28TECH
Become A Better Developer

PROGRAM IN C++

Khuôn mẫu chung của một chương trình

INPUT → 

→ OUTPUT

Chương trình C++ đầu tiên

```
#include <iostream>

using namespace std;

int main () {
    cout << "Hello world !" << endl;
    return 0;
}
```

Khai báo thư viện cần thiết

Thêm namespace

Hàm main: là nơi bắt đầu thực thi của chương trình

Những phần chính của một chương trình C++:

/01 Thư viện mà chương trình sử dụng.

/02 Namespace.

/03 Chương trình chính với mã nguồn.

Những chú ý khi viết chương trình C++

▶▶▶ Các câu lệnh kết thúc bằng dấu " ; "

Không hợp lệ

cout << “Hello 28tech” <<

Hợp lệ

cout << “Hello 28tech” << ;

Những chú ý khi viết chương trình C++

- Các câu lệnh kết thúc bằng dấu " ; "
- Luôn thụt lề các câu lệnh so với hàm main

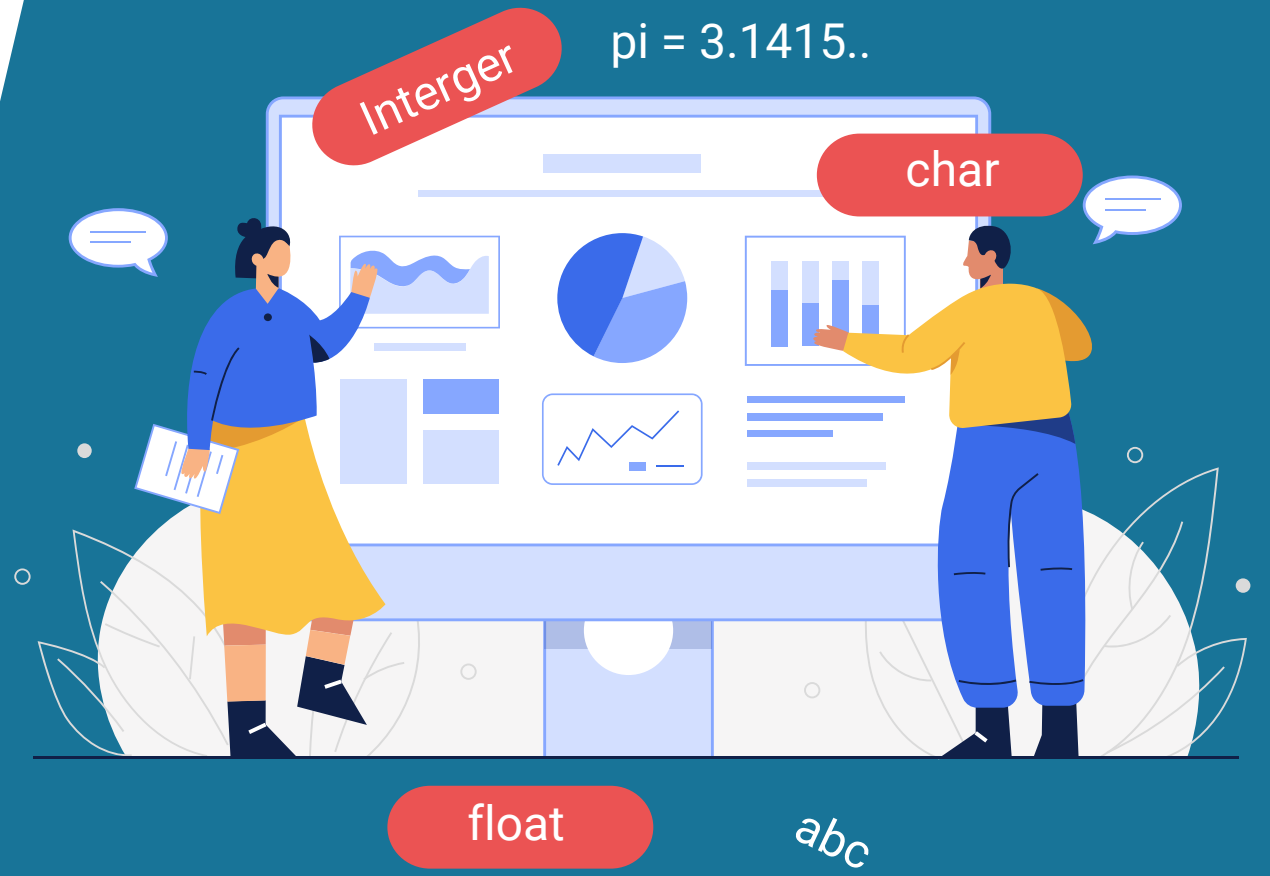
Không chấp nhận

```
int main () {
cout << "Hello world !" << endl;
return 0;
}
```

Chấp nhận

```
int main () {
    cout << "Hello world !" << endl;
    return 0;
}
```

KIỂU DỮ LIỆU (DATA TYPE)



KIỂU DỮ LIỆU SỐ NGUYÊN

1 byte = 8 bit

Đối với số nguyên ta chia làm số nguyên không dấu và số nguyên có dấu, từ số byte lưu trữ ta có thể suy ra số bit cần để biểu diễn số nguyên đó, quy tắc xác định giá trị của 1 số nguyên.

Giả sử số nguyên có K bit

-Số nguyên có dấu : -2^{K-1} tới $2^{K-1} - 1$

-Số nguyên không dấu : 0 tới $2^{K} - 1$

KIỂU DỮ LIỆU SỐ NGUYÊN

Kiểu dữ liệu	Ý nghĩa	Số Byte	Giải giá trị có thể lưu
short	Số nguyên có dấu	2 byte	-32768 đến 32767
unsigned short	Số nguyên không dấu	2 byte	0 đến 65535
int	Số nguyên có dấu	4 byte	-2147483648 đến 2147483647
unsigned int	Số nguyên không dấu	4 byte	0 đến 4294967295
long long	Số nguyên	8 byte	-9223372036854775808 đến 9223372036854775807
unsigned long long	Số nguyên không dấu	8 byte	0 đến 2^64 - 1

KIỂU DỮ LIỆU SỐ THỰC

Kiểu dữ liệu	Ý nghĩa	Số Byte	Giải giá trị có thể lưu
float	Số thực độ chính xác đơn	4 byte	1.17549e-038 đến 3.40282e+038 Độ chính xác : 7 chữ số sau dấu phẩy
double	Số thực độ chính xác kép	8 byte	2.22507e-308 đến 1.79769e+308 Độ chính xác : 15 chữ số sau dấu phẩy

Ưu tiên sử dụng double vì double có độ chính xác cao hơn float

KIỂU DỮ LIỆU ĐÚNG SAI

Kiểu dữ liệu	Ý nghĩa	Số Byte	Giải giá trị có thể lưu
bool	Kiểu dữ liệu luận lý	1 byte	true hoặc false true : Đúng false : Sai

KIỂU DỮ LIỆU KÝ TỰ

Kiểu dữ liệu	Ý nghĩa	Số Byte	Giải giá trị có thể lưu
char	Kiểu dữ liệu ký tự	1 byte	-128 tới 127

Tổng quan về kiểu dữ liệu:

/01 Kiểu số nguyên có int và long long.

>> Sử dụng long long cho các kết quả là số lớn

/02 Kiểu số thực có float và double.

>> Sử dụng double vì có độ chính xác cao

/03 Kiểu đúng sai có bool.

/04 Kiểu ký tự có char.

BIẾN (VARIABLE)

BIẾN (VARIABLE)

Khái niệm

- Biến được sử dụng để lưu các giá trị trong quá trình tính toán của chương trình. Tùy theo kiểu dữ liệu của biến, một ô trong bộ nhớ sẽ được cấp phát để lưu trữ giá trị của biến này.

Cú pháp

- [Kiểu dữ liệu] [Tên biến];

Ví dụ: int x; long long b; char ki_tu; bool check; double dienTich; ...

BIẾN (VARIABLE)

Quy tắc đặt tên biến:	Ví dụ cách đặt sai:
01 Không đặt tên biến bắt đầu bằng chữ số	1dientich, 2chuvi, 222bankinh, 9a,...
02 Tên biến không được chứa dấu cách và các kí tự đặc biệt	ban kinh, dien#tich, chu@vi,...
03 Tên biến không được trùng với tên từ khóa trong C++	int, main, for, while, ...
04 Tên biến trong C++ là phân biệt hoa thường	banKinh và BanKinh là 2 biến khác nhau
05 Không được đặt 2 biến có cùng tên trong cùng một phạm vi	int a; float a;

CHÚ THÍCH TRONG C++

//

/* */

CHÚ THÍCH

Khái niệm

Chú thích (Comment) là một giải pháp bổ sung thông tin vào code của bạn, nhằm làm rõ nội dung, giải thích câu lệnh, mục đích của code...

Giúp người đọc code có thể dễ nắm bắt nội dung code và thuận lợi trong việc bảo trì code

Các chú thích sẽ không được coi là câu lệnh và sẽ được loại bỏ khi chương trình thực thi

CHÚ THÍCH

Cách chú thích

Để chú thích trên 1 dòng ta dùng //

Ví dụ :

```
//Đây là chú thích
//Chú thích giúp code rõ ràng hơn
```

Để chú thích trên nhiều dòng ta dùng /* ND cần chú thích */

```
/*
Đây là
chú thích trên nhiều dòng
*/
```

MỘT SỐ VÍ DỤ

Khai báo biến

```
#include <iostream>
using namespace std;

int main(){
    int a;   // biến a có kiểu int
    int x, y, z; // khai báo 3 biến cùng kiểu int
    long long dien_tich;  // biến dien_tich có kiểu ll
    bool check;  // biến check kiểu đúng sai
    char kitu;  // biến kitu kiểu char
}
```

MỘT SỐ VÍ DỤ

Hiển thị giá trị của biến ra màn hình

Cú pháp: cout << var ;

Hiển thị giá trị của biến số thực var với độ chính xác k số sau dấu phẩy:

Cú pháp:

cout << fixed << setprecision(k) << var;

Phải khai báo thư viện iomanip !

```
#include <iostream>
#include <iomanip>
using namespace std;

int main(){
    int a = 28;   // biến a có kiểu int và có giá trị là 28
    cout << a << endl;  //xuat biến a ra màn hình
    int x = 100, int y = 200; //khởi tạo nhiều biến
    cout << x << ' ' << y << ' ' << endl; //xuất cùng lúc
    double res = 3.14;
    cout << fixed << setprecision(5) << res << endl;
   //output: 3.14000
}
```

MỘT SỐ VÍ DỤ

Nhập giá trị cho biến từ bàn phím

Cú pháp: cin >> var ;

Chú ý : Khi nhập xong giá trị các bạn ấn enter để cin bắt đầu nhập giá trị cho biến, trong trường hợp bạn nhập giá trị cho nhiều số, có thể nhập từng số rồi enter hoặc nhập 1 lúc nhiều số rồi ấn enter 1 lần.

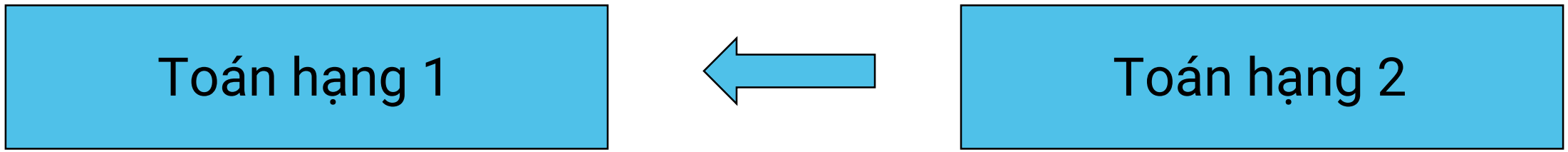
```
#include <iostream>
using namespace std;

int main(){
    int a;   // biến a có kiểu int
    cin >> a; //nhập giá trị cho a
    cout << a << endl; //xuất ra giá trị vừa nhập
    int x, y, z; // khai báo 3 biến cùng kiểu int
    cin >> x >> y >> z;
    cout << x << " " << y << " " << z << " " << endl;
}
```


TOÁN TỬ (OPERATOR)

TOÁN TỬ GÁN: (ASSIGNMENT OPERATOR)

Cú pháp: [Toán hạng 1] = [Toán hạng 2]



Ý nghĩa:Gán giá trị của toán hạng 2 cho toán hạng 1.

VÍ DỤ:

```
ban_kinh = 100;   // Gán giá trị 100 cho biến ban_kinh
Chu_vi = 1000;   // Gán giá trị của chuvi là 1000
```

TOÁN TỬ TOÁN HỌC: (ARITHMETIC OPERATOR)

Toán tử	Ý nghĩa	Ví dụ
+	Cộng 2 toán hạng	X = Y + Z X = 100 + 200; // 300
-	Trừ 2 toán hạng	X = Y - Z X = 200 - 100; // 100
*	Nhân 2 toán hạng	X = Y * Z X = 10 * 50; // 500
/	Chia 2 toán hạng	X = Y / Z X = 300 / 200; // 1
%	Chia dư 2 toán hạng	X = Y % Z X = 300 % 200; // 100

TOÁN TỬ TOÁN HỌC: (ARITHMETIC OPERATOR)

Chú ý 1: Nếu bạn chia 2 số nguyên (int, long long) cho nhau thì phép chia ở trên sẽ là phép chia nguyên, tức là nó chỉ lấy phần nguyên và bỏ phần thập phân ở thương

- Nếu muốn kết quả ở số thập phân thì ít nhất 1 trong 2 số phải ở kiểu số thực và biến thương phải ở dạng số thực

+) Ép kiểu

+) Nhân thêm 1 số thực khi tính toán

```
int a = 100, b = 30;
float thuong = (float) a / b;
cout << fixed << setprecision(2) <<
thuong << endl; //3.33
```

TOÁN TỬ TOÁN HỌC: (ARITHMETIC OPERATOR)

Chú ý 1: Nếu bạn chia 2 số nguyên (int, long long) cho nhau thì phép chia ở trên sẽ là phép chia nguyên, tức là nó chỉ lấy phần nguyên và bỏ phần thập phân ở thương

- Nếu muốn kết quả ở số thập phân thì ít nhất 1 trong 2 số phải ở kiểu số thực và biến thương phải ở dạng số thực

+) Ép kiểu

+) Nhân thêm 1 số thực khi tính toán

```
int a = 100, b = 30;
float thuong = 1.0 * a / b;
cout << fixed << setprecision(2) <<
thuong << endl; //3.33
```

TOÁN TỬ TOÁN HỌC: (ARITHMETIC OPERATOR)

Chú ý 2: Nếu bạn nhân 2 số nguyên int với nhau mà kết quả của tích vượt giới hạn lưu của số int thì kết quả sẽ bị tràn, ngay cả khi bạn sử dụng biến long long để lưu biến tích. Việc cần xử lý ở đây là can thiệp vào phép nhân

Cách xử lí sai

```
int a = 1000000, b = 1000000;
long long tich = a * b; //sai
long long tich = (long long) (a  * b);
 // vẫn sai
cout << tich << endl; //tràn số
```

Cách xử đúng

```
int a = 1000000, b = 1000000;
// ép 1 trong 2 số thành ll
long long tich = (long long) a  * b;
long long tich = 1ll * a * b; //1ll ?
cout << tich << endl; //10^12
```

TOÁN TỬ SO SÁNH: (COMPARISION OPERATOR)

Khi bạn sử dụng các toán tử so sánh để so sánh 2 toán hạng thì kết quả của phép so sánh sẽ trả về đúng hoặc sai.

Toán tử	Ý nghĩa	Ví dụ
>	Lớn hơn	10 > 50 : false
>=	Lớn hơn hoặc bằng	20>=10 : true
<	Nhỏ hơn	10 < 50 : true
<=	Nhỏ hơn hoặc bằng	20 <= 20 : true
!=	So sánh khác	10 != 20 : true
==	So sánh bằng	10 == 10 : true

TOÁN TỬ SO SÁNH: (COMPARISION OPERATOR)

VÍ DỤ:

Ví dụ	Kết quả
10 == 20	Sai
10 <= 20	Đúng
10 == 10	Đúng
50 != 50	Sai
100 <= 100	Đúng

TOÁN TỬ LOGIC: (LOGICAL OPERATOR)

Trong trường hợp bạn muốn kết hợp nhiều phép so sánh lại với nhau, ta sử dụng 3 cổng logic cơ bản của máy tính là AND, OR, NOT

Toán tử	Ký hiệu	Ví dụ
AND	&&	(x >= 10) && (x <= 50)
OR	||	(a >= 10) || (a % 2 == 0)
NOT	!	!(a <= 10)

Để tính toán giá trị cuối cùng của biểu thức các bạn xác định giá trị của từng mệnh đề thành phần, sau đó áp dụng bảng chân lý của các cổng logic để tìm giá trị của biểu thức ban đầu.

TOÁN TỬ LOGIC: (LOGICAL OPERATOR)

Bảng chân lý của cổng AND

A	B	A && B
0	0	0
0	1	0
1	0	0
1	1	1

Chú ý: Cổng AND chỉ cho kết quả đúng khi mọi mệnh đề thành phần đều có giá trị đúng, sai trong các trường hợp còn lại.

TOÁN TỬ LOGIC: (LOGICAL OPERATOR)

Bảng chân lý của cổng OR

A	B	A || B
0	0	0
0	1	1
1	0	1
1	1	1

Chú ý: Cổng OR chỉ cho kết quả sai khi mọi mệnh đề thành phần đều có giá trị sai, đúng khi chỉ cần ít nhất 1 mệnh đề thành phần có giá trị đúng

TOÁN TỬ LOGIC: (LOGICAL OPERATOR)

Bảng chân lý của cổng NOT

A	NOT A
0	1
1	0

TOÁN TỬ TĂNG GIẢM: (INCREMENT, DECREMENT OPERATOR)

Để tăng hoặc giảm giá trị của một biến đi 1 đơn vị ta có thể sử dụng toán tử tăng, giảm này sẽ thuật tiện hơn.

Toán tử	Ý nghĩa	Ví dụ
++	Tăng trước 1 đơn vị	++a
++	Tăng sau 1 đơn vị	a++
--	Giảm trước 1 đơn vị	--a
--	Giảm sau 1 đơn vị	a--

TOÁN TỬ TĂNG GIẢM: (INCREMENT, DECREMENT OPERATOR)

VÍ DỤ:

Ví dụ	Kết quả	Giải thích

int a = 100;
int b = a++;
cout << a << " " << b << endl;
	101 100	Đây là tăng sau, tức ban đầu câu lệnh sẽ gán giá trị của a là 100 cho b, sau đó mới tăng giá trị của a lên 101

int a = 100;
int b = ++a;
cout << a << " " << b << endl;
	101 101	Đây là tăng trước, giá trị của a sẽ được tăng ngay lập tức lên 101, sau đó mới lấy giá trị đó và gán cho b

TOÁN TỬ 3 NGÔI: (CONDITIONAL OPERATOR)

Cú pháp

▶▶▶[Biểu thức so sánh] ? [Giá trị trả về khi biểu thức đúng] : [Giá trị trả về khi biểu thức sai]

Ví dụ	Kết quả	Giải thích
int x = 10 < 20 ? 10 : 20;	x = 20	Nếu 10 < 20 thì vế phải sẽ trả về 10, ngược lại sẽ trả về 20. Sau đó giá trị này được gán cho x
int y = (10 < 20) && (20 > 20) ? 5 : 10;	y = 10	Nếu 10 < 20 và 20 > 20 thì sẽ gán 5 cho y, ngược lại gán 10 cho y

CHÚ Ý Ở PHẦN TOÁN TỬ

Các toán tử sẽ có thứ tự ưu tiên nhất định, ví dụ như nhân chia trước, cộng trừ sau hoặc cùng mức độ ưu tiên thì thực thi từ trái qua phải

Nhưng dấu đóng mở ngoặc tròn luôn có độ ưu tiên cao nhất, vì thế khi viết biểu thức thì các bạn nên sử dụng dấu ngoặc để biểu thức được thực thi theo đúng mong muốn của mình.

CẤU TRÚC RẺ NHÁNH

IF...ELSE

SWITCH CASE

1. CÂU LỆNH IF

- Câu lệnh if được sử dụng trong trường hợp bạn muốn chương trình của mình thực hiện 1 hoặc 1 nhóm câu lệnh khi một điều kiện nào đó thỏa mãn.
- VD: Nếu chỉ số máu của nhân vật bằng 0 thì nhân vật sẽ chết, vậy điều kiện ở đây là “chỉ số máu của nhân vật bằng 0”, và hành động được thực hiện ở đây là sẽ có một câu lệnh thực thi làm nhân vật bị chết.

SYNTAX

```
if (condition) {
    // code
}
```

1. CÂU LỆNH IF

VÍ DỤ:

Ví dụ	Kết quả	Ý nghĩa

int a = 100, b = 200;
if (a < b){
cout << “OK”;
}
	OK	Vì a < b có giá trị là đúng nên câu lệnh bên trong if được thực thi

int a = 100, b = 200;
if ((a % 2) == 0){
cout << “Even”;
}
	Even	Nếu a chia dư cho 2 bằng 0, thì a là số chẵn

Chú ý: Các bạn nhớ thụt lề các câu lệnh bên trong if so với if nhé, trong trường hợp bên trong if chỉ có 1 câu lệnh, các bạn có thể bỏ dấu đóng mở ngoặc nhọn.

2. CÂU LỆNH IF ELSE

- If được sử dụng khi bạn muốn thực thi code với điều kiện nào đó đúng, trong trường hợp điều kiện đó sai bạn muốn thực thi một đoạn code khác thì cấu trúc if else sẽ được sử dụng

SYNTAX

```
if (condition) {
    // code if
}
else {
    // code else
}
```

2. CÂU LỆNH IF ELSE

VÍ DỤ:

Ví dụ	Kết quả	Ý nghĩa

int a = 100;
if((a % 2) == 0){
  cout << “Chan”;
}
else{
  cout << “Le”;
}
	Chan	Nếu a chia 2 dư 0, tức điều kiện trong if đúng thì câu lệnh in ra “Chan”, ngược lại nếu điều kiện đó sai thì in ra “Le”

3. CÂU LỆNH IF VÀ ELSE IF

- Nếu bạn muốn kiểm tra nhiều điều kiện khác nhau thì sử dụng cấu trúc else if sẽ hiệu quả hơn so với sử dụng nhiều câu lệnh if else lồng nhau.

SYNTAX

```
if (condition1) {
    // code
}
else if (condition2){
    // code
}
....
else if(conditionN){
   //code
}
else{
   // code
}
```

3. CÂU LỆNH IF VÀ ELSE IF

Chú ý: Nếu một điều kiện nào trong N điều kiện trong cấu trúc trên đúng và câu lệnh bên trong nhánh đó được thực hiện thì khối lệnh if else if này sẽ kết thúc ngay lập tức. Ví dụ, nếu condition2 đúng thì khối lệnh bên trong nhánh đó được thực thi, sau đó cấu trúc này sẽ kết thúc ngay mà không kiểm tra các điều kiện còn lại cũng như trong else.

Good	Bad

int day; cin >> day;
if(day == 1) cout << “Chu nhat” << endl;
else if(day == 2) cout << “Thu hai” << endl;
else if(day == 3) cout << “Thu ba” << endl;
else if(day == 4) cout << “Thu tu” << endl;
else if(day == 5) cout << “Thu nam” << endl;
else if(day == 6) cout << “Thu sau” << endl;
else if(day == 7) cout << “Thu bay” << endl;
	int day; cin >> day;
	if(day == 1) cout << “Chu nhat” << endl;
	if(day == 2) cout << “Thu hai” << endl;
	if(day == 3) cout << “Thu ba” << endl;
	if(day == 4) cout << “Thu tu” << endl;
	if(day == 5) cout << “Thu nam” << endl;
	if(day == 6) cout << “Thu sau” << endl;
	if(day == 7) cout << “Thu bay” << endl;

4. CÂU LỆNH SWITCH CASE

- Switch case cũng giúp bạn kiểm tra nhiều điều kiện khác nhau.
- Ý nghĩa : Giá trị của val sẽ được so sánh lần lượt với các giá trị trong các case, nếu giá trị của val bằng giá trị tại 1 case nào đó thì câu lệnh bên trong case đó được thực thi. Nếu val không giống bất cư một giá trị trong các case nào thì câu lệnh bên trong default được thực thi.
- Chú ý : Giá trị của val có thể là số, kí tự, xâu kí tự (sẽ học sau) .Các khối lệnh bên trong các case sẽ được kết thúc bằng câu lệnh break.

SYNTAX

```
switch (val) {
        case 1:
                // code
                break;
        case 2:
                //code
                break;
        .....
        case n:
                // code
                break;
        default :
                // code
}
```

BẢNG MÃ ASCII VÀ CÁC LỆNH LIÊN QUAN

BẢNG MÃ ASCII

BẢNG MÃ ASCII

Bảng mã này có 256 kí tự, mỗi kí tự được gán với 1 mã nhất định gọi là mã ASCII.

Bạn có thể coi kiểu dữ liệu char như số hoặc kí tự đều được, tức là bạn hoàn toán có thể sử dụng nó để cộng, trừ, nhân, chia

Chú ý: Hãy luôn nhớ khi cộng, trừ, nhân, chia một kí tự nào đó thì mã ASCII của nó sẽ được sử dụng.

Một vài dải kí tự cần lưu ý:

Dải kí tự	Dải mã ASCII
A-Z	65-90
a-z	97-122
0-9	48-57

Một vài câu lệnh kiểm tra kiểu ký tự cần nắm vững:

Câu lệnh	Ý nghĩa
char c; if ((c >= ‘a’) && (c <= ‘z’))	Kiểm tra kí tự in thường
char c; if ((c >= 97) && (c <= 122))	Kiểm tra kí tự in thường
char c; if ((c >= ‘A’) && (c <= ‘Z’))	Kiểm tra kí tự in hoa
char c; if ((c >= 65) && (c <= 90))	Kiểm tra kí tự in hoa
char c; if ((c >= ‘0’) && (c <= ‘9’))	Kiểm tra kí tự là chữ số
char c; if ((c >= 48) && (c <= 57))	Kiểm tra kí tự là chữ số
char c = ‘A’; c += 32;	Chuyển kí tự c thành dạng in hoa tương ứng
char c = ‘a’; c -=32;	Chuyển kí tự c thành dạng in thường tương ứng

KẾT THÚC BUỔI HỌC