آموزش برنامه نویسی با C++

هر برنامه C++ شامل یک یا چند تابع است. که در کد های برنامه در این توابع گنجانده شده اند. یک برنامه C++ حداقل شامل یک تابع به نام main است. آخر هر سطر فرمان، یک سمی کالن(; = نقطه ویرگول) قرار می گیرد.

اولین سطر برنامه

اولین سطر برنامه می تواند توضیح در مورد برنامه باشد. توضیح ها به دو صورت نوشته می شوند:

- توضیح شامل یک یا چند سطر است : برای اضافه کردن توضیح چند سطری از علائم */ و * استفاده می شود. مانند:

```
/*
  EX01.CPP
  A Simple Program Example
*/
```

- توضیح شامل یک سطر است : برای اضافه کردن یک توضیح جلوی یک دستور می توان از این روش استفاده کرد. مانند:

```
cout << endl;     // Start output on a new line
```

سطر دوم برنامه

سطر دوم برنامه (که می تواند اولین سطر نیز باشد.) دستور include است که مواردی را که برنامه نیاز دارد، فراخوانی می کند. این موارد را اصطلاحا Header می گویند. برای مثال برای گرفتن ورودی و خروجی (دستورات cin و cout) به فایل هدر iostream نیاز داریم که آن را به صورت زیر به برنامه اضافه می کنیم:

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

پسوند ".h"، پسوند فایل های هدر است.

سطر سوم برنامه

پس از فراخوانی موارد مورد نیاز برنامه، نوبت به تعریف کردن عناصر می رسد. مقادیر در C++ به شش دسته زیر تقسیم می شوند:

- **Long:** برای مقادیر عددی بزرگ استفاده می شود.
- **Int:** برای مقادیر عددی صحیح استفاده می شود.
- **Short:** برای مقادیر عددی صحیح کوچک استفاده می شود.
- ***Char:** برای یک عبارت شامل چند کاراکتر استفاده می شود.(رشته)
- **Char:** برای یک عبارت شامل یک کاراکتر استفاده می شود.(کاراکتر)
- **Float:** برای مقادیر اعشاری استفاده می شود.

تمامی مقادیری در بالا ذکر شده اند، مقادیر صحیح اند به جز Float که مقادیر اعشاری دریافت می کند.
تعریف یک تابع به صورت زیر انجام می شود:

Data-type Data-name;

که در آن Data-type نوع داده و Data-name نام داده است.

برای مثال این دستور یک متغیر به نام apple و از نوع Int ایجاد می کند :

```
Int apple;
```

برای تعریف چند تابع که همگی از یک نوع هستند می توان مانند زیر عمل کرد :

```
Int apple, orange, cherry;
```

اگر بخواهیم یک متغیر با مقدار اولیه ایجاد کنیم از دستور زیر استفاده می کنیم :

Data-type Data-name = Value;

برای تعریف ثابت ها از دستور Const استفاده می کنیم :

Const Data-type Data-name = Value;

برای مثال دستور زیر یک ثابت به نام kilo از نوع Int را برابر با ۱۰۰۰ قرار می دهد :

```
Const int kilo = 1000;
```

قواعد نگارشی! (چگونه خوب برنامه بنویسیم؟)
نام ثابت ها را باحروف بزرگ تایپ کنید و متغیر ها را با حروف کوچک تایپ کنید. اگر شناسه(نام) شامل چند کلمه است، آن ها را با علامت _ از یک دیگر جدا کنید.
به این ترتیب خواننده برنامه قادر است ثابت ها را به راحتی از متغیر ها تشخیص دهد.

نکته : برای نام گذاری نمی توانید از کلمات رزرو شده استفاده کنید. کلمات رزرو شده کلماتی هستند که در C++ معنی خاص دارند. برای مثال کلمه char که نوع داده کاراکتر را مشخص می کند.

پس از اینکه یک متغیر را تعریف کردیم می توانیم به آن مقدار بدهیم:

Variable = Expression;

Variable همان نام متغیر و Expression مقداری است که قرار است به آن تعلق گیرد.

مثال :

```
Int apple;
apple = 2;
```

عملگر ها در C++

+	مثبت – جمع
-	منفی – تفریق
*	ضرب
/	تقسیم(تقسیم اعشاری با داده و باقیمانده با اعشار – تقسیم صحیح با داده صحیح و با قیمانده بدون اعشار)
%	مانده حاصل از تقسیم صحیح(هم نهشتی)
++	دستور افزایش یک واحد
--	دستور کاهش یک واحد
&&	AND منطقی
||	OR منطقی
!	NOT منطقی
=!	نا مساوی
==	مساوی
<	کوچکتر
>	بزرگتر
=<	بزرگتر و مساوی
=>	کوچکتر و مساوی

نکته : برای مقدار دهی به یک متغیر از = استفاده می شود و برای سنجیدن مساوی بودن یا نبودن دو عبارت از == استفاده می شود.

مثال :

```
apple++;
++apple;
```

دو دستور فوق با دستور ذیل برابر هستند :

```
apple = apple + 1;
```

دستور زیر اگر a=2 و b=3 ، مقدار صحیح(True) را برمی گرداند :

```
a==2 && b==3;
```

دستور زیر اشتباه است . زیرا a را برابر ۲ قرار می دهد و b را برابر ۳ و در نتیجه جواب شرط همیشه درست است :

```
a=2 && b=3;
```

دستور زیر نیز اشتباه است ولی برنامه نمی تواند آن را تشخیص دهد. چون از نظر دستوری مشکلی ندارد ولی معنای آن اشتباه است :

```
a==2 || 3;
```

شکل صحیح این دستور به این صورت است :

```
a==2 || a==3;
```

هنگامی که از شکل نادرست استفاده کنیم، کامپیوتر ۳ را یک طرف شرط می گیرد که همیشه درست است و چه a برابر با ۲ باشد و یا نباشد ، عبارت مقدار True را برمی گرداند.

اولویت عملگر ها

() **!** *** / %** **+ -** **< <= > >=** **== !=** **&&** **||** **=**	بالاترین اولویت ↓ پایین ترین الویت

فراخوانی توابع

توابع به سه دسته کلی :

- **با مقدار برگشتی**
- **کتابخانه ای**
- **تهی(بدون مقدار برگشتی)**

تقسیم می شوند.

تعریف یک تابع با مقدار برگشتی شبیه تعریف متغیرها است :

Int Function-name(Parameter-type);

تابع تهی تابعی است که هیچ مقداری را بازنمی گرداند. فقط یک عمل را انجام می دهد و پایان می پذیرد. برای تعریف تابع تهی به جای نوع داده از کلمه void استفاده می کنیم :

Void Function-name(Parameter-type);

برای فراخوانی توابع از دستور زیر استفاده می کنیم :

Function-name(Parameter-list);

- مثال اول :

```
Int cube(int n);
Cube(2)
Int cube(int n) {
Return n * n * n
}
```

سطر اول یک تابع با نوع داده int را ایجاد می کند.
سطر دوم تابع را فراخوانی می کند.
سطر سوم تا آخر نیز عملی که تابع انجام می دهد را تعریف می کند.(که در اینجا عدد را به توان ۳ می رساند.)

- مثال دوم:

```
Void salary(float, float, float);
salary (a, b, c);
Void salary (float a, float b, float c) {
//code
}
```

توابع کتابخانه ای

توابعی مانند جذر گیری، محاسبه سینوس و ... نیازمند و ضروری هستند. همچنین نوشتن برنامه هایی برای محاسبه این گونه محاسبات توسط هر برنامه نویس نتیجه ای جز اتلاف وقت ندارد. بنابر این کتابخانه های استانداردی با مجموعه ای بزرگ از توابع، نوشته شده اند. این کتابخانه ها با استفاده از راهنمای #include به برنامه اضافه می شوند. پس از آن این گونه توابع همانند توابع با مقدار برگشتی فراخوانی می شود.

ورودی و خروجی

برای گرفتن ورودی و نوشتن متن در صفحه، نیاز به دستورات ورودی و خروجی و فایل هدر iostream داریم.

- cin : دستور گرفتن ورودی
- cout : دستور خروجی

Cin >> variable >> variable >> ... ;
Cout << "Expression" or Variable << "Expression" or Variable << ... ;

مثال :

```
Cout << "How many apple do you want?" << endl;
cin >> apple;
cout <<  endl << "You want " << apple << " apples." << endl;
```

دستور endl (End Line خوانده می شود.) یک سطر جدید ایجاد می کند. مانند کلید Enter در صفحه کلید. کاراکتر خط جدید (/n)نیز همین عمل را انجام می دهد. این کاراکتر از دو علامت تشکیل شده است ولی یک کاراکتر به حساب می آید.
در سطر سوم، از متغیر apple استفاده کردیم. وقتی که از یک متغیر و یا تابع در خروجی استفاده می کنیم، مقداری که آن متغیر یا تابع بر می گرداند، چاپ می شود. برای مثال اگر کاربر پس از دستور cin >> apple; عدد ۳ را وارد کند. در سطر بعدی نوشته می شود : .You want 3 apples

مثال :

```
Cout << "Hello/n";
```

دستور فوق با دستور زیر یکسان است :

```
Cout << "Hello" << endl;
```

نکته : برای اینکه جهت علامت های >> و << را اشتباه نکنید، به جهت انجام عمل دقت کنید. برای مثال دستور cin مقداری را به apple نسبت می دهد. پس جهت به طرف apple است(<<).

خواندن کاراکتر ها با تابع get

برای گرفتن ورودی می تواند از تابع get استفاده نمود. فرض کنید ch1 و ch2 دو نوع متغیر از نوع کاراکتر هستند و دستور زیر را در برنامه نوشتیم :

```
Cin >> ch1 >> ch2;
```

اگر کاربر عبارت زیر را بنویسد :

```
R 1
```

عملگر دریافت کننده، 'R' را در ch1 ذخیره می کند. سپس از فاصله (فضای خالی = blank) صرف نظر می کند و سپس کاراکتر '1' را در ch2 ذخیر می کند.

نکته : توجه کنید که کاراکتر '1' با عدد صحیح 1 تفاوت دارد و نحوه ذخیر آن ها در حافظه نیز متفاوت است.

حال اگر بخواهیم که کاراکتر فضای خالی را نیز در یک متغیر ذخیر کنیم مجبوریم از تابع get استفاده کنیم.

```
Cin.get(ch1);
Cin.get(ch2);
Cin.get(ch3);
```

اگر ورودی همان ورودی قبلی باشد، کاراکتر 'R' را در ch1 ذخیر می کند. سپس کاراکتر ' ' را در ch2 قرار می دهد. در انتها کاراکتر '1' در ch3 قرار می گیرد.

صرف نظر کردن از کاراکتر ها با تابع ignore

اگر بخواهیم از تعدادی کاراکتر صرف نظر کنیم و یا تا رسیدن به کاراکتر خاصی صبر کنیم، از این تابع استفاده می کنیم.

مثال :

```
Cin.ignore (200, '/n');
```

دستور فوق از ۲۰۰ کاراکتر صرف نظر می کند یا تا رسیدن به کاراکتر خط جدید از کاراکتر های ورودی صرف نظر می کند. (صبر می کند تا یکی از شرایط انجام شود.)

تصمیم گیری و حلقه

شرط

هنگامی که می خواهیم در صورت درست بودن یک شرط کاری انجام شود از دستور **if** استفاده می کنیم. با عبارت **else** هم می توان در صورت درست نبودن شرط کاری را انجام داد.

```
If (condition) {
//code1
}
Else {
//code2
}
```

در عبارت فوق condition همان عبارت شرط است. اگر شرط درست باشد code1 انجام می شود و اگر شرط اشتباه باشد، code2 انجام می شود. اگر دستور یک خط است می توان {} را حذف نمود.

عملگر شرطی

این عملگر کار If را انجام می دهد. ولی در مواقعی که نمی توان از If استفاده کرد (مانند cout)، استفاده می شود.

```
Condition ? True-expression : False-expression
```

در عبارت فوق True-expression عبارتی است که وقتی شرط درست باشد، انجام می شود و False-expression عبارتی است که وقتی شرط اشتباه باشد، انجام می شود. عبارت فوق با عبارت زیر برابر است :

```
If (condition)
//True-expression;
Else
//False-expression;
```

If تودرتو

هنگامی که می خواهیم در صورت اشتباه بودن شرط، یک عبارت شرطی دیگر امتحان شود، از If تو در تو استفاده می کنیم.

مثال :

```
If (day == 1)
Cout << "sat" << endl;
Else
If (day == 2)
Cout << "sun" << endl;
Else
If (day == 3)
Cout << "Mon" << endl;
Else
Cout << "another day!" << endl;
```

حلقه

اگر بخواهیم تا زمانی که یک شرط درست است، کاری انجام شود، از حلقه استفاده می کنیم.

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

حلقه For

```
For (initializing-expression; test-expression; increment-expression)
//code
```

در عبارت فوق initializing-expression برابر با مقدار اولیه، test-expression برابر با آخرین مقدار و increment-expression برابر با قدر نسبت است.

مثال :

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