داده ها

همانطور که در بخش های قبل بیان کردیم؛ یک متغییر مکانی از حافظه است که یک مقدار می تواند در آن ذخیره شود. هر متغیر پیش از آنکه دریک برنامه استفاده گردد ابتدا باید اعلان گردد. اعلان یک متغیر بدین معنی است که نوع آن متغییر را مشخص شود.

نوع داده	توضیح	اندازه(بیت)	دامنه
char	کاراکتر	۸	-۱۲۸ تا ۱۲۷
int	عدد صحیح	۱۶	-32768 تا ۳۲۷۶۷
float	عدد اعشاری	۳۲	3.4e-38 تا 3.4e+38
double	عدد اعشاری با دقت مضاعف	۶۴	1.7e-308 تا 1.7e308

تعریف متغیر

برای تعریف متغیرها به شکل زیر عمل می کنیم:

<type> <variable-list> ;

که type یکی از نوع داده های گفته شده و variable-list لیستی از متغیرها است که با کاما از یکدیگر جدا شده اند. بعنوان مثال :

```
int   number;
float  average;
double   a, b, c ;
```

علاوه براین می توان در هنگام تعریف متغیر به آن مقدار اولیه نیز داد. مثال :

```
int  d = 0 ;
```

عملگر ها

عملگر، نمادی است که به کامپایلر می گوید تا عملیات محاسباتی یا منطقی خاصی را برروی یک یا چند عملوند، انجام دهد. به عملگرهایی که فقط یک عملوند دارند، عملگر یکانی می گوییم و همواره عملگر در سمت چپ عملوند قرار می گیرد(مانند عدد -125). اما عملگرهایی که برروی دو عملوند اثر می کنند را عملگر دودویی نامیده و عملگر را بین دو عملوند قرار می دهیم (مانند 23+86). هر ترکیب درستی از عملگرها و عملوندها را یک عبارت می نامیم.

عملگرها به چند دسته اصلی تقسیم می گردند که آنها را به ترتیب بررسی می کنیم.

عملگر محاسباتی

عمل در C++	عملگر در C++	نوع عمل	عمل جبری	مثال
منفی کردن	-	یکانی	-b	-4
جمع	+	دودویی	a + b	1+3
تفریق	-	دودویی	a - b	1-3
ضرب	*	دودویی	a * b	1*3
تقسیم	/	دودویی	a / b	1/3
باقی مانده	%	دودویی	a % b	1%3

عملگر محاسباتی

عملگرهای فوق همه اعمال متداول ریاضی هستند که برروی همه انواع داده های C عمل می کنند بجز عملگر % که فقط برروی نوع داده های صحیح عمل می کند و برروی داده های اعشاری تعریف نشده است. اما نکته مهمی که باید به آن اشاره کرد، نحوه کار عملگر تقسیم برروی نوع داده های مختلف است. درصورتیکه هردو عملوند صحیحی باشند، تقسیم بصورت صحیح بر صحیح انجام خواهد شد. اما اگر یکی یا هر دو عملوند اعشاری باشند، تقسیم بصورت اعشاری انجام خواهد پذیرفت.

$c * (a + d)$

$(a / (w - x))$ پرانتز تو در تو

نکته بسار مهم دیگری که باید در هنگام کار با عبارات محاسباتی به آن توجه کرد، اولویت عملگرها است. یعنی در عبارتی که شامل چندین عملگر است، کدامیک در ابتدا اعمال خواهد گردید. در جدول زیر اولویت عملگرهای محاسباتی از بالا به پایین نشان داده ایم.

عملگر محاسباتی

توضیحات	عمل	عملگر در C++
اولویت اول	عملگر یکانی	-
اولویت دوم	پرانتز	( )
اگر عبارت شامل پرانتزهای تو در تو باشد ابتدا داخلی ترین پرانتز محاسبه می شود		
اولویت سوم	ضرب، تقسیم و باقی مانده	*, /, %
اگر این ها باهم در یک عبارت محاسباتی قرار بگیرند از چپ به راست ارزیابی می شوند		
اولویت آخر	جمع و تفریق	+, -
اگر این ها باهم در یک عبارت محاسباتی قرار بگیرند از چپ به راست ارزیابی می شوند		

عملگر محاسباتی

چنانچه اولویت دو عملگر یکسان باشد، این عملگرها از چپ به راست محاسبه خواهند شد. چنانچه بخواهیم یک عمل با اولویت پایین زودتر انجام شود، باید از پرانتز استفاده کنیم. در مورد پرانتزهای متداخل، ابتدا پرانتز داخلی محاسبه می شود؛ اما در مورد پرانتزهای هم سطح، ابتدا پرانتز سمت چپتر محاسبه می گردد.

```
a=5;
b = c + 2 * d;
```

این عملگر باعث می شود که عبارت سمت راست در عبارت سمت چپ قرار گیرد. توجه کنید که مقدار سمت چپ باید عبارتی باشد که بتوان به آن یک مقدار را نسبت داد(مانند یک متغیر) بنابراین یک ثابت نمی تواند در سمت چپ قرار گیرد. نکته دیگر اینکه اولویت عملگر = از عملگرهای ریاضی کمتر است و درنتیجه ابتدا آن عملیات انجام شده و در پایان حاصل در عبارت سمت

عملگر محاسباتی

چپ ریخته می شود. لازم به ذکر است که در هنگام انتساب، درصورت لزوم نوع عبارت سمت راست به نوع عبارت سمت چپ تبدیل می شود. مثال:

```
int  a;
a = 2.5 * 5.0;
```

که دراینصورت عدد ۱۲ در a ذخیره خواهد شد.

شرکت پذیری این عملگر از راست به چپ می باشد، بدین معنا که چنانچه چندین عملگر نسبت دهی داشته باشیم، این عملگرها از راست به چپ محاسبه می شوند. مثلا پس از اجرای دستور زیر :

```
a = b = c = 10;
```

عملگر انتسابی

عملگر	نام	اولویت	مثال	توضیحات
++	پیش افزایش	راست به چپ	++a	ابتدا a را افزایش داده و سپس از آن در عبارت استفاده می کند.
++	پس افزایش	چپ به راست	a++	ابتدا از مقدار فعلی a در عبارت موردنظر استفاده کن و سپس آن را افزایش بده
--	پیش کاهش	راست به چپ	--a	ابتدا a را کاهش داده و سپس از آن در عبارت استفاده می کند.
--	پس کاهش	چپ به راست	a--	ابتدا از مقدار فعلی a در عبارت موردنظر استفاده کن و سپس آن را کاهش بده

عملگر انتسابی

```
#include<iostream>
using namespace std;

void main()
{
        int a=5;
        int b,c,d,e,g;
        b=a++;
        cout<<"a is: "<<a<<"\n";
        cout<<"b is: "<<b<<"\n\n";
        c=++a;
        cout<<"a is: "<<a<<"\n";
        cout<<"c is: "<<c<<"\n\n";
        d=a--;
        cout<<"a is: "<<a<<"\n";
        cout<<"d is: "<<d<<"\n\n";
        e=--a;
        cout<<"a is: "<<a<<"\n";
        cout<<"e is: "<<e<<"\n";
}
```

```
a is: 6
b is: 5

a is: 7
c is: 7


a is: 6
d is: 7


a is: 5
e is: 5
```

عملگر انتسابی

int a=12;

عملگر	مثال	توسعه یافته عملگر	مقدار متغییر پس از اعمال عملگر
		49	
+=	a+=4	a = a + 4	a=16
-=	a-=5	a = a - 5	a=7
*=	a*=3	a = a * 3	a=36
/=	a/=4	a = a / 4	a=3
%=	a%5	a = a %5	a=2

عملگر مقایسه ای

عملگر	مفهوم عملگر	مثال
>	بزرگتر (>)	a > b
<	کوچکتر (<)	a < b
>=	بزرگتر یا مساوی (≥)	a >= b
<=	کوچکتر یا مساوی (≤)	a <= b
==	مساوی (=)	a == b
!=	نامساوی (≠)	a != b

عملگر مقایسه ای

نکته مهمی که باید به آن دقت کرد عملگر مساوی (= =) است، چرا که یک اشتباه بسیار متداول برنامه نویسان C استفاده اشتباه از عملگر انتساب (=) بجای عملگر تساوی (= =) است که باعث ایجاد خطا در برنامه می شود.

عامل دیگری که باید در بکارگیری عملگرهای مقایسه ای دقت کرد این است که عملگر =< و => را نباید بصورت <= و >= بکار برد زیرا یک خطای نحوی است.

اولویت این عملگرها از بالا به پایین بشرح زیر است:

۱- عملگرهای <، > ، =< و =>

۲- عملگرهای == و =!

عملگر منطقی

عملگر	مفهوم عملگر	نحوه کار
&&	منطقی AND	اگر هر دو عملوند درست باشند, درست و در غیر اینصورت نادرست باز می گرداند.
||	منطقی OR	اگر هر دو عملوند نادرست باشند, نادرست و در غیر اینصورت درست باز می گرداند.
!	منطقی NOT	اگر عملوند درست باشد، نادرست و اگر نادرست باشد، درست برمی گرداند.

۱- عملگر !　　۲- عملگر &&　　۳- عملگر ||

عملگر شرطی

گاهی لازم است که ابتدا یک شرط بررسی شده و سپس برمبنای نتیجه (درست یا نادرست بودن آن) یکی از دو عبارت ممکن بازگردانده شود. گرچه معمولا برای اینکار از دستور if (که در فصل بعدی بحث خواهد شد) استفاده می شود، اما یک عملگر ۳تایی (با ۳ عملوند) نیز برای آن وجود دارد. شکل کلی این عملگر بصورت زیر است:

(شرط) ? عبارت۱ : عبارت۲

```
a = (k<10) ? 100 : 50;
```

که این عبارت معادل دستور زیر است:

```
if (k<10) a=100;
else a=50;
```

دریافت داده از کاربر و نمایش اطلاعات

```
// Comparing integers using if statements, relational operators
// and equality operators.
#include <iostream> // allows program to perform input and output

using namespace std;

// function main begins program execution
int main()
{
  int number1; // first integer to compare
  int number2; // second integer to compare

  cout << "Enter two integers to compare: "; // prompt user for data
  cin >> number1 >> number2; // read two integers from user

  if ( number1 == number2 )
    cout << number1 << " == " << number2 << "\n";

  if ( number1 != number2 )
    cout << number1 << " != " << number2 << "\n";
```

دریافت داده از کاربر و نمایش اطلاعات

```
   if ( number1 < number2 )
      cout << number1 << " < " << number2 << "\n";

   if ( number1 > number2 )
      cout << number1 << " > " << number2 << "\n";

   if ( number1 <= number2 )
      cout << number1 << " <= " << number2 << "\n";

   if ( number1 >= number2 )
      cout << number1 << " >= " << number2 << "\n";
} // end function main
```

```
Enter two integers to compare: 3 7
3 != 7
3 < 7
3 <= 7
```

```
Enter two integers to compare: 22 12
22 != 12
22 > 12
22 >= 12
```

```
Enter two integers to compare: 7 7
7 == 7
7 <= 7
7 >= 7
```

عملگر کاما

این عملگر برای به زنجیر در آوردن چندین عملیات مختلف استفاده میشود.بدین شکل که ارزش لیستی از عبارتها که بوسیله کاما از هم جدا میشوند برای آن عبارتی منظور میشود که در سمت راست بقیه قرار دارد.دستور زیر نحوه استفاده از این عملگر را نشان میدهد:

```
value=(count,99,3,100);
```

عملگر sizeof

این عملگر طول یک متغییر یا یک نوع داده را مشخص میکند:

sizeof متغییر ;

sizeof(نوع داده) ;

عملگر بیتی انتقال

عملگرهای بیتی انتقال، عبارتند از عملگر بیتی انتقال به چپ و عملگر انتقال بیتی به راست که آنها را به شکل زیر نشان میدهند:

کاربرد	نماد در c++
عملگر بیتی انتقال به چپ	<<
عملگر انتقال بیتی به راست	>>

عملگرهای بیتی انتقال به چپ << باعث انتقال بیتهای عملوند متغییر چپ به اندازه تعداد بیت های تعیین شده بوسیله مقدار طرف راست عملگر به سمت چپ مشود.در اینصورت بیتهای سمت چپ از دست رفته وبیت های خالی شده سمت راست با صفر پر میگردند.

عملگر بیتی انتقال

X=0x7ca4;

X=0111,1100,1010,0100

Y=x<<4;

Y=1100,1010,0100,0000

در نتیجه خواهیم داشت:

Y=0xca40;

عملگر بیتی انتقال

عملگرهای بیتی انتقال به راست << باعث انتقال بیتهای عملوند متغییر راست به اندازه تعداد بیت های تعیین شده بوسیله مقدار طرف راست عملگر به سمت راست مشود.در اینصورت بیتهای سمت راست از دست رفته وبیت های خالی شده سمت چپ با صفر پر میگردند.

علاوه بر چند عملگر بالا میتوان از عملگرهای نیز برای عملیات بیتی استفاده نمود:

عمل	نماد در C++
نقیض بیتی	!
یای بیتی	|
و بیتی	&
یای انحصاری بیتی	^

عملگر بیت انتقال

مثال: اگر a=10 و b=15 باشد مقدار c=a&b به صورت زیر محاسبه میشود:

	بایت پر ارزش								بایت کم ارزش							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
a	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
b	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
c	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

در اینصورت خروجی برابر با c=10 خواهد بود.

عملگر توصیف کننده نوع داده

با استفاده از یک توصیف کننده نوع داده میتوان انواع داده را ایجاد نمود که با نیازهای خاص به طور دقیق انطباق داشته باشد.

توصیف کننده های نوع داده عبارتند از:

signed

unsigned

long

short

توصیف کننده long بزرگی داده ها را تقریبا دوبرابر میکند وتوصیف کننده short طول داده را تقریبا نصف میکنند.

توصیف کننده unsigned را میتوان روی char و int اعمال نمود.این توصیف کننده را میتوان همراه long وshort نیز مورد استفاده قرار داد. این توصیف کننده برای ایجاد یک عدد صحیح بدون علامت استفاده میشود.اعداد صحیح بدون علامت از تمامی بیت های جهت نگهداری مقادیر استفاده میکنند و همیشه عددی مثبت خواهندبود.

عملگر توصیف کننده نوع داده

```
#include<iostream.h>
#include<iomanip.h>
#include<conio.h>
main(){
   cout << "sizeof(char) is " << sizeof(char) << " bytes " << endl;
 cout << "sizeof(short) is " << sizeof(short) << " bytes " << endl;
 cout << "sizeof(int) is " << sizeof(int) << " bytes " << endl;
 cout << "sizeof(long) is " << sizeof(long) << " bytes " << endl;
 cout << "sizeof(long long) is " << sizeof(long long) << " bytes " << endl;
 cout << "sizeof(float) is " << sizeof(float) << " bytes " << endl;
 cout << "sizeof(double) is " << sizeof(double) << " bytes " << endl;
 cout << "sizeof(long double) is " << sizeof(long double) << " bytes " << endl;
 cout << "sizeof(bool) is " << sizeof(bool) << " bytes " << endl;
}
```

```
sizeof(char) is 1 bytes
sizeof(short) is 2 bytes
sizeof(int) is 4 bytes
sizeof(long) is 4 bytes
sizeof(long long) is 8 bytes
sizeof(float) is 4 bytes
sizeof(double) is 8 bytes
sizeof(long double) is 12 bytes
sizeof(bool) is 1 bytes
```

عملگر توصیف کننده نوع داده

نوع داده	اندازه(بایت)	مینیمم	ماکزیمم
int	4 (2)	-147483648	2147483647
unsigned int	4 (2)	0	4294967295
char	1	-128	127
unsigned char	1	0	255
short (or short int)	2	-32768	32767
unsigned short	2	0	65535
long (or long int)	4 (8)	-2147483648	2147483647
unsigned long	4 (8)	0	مانند بالا
long long (or long long int)	8	-2^{63}	$2^{63}-1$
unsigned long long	8	0	$2^{64}-1$
float	4	3.4e38	3.4e-38
double	8	1.7e308	1.7e-308
long double	12		
bool	1	false (0)	true (1 or non-zero)
wchar_t	2 (4)		

قالب بندی نوع داده

نوع داده را میتوان به طور موقت تغییر داد .شکل کلی این عمل به صورت زیر خواهد بود.

متغییر (نوع داده) ;

مثال:

```
float f;
f=100.2;
cout<<(int) f;//100چاپ
int i=10;
cout<<i/3<<'\n';
cout<<(float) i/3<<'\n';
```

قالب بندی خروجی

برای چاپ کمیت ها به شکل مورد نظر بااستفاده از cout میتوان آنها راقالب بندی نمود.برای این کار از کمیتهای به شکل زیر که در فایل سرآیندiomanip قرار دارند، میتوان استقاده نمود.

دستور	عملکرد
setw(d)	به اندازه d فضای خالی چاپ میشود
Setprecision(d)	تا دقت dرقم عددراچاپ میکند

قالب بندی خروجی

```
#include <iomanio.h>
main() {
   // Floating point numbers
   double pi = 3.14159265;
   cout << fixed << setprecision(4); //fixed format with 2 decimal
places
   cout << pi << endl;
   cout << "|" << setw(8) << pi << "|" << setw(10) << pi << "|" <<
endl;
      // setw() is not sticky, only apply to the next operation.
   cout << setfill('-');
   cout << "|" << setw(8) << pi << "|" << setw(10) << pi << "|" <<
endl;
   cout << scientific;  // in scientific format with exponent
   cout << pi << endl;

   // booleans
   bool done = false;
   cout << done << endl;  // print 0 (for false) or 1 (for true)
   cout << boolalpha;     // print true or false
   cout << done << endl;


}
```

قالب بندی خروجی

در این صورت خروجی به شکل زیر خواهد بود:

```
3.1416
|  3.1416|    3.1416|
|--3.1416|----3.1416|
3.1416e+000
0
false
```

تبدیل نوع داده

یک بخش از قواعد تبدیل در C++ را ترفیع نوع (type casting) مینامند.در C++ هر جا یک char یا short در عبارتی به کاررفته باشد،مقدار آن در طی ارزیابی آن عبارت به طور اتوماتیک به int ارتقا داده میشود.

باید توجه نمود که ترفیع نوع فقط در طی ارزیابی آن عبارت موثر میباشد و آن متغییر از نظر فیزیکی بزرگتر نمیشود.در اصل کامپایلر از کپی موقتی از مقدار آن متغییر استفاد خواهد نمود. پس از آنکه ترفیع نوع های اتوماتیک اعمال گردید کامپایلر همه عملوندها را به نوع بزرگترین عملوند تبدیل میکند.

اگر در یک دستور انتساب که نوع داده سمت راست آن با نوع داده سمت چپ آن تفاوت داشته باشد ،نوع داده سمت راست به نوع داده سمت چپ تبدیل میگردد.وقتی نوع داده سمت چپ بزرگتر از نوع داده سمت راست باشداین فرایندمشکلی ایجاد نمی کند.اما وقتی نوع داده سمت چپ از نوع داده سمت راست کوچکتر باشد ممکن است مقداری از داده ها گم شود .

تبدیل نوع داده

مثال: در عمل تبدیل نوع داده از یک مقدار اعشاری به یک مقدار صحیح ،جز اعشاری عدد ازبین میرود:

```
#include <iostream>
#include <iomanip>


main() {
   int i;
   double d;



   i = 3;
   d = i;     // انتساب یک متغییر صحیح به یک متغییر اعشتری
   cout << "d = " << d << endl;   // خروجی3.0

   d = 5.5;
   i = d;     // انتساب یک متغییر اعشاری به یک متغییر صحیح
   cout << "i = " << i << endl;   // خروجی 5

   i = 6.6;   // انتساب یک مقدار اعشاری به یک متغییر صحیح
   cout << "i = " << i << endl;   // خروجی6

   }
```

تبدیل نوع داده

مثال: خروجي قطعه كد زير برابر ۲۴- ميباشد:

```
#include <iostream.h>

main(){

char ch;

int i; i=1000;

ch=i;

i=ch;

cout<<i;}
```

رفتار با ثابت ها

به طور پیش فرض ،کامپایلر یک عدد ثابت عددی را در کوچکترین نوع داده ای که میتواند آن داده را جای دهد ،نگه میدارد.بنابراین اگر فرض کنیم که اعدا صحیح ۱۶ بیتی هستند ،به طور پیش فرض ۱۰ یک int است اما ۴۶۰۰۰ یک Unsigned و 100001 یک long است.گرچه مقدار ۱۰ را در یک char هم میشود جای داد اما کامپایلر چنین کاری نمی کند چون این کار به معنای شکستن حدود نوع داده است.

تنها استثنا در قاعده کوچکترین نوع داده زمانی است که ثابت های مورد بحث اعشاری هستند، که در این صورت از نوع double خواهند بود.

رفتار با ثابت ها

مثال: با توجه به توضیحات بالا خروجی قطعه کد زیر

```
cout<<"enter a number";
cin>>i;
if(i&32768) cout<<"N";
else cout<< "P" ;
```

اگر عدد i مثبت باشد P و اگر منفی باشدN خواهد بود.

در مواردی که فرض C++ در مورد یک ثابت عددی ،همان چیزی نیست که شما میخواهید،میتوانید با بکار بردن یک پسوند نوع دقیق آن ثابت عددی را مشخص کنید.

برای نوع اعشاری ،اگر پس از عدد مورد نظر خود یک 'F' قرار دهید با آن عدد به مثابه یک float رفتار میشود.

اگر پس از آن عددیک L قرار دهید،آن عدد یک long doubleتلقی میشود. برای انواع صحیح پسوند 'U' به معنای unsigned و'L' به معنای long است.

تمرین ها

۱-با اجرای هریک از دستورهای زیر چه چیزی در خروجی چاپ می شود. فرض کنید x=2 و y=3 باشد و در صورتی که خروجی ندارد بنویسید "هیچ".

A. cout<<x;
B. cout<<x+x;
C. cout<<"x=";
D. cout<<"x= "<<x;
E. z=x+y;
F. // cout<<x;

۲-کد زیر چه چیزی را در خروجی چاپ می کند.

```
cout<<"*\n**\n***\n****\n*****\n";
```

تمرین ها

۳-در دستورهای C++ زیر ترتیب ارزیابی عملگرها را مشخص کنید و مقدار x را پس از اجرای هر دستور نشان دهید:

A. x=7+3*6/2-1;
B. x=7%8+3*5-7/8;
C. x=(3*9*(3+(9*3/(3))));

۴-خروجی را بیابید:

```
int a,b=5;
a=(b++)*(++b);
cout<<a<<b;
```

۵-خروجی را بیابید:

```
int a=6;
int b=2;
c=++a*( b++)+a-a++;
cout<<a<<b<<c;
```