عملگرهای منطقی

عملگرهای منطقی عبارتند از ! ، || و && . نتیجه عملگر ! (NOT) وقتی درست است که عبارتی که این عملگر بر روی آن عمل می کند نادرست باشد و نتیجه هنگامی نادرست است که عملوند آن درست باشد. ضمناً این عملگر تنها یک عملوند دارد. در حقیقت این عملگر نقیض عملوند خود را به عنوان نتیجه می دهد.

به مثال های زیر توجه کنید:

```
!(5 == 5)     ----> نادرست
!(6 <= 4)     ----> درست
!0            ----> درست
!1            ----> نادرست
```

عملگرهای &&(AND) و ||(OR) هنگامی مورد استفاده قرار می گیرند که بخواهیم از دو عبارت یک نتیجه را بدست آوریم. نتیجه این عملگرها بستگی به ارتباط بین دو عملوندشان طبق جدول زیر دارد:

عملوند اول b	عملوند دوم a	نتیجه a&&b	نتیجه a||b
درست	درست	درست	درست
درست	نادرست	نادرست	درست
نادرست	درست	نادرست	درست
نادرست	نادرست	نادرست	نادرست

به مثال های زیر توجه نمائید:

```
((5==5)&&(3>6))     -----> نادرست
((5==5)||(3>6))     -----> درست
((3-3)&&(3<5))      -----> نادرست
((3-3)||(3<5))      -----> درست
```

در مثال های زیر به جای اعداد از متغیر نیز استفاده شده است (فرض کنید a=1 و b=2 وc=3)

```
((b-2*a)&&(c==3))          -----> نادرست
((b==2*a)&&(c!=4))         -----> نادرست
```

```
((c==a+b)||(b<a))          -----> درست
((b-c==-a)||(b-c==a))      -----> درست
```

عملگر شرطی

این عملگر یک عبارت را مورد ارزیابی قرار می دهد و براساس عبارت ارزیابی شده مقادیر متفاوتی را به عنوان نتیجه بر می گرداند. ساختار این عملگر به صورت زیر می باشد:

```
شرط ? نتیجه ۱ : نتیجه ۲
```

اگر شرط برقرار باشد نتیجه ۱ به عنوان خروجی خواهد بود در غیر این صورت نتیجه ۲ به عنوان خروجی در نظر گرفته می شود. به مثال های زیر توجه نمایید:

```
7==6?4:3     --->خروجی عدد۳ می باشد چون۷ مساوی ۶ نمی باشد
8==6+2?4:3   --->خروجی عدد۴ می باشد چون۸ مساوی ۶+۲می باشد
6>3?a:b      ---> خروجی a است چون ۶ از ۳ بزرگتر می باشد
a>b?a:b      ---> خروجی a یا b عدد بزرگتر می باشد
```

همانطور که در عملگرهای محاسباتی دیدیم درک تقدم عملگرها، اهمیت ویژه ای داشت در اینجا نیز دانستن این تقدم از اهمیت خاصی برخوردار می باشد، تقدم عملگرهای رابطه ای ، منطقی و شرطی به ترتیب عبارتند از:

۱- !
۲- > >= < <=
۳- == !=
۴- &&
۵- ||
۶- ?:

به عنوان مثال مراحل بررسی عبارت مقابل به صورت زیر می باشد:

```
2 >= 3 && 2 == 2 || 2 != 3
  1     4     2     5     3

1  نادرست
2  درست
```

```
3    درست
4    نادرست  &&  درست    ---->    نادرست
5    نادرست || درست    ---->        درست

جواب نهایی درست می باشد
```

پیشنهاد می شود برای جلوگیری از پیچیدگی فهم عبارتهای منطقی و یا محاسباتی تقدم های مورد نظر را با به کار بردن پرانتز کاملاً مشخص کنیم ، به عنوان مثال عبارت فوق را به صورت زیر مورد استفاده قرار دهیم:

```
(((2 >= 3)  &&  (2 == 2))  ||  (2 != 3))
```

دستورات ورودی و خروجی

همانطور که در سازمان کامپیوتر گفته شد کامپیوتر دارای واحدهای ورودی و خروجی می باشد. واحد ورودی که ما در اینجا استفاده می کنیم صفحه کلید می باشد و واحد خروجی مورد استفاده نیز صفحه نمایش خواهد بود.

برای دریافت اطلاعات از صفحه کلید ، زبان C++ دستوری به نام **cin** را در اختیار ما قرار داده است، و برای ارسال اطلاعات به صفحه نمایش دستور **cout** موجود می باشد. توسط این دو دستور شما می توانید با نمایش اطلاعات بر روی صفحه نمایش و دریافت اطلاعات از صفحه کلید با کاربری که از برنامه شما استفاده می کند، در ارتباط باشید.

دستور خروجی cout

دستور cout همراه علامت << به کار می رود.

```
cout << "This is a test";
```

دستور فوق عبارت This is a test را بر روی صفحه نمایش چاپ می کند.

```
cout << 5120;
```

دستور فوق عدد ۵۱۲۰ را بر روی صفحه نمایش ظاهر می سازد.

```
cout << x;
```

دستور فوق محتویات متغیر x را به صفحه نمایش می فرستد.

علامت >> با نام عملگر درج شناخته می شود و اطلاعاتی که بعد از این علامت قرار می گیرند به واحد خروجی منتقل می شوند. در مثال های فوق یک عبارت رشته ای (This is a test) یک عدد (۵۱۲۰) و یک متغیر (x) به واحد خروجی ارسال شدند. توجه داشته باشید که در اولین مثال عبارت This is a test بین دو علامت (") قرار گرفت ، چون این عبارت حاوی رشته ای از حروف می باشد؛ هرگاه که بخواهیم رشته ای از حروف را به کار ببریم باید آنها را بین دو علامت (") قرار دهیم تا با نام متغیرها به اشتباه گرفته نشوند. به عنوان مثال، دستور زیر:

```
cout << " Hello";
```

عبارت Hello را بر روی صفحه نمایش ظاهر می سازد ولی دستور زیر:

```
cout << Hello;
```

محتویات متغیری با نام Hello را بر روی صفحه نمایش چاپ می کند.

عملگر درج ممکن است بیش از یک بار در یک جمله به کار رود، به عنوان مثال دستور زیر:

```
cout << "Hello," << "I am" << "new in C++";
```

پیغام Hello, I am new in C++ را بر روی صفحه نمایش نشان می دهد.

تکرار استفاده از عملگر درج در یک دستور به ما این امکان را می دهد که ترکیبی از متغیر و رشته حروف را در کنار هم استفاده کنیم.

```
cout << "Hello, my code is" << code
     << "and I am" << age << "years old.";
```

به عنوان مثال دستور فوق با فرض اینکه متغیر code حاوی عدد ۱۱۶۲۲۳ و متغیر age حاوی عدد ۱۶ باشد عبارت زیر را در صفحه نمایش ظاهر می سازد:

```
Hello, my code is 116223 and I am 16 years old.
```

توجه داشته باشید که دستور cout عبارات را به صورت خودکار به خط بعد منتقل نمی کند، به عنوان مثال دستورهای زیر:

```
cout << "This is a text.";
cout << "This is another text.";
```

علارقم اینکه از دستور cout در دو خط استفاده شده است، به صورت زیر در صفحه نمایش نشان داده خواهد شد:

```
This is a text. This is another text.
```

برای اینکه عبارتی را در چند خط نمایش دهیم، برای انتقال به هر خط جدید از علامت \n استفاده می کنیم. به عنوان مثال دستورات زیر:

```
cout << "First sentence.\n";
cout << "Second sentence.\n Third sentence.";
```

به شکل زیر در صفحه نمایش دیده خواهد شد:

```
First sentence.
Second sentence.
Third sentence.
```

علاوه بر علامت \n می توان از دستور endl برای انتقال به خط جدید استفاده کرد به عنوان مثال دستورات :

```
cout << "First sentence." << endl;
cout << "Second sentence." << endl;
```

در صفحه نمایش به صورت زیر دیده می شوند:

```
First sentence.
Second sentence.
```

دستور ورودی cin

دستور cin همراه علامت << به کار می رود.

```
int age;
cin >> age;
```

دستورات فوق ابتدا فضایی در حافظه برای متغیر age در نظر می گیرند، سپس برنامه منتظر وارد کردن عددی از صفحه کلید می ماند تا عدد وارد شده را در متغیر age قرار دهد. cin هنگامی قادر به دریافت اطلاعات از صفحه کلید خواهد بود که، کلید Enter بر روی صفحه کلید فشرده شود. به عنوان مثال اگر بخواهیم عدد ۱۶ در متغیر age قرار گیرد ابتدا عدد ۱۶ را تایپ کرده سپس دکمه Enter را فشار می دهیم.

علامت >> با نام عملگر استخراج شناخته می شود، و اطلاعاتی که از واحد ورودی دریافت می شود در متغیری که بعد از این علامت می باشد، قرار می گیرند. ضمناً شما می توانید توسط یک دستور cin بیش از یک متغیر را مقدار دهی کنید.

به عنوان مثال دستورات زیر معادل یکدیگر می باشند:

```
cin >> a >> b;
cin >> a;
cin >> b;
```

برنامه چاپ یک متن

در مباحث قبلی با مفاهیم حافظه و انواع داده، اعمال محاسباتی، عبارات منطقی ودستورات ورودی وخروجی آشنا شدیم. با این مقدمات می توان نوشتن اولین برنامه را آغاز کرد و با این برنامه نحوه اجرا و سایر جزئیات را مورد بررسی قرار خواهیم داد.

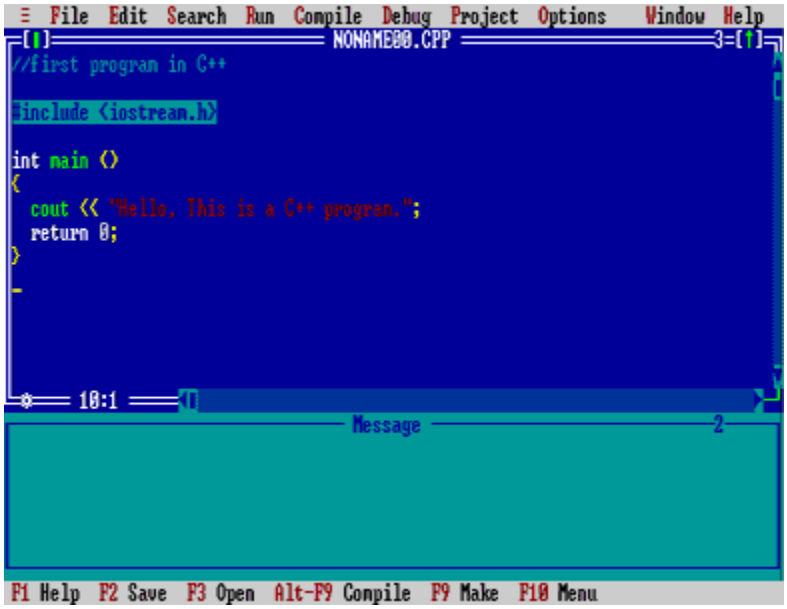
```
//first program in C++

#include <iostream.h>

int main ()
{
  cout << "Hello, This is a C++ program.";
  return 0;
}
```

قبل از هر گونه توضیح اضافی به شیوه نوشتن این برنامه در ویرایشگر زبان C++ و نحوه اجرای آن می پردازیم.

برای اجرای این برنامه شما به کامپایلر زبان **C++** نیاز دارید ، که باید آنرا نصب کنید. پس از نصب برنامه به آدرس "c:\tcp\bin" بر روی کامپیوتر خود مراجعه کرده و فایل **tc** را اجرا نمایید. با اجرای این فایل وارد محیط ویرایشگر برنامه **C++** خواهید شد.



حال در این محیط می توانید برنامه ذکر شده در ابتدای این مبحث را بنویسید. در نوشتن برنامه دقت لازم را به کار ببرید ، چون در صورت وجود اشتباهات تایپی ممکن است اجرای برنامه با مشکل مواجه شود. پس از این که برنامه را نوشتید یک بار دیگر آن را بررسی کنید تا مطمئن شوید اشتباهی ندارد، سپس دکمه **Ctrl** را از صفحه کلید فشار داده و آن را نگه دارید ، سپس همزمان دکمه **F9** از صفحه کلید را فشار دهید(**Ctrl + F9**). با این کار برنامه اجرا خواهد شد.
همانطور که دیدید صفحه نمایش به سرعت دوباره به صفحه ویرایشگر زبان **C++** بر می گردد. برای اینکه خروجی برنامه خود را ببینید دکمه **Alt** را همراه با **F5** را فشار دهید (**Alt + F5**) . با این کار به محیط خروجی برنامه خواهید رفت و پیغام:

```
Hello, This is a C++ program.
```

را در یک صفحه سیاه خواهید دید. تبریک می گوئیم ، شما اولین برنامه خود با زبان **C++** را با موفقیت به مرحله اجرا گذاشتید؛ به همین سادگی.

اگر به یاد داشته باشید در مبحث معرفی ساختاری زبان **C++** روال رسیدن به مرحله اجرا را شش مرحله ذکر کردیم، در اینجا مرحله اول را که نوشتن برنامه در ویرایشگر بود، شما به اجرا گذاشتید و با فشار کلید (**Ctrl + F9**) مراحل دوم تا ششم به صورت خودکار انجام گرفت، پس شش مرحله ذکر شده همچنان برقرار است و ما نیز نیازی نداریم خود را با جزئیات مراحل دوم تا ششم درگیر کنیم.

پس برای اجرای هر برنامه کافی است برنامه را در محیط ویرایشگر زبان C++ بنویسیم سپس دکمه (**Ctrl + F9**) را فشار دهیم.

برنامه ای که ما در اینجا نوشتیم یکی از ساده ترین برنامه هایی است که می توانیم به زبان **C++** بنویسیم ، ضمن اینکه شامل دستوراتی است که تقریباً هر برنامه **C++** به آنها نیاز دارد. در اینجا به بررسی تک به تک دستورات برنامه فوق می پردازیم.

```
//first program in C++
```

دستور فوق شامل توضیحات می باشد و تأثیری بر نحوه اجرای برنامه نخواهد داشت. هر نوشته ای که بعد از علامت // در زبان **C++** قرار گیرد به عنوان توضیحات در نظر گرفته می شود. توضیحی که در اینجا مورد استفاده قرار گرفته به ما می گوید که این اولین برنامه ما به زبان **C++** می باشد. علاوه بر علامت // ، توضیحات را می توان بین */* و */** قرار داد. از شیوه جدید هنگامی استفاده می کنیم که توضیحات ما بیش از یک خط باشد.

```
/* This is a comment line.
   This is another comment line. */
```

قرار دادن دستورات فوق در برنامه تأثیری بر خروجی ندارد و تنها توضیحاتی برای فهم بهتر برنامه می باشد.

```
#include <iostream.h>
```

خطوطی که با علامت پوند # شروع می شوند دستوراتی برای پیش پردازنده می باشند. این دستورات جزء خطوط اجرایی برنامه نمی باشند و نشانه هایی برای کامپایلر می باشند. در اینجا دستور فوق به پیش پردازنده می گوید که تعدادی از دستورات مورد استفاده در این برنامه در فایل کتابخانه ای **iostream.h** قرار دارند. در این مورد خاص فایل **iostream.h** شامل دستورات ورودی و خروجی (مانند **cin** و **cout**) می باشد.

```
int main( )
```

این دستور شروع بدنه اصلی برنامه را مشخص می کند. تابع **main** اولین جایی از برنامه است که زبان **C++** شروع به اجرای آن می کند. فرقی ندارد که تابع **main** را در کجا مورد استفاده قرار دهیم. ابتدا وسط یا انتهای کدهای برنامه نویسی، در هر کجا که تابع **main** را قرار دهیم ، زبان **C++** ابتدا به این تابع مراجعه می کند. بعد از کلمه **main** یک جفت پرانتز ( ) قرار می دهیم، چون **main** یک تابع است. در زبان **C++** تمام توابع دارای یک جفت پرانتز می باشند (در مبحث توابع به طور مفصل در باره نحوه ایجاد تابع و آرگومانها و ... صحبت خواهیم کرد). محتویات تابع **main** همانطور که در برنامه دیدید بین دو علامت { } قرار می گیرند.

```
cout << "Hello, This is a C++ program.";
```

این دستور کار اصلی مورد نظر در این برنامه را که چاپ عبارت داخل کوتیشن " " بر روی صفحه نمایش است را انجام می دهد. همانطور که گفته شد هنگامی که می خواهیم از دستورات ورودی و خروجی استفاده کنیم باید در ابتدای برنامه از دستور #include<iostream.h> استفاده کنیم. توجه داشته باشید که بعد از هر دستور زبان **C++** ملزم به استفاده از علامت (**;**) می باشیم.

```
return 0;
```

این دستور باعث می شود که تابع **main** به پایان برسد و عدد صفر را به عنوان خروجی تابع بر می گرداند. این مرسوم ترین روش برای پایان دادن به برنامه بدون دریافت هیچگونه پیغام خطا می باشد. همانطور که در برنامه های بعدی خواهید دید، تقریباً همه برنامه های زبان **C++** با دستوری مشابه دستور فوق به پایان می رسند.

نکته: بهتر است هر دستور زبان **C++** را در یک خط جداگانه بنویسیم. البته در انجام اینکار الزامی نداریم ولی برای خوانایی بیشتر برنامه توصیه می شود از این شیوه استفاده کنید.

برنامه جمع دو عدد

در مبحث قبلی برنامه ای را نوشتیم که در آن تنها از دستور cout استفاده شده بود و رشته ای را بر روی صفحه نمایش چاپ می کرد. برای اینکه با نحوه کاربرد متغیرها و شیوه مقدار دهی به آنها و نیز دستور cin آشنا شوید، در اینجا برنامه جدیدی را می نویسیم که دو عدد را از ورودی دریافت کرده و سپس آنها را جمع نموده و حاصل را در خروجی نمایش می دهد.

```
// Addition program.
#include <iostream.h>

// function main begins program execution
int main()
{
  int integer1; // first number to be input by user
  int integer2; // second number to be input by user
  int sum; // variable in which sum will be stored

  cout << "Enter first integer\n"; // prompt
  cin >> integer1; // read an integer

  cout << "Enter second integer\n"; // prompt
  cin >> integer2; // read an integer

  sum = integer1 + integer2; //assignment result to sum
```

```
  cout << "Sum is " << sum << endl; // print sum

  return 0; // indicate that program ended successfully

} // end function main
```

همانطور که در این برنامه نیز می بینید، تعدادی از دستورات برنامه قبلی تکرار شده اند. در اینجا به بررسی و توضیح دستورات جدید می پردازیم:

```
  int integer1; // first number to be input by user
  int integer2; // second number to be input by user
  int sum; // variable in which sum will be stored
```

سه دستور فوق وظیفه تخصیص حافظه به سه متغیر integer1 و integer2 و sum از نوع عدد صحیح را دارند. انواع داده در مبحث مفاهیم حافظه و انواع داده توضیح داده شده اند. در ضمن به جای استفاده از سه دستور فوق می توانستیم از دستور زیر نیز استفاده کنیم:

```
int integer1, integer2, sum;
```

نکته :

- بعضی از برنامه نویسان ترجیح می دهند که هر متغیر را در یک خط تعریف کنند و توضیحات لازم را در جلوی آن بنویسند.
- متغیر را می توان در هر جایی از برنامه تعریف کرد، ولی حتماً این کار باید قبل از اولین استفاده از متغیر صورت گیرد، به عنوان مثال برنامه فوق را می توان به صورت زیر نوشت:

```
•   #include <iostream.h>
•
•   int main()
•   {
•     cout << "Enter first integer\n";
•
•     int integer1;
•     cin >> integer1;
•
•     cout << "Enter second integer\n";
•
•     int integer2;
•     cin >> integer2;
•
```

```
    int sum;
    sum = integer1 + integer2;

    cout << "Sum is " << sum << endl;

    return 0;
}
```

- اگر تعریف متغیر را در بین دستورات اجرایی برنامه انجام می دهید، یک خط خالی قبل از آن بگذارید تا تعریف متغیر مشخص باشد. اینکار به وضوح برنامه کمک می کند.
- اگر تعریف متغیرها را در ابتدای برنامه انجام می دهید، یک خط خالی بعد از آنها بگذارید تا از دستورات اجرایی جدا شوند. اینکار نیز به وضوح برنامه و سهولت اشکال زدایی کمک می کند.

```
cout << "Enter first integer \n";
```

دستور فوق رشته Enter first integer را بر روی صفحه نمایش نشان می دهد و به ابتدای سطر جدید می رود.

```
cin >> integer1;
```

دستور فوق با وارد کردن هر عدد و فشردن کلید Enter عدد وارد شده را در متغیر integer1 قرار می دهد.

```
cout << "Enter second integer \n";
cin >> integer2;
```

دو دستور فوق نیز ابتدا عبارت integer Enter second را بر روی صفحه نمایش چاپ کرده و سپس در خط بعد عدد وارد شده از صفحه کلید را پس از فشردن کلید Enter در متغیر integer2 قرار می دهد.

```
sum = integer1 + integer2;
```

دستور فوق حاصل جمع متغیرهای integer1 و integer2 را محاسبه و نتیجه را توسط عملگر انتساب (=) در متغیر sum قرار می دهد.

```
cout << "Sum is " << sum << endl;
```

و در نهایت دستور فوق باعث نمایش حاصل جمع بر روی صفحه نمایش می شود.

ساختار انتخاب if

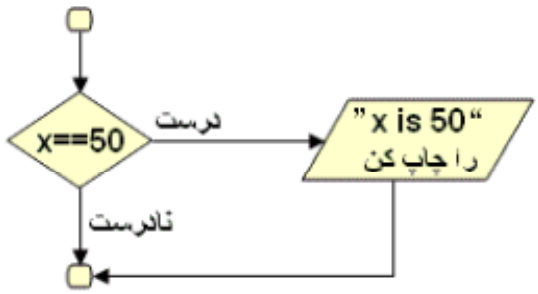
در برنامه نویسی مواردی پیش می آید که بخواهیم دستور یا دستوراتی، هنگامی که شرط خاصی برقرار است، توسط برنامه به اجرا در آید. این مورد در زندگی روزمره نیز دیده می شود؛ به عنوان مثال " اگر فردا باران نیاید، من به کوه خواهم رفت." شرط مورد نظر نیامدن باران است و عملی که قرار است انجام شود رفتن به کوه می باشد. شیوه پیاده سازی ساختار انتخاب **if** به صورت زیر می باشد:

```
if  ( شرط مورد نظر )
        دستور مورد نظر  ;
```

به مثال زیر توجه کنید:

```
if (x == 50)
    cout << "x is 50";
```

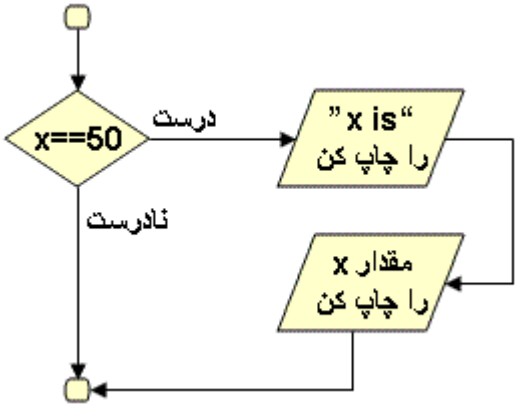
اگر از دستور فوق در برنامه استفاده کنیم، اگر مقدار متغیر **x** قبل از رسیدن به شرط فوق برابر **۵۰** باشد عبارت " **x is 50**" بر روی صفحه نمایش ظاهر خواهد شد وگرنه دستور **cout << "x is 50" ;** نادیده گرفته می شود و برنامه خط بعدی را اجرا می کند.



توجه داشته باشید که شرط مورد استفاده در دستور **if** هر عبارت منطقی می تواند باشد. در مبحث عبارات منطقی ، اینگونه عبارات و شیوه کاربرد آنها را به طور کامل بررسی کردیم. اگر بخواهیم هنگامی که شرط برقرار می شود، بیش از یک دستور اجرا شود، باید دستورات مورد نظر را با علامت { } دسته بندی کنیم، به مثال زیر توجه کنید:

```
if  ( x==50 )
  {
    cout << "x is ";
    cout << x;
  }
```

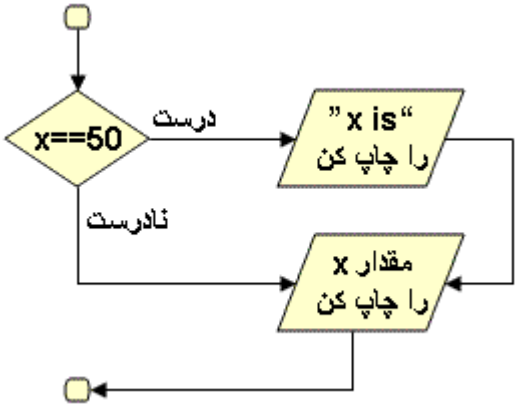
قطعه کد فوق هنگامی که مقدار **x** عدد ۵۰ باشد، عبارت "**x is 50**" را در صفحه نمایش چاپ می کند.



ولی در دستورات زیر:

```
if ( x == 50)
     cout << "x is ";
     cout << x ;
```

خط آخر برنامه به هر جهت اجرا می شود. به عنوان مثال اگر فرض کنیم **x** برابر ۵۰ است برنامه به درستی عبارت " **x is 50**" را چاپ می کند، اما اگر مثلاً **x** برابر ۲۰ باشد عدد ۲۰ بر روی صفحه نمایش ظاهر خواهد شد. چون عبارت **cout;** <<**x** جز دستورات **if** قرار ندارد و یک دستور مجزا می باشد.



مورد اخیر که توضیح داده شد یکی از مواردی است که بعضی از برنامه نویسان به اشتباه مرتکب آن می شوند. پس در هنگام نوشتن برنامه های خود به دسته بندی دستورات دقت کنید.

ساختار انتخاب if/else

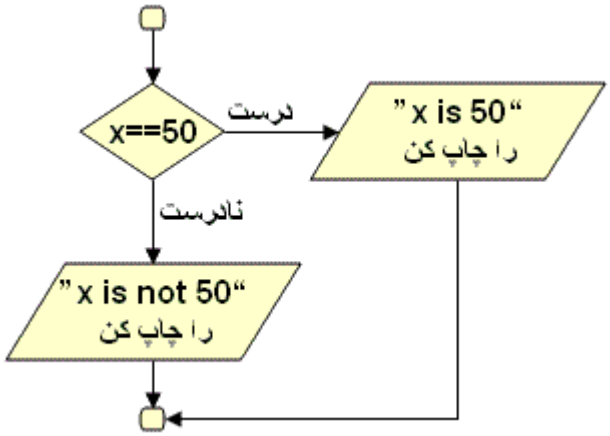
گاهی اوقات نیاز داریم که در صورت برقرار بودن شرط خاصی یک سری دستورات اجرا و در صورت برقرار نبودن شرط دسته ای دیگر از دستورات اجرا گردند. به عنوان مثال اگر فردا باران بیاید من به کوه نمی روم در غیر این صورت من به کوه خواهم رفت؛ زبان C++ برای پیاده سازی چنین ساختاری شیوه زیر را در اختیار ما قرار داده است.

```
if  (شرط مورد نظر)
        دستور۱ ;
else
        دستور۲ ;
```

اگر شرط برقرار باشد دستور۱ اجرا می گردد و در غیر این صورت دستور۲ اجرا می شود. به مثال زیر توجه کنید:

```
if  ( x = = 50 )
      cout << "x is 50";
else
      cout << "x is not 50";
```

اگر مقدار متغیر قبل از رسیدن به شرط فوق برابر ۵۰ باشد عبارت "**x is 50**" برروی صفحه نمایش چاپ می شود در غیر این صورت عبارت "**۵۰ x is not**" چاپ می شود.



بیاد داشته باشید اگر می خواهید از بیش از یک دستور استفاده کنید، حتماً آنها را با { } دسته بندی نمایید. به عنوان مثال:

```
if  ( x > 50 )
    {
      cout << x;
      cout << "is greater than 50";
```

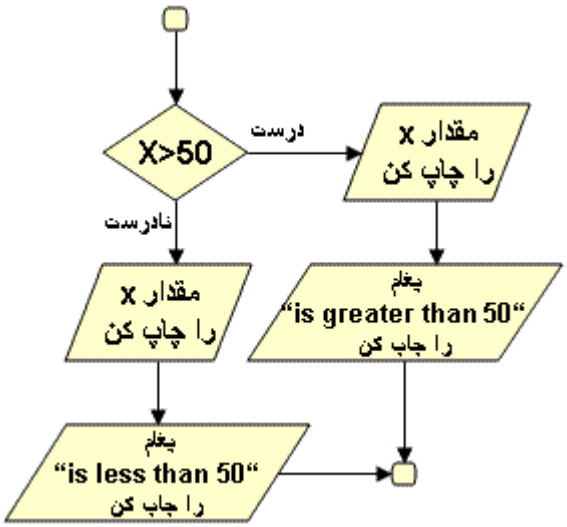
```
    }
else
    {
      cout << x;
      cout << "is less than 50";
    }
```

اگر متغیر **x** حاوی عدد **۱۰۰** باشد خروجی به صورت زیر می باشد:

```
100 is greater than 50
```

و اگر متغیر **x** عدد **۱۰** باشد خروجی به صورت زیر است:

```
10 is less than 50
```



از ساختارهای **if/else** های تو در تو می توان برای بررسی حالتهای چندگانه استفاده کرد. برنامه زیر در همین رابطه است:

```
#include <iostream.h>
int main( )
{
  int x;
  cout << "Please enter a number:";
  cin >> x;

  if ( x > 0 )
     cout << x << "is positive.";
```