



28TECH

Become A Better Developer



Kiểu dữ liệu & Biến

Toán tử

Cấu trúc rẽ nhánh



1. Kiểu dữ liệu và biến:

a. Kiểu dữ liệu (Data type):



Ngôn ngữ lập trình Java cung cấp các kiểu dữ liệu để biểu diễn dữ liệu kiểu số nguyên, số thực, kí tự, đúng sai, xâu ký tự...

Kiểu dữ liệu	Kích thước (byte)	Giá trị
byte	1 byte	-128 đến 127
short	2 byte	-32768 đến 32767
int	4 byte	-2^{31} đến $2^{31} - 1$
long	8 byte	-2^{63} đến $2^{63} - 1$
float	4 byte	Lưu số thực độ chính xác từ 6 tới 7 chữ số sau dấu phẩy
double	8 byte	Lưu số thực độ chính xác tới 15 chữ số sau dấu phẩy
char	2 byte	Lưu các kí tự
boolean	1 byte	Lưu giá trị true (đúng) hoặc false(sai)



1. Kiểu dữ liệu và biến:

b. Biến (Variable):



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 KHAI BÁO BIẾN VÀ VÍ DỤ:

Cú pháp

[Kiểu dữ liệu] [Tên biến] ;

EXAMPLE

```
int a;  
long b;  
float chuVi;  
double dienTich;  
boolean check;  
char kiTu;
```



1. Kiểu dữ liệu và biến:

b. Biến (Variable):

Quy tắc đặt tên biến:

Ví dụ cách đặt sai:

1. Không được đặt tên biến bắt đầu bằng chữ số.

1dientich, 2chuvi, 222bankinh,...

2. Tên biến không được chứa dấu cách và các kí tự đặc biệt.

ban kinh, dien#tich, chu@vi,...

3. Tên biến không được trùng với tên từ khóa trong Java.

int, main, for, while, ...

4. Tên biến trong Java là phân biệt hoa thường.

banKinh và BanKinh



1. Kiểu dữ liệu và biến:

b. Biến (Variable):

NAME CONVENTION



Trong Java khi đặt tên biến bạn cần phải tuân theo quy chuẩn đặt tên chung.



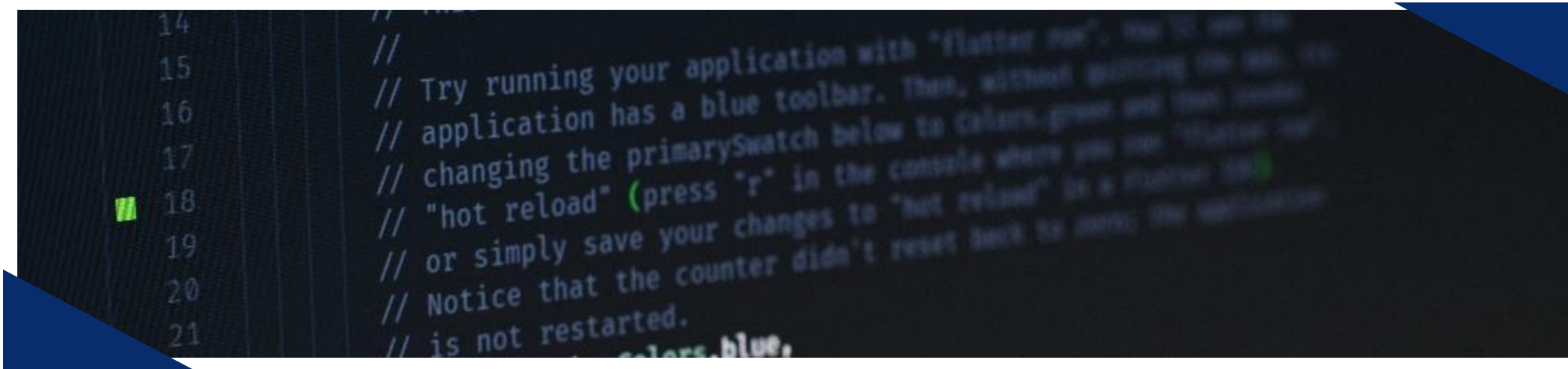
Tên biến được chia thành các từ, ngoại trừ từ đầu tiên thì các từ tiếp theo bạn viết hoa chữ cái đầu và viết thường chữ cái sau.

Ví dụ: banKinh, dienTich, luongNhanVien, diemTrungBinh,...





2. Chú thích trong Java:



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.





2. Chú thích trong Java:

Chú thích trên một dòng:

Để chú thích trên 1 dòng ta dùng //

EXAMPLE

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



Chú thích trên nhiều dòng:

Để chú thích trên nhiều dòng ta đặt nội dung cần chú thích giữa /* và */

EXAMPLE

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



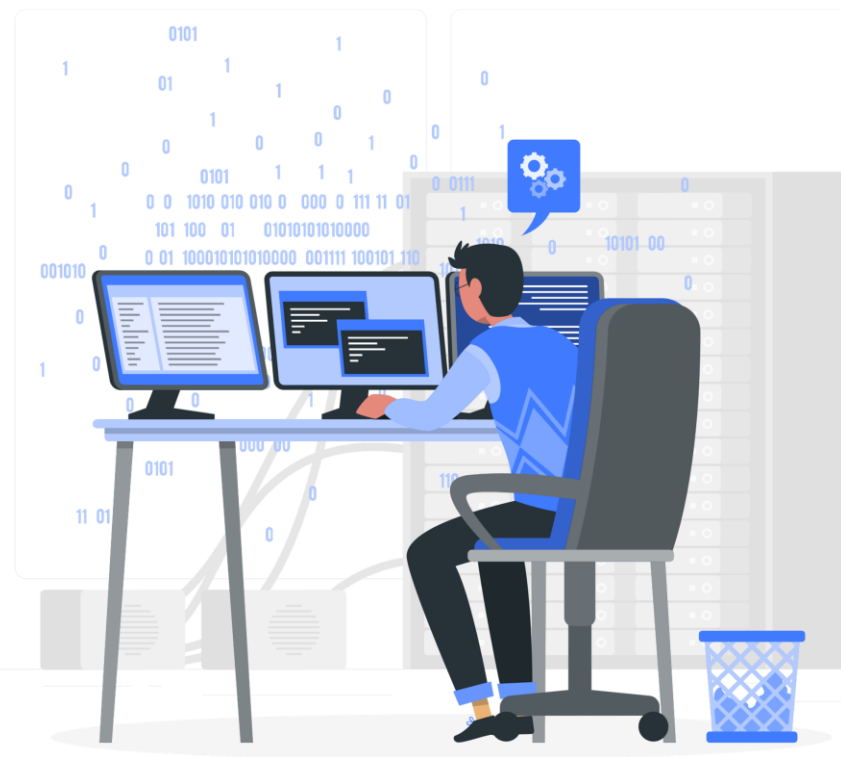


3. Ví dụ về khai báo biến, nhập và hiển thị giá trị của biến:

Ví dụ về khai báo biến:

EXAMPLE

```
public class Main {  
    public static void main(String[] args) {  
        int banKinh = 100;  
        long dienTich = 10182383812838123L;  
        float PI = 3.14F;  
        double temp = 30.89129391293D;  
        boolean kiemTra = true;  
        char kiTu = '@';  
    }  
}
```





3. Ví dụ về khai báo biến, nhập và hiển thị giá trị của biến:

Hiển thị giá trị của biến ra màn hình:

Để hiển thị giá trị của biến ta sử dụng `System.out.println()`.

EXAMPLE

```
public class Main {  
    public static void main(String[] args) {  
        long dienTich = 10182383812838123L;  
        System.out.println(dienTich);  
        System.out.println("Gia tri cua dien tich la :" + dienTich);  
    }  
}
```

OUTPUT

```
10182383812838123  
Gia tri cua dien tich la :10182383812838123
```





3. Ví dụ về khai báo biến, nhập và hiển thị giá trị của biến:

Hiển thị giá trị của biến ra màn hình:

Hiển thị giá trị của biến số thực với độ chính xác cho trước:

EXAMPLE

```
public class Main {  
    public static void main(String[] args) {  
        float chuVi = 3.18238f;  
        double dienTich = 39.91293912391293d;  
        System.out.printf("%.2f\n", chuVi);  
        System.out.printf("%.10f\n", dienTich);  
    }  
}
```

OUTPUT

```
3.18  
39.9129391239
```





3. Ví dụ về khai báo biến, nhập và hiển thị giá trị của biến:

Nhập giá trị cho biến từ bàn phím thông qua đối tượng của lớp Scanner

Để nhập giá trị cho các biến từ bàn phím bạn có thể sử dụng các hàm: **nextInt()**, **nextFloat()**, **nextDouble()**, **nextLine()**... của lớp Scanner.

EXAMPLE

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        long m = sc.nextLong();
        float f = sc.nextFloat();
        char kiTu = sc.nextLine().charAt(0);
    }
}
```

CHÚ Ý

Trong Java nếu bạn nhập dữ liệu không hợp lệ sẽ bị lỗi **InputMismatchException**. Ví dụ khi bạn sử dụng **nextInt()** để nhập số nguyên nhưng lại nhập số thực, kí tự... không phải số nguyên.





4. Toán tử:

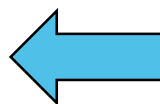
a. Toán tử gán:



Cú pháp:

[Toán hạng 1] = [Toán tử 2]

Toán hạng 1



Toán hạng 2

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

EXAMPLE

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





4. Toán tử:

b. Toán tử toán học:

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





4. Toán tử:

b. Toán tử toán học:

— **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.

EXAMPLE

```
public class Main {  
    public static void main(String[] args) {  
        //Chia nguyên  
        int a = 300, b = 200;  
        int c = a / b;  
        System.out.println(c);  
        //Chia lấy phần thập phân  
        float d = (float) a / b;  
        System.out.printf("%.2f", d);  
    }  
}
```

OUTPUT

1

1.50





4. Toán tử:

b. Toán tử toán học:

— **Chú ý 2:** Nếu bạn nhân 2 hoặc cộng số nguyên int với nhau mà kết quả của tích (tổng) 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.

EXAMPLE

```
public class Main {  
    public static void main(String[] args) {  
        //Xử lý chưa đúng  
        int a = 1000000, b = 1000000;  
        long c = a * b;  
        System.out.println(c);  
        //Xử lý đúng cách 1  
        long d = (long) a * b;  
        System.out.println(d);  
        //Xử lý đúng cách 2  
        long e = 1L * a * b;  
        System.out.println(e);  
    }  
}
```

OUTPUT

```
-727379968  
10000000000000  
10000000000000
```





4. Toán tử:


b. Toán tử toán học:

— **Chú ý 3:** Các toán tử toán học có thứ tự ưu tiên nhân chia trước cộng trừ sau. Các toán tử có cùng thứ tự ưu tiên thì phép toán được thực hiện từ trái qua phải. Tốt nhất các bạn nên sử dụng đóng mở ngoặc tròn để có thể thực hiện biểu thức theo ý muốn của mình.



4. Toán tử:

c. Toán tử so sánh:

 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



4. Toán tử:

c. Toán tử so sánh:



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.

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





4. Toán tử:

d. Toán tử logic:



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	&&	$(x \geq 10) \text{ and } (x \leq 50)$
OR		$(a \geq 10) \text{ or } (a \% 2 == 0)$
NOT	!	$!(a \leq 10)$



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.



4. Toán tử:

d. Toán tử logic:

Bảng chân lý của cổng AND

A	B	A and 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.

4. Toán tử:

d. Toán tử logic:

Bảng chân lý của cổng OR

A	B	A or 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

4. Toán tử:

d. Toán tử logic:

Bảng chân lý của cổng NOT

A	NOT A
0	1
1	0



4. Toán tử:

e. Toán tử tăng giảm:



Để tăng hoặc giảm giá trị của một biến đi 1 đơn vị ta có thể sử dụng toán tử tăng, giảm này sẽ thuận 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--

EXAMPLE

```
public class Main {  
    public static void main(String[] args) {  
        int a = 100;  
        int b = a++;  
        System.out.println(a + " " + b);  
        int n = 100;  
        int m = ++n;  
        System.out.println(n + " " + m);  
    }  
}
```

OUTPUT

```
101 100  
101 101
```





4. Toán tử:

f. Toán tử ba ngôi:



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

Không nên lạm dụng toán tử 3 ngôi





4. Toán tử:

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.






5. Cấu trúc rẽ nhánh:

a. Câu lệnh if:

 **Câu lệnh if** được sử dụng trong lập trình 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.

 **Ví dụ:** 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.

Cú pháp

```
if(condition){  
    //Các câu lệnh khi điều kiện  
    //condition có giá trị đúng  
}
```



5. Cấu trúc rẽ nhánh:

a. Câu lệnh if:

Ví dụ	Kết quả	Ý nghĩa
<pre>int a = 100, b = 200; if (a < b){ System.out.println("OK");}</pre>	OK	Vì $a < b$ có giá trị là đúng nên câu lệnh bên trong if được thực thi
<pre>int a = 100, b = 200; if ((a % 2) == 0){ System.out.println("Even");}</pre>	Even	Nếu a chia dư cho 2 bằng 0, thì a là số chẵn
<pre>int a = 100, b = 200; if((a > 50) && (a < 200){ System.out.println("OK"); }</pre>	OK	Nếu a lớn hơn 50 và nhỏ hơn 200 thì in OK



Các bạn nhớ thật 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.



5. Cấu trúc rẽ nhánh:

b. 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.

Cú pháp

```
if ( condition){  
    //Code khi condition có giá trị đúng  
}  
else{  
    //Code khi condition có giá trị sai  
}
```





5. Cấu trúc rẽ nhánh:

b. Câu lệnh if else:

Ví dụ	Kết quả	Ý nghĩa
<pre>int a = 100; if((a % 2) == 0){ System.out.println("Chan"); } else{ System.out.println("Le"); }</pre>	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"





5. Cấu trúc rẽ nhánh:

c. 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.



Cú pháp

```
if (condition1){  
    //Code nếu condition1 có giá trị đúng  
}  
else if(condition2){  
    //Code nếu condition1 sai và condition2 đúng  
}  
...  
else if(conditionN){  
    //Code nếu N - 1 điều kiện trước sai và conditionN đúng  
}  
else{  
    //Code nếu cả N điều kiện trong if và else if đều sai  
}
```



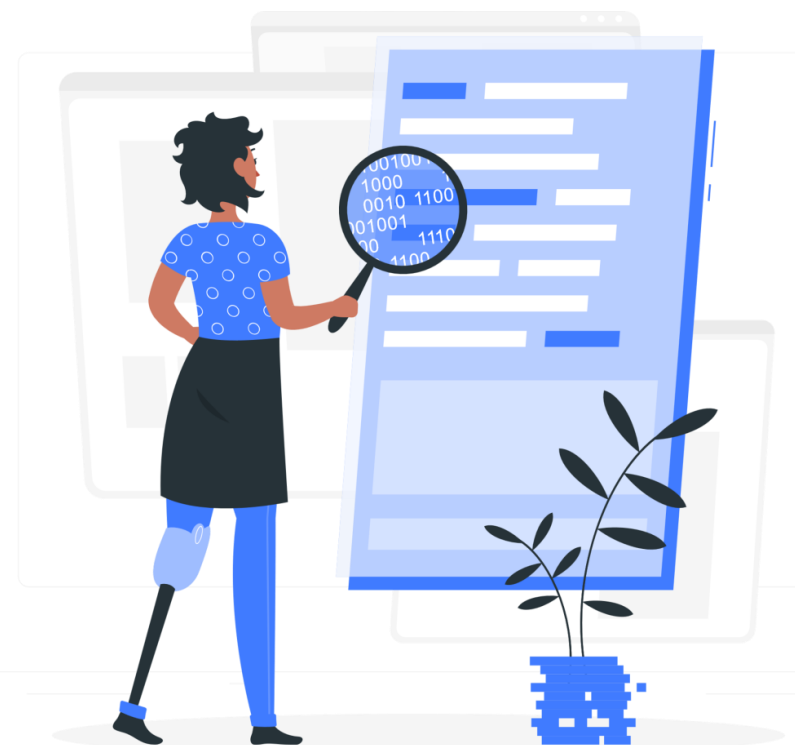


5. Cấu trúc rẽ nhánh:

c. 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.





5. Cấu trúc rẽ nhánh:

d. Cấu trúc switch case:



Tương tự như if và else if thì **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 cứ 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.

Cú pháp

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





6. Bảng mã ASCII và các câu lệnh kiểm tra liên quan tới kí tự:

0		20	¶	40	<	60	<	80	P	100	d	120	x	140	î	160	á	180	ı	200	ƒ	220	ƒ	240	≡
1	␣	21	␣	41)	61	=	81	Q	101	e	121	y	141	ï	161	â	181	ı	201	ı	221	ı	241	±
2	␣	22	␣	42	*	62	>	82	R	102	f	122	z	142	â	162	â	182	ı	202	ı	222	ı	242	±
3	␣	23	␣	43	+	63	?	83	S	103	g	123	<	143	â	163	â	183	ı	203	ı	223	ı	243	±
4	␣	24	␣	44	,	64	@	84	T	104	h	124	>	144	â	164	â	184	ı	204	ı	224	ı	244	±
5	␣	25	␣	45	.	65	A	85	U	105	i	125	~	145	â	165	â	185	ı	205	ı	225	ı	245	±
6	␣	26	␣	46	-	66	B	86	V	106	j	126	Δ	146	â	166	â	186	ı	206	ı	226	ı	246	±
7	␣	27	␣	47	/	67	C	87	W	107	k	127	Ç	147	â	167	â	187	ı	207	ı	227	ı	247	±
8	␣	28	␣	48	0	68	D	88	X	108	l	128	ç	148	â	168	â	188	ı	208	ı	228	ı	248	±
9	␣	29	␣	49	1	69	E	89	Y	109	m	129	ü	149	â	169	â	189	ı	209	ı	229	ı	249	±
10	␣	30	␣	50	2	70	F	90	Z	110	n	130	é	150	â	170	â	190	ı	210	ı	230	ı	250	±
11	␣	31	␣	51	3	71	G	91	[111	o	131	à	151	â	171	â	191	ı	211	ı	231	ı	251	±
12	␣	32	␣	52	4	72	H	92	\	112	p	132	â	152	â	172	â	192	ı	212	ı	232	ı	252	±
13	␣	33	␣	53	5	73	I	93]	113	q	133	â	153	â	173	â	193	ı	213	ı	233	ı	253	±
14	␣	34	␣	54	6	74	J	94	^	114	r	134	â	154	â	174	â	194	ı	214	ı	234	ı	254	±
15	␣	35	␣	55	7	75	K	95	_	115	s	135	â	155	â	175	â	195	ı	215	ı	235	ı	255	±
16	␣	36	␣	56	8	76	L	96	`	116	t	136	â	156	â	176	â	196	ı	216	ı	236	ı	255	±
17	␣	37	␣	57	9	77	M	97	a	117	u	137	â	157	â	177	â	197	ı	217	ı	237	ı	255	±
18	␣	38	␣	58	:	78	N	98	b	118	v	138	â	158	â	178	â	198	ı	218	ı	238	ı	255	±
19	␣	39	␣	59	;	79	O	99	c	119	w	139	â	159	â	179	â	199	ı	219	ı	239	ı	255	±





6. Bảng mã ASCII và các câu lệnh kiểm tra liên quan tới kí tự:



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



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



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

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





6. Bảng mã ASCII và các câu lệnh kiểm tra liên quan tới kí tự:



Bảng bên là một vài câu lệnh kiểm tra kí tự cần nắm vững.

Về phần này, đều có các hàm có sẵn để kiểm tra loại kí tự, nhưng ở thời điểm hiện tại, các bạn hãy hiểu bản chất của vấn đề trước khi sử dụng các hàm có sẵn.

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

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

