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>
```

به جای <> می تواند از " " استفاده کرد. ولی تفاوت آن ها در این است که هنگام استفاده <>، کامپایلر در دایرکتوری استاندارد include به دنبال فایل می گردد. ولی هنگامی که از " " استفاده می کنیم، کامپایلر در دایرکتوری جاری به دنبال فایل می گردد.

پسوند ".h"، پسوند فایل های هدر است. البته در بعضی کامپایلر ها از پسوند های hpp و hxx استفاده می شود.

ورودی - 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.)

داده ها – متغیر ها - محاسبات

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 :

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
+=	جمع و انتساب (a+=b با a=a+b برابر است)
-=	تفریق و انتساب
*=	ضرب و انتساب
/=	تقسیم و انتساب
&&	AND منطقی
||	OR منطقی
!	NOT منطقی
=!	نا مساوی
==	مساوی
<	کوچکتر
>	بزرگتر
>=	بزرگتر و مساوی
<=	کوچکتر و مساوی
Sizeof	اندازه عملوند
? :	عملگر شرطی (در ادامه در مورد آن توضیح داده می شود.)
در قسمت زیر عملوند ها (اپراتورها) باید صحیح باشند.	
<<	شیفت چپ
>>	شیفت راست
&	AND بیت با بیت
|	OR بیت با بیت
^	OR مانع الجمع بیت با بیت
~	معکوس بیت
%=	مدول و انتساب
<<=	شیفت چپ و انتساب
>>=	شیفت راست و انتساب
&=	AND بیتی و انتساب
|=	OR بیتی و انتساب
=^	OR مانع الجمع و انتساب

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
راست به چپ	**++ -- + - ! sizeof**	
چپ به راست	*** / %**	
چپ به راست	**+ -**	
چپ به راست	**< <= > >=**	
چپ به راست	**== !=**	
چپ به راست	**&&**	
چپ به راست	**||**	
	? :	پایین ترین الویت
راست به چپ	**= += -=**	

فراخوانی توابع

توابع به سه دسته کلی تقسیم می شوند :

- **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 Parameter-name);

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 Parameter-name);

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 اضافه شدن، توابع همانند توابع دیگر فراخوانی می شوند.

توابع

هر تابع شامل دو قسمت است : عنوان تابع و بدنه تابع.
بدنه تابع : مجموعه دستوراتی است که تابع انجام می دهد.
الگوی تابع : اعلان تابع بدون بدنه تابع را الگوی تابع می گویند.
تعریف تابع : اعلان تابع به همراه بدنه تابع را تعریف تابع می گویند.
پارامتر هایی که در فراخوانی تابع استفاده می شوند را **پارامترهای واقعی (Actual Parameters)** می گویند.
پارامتر هایی که در عنوان تابع ذکر می شوند را **پارامترهای صوری (Formal Parameters)** می گویند.
متغیر محلی: متغیری است که در داخل یک بلوک اعلان می شود و در خارج از آن قابل دسترسی نیست.
متغیر اتوماتیک : متغیری است که هنگام ورود به بلوک به آن حافظه اختصاص داده می شود و هنگام خروج از بلوک، آن محل از حافظه آزاد می شود.
متغیر استاتیک : متغیری است که حافظه اختصاص داده شده به آن تا انتهای برنامه در اختیارش باقی می ماند. همه متغیر های جهانی استاتیک هستند.
دستور Return : این دستور برای برگرداندن یک مقدار به فراخواننده استفاده می شود. در توابع تهی که مقداری را بازنمی گردانند، از این دستور برای خارج شدن از تابع استفاده می شود.

مثال :

```
void somefunc(int n)
{
if (n > 50) {
cout << "The value is out of range!";
return;
}
n = 412 * n;
cout << n;
}
```

دستور فوق با دستور زیر برابر است :

```
void somefunc(int n)
{
if (n > 50) {
cout << "The value is out of range!!";
}
else
{
n = 412 * n;
cout << n;
}
}
```

پارامترهای مقداری (Value Parameters) : پارامتری که کپی محتویات واقعی نظیر را دریافت می کند.
در اینگونه پارامتر ها، هنگامی که یک تابع را فراخوانی می کنیم، یک کپی از پارامتر واقعی در پارامتر صوری قرار می گیرد. بنابراین هر شیء که مقدار داشته باشد می تواند به عنوان پارامتر مقداری مورد استفاده قرار گیرد.
همانطور که گفته شد، پارامتر مقداری یک کپی از پارامتر واقعی را دریافت می کند. در نتیجه پارامتر واقعی به طور مستقیم در دسترس نیست و نمی توان آن را تغییر داد. همچنین بعد از خارج شدن از تابع (بازگرداندن کنترل)، محتویات پارامتر های مقداری پاک می شوند. با توجه به این موارد، اگر بخواهیم از پارامتر ها به عنوان اطلاعات برگشتی به تابع استفاده کنیم باید از پارامتر های عطفی استفاده کرد.

پارامترهای عطفی (Reference Parameters) : پارامتری که موقعیت پارامتر واقعی نظیر را دریافت می کند. در اینگونه پارامتر ها، هنگام اعلان تابع، بعد از نوع داده علامت & (آمپرسند) قرار می گیرد.

مثال :

```
void somefunc (int & parameter)
```

پارامترهای عطفی می توانند پارامتر واقعی نظیر را به طور مستقیم در اختیار تابع قرار بدهد. همچنین می تواند در مقدار پارامتر واقعی تابع فراخواننده دخل و تصرف کرده و مقدار آن را تغییر دهد.
فقط یک متغییر می تواند به عنوان پارامتر واقعی به پارامتر عطفی ارسال شود.

اولویت نام (Name Precedence)

اگر در یک بلوک، شناسه ای هم نام با یک شناسه جهانی اعلان شود، از تابع جهانی در بلوک صرف نظر می شود. به عبارت دیگر، شناسه محلی بر شناسه جهانی اولویت پیدا می کند. به این اولویت، اولویت نام می گویند.

قوانین دامنه

قوانینی که دامنه شناسه ها را تعیین می کند را قوانین دامنه می گویند. این قوانین عبارتند از :

۱. نام تابع دامنه جهانی دارد. تعریف تابع نمی تواند در تعریف تابع دیگر محصور گردد.
۲. دامنه پارامتر های صوری نظیر دامنه متغیر های محلی است که در خارجی ترین بلوک بدنه تابع اعلان شده است.
۳. دامنه متغیر یا ثابت جهانی از نقطه اعلان تا پایان فایل گسترش دارد.(به جز موارد بند ۵)
۴. دامنه متغیر یا ثابت محلی از نقطه اعلان تا پایان بلوک گسترش دارد. این دامنه شامل بلوک های داخلی می شود. (به جز موارد بند ۵)
۵. اگر در دامنه یک شناسه یک بلوک وجود داشته باشد که شامل شناسه ای محلی هم نام با آن شناسه باشد، این بلوک و این شناسه محلی جزء دامنه آن شناسه نمی باشد.(شناسه های محلی اولویت نام دارند.)

متغیر جهانی

اگر بخواهیم یک متغیر را در فایل دیگری استفاده کنیم، باید به ابتدای دستور اعلان متغیر، کلمه extern را اضافه کنیم.

Extern Data-type Data-name;

مثال :
فایل سر فصل iostream که در آن متغیر های cin و cout از نوع istream و ostream اعلان شده اند :

```
extern istream cin;
extern ostream cout;
```

تصمیم گیری - حلقه

تصمیم گیری با شرط

عبارت IF

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;
```

عبارت Switch

هنگامی که می خواهیم از بین چندین حالت یک را انتخاب کنیم، از این دستور استفاده می کنیم :

```
Switch () {
Case 1://code;
Case 2://code;
....
default://code;
}
```

در عبارت فوق 1 و 2 شرط هایی هستند که در صورت درست بودن، case مربوطه اجرا می شود.

مثال ۱:

```
cin >> letter;
switch (letter) {
case 'a' :  cout  << "This is a vowel letter." << endl;
case 'b' : cout << "This is not a vowel letter." << endl;
default : cout << "I couldn't understand!" << endl;
}
```

مثال ۲:

```
cin >> choice;
switch (choice) {
case 1 : cout << "Hey" << endl;
case 2 : cout << "Hello" << endl;
case 3 : cout << "How are you?" << endl;
default : cout << "Who are you?" << endl;
}
```

دستور Break

این دستور به منظور خروج از حلقه استفاده می شود.

دستور Continue

از این دستور به منظور رها کردن جریان در حال اجرا استفاده می شود. یعنی به انتهای حلقه رفته و دوباره از اول حلقه شروع می کند.

تصمیم گیری بدون شرط

بعضی مواقع نیاز که یک عمل را بدون شرط انجام داد. تصمیم گیری در این مواقع را تصمیم گیری بدون شرط گفته می گویند.

دستور Goto

برای شعبه دادن به قسمتی از برنامه می بایست از دستور goto استفاده کرد. برای استفاده از دستور goto باید محل مورد نظر توسط یک label مشخص شود.

```
Label-name://code;
....
Goto Label-name;
```

مثال ۱:

```
mylabel: cout << "This is a label" << endl;
//code
goto mylabel;
```

حلقه

اگر بخواهیم دستوری چند بار تکرار شود، از حلقه استفاده می کنیم.

حلقه While

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

حلقه Do-While

این نوع حلقه مانند حلقه while است با این تفاوت که در این نوع حلقه، پس از انجام مجموعه دستورات، شرط امتحان می شود.

```
Do {
//code
}while(Condition);
```

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 ۱:

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

مثال ۲:

```
while (a==2) {
cout << "Hello" << endl;
}
```

مثال ۳:

```
Do {
cout << "Hello" << endl;
}while(a==2);
```

حلقه تو در تو

با قرار دادن چند حلقه در یک دیگر می توان یک حلقه تو در تو ایجاد کرد. حلقه هایی که داخل هم قرار می گیرند، می توانند از انواع متفاوت باشند.

ساختمان داده ها

داده ساده(اتمی)
یک مقدار در نوع داده ساده، یک موجود است و نمی تواند به اجزایی تقسیم شود. به این نوع داده، اتمی نیز می گویند.

داده های از پیش ساخته شده
انواع داده ساخته شده عبارتند از :
- **نوع صحیح**
- **نوع اعشاری**

نوع داده صحیح
نوع های زیر همگی از نوع داده صحیح هستند :
- **char**
- **unsigned char**
- **short**
- **unsigned short**
- **int**
- **unsigned int**
- **long**
- **unsigned long**

نکته : کلمه رزرو شده unsigned قبل از هر یک از نوع های داده، معرف مجموعه ای از اعداد غیر منفی است که از صفر شروع می شوند و تا حداکثر مجاز در کامپیوتر ادامه می یابند.
برای مثال در یک کامپیوتر، نوع داده char می تواند مقداری بین ۱۲۷- تا ۱۲۷ بگیرد. ولی نوع داده unsigned char می تواند مقداری از ۰ تا ۲۵۵ بگیرد.

کاراکتر ها
به جای کاراکتر ها می توان از عدد معادل آن ها استفاده کرد.(اعداد معادل کاراکتر ها در ضمیمه آورده شده است.)
کاراکتر های خاص در c++ عبارتند از :

خط جدید	\n
Tab افقی	\t
Tab عمودی	\v
برگشت	\b
Enter	\r
تغذیه فرم	\f
صدای زنگ	\a
ممیز وارون (Backslash)	\\
آپستروف	\’
گیومه	\”
یک کاراکتر تهی که همه بیت های آن صفر است.	\0
نمایش معادل اکتال یک عدد	\ddd
نمایش معادل هگزادسیمال یک عدد	\xddd

مثال :
```
Cout << "/a Hi";
```
صدای بوق شنیده می شود و پیام Hi روی صفحه ظاهر می شود.

نوع داده ساده تعریف شده توسط کاربر
در C++ این امکان به کاربر داده شده است که نوع داده مورد نظر خود را ایجاد کند. در ادامه می آموزیم که چگونه نوع داده دلخواه خود را تعریف کنیم.
دستور typedef
برای معادل سازی یک نام با یک نوع داده از این دستور استفاده می کنیم :

```
typedef type-name new-name;
```

در دستور فوق، type-name برابر با نوع داده مورد نظر و new-name نام جدیدی است که می خواهیم معادل سازی شود.

مثال :

```
typedef int Boolean;
```

پس از دستور فوق، می توان به جای int از Boolean استفاده کرد.

دستور enum

برای تعریف نوع داده ساده، به صورت مجموعه با ذکر کلیه عناصر آن مجموعه از این دستور استفاده می شود :

```
enum name {enum1 , enum2 , ...};
```

تبدیل نوع داده

در موارد زیر وقتی مقادیر با نوع های مختلف به کار می روند، C++ تبدیل نوع انجام می دهد :

۱. عبارات محاسباتی و رابطه ای
۲. انتساب
۳. ارسال پارامتر
۴. برگشت مقدار تابع از یک تابع با مقدار برگشتی

قانون تبدیل نوع در مورد ۱ با ۳ مورد دیگر تفاوت دارد.

انبساط نوع (Promotion)

در عبارات محاسباتی و رابطه ای اگر عملوند ها از نوع متفاوتی باشند، تبدیل نوع با استفاده از قانون انبساط انجام می گیرد.
تبدیل مقدار از نوع پایین تر به نوع بالاتر را **انبساط نوع** می گویند.
جدول زیر نتیجه بسط نوع را نشان می دهد(توجه کنید که نوع صحیح شامل char، short، int و ... می شود.) :

از	به	نتیجه بسط
Double	Long Double	همان مقدار ، گرفتن جای بیشتر در حافظه
Float	Double	همان مقدار ، گرفتن جای بیشتر در حافظه
نوع صحیح	نوع اعشاری	مقدار اعشاری نظیر مقدار صحیح ، قسمت اعشاری صفر است.
نوع صحیح	نوع صحیح بدون علامت نظیر	اگر عدد اولیه غیر منفی باشد همان مقدار. ولی اگر عدد اولیه منفی باشد آنگاه تغییر اساسی رخ می دهد.
نوع صحیح علامت دار	نوع صحیح علامت دار long	همان مقدار ، گرفتن جای بیشتر در حافظه
نوع صحیح بی علامت	نوع صحیح long	همان مقدار ، گرفتن جای بیشتر در حافظه

انقباض نوع

برای تبدیل نوع در انتساب، ارسال پارامتر و برگشت مقدار تابع از قانون انقباض نوع داده استفاده می شود. به طور کلی انبساط باعث از دست رفتن اطلاعات نمی شود ولی انقباض مقادیر، سبب از دست رفتن اطلاعات می شود.
تبدیل مقدار از نوع بالا تر به نوع پایین تر را **انقباض نوع** می گویند.
از دست رفتن اطلاعات در انقباض نوع به صورت زیر است :

- انقباض نوع صحیح بالاتر به نوع صحیح پایین تر (مثل long به int) سبب از بین رفتن بیت های انتهایی سمت چپ نمایش دودویی عدد می گردد.
- انقباض نوع اعشاری به نوع صحیح سبب از میان رفتن قسمت اعشاری می شود.
- انقباض نوع اعشاری بالاتر به نوع اعشاری پایین تر (مثل double به float) سبب کاهش دقت می شود.

داده ساختیافته

در این نوع داده، یک مقدار، مجموعه ای از اجزا را شامل می شود. نوع داده ساختیافته شامل انواع زیر است :

۱. **آرایه**
۲. **struct**
۳. **union**
۴. **کلاس class**

آرایه ها

آرایه مجموعه ای از عناصر همگن(همگی از یک نوع) است که به صورت Nبعدی مرتب شده اند.(N>1) هر عنصر با N عبارت شاخص(راهنما) قابل دسترسی است.

آرایه های یک بعدی

مجموعه ای از عناصر با نام مشترک هستند. برای تعریف آرایه از دستور زیر استفاده می کنیم :

Data-type Array-name[Const-Int-Expression];

در دستور فوق، array-name برابر با نام آرایه و const-int-expression عبارتی صحیح است که می تواند ثابت نامدار نیز باشد.
برای دسترسی به اعضای آرایه از دستور زیر استفاده می کنیم :

Array-name [index-expression]

index-expression یا عبارت شاخص(راهنما) باید از نوع داده صحیح باشد.

مثال :

```
int array [1000];
```

دستور فوق آرایه ای با ۱۰۰۰ عضو (از ۰ تا ۹۹۹) ایجاد می کند.

```
Array [12] = 13;
```

این دستور عضو ۱۳ آرایه را برابر با ۱۳ قرار می دهد.

هنگام اعلان آرایه می توان عناصر آن را مقدار دهی کرد. برای این کار لیست مقادیر را پشت سر هم و جدا شده با کاما می نویسیم و آن ها را در یک جفت آکولات محصور می کنیم.

مثال :

```
int age[5] =  {20, 5, 15, 32, 12};
```

عملگر جمعی : عملگری که روی کل ساختمان داده اعمال می شود.
عملگر جمعی را نمی توان در آرایه های به کار برد.

برای مثال اگر x و y به صورت زیر اعلان شده باشند :

```
int x[50];
int y[50];
```

عمل زیر اشتباه است :

```
x = y;
```

آرایه های کاربردی : لیست ها و رشته ها

لیست : مجموعه ای خطی یا متوالی از عناصر همگن با طول متغیر را لیست می نامند.
طول : تعداد واقعی مقادیر ذخیره شده در لیست را طول لیست می گویند.
لیست ها می توانند توسط آرایه و یا کلاس ایجاد شوند. ولی استفاده از آرایه ها بسیار متداول و معمول است.
رشته : مجموعه ای از کاراکتر ها که به عنوان یک موجود تصور می شود را رشته می نامند. در C++ دنباله ای از کاراکترها با کاراکتر تهی در آرایه ای از نوع char ذخیره می شوند.

مثال :

```
char message[6] = "Hello";
```

دستور فوق همانند دستور زیر است ولی دستور فوق را نمی توان برای دیگر انواع آرایه ها به کار برد. فقط برای رشته ها امکان پذیر است :

```
char message[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

آرایه ها را نمی توان به صورت جمعی چاپ کرد. ولی یک استثنا وجود دارد و آن هم رشته ها هستند. آرایه هایی که از نوع رشته هستند می توانند با دستور cout چاپ شوند . زیرا رشته ها می توانند متغیر باشند.

مثال :

```
char msg[6] = "Hello";
cout << msg;
```

آرایه های چند بعدی
استفاده از typedef در آرایه ها
از دستور typedef برای نامگذاری آرایه ها می توان استفاده کرد. برای مثال :

```
typedef float float-array[20];
```

این عبارت می گوید که flat-array یک آرایه ۲۰ عنصری از نوع اعشاری است. بعد از این دستور می توانیم متغیر هایی از نوع داده float-array ایجاد کنیم.

مثال :

```
typedef float float-array[20];
```

پس از این عبارت

```
float-array angle;
```

دستور فوق با دستور زیر برابر است :

```
float angle[20];
```

آرایه دو بعدی : مجموعه ای از عناصر از یک نوع داده، با ساختاری دو بعدی است. هر عنصر با دو عبارت شاخص(راهنما) قابل دسترسی است که هر عبارت شاخص(راهنما) موقعیت عنصر را در یک بعد نشان می دهد.

اعلان آرایه های چند بعدی مانند آرایه های یک بعدی است :

Data-type Array-name [Const-Int-Expression] [Const-Int-Expression] ... ;

دسترسی به عناصر آرایه های چند بعدی نیز مانند آرایه های یک بعدی است :

Array-name [Index-Expression] [Index-Expression] ...

روش دیگر تعریف آرایه های چند بعدی
آرایه های دو بعدی را می توان آرایه ای از آرایه ها تعریف کرد.

مثال :

```
typedef char string10[10];
string10 student[20];
```

سطر اول می گوید که string10 ، آرایه ای ۱۰ عنصری از نوع char است.
سطر دوم آرایه ای دو بعدی با ۲۰۰ خانه ایجاد می کند. (۲۰×۱۰)