فهرست مطالب

فصل ۱ :

انواع داده ها – انواع عملگرها – دستورات ورودی و خروجی

مقدمه

زبان C در سال ۱۹۷۰ توسط دنیس ریچی طراحی گردید. این زبان، عناصر زبانهای سطح بالا را با خصوصیات تابعی زبان اسمبلی درهم آمیخته است و به همین علت زبان C را یک زبان میانی می‌نامند.

زبان c++ از شیوه برنامه نویسی شی ء گرا (OOP) ، استفاده می کند. در این روش از شیء (object) استفاده می شود. یک شیء دارای یک سری متغیر و توابع است که برای شی ء های دیگر شناخته شده نیست. زبان c++ در سال ۱۹۸۳ توسط استروستراپ ابداع شد و در سال ۱۹۹۴ توسط ANSI استاندارد شد. هر زبان oop دارای سه ویژگی مهم به قرار زیر است که در فصل های بعدی به آنها می پردازیم :

۱- **پلی مورفیسم** (Polymorphism)

استفاده از یک نام در چندین مورد مربوط به هم

۲- **کپسوله سازی** (Encapsulation)

کنار هم قرار دادن کد و داده هایی که این کدها با آنها کار می کنند و حفظ آنها از دخالتهای خارجی.

۳- **ارث بری** (Inheritance)

به دست آوردن خصوصیات یک شیء توسط شیء دیگر.

انواع داده ها

در زبان C پنج نوع داده اصلی وجود دارد که عبارتند از: void, double, float, int, char . برای ذخیره مقادیر صحیح از نوع int و برای مقادیر اعشاری از نوع double,float و برای ذخیره کاراکترها از نوع char استفاده می‌شود. در رابطه با void در فصل تابع و اشاره گر صحبت خواهد شد.

تذکر: با استفاده از کلماتی مانند signed, unsigned, long , short می توان نوع های دیگری نیز ساخت. تمامی این اصلاح کننده ها می توانند با نوع int به کار روند. بر روی نوع char اصلاح کننده های signed وunsigned و بر روی double فقط long را می توان به کار برد.

در زبان C++ داده ای به نام bool برای ذخیره مقادیر منطقی وجود دارد.

در c++ از نوع wchar_t برای ذخیره کاراکترهای ۱۶ بیتی می توان استفاده کرد.

جدول زیر انواع داده ها را نمایش می‌دهد:

نوع	اندازه(بایت)	بازه قابل قبول
char	1	127 تا -128
unsigned char	1	0 تا 255
int	2 یا 4	32767+ تا -32768
unsigned int	2 یا 4	0 تا 65535
long int	4	تقریباً ۲- میلیارد تا ۲+ میلیارد
float	4	
double	8	
long double	10	

- اندازه int در محیطهای ۳۲ بیتی برابر ۴ بایت است.
- نوع signed int همان int است.
- نوع signed char همان char است.
- نوع float دارای ۶ رقم دقت و نوع double دارای ۱۰ رقم دقت است.
- می توان چند اصلاح کننده را با هم ترکیب کرد. مثلا نوع unsigned long int یک عدد ۴ بایتی بدون علامت است.
- می توان کاری کرد که عدد کوچکی مانند ۲۰ به عنوان یک long محسوب شود. کافی است که از حرف L بعد از عدد به صورت long int a=20L; استفاده کنیم.

اعلان ثوابت

ثابت مقداری است که در طول اجرای برنامه قابل تغییر نمی باشند. برای اعلان ثابت به دو روش عمل می شود:

۱- توسط دستور const : const int x =5;

۲- توسط دستور #define : #define Y 2

مشاهده کردید که در انتهای دستور define نیازی به سمی کالون نمی باشد. دستور define باعث می شود که قبل از ترجمه برنامه توسط کامپایلر، مقدار ۲ به جای ثابت Y در برنامه قرار بگیرد و این دستور در زمان اجرا وجود ندارد.

تذکر: به ثوابتی که با دستور define تعریف می شوند، ماکرو می گویند و برای تفکیک آنها از متغیرهای برنامه، آنها را با حروف بزرگ نمایش می دهیم.

تبدیل نوع

وقتی که متغیرهایی با نوعهای مختلف در یک عبارت با یکدیگر ترکیب می‌شوند، نوع هایی با طول کوچکتر بـه نـوع هـایی بـا طول بزرگتر تبدیل می‌شوند. تبدیل نوع هم در عبارات و هم در احکام انتسابی ممکن است رخ دهد. در تبدیل نوع ممکن است اطلاعاتی از بین بروند. مثلاً وقتی یک متغیرint به یک متغیر char انتساب داده می‌شود، بایت کـم ارزش متغیـر int بـه یـک متغیر char منتقل شده ولی بایت با ارزش از بین می‌رود.

برای ذخیره سازی کاراکترها، کد اسکی آنها ذخیره می شود. مثلا با تعریف char ch='5'; مقدار ۳۵ (کد اسکی کـاراکتر ۵) در متغیر ch ذخیره می شود.

*نوع نتیجه عبارت زیر را مشخص نمایید؟

```
float     f;
char      ch;
double  d;
(f+ch)-(f*d)
```

حل: نوع نتیجه عبارت، double است:

$$\underbrace{\underbrace{(f+ch)}_{\text{float}}-\underbrace{(f\times d)}_{\text{double}}}_{\text{double}}$$

* بعد از اجرای دستورات زیر چه مقداری در ch قرار می‌گیرد؟

```
int     i=65; char  ch;
ch= i;
```

حل: کاراکتر A قرار می‌گیرد.(کد اسکی کاراکتر A برابر ۶۵ است.)

عملگرها

عملگرها نمادهایی هستند که اعمال خاصی را انجام می دهند. انواع عملگرها عبارتند از :

۱- محاسباتی (++ , --, ٪ , /,*, -, +)

۲- رابطه ای (<, >, <=, >=, ==, !=)

۳- منطقی (&&, ||, !)

۴- بیتی (&, |, ~, ^, >>, <<)

۵- عملگر کاما (,)

۶- عملگر شرطی (?)

۷- عملگر ترکیبی (ترکیب عملگرهای محاسباتی و عملگر =)

تذکر: البته عمگرهای دیگری مانند & ، * وجود دارد، که در مبحث اشاره گر ها بررسی می شوند.

مقدار صفر به منزله FALSE و هر مقدار غیر صفر به منزله TRUE در نظر گرفته می شود.

حاصل and (عملگر &&) دو عبارت وقتی درست است، که هر دو عبارت درست باشند و حاصل or (عملگر ||) دو عبارت وقتی درست است، که حداقل یکی از آنها درست باشد.

اولویت عملگرهای رابطه ای بالاتر از عملگرهای منطقی است.

جدول تقدم عملگرها

```
[ ]   ( )                                   بالاترین اولویت
!     ~     ++    --    *    &    sizeof
*     /     %
+     -
<<    >>
<     <=    >     >=
==    !=
&
^
|
&&
||
?
=     +=    -=    *=    /=
,                                           کمترین اولویت
```

مثالهایی از کاربرد عملگرهای محاسباتی

✱ بعد از اجرای دستورات زیر مقدار x و y چه خواهد شد؟

```
x  = 5;
y = --x;
```

حل: ابتدا مقدار x یک واحد کم شده و سپس در متغیر y ذخیره می شود. بنابراین در هر دو متغیر مقدار ۴ ذخیره می شود.

■

✱ بعد از اجرای دستورات مقدار x و y چه خواهد شد؟

```
x  = 5;
y = x--;
```

حل: ابتدا مقدار x به y منتقل شده و سپس یک واحد از x کم می‌شود. در نتیجه مقدار y برابر ۵ و مقدار x برابر ۴ خواهد شد.

■

✱ بعد از اجرای دستورات زیر مقدار x وy چه خواهد شد؟

```
x  = 5;
y = ++ x * x;
```

حل: ابتدا مقدار x برابر ۶ شده و سپس حاصل ۶×۶ در y قرار می گیرد. در نتیجه مقدار x برابر ۶ و مقدار y برابر ۳۶ خواهد شد.

■

✱ بعد از اجرای دستورات زیر مقدار x وy چه خواهد شد؟

```
x  = 5;
y = ++ x * ++x;
```

حل: ابتدا مقدار x برابر ۶ شده و سپس برابر ۷ شده و در نهایت حاصل ۷×۷ یعنی ۴۹ در y قرار می گیرد.

■

✱ بعد از اجرای دستورات زیر مقدار x وy چه خواهد شد؟

```
x  = 5;
y =  ++ x * x++;
```

حل: ابتدا مقدار x برابر ۶ شده و سپس حاصل ۶×۶ در y قرار گرفته و در نهایت مقدار x برابر ۷ خواهد شد.

■

✱ حاصل اجرای دستورات زیر در مقابل آنها نوشته شده است:

```
4 % 3 = 1  , 4 % -3 = 1 , -4 % 3 = -1 , -4 % -3 = -1
```

تذکر: علامت نتیجه عملگر % ، با علامت عملوند اول یکسان است و به علامت عملوند دوم بستگی ندارد.

■

مثالهایی از کاربرد عملگرهای بیتی

✱ حاصل 3 | 129 را بدست آورید؟(عملگر or بیتی)

```
129 = 10000001
  3 = 00000011
---------------------
      10000011=131
```

حل: نتیجه or برای ۲ بیت وقتی یک است که حداقل یکی از آنها یک باشد.

■

✱ حاصل 3 ^ 129 را بدست آورید؟ (عملگر xor بیتی)

```
129 = 10000001
  3 = 00000011
---------------------
      10000010=130
```

حل: نتیجه xor برای ۲ بیت وقتی یک است که آن دو بیت مانند هم نباشند.

■

✱ حاصل 3 & 129 را بدست آورید؟ (عملگر and بیتی)

```
129 = 10000001
  3 = 00000011
      ------------
      00000001=1
```

حل: نتیجه and برای ۲ بیت وقتی یک است که هر دو بیت یک باشند.

■

✱ بعد از اجرای دستورات زیر مقدار x چه خواهد شد؟ (عملگر شیفت به راست)

```
x = 96;
x=x  >> 2 ;
```

حل: هر شیفت به راست معادل تقسیم بر ۲ است. بنابراین ۹۶ بر ۴ تقسیم شده و حاصل برابر ۲۴ خواهد شد.

■

✱ بعد از اجرای دستورات زیر مقدار x چه خواهد شد؟(عملگر شیفت به چپ)

```
x= 14;
x=x<<3;
```

حل: هر شیفت به چپ معادل ضرب در ۲ است و سه شیفت به چپ معادل ضرب در ۸ است. بنابراین ۱۴ در ۸ ضرب شده و x برابر ۱۱۲ می شود.

■

عملگر شرطی

این عملگر با تست یک شرط، مقداری را به یک متغیر نسبت می‌دهد:

```
x = exp1 ? exp 2: exp3;
```

یعنی ابتدا عبارت exp1 ارزیابی می‌شود و اگر دارای ارزش درست بود، مقدار exp2 بعد از ارزیابی در x قرار می‌گیرد و اگر نادرست بود، مقدار exp3 بعد از ارزیابی در x قرار می گیرد.

✱ بعد از اجرای دستورات زیر چه مقداری در x قرا ر می‌گیرد؟

```
x = 5 > 2 ?  100: 20 ;
```

حل: حاصل عبارت 5>2 ، درست است و مقدار عبارت اول یعنی ۱۰۰ در x قرار می گیرد.

■

عملگر کاما

برای انجام چند عمل در یک دستور از عملگر کاما باید استفاده کرد :

```
x = (exp1 , exp2);
```

ابتدا exp1 ارزیابی شده و سپس نتیجه ارزیابی exp2 به متغیر x منتقل می‌شود. در این نوع موارد معمولاً exp1 و exp2 با یکدیگر ارتباط دارند. بعد از اجرای دستور زیر، مقدار ۵ در متغیر x قرار می‌گیرد.

```
x = (a=2 , a+3);
```

✱ حاصل ارزیابی عبارت زیر چیست؟

```
9 > 4 || !(5 <7 )  &&  8 <=9
```

حل: حاصل ارزیابی عبارت ، (True) است:

$$\underbrace{9>4}_{T} \;\|\; \underbrace{!(5<7)}_{F} \;\&\&\; \underbrace{8<=9}_{T}$$

$$\underbrace{\qquad\qquad}_{F}$$

$$T$$

■

کوتاه نویسی

می‌توان از کوتاه نویسی استفاده کرد. مثلاً به جای دستور x= x+3; از دستور x+ =3; استفاده کرد.

✱ با فرض x=5 و y=2 ، مقدار متغیر y بعد از اجرای دستور ; y *= x-2 چه خواهد شد؟

حل: در واقع عبارت به صورت زیر ارزیابی می‌شود و حاصل برابر ۶ خواهد شد:

```
y = y* (x-2) = 2*(5-2)=6
```

■

دستورات cin , cout

در c++ ، برای خواندن اطلاعات از ورودی از دستور cin و برای نمایش اطلاعات در خروجی از cout استفاده می شود. به عبارتی به جای استفاده از دستور scanf از دستور cin و به جای استفاده از دستور printf از دستور cout استفاده می شود. دو دستور خواندن زیر معادل می باشند:

```
scanf("%d%d",&x,&y);
cin>>x>>y;
```

دو دستور چاپ زیر معادل می باشند:

```
printf("%d%d",x,y);
cout<<x<<y;
```

دستور cin تا رسیدن به space ، داده ها را از ورودی می خواند.

دستور cin.get( ) تا رسیدن به enter و یا کاراکتر مشخص شده، داده ها را از ورودی می خواند. تابع get( ) ، عضو شیء cin است.

مثال‌هایی برای cin :

✱اگر از ورودی ↵ ali reza وارد شود، فقط رشته ali در متغیر s ذخیره می شود:

```
char s[10]; cin>>s;
```

اگر بخواهید کل رشته ali reza در متغیر s ذخیره شود، باید از دستور cin.get استفاده کرد.

■

✱ اگر از ورودی ↵ abcd وارد شود، فقط رشته ab در متغیر s ذخیره می شود:

```
char s[10]; cin.get(s,10,'c');
```

■

✱ در صورت ورود کاراکتر A ، کد اسکی آن یعنی 65 در x ذخیره می شود:

```
int x; x=cin.get( );
```

■

✱در صورت ورود رشته ↵ ali، فقط کاراکتر اول آن یعنی a در متغیر s ذخیره می شود:

```
char s; s=cin.get( );
```

■

✱ در صورت وارد کردن Enter خروجی 10 است :

```
cout<<cin.get( );
```

تذکر: پس از وارد کردن ctrl+z ، خروجی 1- می باشد.

■

✱ در برنامه ی روبرو، پس از وارد کردن سه Enter خروجی کدام است؟

```
cout<<(cin.get( ) +cin.get( ) +cin.get( ) );
```

حل : خروجی 30 می باشد.

تذکر: بعد از وارد کردن یک ctrl+z ، خروجی 3- می باشد.

■

مثال‌هایی برای cout :

✱ توسط دستور زیر یک رشته ali چاپ شده و در خط بعد رشته reza می باشد:

```
cout<<"ali"<<endl<<"reza";
```

تذکر: به جای endl می توان از "\n" استفاده کرد.

■

✱ توسط دستور زیر رشته ok با ۴ فاصله قبل از آن چاپ می شود:

```
cout.width(6);
cout<<"ok";
```

تذکر: اگر از دستور ;('-')cout.fill قبل از دستورات بالا استفاده کنید، به جای فاصله خالی، کاراکتر خط تیره چاپ خواهد شد.

■

✱ خروجی دستور زیر 1.20000 می باشد:

```
cout.setf(ios::showpoint);
cout<<1.2;
```

تذکر: اگر از دستور اول استفاده نشود، خروجی 1.2 خواهد بود.

■

✱ برای چاپ عدد ۱۰ در مبنای ۱۶ (یعنی A) از دستور زیر استفاده می کنیم:

```
cout<<hex<<10;
```

البته می توان از دستورات زیر نیز استفاده کرد:

```
cout.setf (ios::hex);
cout<<10;
```

■

✱ برای چاپ عدد ۱۰ در مبنای ۸ (یعنی ۱۲) از دستور زیر استفاده می کنیم:

```
cout<<oct<<10;
```

■

✱ خروجی دستور زیر 1- می باشد:

```
cout<<EOF;
```

■

✱ خروجی چه می باشد؟

```
char  c=EOF;
cout<<c<<'a';
```

حل: چاپ کاراکتر **a** با یک فاصله خالی قبل .

■

✱ خروجی چه می باشد؟

```
cout<<EOF<<'a';
```

حل: خروجی 1a- می باشد.

■

تذکر: تابع ( )put ، عضو شیء cout است. خروجی دستور ;('a')cout.put برابر a می باشد.

دستور cin، شیئی از کلاس istream و دستور cout، شیئی از کلاس ostream می باشد. هنگام استفاده از این دستورات باید فایل <iostream.h> را در ابتدای برنامه include کرد.

عبارت دستور cout از راست به چپ پردازش شده و از چپ به راست نوشته می شود.

✱ خروجی چیست؟

```
int  x=3;
cout<<(x=0);
```

حل: خروجی 0 است.(اگر پرانتز برداشته شود، خطا دارد.)

■

✱ خروجی چیست؟

```
int  x=3;
cout<<(x=0)<<(x==0);
```

حل: خروجی 00 است.(اگر پرانتزها برداشته شود، خطا دارد.)

■

✱ خروجی چیست؟

```
int  x=0;
cout<<(x=0)<<(x==0);
```

حل: خروجی 01 است.

■

✱ خروجی چیست؟

```
int  x=4;
cout<<(x==0)<<(x=0);
```

حل: خروجی 10 است.

■

✱ خروجی چیست؟

```
int  x=0;
cout<<(x==0)<<(x=0);
```

حل: خروجی 10 است.

■

در جدول زیر کاراکترهای کنترلی که در cout می توان از آنها استفاده کرد، آورده شده است:

کاراکتر	عملکرد	کاراکتر	عملکرد
\n	انتقال کنترل به خط بعد	**\b**	حذف کاراکتر قبلی
\r	انتقال کنترل به اول خط	\"	چاپ دابل کوتیشن (")
\t	انتقال کنترل به ۸ محل بعدی	\\	چاپ کاراکتر \
\v	انتقال کنترل به ۸ سطر بعد	\:	چاپ کاراکتر :

✱خروجی هر دستور در مقابل آن نوشته شده است:

```
cout<<"abc'"<<'\r'<<'d';      ⇒  dbc
cout<<"abc'"<<'\b'<<'d';      ⇒  abd
cout<<'a'<<'\t'<<'b';         ⇒  a      b
```

■

✱ خروجی چه می باشد؟

```
cout<<strlen("'\n");
```

حل: تابع strlen طول رشته ورودی را محاسبه می کند. بنابراین خروجی 3 می باشد. (\n یک کاراکتر محسوب می شود.)

■