spt2]…[sptn];

**Diễn giải**:

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

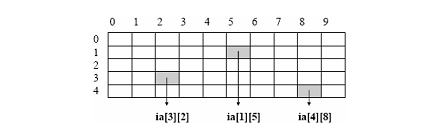
- tên\_biến: Tên của biến mảng, tên được đặt theo qui tắc đặt tên của C

- 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]



**Nhập dữ liệu cho mảng 2 chiều**

Mảng 2 chiều giống như một ma trận hai chiều, có kích thước bằng số dòng nhân số cột, để nhập dữ liệu cho các phần tử trong ma trận bạn cần duyệt qua từng phần tử trên từng hàng trong ma trận và đưa dữ liệu vào cho phần tử tương ứng như một biến thông thường

Ví dụ 2:

for (int i = 0; i < 5; i++) //vòng for có giá trị i chạy từ 0 đến 4 cho dòng

//tương ứng với mỗi dòng sẽ duyệt qua từng phần tử trong mỗi dòng tương ứng

for (int j = 0; j < 10; j++) //vòng for có giá trị j chạy từ 0 đến 9 cho cột

{

printf("Nhap vao phan tu ia[%d][%d]: ", i + 1, j + 1);

scanf("%d", &ia[i][j]);

}

@Lưu ý:Phép lấy địa chỉ đối với các phần tử mảng hai chiều chỉ có thể áp dụng khi các phần tử mảng hai chiều có kiểu nguyên, còn lại thì phép lấy địa chỉ cho các phần tử mảng nhiều chiều là không thực hiện được.

Chúng ta sửa lại  ví dụ trên để có thể nhập được kiểu dữ liệu khác cho mảng hai chiều bằng cách dừng biến trung gian như sau:

float  fa[5][10], tg;

for (int i = 0; i < 5; i++)

for (int j = 0; j < 10; j++)

{

printf("Nhap vao phan tu ia[%d][%d]: ", i + 1, j + 1);

scanf("%f", &tg);

fa[i][j]=tg;

}

**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

Ví dụ 3: in giá trị các phần tử mảng 2 chiều ra màn hình.

for (int i = 0; i < 5; i++) //vòng for có giá trị i chạy từ 0 đến 4 cho dòng

{

for (int j = 0; j < 10; j++) //vòng for có giá trị j chạy từ 0 đến 9 cho cột

printf("%3d ", ia[i][j]);

printf("\n"); //xuống dòng để in hàng kế tiếp

}

Ví dụ 4: Viết chương trình nhập vào 1 ma trận số nguyên n x n. In ra ma trận vừa nhập vào và in theo chiều ngược lại của mỗi dòng .

#include <stdio.h>

#include <conio.h>

#define MAX 50;

void main(void)

{

int ia[MAX][MAX], i, j, n;

printf("Nhap vao cap ma tran: ");

scanf("%d", &n);

//Nhap du lieu vao ma tran

for (i = 0; i < n; i++)

for (j = 0; j < n; j++)

{

printf("Nhap vao phan tu ia[%d][%d]: ", i + 1, j + 1);

scanf("%d", &ia[i][j]);

}

//In ma tran

for (i = 0; i < n; i++)

{

for (j = 0; j < n; j++)

printf("%3d ", ia[i][j]);

printf("\n"); //xuống dòng để in hàng kế tiếp

}

printf("\n"); //Tao khoang cach giua 2 ma tran

//In ma tran theo thu tu nguoc

for (i = 0; i< n; i++)

{

for (j = n-1; j >= 0 j--)

printf("%3d ", ia[i][j]);

printf("\n"); //xuống dòng để in hàng kế tiếp

}

getch();}

**Bài tập thực hành**

1. Viết hàm cho phép nhập vào một ma trận vuông kích thước NxN

2. Viết hàm tính tổng các phần tử trên đường chéo chính

3. Viết hàm tính tổng các phần tử trên đường chéo phụ

4. Viết hàm in ra tổng của từng dòng trong ma trận

5. Viết hàm in ra tổng của từng cột trong ma trận

6. Viết hàm kiểm tra ma trận có phải là ma trận đơn vị không

7. Viết hàm kiểm tra ma trận có phải là ma trận chéo không

8. Viết hàm kiểm tra ma trận có phải là ma trận tam giác trên không

9. Xây dựng menu và hàm main để gọi thực hiện các câu 1 đến câu 8