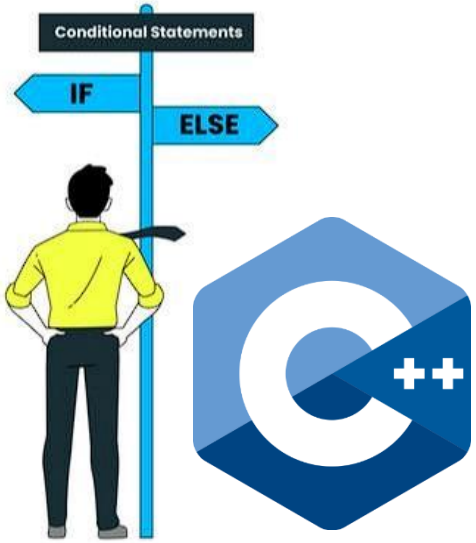
مبانی کامپیوتر و برنامه سازی

اسلاید هشتم

«برنامه نویسی مقدماتی به زبان C++»

بخش دوم : دستورات شرطی و انتخاب



محمد سعید صفایی صادق

درس

مبانی کامپیوتر

۹

۱۴۰۲

تفاوت void و int

درواقع int و void نوع تابع هستند.

که **void** به معنی **پوچ** هست.

یعنی اگر تابعی از نوع void تعریف شود نمیتواند مقداری را برگرداند.

و **int** نوع **عدد صحیح** است.

یعنی تابعی که از نوع int تعریف شده باشد باید یه مقدار صحیح را برگرداند.

اینها فقط مربوط به تابع main نیستند و در تعریف توابع هم باید به این صورت تعریف شوند.

تفاوت void و int

تابع main همیشه باید یک عدد صحیح برگرداند ، این عدد بیان کننده این هست که برنامه درست اجرا شده و به پایان رسیده یا نه ، اگر عدد بازگشتی ۰ باشد سیستم عامل میفهمد که اجرای برنامه بدون مشکل تمام شده است.

ولی اگر عددی غیر از ۰ باشه یعنی مشکلی در برنامه رخ داده بوده ، این مقدار بازگشتی اصولا در آخر برنامه ها با return 0 تعیین میشود.

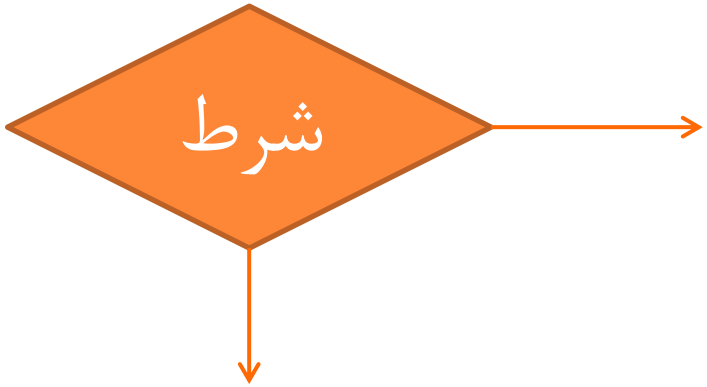
تفاوت void و int

اگر این خط را پاک کنید باز هم مشکلی در اجرا به وجود نمیاد چون کامپایلر C++ به صورت خودکار در زمان کامپایل این مقدار بازگشتی return 0 را به فایل اجرایی نهایی اضافه میکند .

در C++ استاندارد شما قادر نیستند مقدار بازگشتی تابع main برنامه را void در نظر بگیرید.

Void در بعضی کامپایلرهای خاص قابل قبول هست که تعدادشون خیلی خیلی کمه و کلا از لحاظ فنی مورد قبول نیست ،

پس همیشه نوع بازگشتی تابع اصلی برنامه رو int در نظر بگیرید .



دستورات شرطی و انتخاب
If - else

دستورات شرطی

دستورات شرطی برای انجام کار ها و تصمیم گیری های مختلف بر اساس شرایط مختلف به کار می رود.

در ادامه درس انتظار می‌رود پس از پایان این جلسه بتوانید نحو دستور if را شناخته و آن را در برنامه‌ها به کار ببرید.

نحو دستور if..else را شناخته و آن را در برنامه‌ها به کار ببرید.

از ساختار else..if در تصمیم‌گیری‌های پیچیده استفاده کنید.

نحو دستور switch را شناخته و خطای «تلهٔ سقوط» را تشخیص دهید.

دستورات شرطی

همهٔ برنامه‌هایی که در جلسات قبل بیان شد، به شکل ترتیبی اجرا می‌شوند، یعنی دستورات برنامه به ترتیب از بالا به پایین و هر کدام دقیقا یک بار اجرا می‌شوند.

در این جلسه نشان داده می‌شود چگونه از دستورالعمل‌های انتخاب جهت انعطاف‌پذیری بیشتر برنامه استفاده کنیم.

دستورات شرطی If

If (*condition*) statement;

Condition که شرط نامیده می‌شود یک عبارت صحیح است (عبارتی که با یک مقدار صحیح برآورد می‌شود) و statement می‌تواند هر فرمان قابل اجرا باشد.

Statement وقتی اجرا خواهد شد که condition مقدار غیر صفر داشته باشد.

دقت کنید که شرط باید درون پرانتز قرار داده شود.

دستورات شرطی if..else

دستور if..else موجب می‌شود بسته به این که شرط درست باشد یا خیر، یکی از دو دستورالعمل فرعی اجرا گردد.

نحو این دستور به شکل زیر است:

```
if (condition) statement1;
else statement2;
```

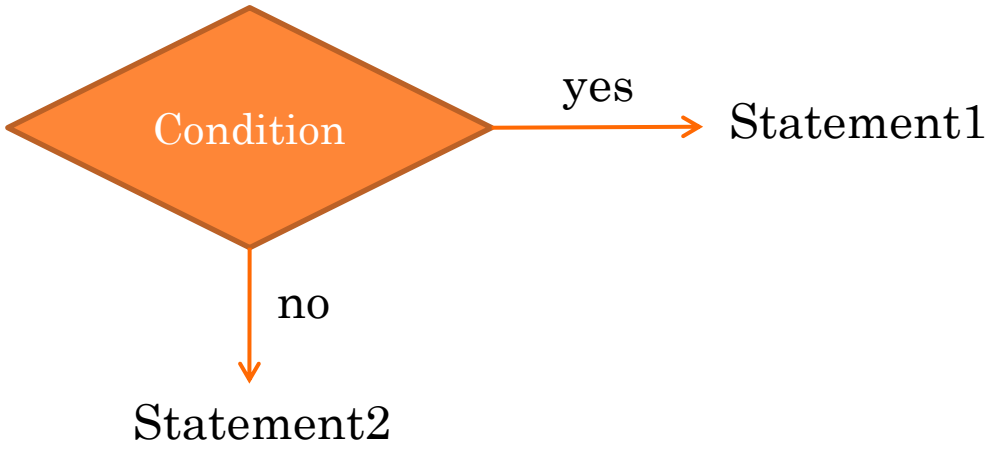
Condition همان شرط مساله است که یک عبارت صحیح می‌باشد و statement1 و statement2 فرمان‌های قابل اجرا هستند.

اگر مقدار شرط، غیر صفر باشد، **statement1** اجرا خواهد شد وگرنه **statement2** اجرا می‌شود.

دستورات شرطی if..else

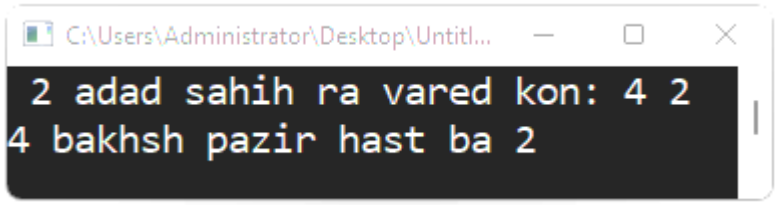
if (*condition*) statement1;

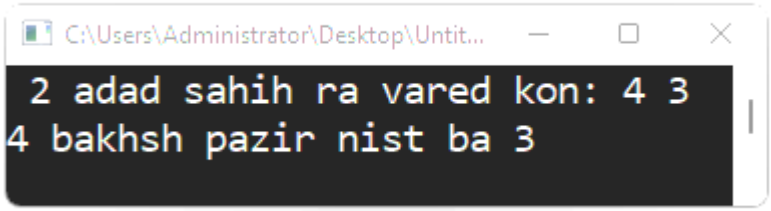
else statement2;



مثال : دو عدد صحیح را از ورودی دریافت، و در آزمون قابلیت تقسیم، نشان دهید به هم بخش پذیر هستند یا خیر؟

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {   int n, d;
5      cout << " 2 adad sahih ra vared kon: ";
6      cin >> n >> d;
7      if (n%d) cout << n << " bakhsh pazir nist ba " << d << endl;
8      else cout << n << " bakhsh pazir hast ba " << d << endl;
9  }
```





مثال : عددی را از ورودی دریافت کن، زوج یا فرد بودن آن را چاپ کن

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {  int n;
5     cout << " adad sahih ra vared kon: ";
6     cin >> n ;
7     if (n%2==0)
8     cout << n << " zoj ast " << endl;
9     else cout << n << " fard ast " << endl;
10 }
```

الگوی کلی دستور if

If (*condition*) statement;

If (*condition*) { statements;}

if (*condition*) statement1;
else statement2;

if (*condition*) {statements1;**}**
else {statements2;**}**

اگر بیشتر از یک دستورالعمل داشته باشد
از بلوک دستورالعمل {} استفاده می شود

یادآوری

عملگرهای مقایسه ای

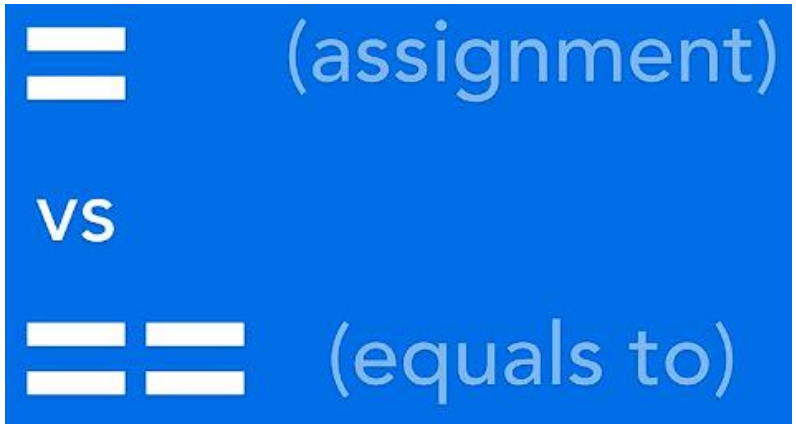
در C++ شش عملگر مقایسه‌ای وجود دارد: < و > و =< و => و == و =! . هر یک از این شش عملگر به شکل زیر به کار می‌روند:

```
x < y  // x کوچکتر از y است
x > y  // x بزرگتر از y است
x <= y // x کوچکتر یا مساوی y است
x >= y // x بزرگ‌تر یا مساوی y است
x == y // x مساوی با y است
x != y // x مساوی با y نیست
```

این‌ها می‌توانند برای مقایسهٔ مقدار عبارات با هر نوع ترتیبی استفاده شوند.
عبارت حاصل به عنوان یک شرط تفسیر می‌شود.
مقدار این شرط صفر است اگر شرط نادرست باشد و غیر صفر است اگر شرط درست باشد.
برای نمونه، عبارت ۵*۶>۸*۷ برابر با **صفر** ارزیابی می‌شود،

به این معنی که این شرط نادرست است.

دقت کنید که در C++ عملگر انتساب با عملگر برابری فرق دارد. عملگر جایگزینی یک مساوی تکی " = " است ولی عملگر برابری، دو مساوی " = = " است.

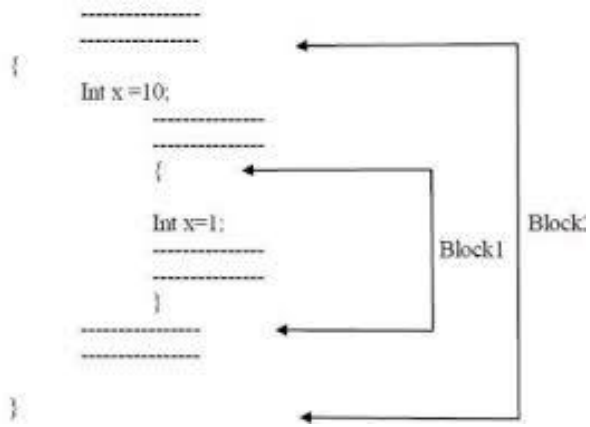


مثلا دستور x = 33; مقدار 33 را در x قرار می‌دهد، ولی دستور x == 33; بررسی می‌کند که آیا مقدار x با 33 برابر است یا خیر. درک این تفاوت اهمیت زیادی دارد.

بلوک دستورالعمل

یک *بلوک دستورالعمل* زنجیره‌ای از دستورالعمل‌هاست که درون براکت {} محصور شده، مانند :

```
{
   int temp=x;
   x = y;
   y = temp;
}
```

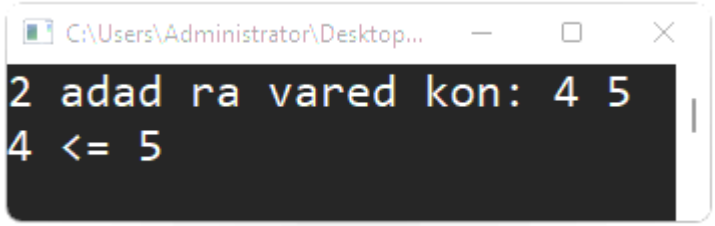


در برنامه‌های C++ یک بلوک دستورالعمل مانند یک دستورالعمل تکی است.

مثال : بلوک دستورالعمل درون یک دستور if

این برنامه دو عدد صحیح را گرفته و به ترتیب بزرگ‌تری، آن‌ها را چاپ می‌کند:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {   int x, y;
5      cout << "2 adad ra vared kon: ";
6      cin >> x >> y;
7      if (x > y)  { int temp = x;
8                    x = y;
9                    y = temp;
10                 }    //x va y ra jabeja kon
11     cout << x << " <= " << y << endl;
12 }
```



خطا و عملگر منطقی

این برنامه دارای یک خطای منطقی است. دقت کنید!

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {  int n1, n2, n3;
5     cout << "Enter three integers: ";
6     cin >> n1 >> n2 >> n3;
7     if (n1 >= n2 >= n3) cout << "max = " << n1;
8  }
```

(n1>=n2 && n2>=n3)

شرط‌هایی مانند n%d و x>=y می‌توانند به صورت یک شرط مرکب با هم ترکیب شوند.
این کار با استفاده ازعملگرهای منطقی

&& and و || or و ! Not

صورت می‌پذیرد.
این عملگرها به شکل زیر تعریف می‌شوند: (مثال رسیدن به خانه)

p && q درست است اگر و تنها اگر هم p و هم q هر دو درست باشند
p || q نادرست است اگر و تنها اگر هم p و هم q هر دو نادرست باشند
!p درست است اگر و تنها اگر p نادرست باشد

برای مثال(n%d || x>=y) نادرست است اگر و تنها اگر n%d برابر صفر و x کوچک‌تر از y باشد.

یادآوری: سه عملگر منطقی && و || و ! معمولا با استفاده از جداول درستی به گونهٔ زیر بیان می‌شوند:

p	q	P&&q
T	T	T
T	F	F
F	T	F
F	F	F

p	q	P||q
T	T	T
T	F	T
F	T	T
F	F	F

P	!P
T	F
F	T

طبق جدول‌های فوق اگر p درست و q نادرست باشد، عبارت p&&q نادرست و عبارت p||q درست است.

عملگرهای && و || به دو عملوند نیاز دارندتا مقایسه را روی آن دو انجام دهند.

جداول درستی نشان می‌دهد که p&&q نادرست است اگر p نادرست باشد.

در این حالت دیگر نیازی نیست که q بررسی شود.

همچنین p||q درست است اگر p درست باشد و در این حالت هم نیازی نیست که q بررسی شود.

در هر دو حالت گفته شده، با ارزیابی عملوند اول به سرعت نتیجه معلوم می‌شود.

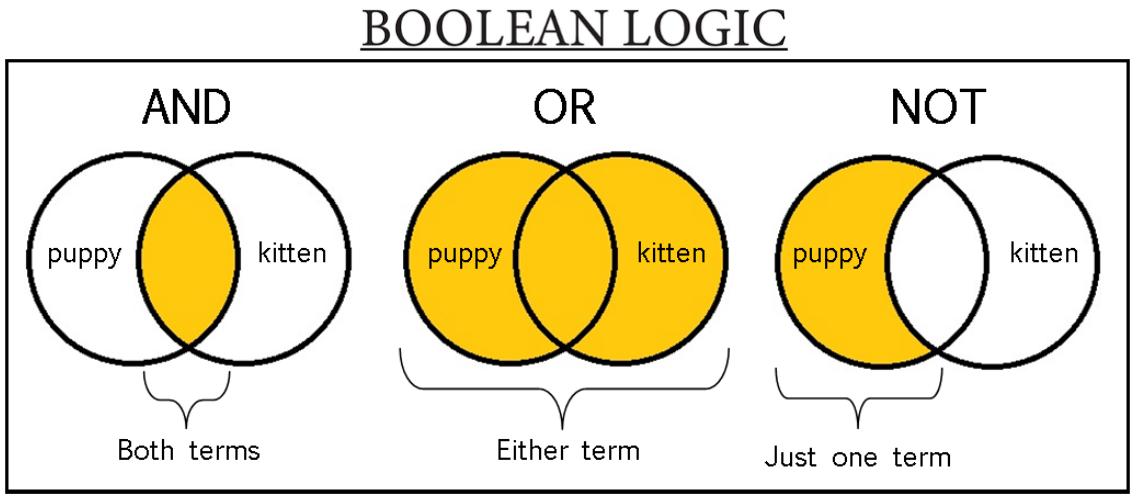
این کار ارزیابی میانبری نامیده می‌شود.

یک عبارت منطقی شرطی است که یا درست است یا نادرست.

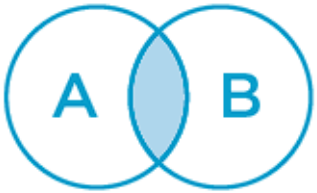
قبلا دیدیم که عبارات منطقی با مقادیر صحیح ارزیابی می‌شوند.

مقدار صفر به معنای نادرست و هر مقدار غیر صفر به معنای درست است.

به عبارات منطقی «عبارات بولی» هم می‌گویند.

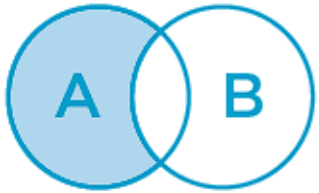


عملگر منطقی



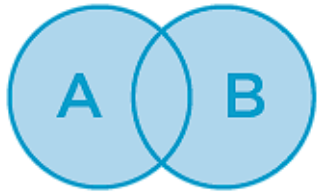
AND

When using **AND**, only the events located within the overlap of all gates defined by **AND** are included. For example, **A AND B** only includes the events that fall within the overlap of both gates.



NOT

Excluded events are defined by **NOT**. For example, **A NOT B** includes all events from gate **A**, but excludes all events in gate **B**.



OR

Events in all of the gates defined with an **OR** will be included in the Boolean gate. For example, **A OR B** includes all events in both gates.

به عبارات منطقی «عبارات بولی» هم می‌گویند

چون همهٔ مقادیر صحیح ناصفر به معنای درست تفسیر می‌شوند، عبارات منطقی اغلب تغییر قیافه می‌دهند.
برای مثال دستور

```
if (n) cout << "n is not zero";
```

وقتی n غیر صفر است عبارت n is not zero را چاپ می‌کند زیرا عبارت منطقی (n) وقتی مقدار n **غیر صفر** است به عنوان درست تفسیر می‌گردد.

کد زیر را نگاه کنید:

```
if (n%d) cout << "n is not a multiple of d";
```

دستور خروجی فقط وقتی که n%d ناصفر است اجرا می‌گردد و n%d وقتی ناصفر است که n بر d بخش‌پذیر نباشد.

گاهی ممکن است فراموش کنیم که عبارات منطقی مقادیر صحیح دارند و این فراموشی باعث ایجاد نتایج غیر منتظره و نامتعارف شود.

دستورهای انتخاب
تو در تو

دستورهای انتخاب می‌توانند مانند دستورالعمل‌های مرکب به کار روند. به این صورت که یک دستور انتخاب می‌تواند درون دستور انتخاب دیگر استفاده شود. به این روش، ***جملات تودرتو*** می‌گویند.

مثال: دستورهای انتخاب تودرتو

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int n, d;
6      cout << "Enter two positive integers: ";
7      cin >> n >> d;
8      if (d != 0)
9          if (n%d == 0) cout << d << " divides " << n << endl;
10         else cout << d << " does not divide " << n << endl;
11     else cout << d << " does not divide " << n << endl;
12 }
```

در برنامهٔ بالا، دستور if..else دوم درون دستور if..else اول قرار گرفته است.

وقتی دستور if..else به شکل تو در تو به کار می‌رود، کامپایلر از قانون زیر جهت تجزیه این دستورالعمل مرکب استفاده می‌کند:

« هر else با آخرین if تنها جفت می‌شود.»

دستور **if..else** تودرتو، اغلب برای بررسی مجموعه‌ای از حالت‌های متناوب یا موازی به کار می‌رود.

در این حالات فقط عبارت **else** شامل دستور **if** بعدی خواهد بود.

این قبیل کدها را معمولا با ساختار **else if** می‌سازند.

استفاده از ساختار else if برای مشخص کردن محدودهٔ نمره. برنامهٔ زیر یک نمرهٔ امتحان را به درجهٔ حرفی معادل تبدیل می‌کند:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {  int score;
5     cout << "Enter your test score: ";  cin >> score;
6     if (score > 100) cout << "Error: that score is out of range.";
7     else if (score >= 90) cout << "Your grade is an A." << endl;
8     else if (score >= 80) cout << "Your grade is a B."  << endl;
9     else if (score >= 70) cout << "Your grade is a C."  << endl;
10    else if (score >= 60) cout << "Your grade is a D."  << endl;
11    else if (score >= 0)  cout << "Your grade is an F." << endl;
12    else cout << "Error: that score is out of range.";
13 }
```

اگر اولی نبود، دومی، اگر نبود، سومی و ....

یادآوری: تمرین فلوچارت ایام هفته

عملگر
عبارت شرطی

عملگر عبارت شرطی یکی از امکاناتی است که جهت اختصار در کدنویسی تدارک دیده شده است. این عملگر را می‌توانیم به جای دستور **if..else** به کار ببریم. این عملگر از نشانه‌های **?** و **:** به شکل زیر استفاده می‌کند:

***condition* ? expression1 : expression2;**

در این عملگر ابتدا شرط ***condition*** بررسی می‌شود. اگر این شرط درست بود، حاصل کل عبارت برابر با **expression1** می‌شود و اگر شرط نادرست بود، حاصل کل عبارت برابر با **expression2** می‌شود.

مثال :

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int x,y,min;
6      cout<<"Put Two int Number: "<<endl;
7      cin>>x>>y;
8      min = ( x<y ? x : y );
9      cout<<"Min: "<<min<<endl;
10 }
```

اگر x<y باشد مقدار xرا درون min قرار می‌دهد و اگر x>y نباشد مقدار y را درون minقرار می‌دهد .

یعنی به همین سادگی و اختصار، مقدار کمینهٔ xو yدرون متغیر min قرار می‌گیرد .

عملگر شرط و انتخاب
Switch-Case

Switch-Case

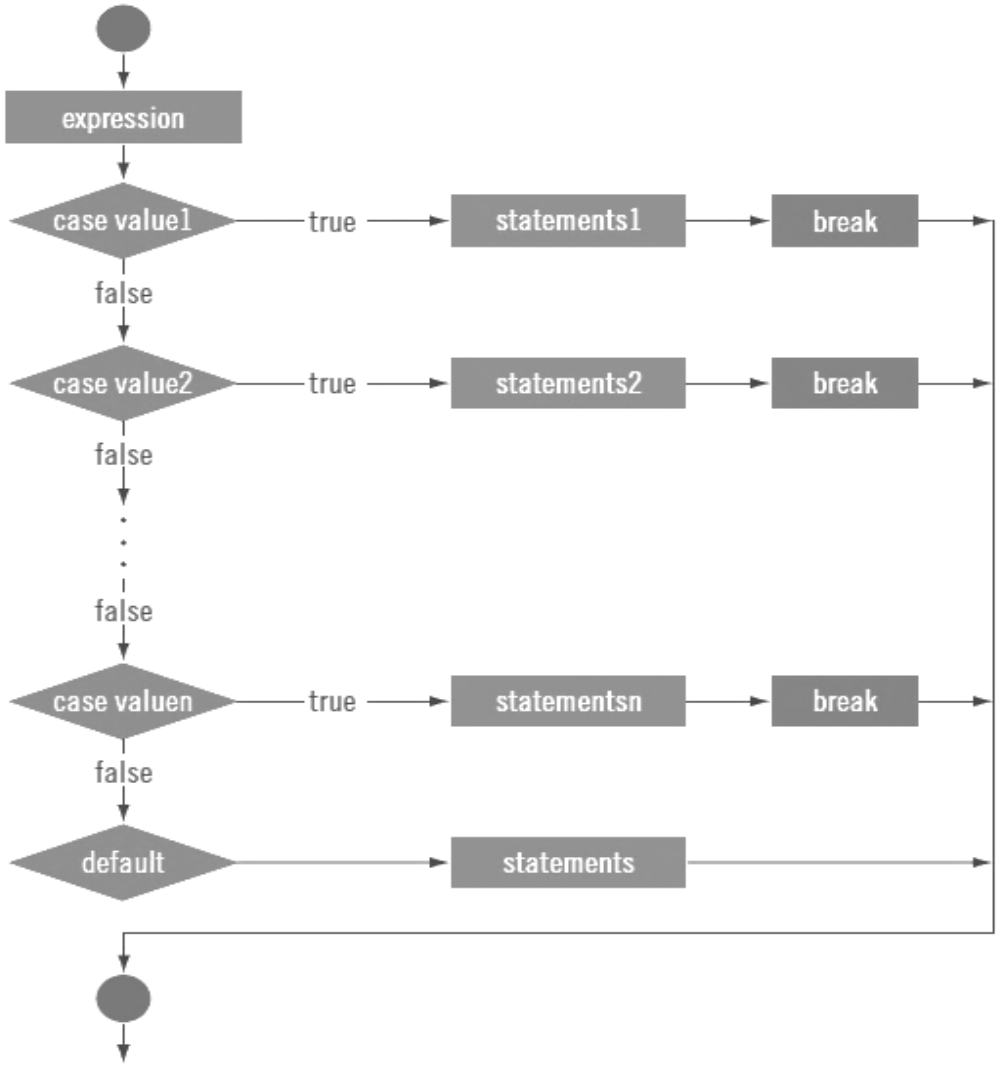
دستور switch می‌تواند به جای ساختار else if برای بررسی مجموعه‌ای از حالت‌های متناوب و موازی به کار رود.

نحو دستور switchبه شکل زیر است:

```
switch (expression)
{
  case constant1: st0atementlist/s1;
  case constant2: statementlist/s2;
  case constant3: statementlist/s3;
       :
       :
  case constantN: statementlistN;
  default: statementlist0;
}
```

Expression = عبارت (که نام متغییر است)
Constant = مقدار ثابت
Statementlist = لیست دستور

Switch-Case



نکات:

۱- متغییر استفاده شده در switch تنها می‌تواند صحیح یا کاراکتر باشد.

۲- دستور Break بر if یا if-else تأثیری ندارد.

۳- عبارت default یک عبارت اختیاری است.

یعنی می‌توانیم در دستور switch آن را قید نکنیم.

Switch-Case

این دستور ابتدا **expression** را برآورد می‌کند و سپس میان ثابت‌های **case** به دنبال مقدار آن می‌گردد.

اگر مقدار مربوطه از میان ثابت‌های فهرست‌شده یافت شد، دستور **statementlist** مقابل آن **case** اجرا می‌شود.

اگر مقدار مورد نظر میان **case** ها یافت نشد و عبارت **default** وجود داشت، دستور **statementlist** مقابل آن اجرا می‌شود.

عبارت **default** یک عبارت اختیاری است.
یعنی می‌توانیم در دستور **switch** آن را قید نکنیم.
expression باید به شکل یک نوع صحیح ارزیابی شود و ***constant*** ها باید ثابت‌های صحیح باشند.

لازم است در انتهای هر **case** دستور **break** قرار بگیرد.

بدون این دستور، اجرای برنامه پس از این که case مربوطه را اجرا کرد از دستور switch خارج نمی‌شود، بلکه همهٔ case های زیرین را هم خط به خط می‌پیماید و دستورات مقابل آن‌ها را اجرا می‌کند.

به این اتفاق، **تلهٔ سقوط** می‌گویند.

case constant1: *statementlist1*;

break;

مثال: عملکرد برنامه زیر را توضیح دهید:

```
1  #include <iostream>
2  #include <conio.h>
3  using namespace std;
4  int main()
5  {
6  char c;
7  c = getch();
8  switch (c) {
9      case 'a':
10         cout<<"its ASCII code is 97" <<endl;
11         break;
12     case 'b':
13         cout<<"its ASCII code is 98" <<endl;
14         break;
15     default:
16         cout<< "not a character"<<endl;
17         }
18 }
```

فایل کتابخانه ای Console Input و Output که برای گرفتن ورودی و خروجی به صورت خاص است.

یک کاراکتر را از کیبرد می گیرد و returnمی کند
ولی این کد کاربرد دیگری نیز دارد بعضی افراد برای اینکه پس از پایان برنامه، برنامه در جا بسته نشود از getchو getch در اخر کد استفاده می کنند.

مثال: عملکرد برنامه زیر را توضیح دهید:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int x;
6      cin >> x;
7      switch(x)
8      {
9          case 5:
10         cout<<"*****"<<endl;
11         break;
12         case 2:
13         cout<<"**"<<endl;
14         break;
15         default:
16         cout<<"error!"<<endl;
17     }
18 }
```

مثال: عملکرد برنامه زیر را توضیح دهید:

```cpp
#include <iostream>
using namespace std;
int main()
{
    int day;
    cin>>day;
    switch(day)
    {
        case 1:
        cout<<"Shanbe"<<endl;
        break;
        case 2:
        cout<<"1 Shanbe"<<endl;
        break;
        case 3:
        cout<<"2 Shanbe"<<endl;
        break;
        case 4:
        cout<<"3 Shanbe"<<endl;
        break;
        case 5:
        cout<<"4 Shanbe"<<endl;
        break;
        case 6:
        cout<<"5 Shanbe"<<endl;
        break;
        case 7:
        cout<<"Jome"<<endl;
        break;
        default:
        cout<<"Error!"<<endl;
        break;
    }
}
```

کلمات کلیدی در C++

اکنون با کلماتی مثل if و case و float آشنا شدیم.
دانستیم که این کلمات برای C++ معانی خاصی دارند.
از این کلمات نمی‌توان به عنوان نام یک متغیر یا هر منظور دیگری استفاده کرد و فقط باید برای انجام همان کار خاص استفاده شوند.
مثلا کلمهٔ float فقط باید برای معرفی یک نوع اعشاری به کار رود.

یک *کلمهٔ کلیدی* در یک زبان برنامه‌نویسی کلمه‌ای است که از قبل تعریف شده و برای هدف مشخصی منظور شده است.

C++ استاندارد اکنون شامل ۷۴ کلمهٔ کلیدی است:

and	and_eq	asm
auto	bitand	Bitor
bool	break	case
catch	char	class
compl	const	const_cast
continue	default	delete
dodouble	dynamic_cast	else
enum	explicit	export
extern	dfalse	float
for	friend	goto

کلمات کلیدی

if	inline	int
long	mutable	namespace
new	not	not_eq
operator	or	or_eq
privat	eprotected	public
register	reinterpret_cast	return
short	signed	sizeof
static	static_cast	struct
swich	template	this
throw	TRUE	try

کلمات کلیدی

typedef	typoid	typename
using	union	unsigned
virtual	void	volatile
wchar_t	while	xor
xor_eq		

دو نوع کلمهٔ کلیدی وجود دارد:

۱- *کلمه‌های رزرو شده*

۲- *شناسه‌های استاندارد.*

یک **کلمهٔ رزرو** شده کلمه‌ای است که یک دستور خاص از آن زبان را نشان می‌دهد.

کلمهٔ کلیدی if و else کلمات رزرو شده هستند.

یک **شناسهٔ استاندارد** کلمه‌ای است که یک نوع دادهٔ استاندارد از زبان را مشخص می‌کند. کلمات کلیدی **bool** و **int** شناسه‌های استاندارد هستند

با تشکر از همراهی شما

محمد سعید صفایی صادق

www.SaeidSafaei.ir