Basic input/output - I/O - دستورات ورودی خروجی در برنامه نویسی C++

هدف از ساخت کامپیوتر و ایجاد برنامه نویسی دادن اطلاعات به ماشین و دریافت جواب بوده که به این روال، جریان ورودی خروجی گفته میشود . برای استفاده از این جریانات ما حداقل به ۲ فایل کتابخانه ای در C++نیاز داریم و دستوراتی برای استفاده از این کتابخانه ها که با ارائه یک مثال، استفاده از این دستورات را نشان می دهم.

```
#include <iostream.h>
#include <conio.h>
int main()
{
   int a;          //a=?
   cin >> a;       //a=value of user input
   cout << "a:";    //Prints a: on screen
   cout << a;      //Prints value of a
   getch();
   return 0;
}
```

```
a:120
```

برای استفاده از کلمات کلیدی cin و cout لازمه که از کتابخانه iostream.h در برنامه خود استفاده کنیم.

دستور cout باعث چاپ اطلاعات در مانیتور می شود به این صورت که اگر بعد از این دستور، عبارت مورد نظر رو تو " " قرار بدیم عینا همون عبارت تو مانیتور نشان داده می شود و معمولا جایی که کاربر قصد نشان دادن عین یک عبارت را در خروجی دارد مورد استفاده قرار می کیرد و اگه بخواهیم مقدار یک متغیر یا ثابتی را نمایش بدهیم باید نام را نوشته و از " " استفاده نکنیم.

دستور cin باعث می شود که از کاربر یا برنامه دیگه ای یک مقدار یا متنی را بگیریم . در واقع کامپایلر با رسیدن به این دستور منتظر ورود اطلاعات از طرف کاربر شده تا زمانی که کاربر Enter را فشار بده و مقدار ورودی کاربر را در متغیری که در جلوش اعلان شده بریزه و همونطور که قبلا اشاره شد موجب مقدار دهی به متغیر مربوطه میشود.

⭐ در برنامه بالا تابعی بنام getch()هست که برای استفاده از اون باید از کتابخانه conio.h استفاده کنیم که باعث میشود بعد از اجرای برنامه و رسیدن کامپایلر به این خط، برنامه متوقف شده و منتظر این باشه که کاربر یک کلید از کیبورد را فشار دهد . با این دستور می توانیم خروجی برنامه را ببینیم چون اگر این تابع نباشد در کسری از ثانیه برنامه اجرا و بلافاصله بسته میشود و دیگر ما قادر به دیدن خروجی نخواهیم بود . (امتحان کنید!)

در مثال بالا فرض بر اینه که کاربر مقدار ۱۲۰ را وارد کرده است.

گاهی اوقات لازمه که ما چند مقدار را با استفاده از دستور cin به برنامه بدهیم، در اینصورت به ازاء هر مقدار از یک >>استفاده می کنیم و همینطور برای دستور cout ازیک . <<

```
int a, b;          //a=?, b=?
cin >> a >> b;
cout << a << b;
```

همونطور که در برنامه های قبلی دیدیم با نوشتن چند دستور چاپ، تمامی اطلاعات در یک خط نوشته شد . در خروجی برای اینکه از خط کنونی به خط بعدی برویم به یکی از صورتهای زیر عمل می کنیم:

```
cout << a << endl;
```

عبارت endl به کامپایلر پایان خط جاری را نشان می دهد و در صورت وجود خروجی دیگه ای در برنامه، نمایش خروجی را از خط بعدی ادامه میدهد.

```
cout << a << "\n";
```

عبارت \nنیز مثل endl خروجی را به خط بعدی هدایت می کند.

```
1  #include <iostream.h>
2  #include <conio.h>
3  int main()
4  {
5     int a, b;                        //a=?,b=?
6     cout << "Please inter value of a:";
7     cin >> a;                         //a=value of user input
8     cout << "Please inter value of b:";
      cin >> b;                         //b=value of user input
```

```
9       cout << "a:" << a << "\n";              //Prints a:(value of a) on screen and go to
10  next line
11      cout << "b:" << b;                  //Prints b:(value of a) on screen
12      return 0;
13  }
```

```
Please inter value of a:
Please inter value of b:
a:120
b:87
```

باید بگم فرض بر این است که مقادیر ۱۲۰ و ۸۷ توسط کاربر وارد شده است . نکته ای که اینجا قابل توجه است اینه که دستور cin خودش باعث میشود که کامپایلر با گرفتن مقدار بطور خودکار به خط بعدی برود و دیگه اینجا نیازی به قید \n یا endl نیست.

عبارات محاسباتی و تقدم عملگرها در برنامه نویسی : C++ (Precedence of operators)

همانطور که قبلا اشاره شد عبارات محاسباتی شامل عملیات یک یا چند عملگر بر روی یک یا چند عملوند هستند. لذا به مبحث تقدم عملگرها در برنامه نویسی C++می پردازیم تا بدانیم که در برخورد با یک عبارت محاسباتی طولانی به چه صورت باید رفتار کنیم.

```
a = b+10*(9%4);
a = b+10*9%4;
```

در بالا دو عبارت محاسباتی شبیه به هم اما با ساختاری متفاوت را مشاهده می کنیم. این دو عبارت جوابی متفاوت نسبت به یکدیگر دارند و تنها دلیل این تفاوت وجود یا عدم وجود پرانتز در هر کدام از این دو می باشد.

بطور کلی در عبارات محاسباتی قوانینی وجود دارد که اولویت عملگری را بر سایر عملگرها مشخص می کند. یعنی در برخورد با یک عبارت محاسباتی، عملگرهای با تقدم بالاتر، زودتر محاسبه شده تا به آخرین عملگر در عبارت برسیم.

جدول زیر تقدم عملگرها را با اولویت از بالا به پایین به ما نمایش میدهد.

1	()
2	! ~ ++ -- sizeof
3	* / %
4	+ -
5	<< >>
6	< <= > >=
7	== !=
8	&
9	^
10	|
11	&&
12	||
13	?
14	= += -= *= /= %=
15	,

همینطور که در جدول بالا مشخص شد بالاترین تقدم عملگرها را پرانتز و پایین ترین تقدم را کاما دارا می باشند.

در جدول بالا ردیف هایی با چند عملگر وجود دارند. این بدین معنی است که از تقدم یکسانی برابر هستند، بنابر این اگر در عبارات محاسباتی به چند عملگر با تقدم یکسان برخورد کردیم تقدم بالاتر به عملگری میرسد که در سمت چپ دیگر عملگرها قرار دارد.

برای درک مطلب به بررسی مثالی می پردازیم:

```
a = 20-3*4+19%(3*(2+1));
20-3*4+19%(3*3)
```

ابتدا عبارت داخل پرانتز به دلیل بالاترین تقدم بررسی می شود. اینجا ما دو پرانتز تودرتو داریم پس از پرانتز داخلی شروع می کنیم.

```
20-3*4+19%9
```

با محاسبه مقدار اولین پرانتز، دوباره عبارت داخل آنرا بدلیل وجود پرانتز دوم و تقدم آن نسبت به دیگر عملگرها محاسبه می کنیم.

```
20-12+19%9
```

حالا از بین عملگرهای موجود * و % از تقدم بالاتری برخوردارند. بدلیل یکسان بودن تقدم این دو از چپ شروع کرده و با رسیدن به هر کدام از این دو عملگر مقدار عبارت را محاسبه میکنیم که در این مثال ابتدا * محاسبه می شود.

```
20-12+1
```

سپس نوبت به % میرسد.

```
8+1
```

اکنون عملگرهای + و – در عبارت باقی می مانند که بدلیل یکسانی تقدم اولین عملگر از چپ یعنی – ابتدا محاسبه می گردد.

```
9
```

و در پایان عملگر + محاسبه شده که در نهایت به جواب ۹ می رسیم.