بنام خدا

سری های آموزشی آشنایی با زبان برنامه نویسی C++

قسمت اول : مبانی C++

ویرایش : 2

آشنایی مقدماتی با زبان C++ :

C++ در اوایل دهه ی 1980 در آزمایشگاه T&AT توسط **بیارنه استراس تروپ (دانمارکی)** به منظور الحاق شیوه ی شی گرایی در زبان برنامه نویسی C طراحی گردید ...

شروع به برنامه نویسی C++:

اولین کاری که باید در برنامه نویسی c++ انجام دهید نوشتن تابع main است . تابع main تابع اصلی برنامه است . کد هایی که در داخل تابع main نوشته می شوند در ابتدای برنامه اجرا می شوند .
صورت کلی تابع main به صورت زیر است :

```
int main()
 {

 دستورات

 return 0;

 }
```

متغیر :

خانه هایی از حافظه هستند که می توانند مقادیر مختلفی را بر اساس نوع آنها در خود نگهداری کنند .

متغیر ها و نحوه ی مقدار دهی به آنها در C++ به این صورت تعریف می شوند :

```
Variable type        variable name;

variable type        variable name1,variable name2,…,variable N;

Variable type        variable name=variable value;
```

مثال : در زیر مشاهده می کنید که متغیر majidonline و majid از نوع int تعریف شده اند و همچنین برای متغیر sadeghkhafan که از نوع int تعریف شده است مقدار اولیه ی 20 نسبت داده شده است .

```
int main()
 {
 int majidonline,majid;

 int sadeghkhafan=20;

 Return 0;
 }
```

*توجه : در طول یک برنامه مقدار یک متغیر می تواند تغییر کند .

در زیر انواع متغیر ها در C++ را مشاهده می کنید :

نوع داده	حافظه (بایت)	محدوده
char	۱	۱۲۸- تا ۱۲۷
signed char	۱	۱۲۸- تا ۱۲۷
unsigned char	۱	۰ تا ۲۵۵
int	۲	برای سیستم ۱۶ بیتی بین ۳۲۷۶۸- تا ۳۲۷۶۷
unsigned int	۲	برای سیستم ۱۶ بیتی بین ۰ تا ۶۵۵۳۵
short int	۲	۳۲۷۶۸- تا ۳۲۷۶۷
unsigned short int	۲	۰ تا ۶۵۵۳۵
long int	۴	۲۱۴۷۴۸۳۶۴۸- تا ۲۱۴۷۴۸۳۶۴۷
unsigned long int	۴	۰ تا ۴۲۹۴۹۶۷۲۹۵
float	۴	اعداد اعشاری کوچک
double	۸	اعداد اعشاری بزرگ
long double	۱۰	اعداد اعشاری خیلی بزرگ

آرایه های یک بعدی : با طور کلی آرایه ها نوعی پیشرفته تر از متغیر ها هستند که می توانند چندین مقدار را در خود ذخیره کنند .

در مورد آرایه ها در قسمت 6 بطور مفصل تری صحبت خواهیم کرد.

ثابت ها : همانطور که از نام آنها معلوم است ، مقادیری هستند ثابت که در برنامه برای راحتی بعضی کار ها از آنها استفاده می کنیم .
برای تعریف ثابت ها از کلمه کلیدی #define استفاده می شود .

صورت کلی تعریف ثابت ها به صورت زیر است :

```
#define   نام ثابت   مقدار ثابت ;
```

مثال : در زیر ثابت tel با مقدار 110 تعیین شده است :

```
#define tel 110;
```

توجه داشته باشید که مقدار یک ثابت در طول برنامه تغییر نمی کند .

سازمان برنامه ها در C++ :

مانند زبان C هر برنامه ی C++ در چند فایل گسترده می شود . نوعی از این فایل ها سرفایل نام دارند که دارای پسوند h. هستند و از آنها برای ذخیرهی اعلان ها استفاده می شود .
سرفایل ها خود نیز دو گروهند :
1- بعضی از سرفایل ها در خود سیستم تعریف شده اند . مانند : <iostream.h>
2- عده ی دیگری از سرفایل ها توسط کاربر تعیین می شود یعنی هرگاه بخواهیم کد های داخل یک فایل را به کدهای برنامه بیفزاییم، از آنهای استفاده می کنیم.

سر فایل ها با استفاده از دستور پیش پردازنده ی include در فایل های مربوطه گنجانده می شوند .

دستور include : هر گاه بخواه کد های داخل یک فایل را به برنامه بیفزاییم ، از دستور اینکداد (include) استفاده می کنیم .

صورت های کلی استفاده از دستور include به صورت زیر است :

```
#include <Parham>

#include "Parham"
```

تفاوت این دو حالت در این است که :

در حال اول کامپایلر (compiler یا همون چیزی که برنامه رو تجزیه و تحلیل کرده و به اجرا در میاره) در دایرکتوری تعریف شده ی include ها دنبال فایل می گردد اما در حالت دوم کامپایلر در داخل دایرکتوری جاری دنبال فایل Header می گردد.
(اگر چیز زیادی متوجه نشدید ، نگران نباشید ! مطمئن باشید در ادامه با برنامه های نمونه و آزمایشات مختلف متوجه این تفاوت ها خواهید شد .)

دستورات ورودی و خروجی :

یکی از تفاوت های مهم C و C++ در همین قسمت دستورات ورودی و خروجی (I/O) می باشد که بجا اینکه مانند C از دستورات printf و scanf استفاده کند ، از دستورات cout (بخوانید : سی اوت) و cin (بخونید : سی این) استفاده می کند .

ساختار کلی برای استفاده از دستورات ورودی در C++ بصورت زیر است :

```
cin>>value1>>value2>>value3...;
```

برای مثال در کد زیر کامپایلر با دیدن دستور cin منتظر ورود مقادیری مانند mambolearn و Ahmadzadeh می باشد :

```
#include <iostream.h>

int main()
   {

   int mambolearn,Ahmadzadeh;
   cin>>mambolearn>>Ahmadzadeh;

   return 0;
   }
```

ساختار کلی برای استفاده از دستورات خروجی در C++ بصورت زیر است :

```
cout<<value1<<value2<<value3...;
```

برای مثال در کد زیر کامپایلر با دیدن دستور cout مقادیری متغیری مانند Mahmoodi را چاپ خواهد کرد :

```
#include <iostream.h>

int main()
   {

   int Mahmoodi=123;
   cout<<Mahmoodi;
   }
```

توجه : در این ساختار ، برای مثال mahmoodi دو متغیر هستند که درواقع مقادیر داخل آنها چاپ می شود .یعنی :

```
123456789
```

مثال : ابتدا متغیر nomre را از نوع کاراکتری می سازیم و بعد محتوای آنرا در جمله ای به کار می بندیم . به دستورات زیر توجه کنید :

```
#include <iostream.h>

int main()
   {
   int nomre;

   cout<<"Lotfan yek nomre vared
konid !! (az 20)";

   cin>>nomre;

   cout<<"Sadegh e Jedari is a
"<<nomre<<"  boy !";

   return 0;
   }
```

در این قسمت ابتدا یک متغیر با نام nomre ساخته ایم . بعد در خروجی چاپ کرده ایم که " لطفا یک نمره نام وارد کنید " . سپس نمره ی مربوط به او را در داخل متغیرnomre قرار داده ایم . و بعد از آن آن را همراه با یک متن چاپ می کنیم . برای مثال اگر نمره ی وارد شده 0 باشد ، در خروجی خواهیم داشت :

```
Sadegh e Jedari is a 0 boy !
```

شاید پرسید این عبارت آخریه چیه ؟ (return 0 ;)

در اون بالا در تعریف main نوشتیم ( )int main این یعنی خروجی این تابع int هست و با دستور return 0 ، مقدار 0 رو به عنوان خروجی تابع در نظر گرفتیم. حالا اگر نخواهید دستور return رو به کار ببرید ، باید در تعریف main به جای نوشته قبلی بنویسید ( )void main.
(این قسمت مربوط به مبحث تابع هاست که فعلا لازم نیست این ها متوجه شوید ! فقط در همین حد بدانید چرا و برای چه این عبارت را نوشته ایم .)

چند تا از علامات بدرد بخور:

توضیح	علامت
رفتن به یک خط پایین تر : میتوان گفت که همان کار دکمه enter را در نرم افزار word انجام میدهد	\n
توضیح برای کد مورد نظر فقط برای یک سطر : با استفاده از این می توانید مانند نمونه ی نشان داده شده عباراتی را در مقابل کد مورد نظرتان بنویسید تا در مشاهده های بعدی راحت تر و سریعتر کار بکنید . مثا اینکه این کدی که انجا نوشته اید برای چه است و ...	//
توضیح برای کد مورد نظر به تعداد سطر های دلخواه : این هم نوعی از همان علامت بالاست که در توضیح های طولانی از آن استفاده می شود و طرز استفاده ی آن با قبلی فرق دارد.	/* */

برای متوجه شدن کابرد دقیق هریک از این نشانه ها به کد زیر توجه فرمایید :

```
#include <iostream.h>
int main()
{
int nomre; // here we made a variable !
cout<<" Lotfan\n";
cout<<" nomreye riazi e khod ra \n";
cout<<" vared konid !! \n";
cin>>nomre; /* here we will give the client a number that
shows his or her mark in mathmatics
and we will use it in futur */
cout<<"Your mark is not very bad ! : "<<nomre<<" \n";
return 0;
}
```

در نهایت اگر نمره ی وارد شده 20 باشد ، در نهایت این چاپ خواهد شد :

```
 Lotfan

 nomreye riazi e khod ra

 vared konid !!

20

Your mark is not very bad ! : 20
```

یک مثال دیگر :

```
#include <iostream.h>
int main()
{
cout<<"Welcome to \n\n\nC++ farsi
\ne-learning!!\n";
return 0;
}
```

نتیجه :

```
Welcome to



C++farsi

e-learning!!
```

پایان قسمت اول !

نویسنده : دانیال خشابی
ویرایش و صحت مطالب : نوید مردوخ روحانی

www.mrh.ir
www.majidonline.com

کپی رایت :: تیر 1385
آخرین تغییرات : مهر 85
ارائه ی این مطلب فقط با ذکر منبع و دو سایت بالا مجاز است !

بنام خدا

سری های آموزشی آشنایی با زبان برنامه نویسی C++

قسمت دوم : مبانی C++

ویرایش : 1

عملگر های ریاضی در C++ :

عملگر های ریاضی در C++ به صورت زیر تعریف می می شوند :

علامت عملگر در زبان C++	نام عملگر
+	جمع
-	تفریق
*	ضرب
/	تقسیم
%	باقیمانده

بدیهی است که حاصل این عملگر ها می تواند صحیح یا اعشاری باشد .

در زبان C++ برای انجام عملیات جمع و تفریق با 1 دو عملگر دیگر نیز وجود دارد که هدف از ایجاد آنها عموم راحت تر کردن کار بوده است.

علامت عملگر در زبان C++	نام عملگر
++	جمع خود محتوای متغیر با 1
--	کم کردن 1 واحد از محتوای متغیر

نکته ی مهم در استفاده از این دو عملگر اخیر توجه به نقش های مختلف آن در اولویت های مختلف است .
یعنی اینکه a++ با ++a تفاوت خواهد داشت .و عملگر -- نیز به همین ترتیب است .

برای اینکه بهتر متوجه منظورم بشوید به مثال های زیر توجه بفرمایید :

```
#include <iostream.h>

int main()
   {
   int c;

c = 5;
cout << c << endl;     // print 5
cout << c++ << endl;   // print 5 then postincrement
cout << c << endl << endl; // print 6

c = 5;
cout << c << endl;    // print 5
cout << ++c << endl;  // pre increment then print 6
cout << c << endl;    // print 6

return 0;
   }
```

در خروجی برنامه ی فوق خواهیم داشت :

```
5
5
6

5
6
6
```

توجه : endl خط فرمان را سطر پایین تر می آورد . فرق آن با \n در این است که \nدر داخل " " نوشته می شود اما endl خودش به تنهایی نگاشته می شود .

مسئله ی 1 : برنامه ای بنویسید که دو عدد را گرفته و مجموع آنها را در خروجی چاپ کند .

```
// program for suming up twe numbers !
#include <iostream.h>
int main()
{
int num1,num2,sum;
cout<<"please enter the first integer number ! : " ;
cin>>num1;
cout<<"please enter the second integer number ! : ";
cin>>num2;
sum=num1+num2;
cout<<" The sum is "<<sum<<" \n";
return 0;
}
```

در بحث عملگر های ریاضی اولویت بعدی عملگر ها خود نیز مسئله ی مهمی به شمار می آید. اولویت عملگر ها به صورت زیر است :

عملگر ها	ترتیب ی اولویت
پرانتز ها ( )	1
* یا /	2
%	3
+ یا -	4

چند نمونه دیگر از عملگر های ریاضی که برای آسانی کار ارائه شده اند :

شکل دیگر عملگر	مثال	علامت عملگر
a=a+5	a+=5	+=
a=a-5	a-=5	-=
a=a*5	a*=5	*=
a=a/5	a*=5	/=
a=a%5	a%=5	%=

برای اینکه با طرز کار با این عمگر ها آشنا شوید ، به مثال زیر توجه فرمایید :

```
#include <iostream.h>

int main()
{
int a;

a=6;
a+=6;
cout<<a<<endl;
a=6;
a-=6;
cout<<a<<endl;
a=6;
a*=6;
cout<<a<<endl;
a=6;
a/=6;
cout<<a<<endl;
a=6;
a%=6;
cout<<a<<endl;

return 0;
}
```

مسئله ی 2 : برنامه ای بنویسید که حقوق پایه و تعداد فرزندان یک کار گر را از ورودی گرفته و حقوق کل وی را از فرمول زیر بدست آورد :

حقوق کل = حقوق پایه + فرزندان * 10

جواب :

```
#include <iostream.h>

int main()
   {
      int  child,salary,wholesalary;

      cout<<" Enter yor child's number: ";

      cin>>child;

      cout<<" Enter your salary :";

      cin>>salary;

      wholesalary=salary+10*child;

      cout<<"your whole salary is : "<<wholesalary<< endl;

return 0;
   }
```

آرایه های چند وجهی :

آرایه های چند بعدی ، نوع پیشرفته تری از آرایه ها هستند که می توانند اطلاعات بیشتری را در خود ذخیره کنند . و در عوض نیز کار های پیشرفته تری را انجام دهند . (فعلا تا همین کافی است ! در قسمت های بعد بیشتر درباره کاربرد های اینها آشناخواهید شد .)

ساختار کلی برای استفاده از آرایه های چند بعدی به صورت زیر است :

```
 Type Arayname
[size1][size2]...[sizeN];
```

برای مثال در در نمونه ی زیر آرایه ی kami دارای 2 بعد می باشد که دارای 2 سطر و ستون است . می خواهیم مقادیر اولیه ی این آرایه را بگیریم و 1 مقدار به آنها اضافه کنیم و بعد مقادیر آرایه ی kami را چاپ کنیم :

```
#include <iostream.h>

int main()
   {

int kami[2][2];

cin>>kami[1][1];
cin>>kami[2][1];
cin>>kami[1][2];
cin>>kami[2][2];

kami[1][1]++;
```

```
kami[2][1]++;
kami[1][2]++;
kami[2][2]++;

cout<<"kami [1][1]:
"<<kami[1][1]<<endl;
cout<<"kami [2][1]:
"<<kami[2][1]<<endl;
cout<<"kami [1][2]:
"<<kami[1][2]<<endl;
cout<<"kami [2][2]:
"<<kami[2][2]<<endl;

return 0;

   }
```

عملگر های منطقی :

در جدول زیر انواع عملگر های مقایسه ای یا منطقی را مشاهده می فرمایید:

نام عملگر	علامت عملگر در زبان C++
AND	&&
OR	||
NOT	!
کوچکتر	<
کوچکتر یا مساوی	<=
بزرگتر	>
بزرگتر یا مساوی	=>
مقایسه	==
نا مساوی	!=
انتصاب شرطی	?:

بیشترین استفاده ی از عملگر های منطقی یا مقایسه ای در ساختار های شرطی است . برای آشنایی با ساختار های شرطی به مطلب بعدی توجه فرمایید .

ساختار های تصمیم گیری در C++ :

1- دستور if-else: زمانی از این ساختار استفاده می شود که شرط ها کم باشد.
2- دستور switch-case: زمانی از این ساختار استفاده می شود که تعداد تصمیم گیری ها زیاد باشد .

ساختار شرطی if :

ساختار کلی شرطی if به صورت زیر است :

```
If (condition1)
{
دستورات قسمت اول ;
}
Else
{
دستورات قسمت دوم;
}
```

توجه کنید که ساختار بالا یک ساختار کلی می باشد و ممکن است در حالات شکل آن تغییر کند . مثل حالات زیر ، آنها را به خاطر بسپارید ! :

1- دستورات قسمت اول یا دستورات قسمت دوم یا هردو فقط شامل ی دستور باشند :

```
If (condition1)
دستور قسمت اول ;
Else
{
دستورات قسمت دوم;
}
```

```
If (condition1)
{
دستورات قسمت اول ;
}
Else
دستور قسمت دوم;
```

```
If (condition1)
دستور قسمت اول ;
Else
دستور قسمت دوم;
```

2- در بعضی مواقع استفاده از قسمت دوم این ساختار تصمیم گیری (else) لازم نیست . یعنی اینکه شما فقط قصد استفاده از قسمت ifرا دارید :

```
If (condition)
{
دستورات ;
}
```

توجه کنید که در صورتی هم که دستورات شما شامل فقط یک دستور باشد ، لازم نیست که از دو آکولاد استفاده کنید :

```
If (condition)
دستور ;
```

مسئله 3 : برنامه ای بنویسید که 2 عدد دریافت کند و بزرگترین آنها را بنویسد .
جواب :

```
#include <iostream.h>
int main()
   {
      int  num1,num2;
      cout<<" Enter your  first  number: ";
      cin>>num1;
      cout<<" Enter your second number : ";
      cin>>num2;

      if (num1>num2)
      cout<<num1<<" is bigger ! ";
      else
      cout<<num2<<" is bigger ! ";

return 0;
   }
```

مسئله 4 : برنامه ای بنویسید که یک عدد را از ورودی گرفته و مشخص کند که آن عدد زوج است یا فرد .

جواب :

```
#include <iostream.h>
int main()
   {
      int  num;
      int rest;

      cout<<" Enter your  number: ";
      cin>>num;
      rest=num % 2;


      if (rest!=0)
      cout<<num<<" is  fard(odd) ! ";
      else
      cout<<num<<" is  zoj(even) ! ";

return 0;
   }
```

تمرینات :

1- (مسئله ی 5) : برنامه ای را بنویسید که سه عدد را گرفته و برگترین آنها را تعیین کند . (فقط با دو if)
2- (مسئله ی 6) : برنامه ای بنویسید که سه عدد را گرفته و بزرگترین و کوچکترین آنها را تعیین کند . (فقط با سه if)
3- (مسئله ی 7) : برنامه ای بنویسید که سه عدد را گرفته و آنها را به ترتیب بزرگتر تا کوچکتر مرتب کرده و در خروجی چاپ کند .

« جواب مسائل در قسمت بعدی »
(اول خودتون فکر کنید .اگر به نتیجه نرسیدید ، در قسمت بعدی جواب هارو بررسی کنید !)

پایان قسمت دوم!

نویسنده : دانیال خشابی
ویرایش و صحت مطالب : نوید مردوخ روحانی

www.mrh.ir
www.majidonline.com

کپی رایت :: تیر 1385
ارائه ی این مطلب فقط با ذکر منبع و دو سایت بالا مجاز است !

بنام خدا

سری های آموزشی آشنایی با زبان برنامه نویسی C++

قسمت سوم : مبانی C++

(ساختار های شرطی و گردشی)

ویرایش : 1

جواب تمرینات شماره ی قبل :

تمرین 1 : برنامه ای را بنویسید که سه عدد را گرفته و برگترین آنها را تعیین کند . (فقط با دو if)

```
#include <iostream.h>
int main()
   {
      int  a,b,c,max;



      cout<<" Enter your  numbers ";
      cin>>a>>b>>c;
    max=a;
      if (b>max)
            max=b;
      if (c>max)
            max=c;
    cout<<" The max is " <<max<<endl;

return 0;
   }
```

تمرین 2 : برنامه ای بنویسید که سه عدد را گرفته و بزرگترین و کوچکترین آنها را تعیین کند . (فقط با سه if)

```
#include <iostream.h>
int main()
   {
      int  a,b,c,max,min;
      cout<<" Enter your  numbers ";
      cin>>a>>b>>c;
    max=a;
      min=a;
      if (b>a)
      {
            max=b; min=a;
      }
      else
      {
            max=a; min=b;
      }
      if (c>max)
            max=c;
      if (c<min)
            min=c;

    cout<<" The max is " <<max<<endl;
   cout<<" The min is " <<min<<endl;
return 0;
   }
```

تمرین 3: برنامه ای بنویسید که سه عدد را گرفته و آنها را به ترتیب بزرگتر تا کوچکتر مرتب کرده و در خروجی چاپ کند .

```
#include <iostream.h>
int main()
   {
   int a,k,b,c;
   cout<<"enter 3 numbers :  ";
   cin>>a>>b>>c;
   if ( b>a)
   {
         k=a; a=b; b=k;
   }
    if ( c>a)
   {
         k=a; a=c; c=k;
   }
    if ( c>b)
   {
         k=b; b=c; c=k;
   }
   cout<<"max number is : "<<a<<endl;
   cout<<"mid number is : "<<b<<endl;
   cout<<"min number is : "<<c<<endl;

return 0;
   }
```

ساختار شرطی swich-case:

زمان که تعداد شرط ها زیاد باشد از این ساختار استفاده خواهیم کرد .
ساختار کلی آن به شکل زیر است :

```
switch (variable)                                  } 1
  {
        case 'value1-1':                           }
        case 'value2-1':                           }
        ...                                        }
        case 'value N-1':                          } 2
                                                   }
              یک یا چند دستور;                    }
              break;                               }

        case 'value1-2':                           }
        case 'value2-2':                           }
        ...                                        }
        case 'value N-2':                          } 3
                                                   }
              یک یا چند دستور;                    }
              break;                               }

        ...

        case 'value1-N':                           }
        case 'value2-N':                           }
        ...                                        }
        case 'value N-N':                          } 4
                                                   }
              یک یا چند دستور;                    }
              break;                               }

       default:                                    }
              دستور یا دستورات;                   } 5
}
```

توضیح :
ساختار کلی دستور شرطی بدین صورت است که ابتدا یک متغیر را در نظر می گیرد = switch (variable) - (قسمت 1) ، سپس شرط می کند اگر مقدار این متغیر برای مثال value1-1 یا value1-2 ، value1-3 یا ... باشد ، یک یا چند دستور را اجرا کند (قسمت 2) . یا اگر مقدار این متغیر برای مثال value2-1 یا value2-2 ، value2-3 یا ... باشد ، یک یا چند دستور دیگر را اجرا کند (قسمت 3) . یا ... (قسمت 4) . که این قسمت ها توسط دستور break; از هم جدا می شوند . در نهایت یک قسمت داریم که اگر مقدار متغیر هیچ یک از مقادیر شرط شده نبود ، آن سری از دستورات را اجرا کند (قسمت 5) .

مسئله 8 : برنامه ای بنویسید که اگر کاربر هر یک از حروف A B C D را وارد کند بنویسد : Big character
اگر حروف a b c d را وارد کند بنویسد : little Character
اگر اعداد 1 تا 4 را وارد کند بنویسد : a figure !
اگر غیر از این حروف را وارد کند ، بنویسد : unknown character !

جواب :

```
#include <iostream.h>
int main()
   {
      char  a;
      cout<<" Enter your  selected character ! :  ";
      cin>>a;

switch (a)
{
      case 'A':
      case 'B':
      case 'C':
      case 'D':
            cout<<"Big character !  ";
            break;
      case '1':
      case '2':
      case '3':
      case '4':
              cout<<" a figure ! ";
            break;
      case 'a':
      case 'b':
      case 'c':
      case 'd':
              cout<<" little character  ! ";
            break;
              default:
            cout<<"unknown character ! ";
            break;
}

return 0;
   }
```

ساختار های حلقه ها در زبان ++C :

در زبان ++C سه نوع ساختار حلقه (گردش) وجود دارد :

1- for
2- do while
3- while

در ادامه به بررسی هر سه حلقه خواهیم پرداخت .

ساختار حلقه ی for :

ساختار کلی حلقه ی for به صورت زیر است :

```
for( مقدار تغییر متغیر ;مقدار نهایی ;مقدار اولیه )
{
        دستور یا دستورات
}
```

توجه : در صورت مجموعه دستورات شما فقط شامل یک دستور باشد می توانید از آکولاد ها صرف نظر کنید .

مسئله 9 : برنامه ای بنویسید که تا اعداد طبیعی کوچکتر از 100 را چاپ کند .

```
#include <iostream.h>
int main()
   {

      cout<<"figures which are less than 100 : \n";

      for(int i=0; i<100; i++)
      {
            cout<<i<<endl;
      }
return 0;
   }
```

مسئله ی 10 : برنامه ای بنویسید که مجموع اعداد 1 تا 99 را بنویسد .

```
#include <iostream.h>
int main()
   {

      int s;
      cout<<"majmo is : ";

      for(int i=0; i<100; i++)
      {
            s=s+i;
      }

      cout<<s;

return 0;
   }
```

مسئله ی 11 : برنامه ای بنویسید که اعداد زوج بین 100 و 1000 را چاپ کند .

```
#include <iostream.h>
int main()
   {

      int s,k;
      cout<<"even numbers between 100 and 1000 are : ";

      for(int i=100; i<1001; i++)
      {
            k=i%2;
            if (k==0)
            cout<<i<<" ";
      }

return 0;
   }
```

مسئله 12 : برنامه ای بنویسید که حاصلضرب اعداد 5 تا 10 را در خروجی چاپ کند .

```
#include <iostream.h>
int main()
   {
      long int  p;

      p=1;
      for(int i=5; i<10; i++)
      {
            p=p*i;

      }

       cout<<" zarbe adade 5 ta 10 : " <<p;


return 0;
   }
```

ساختار حلقه ی while :

```
while(  شرط اجرای حلقه  )
{
 دستورات برنامه
}
```

مسئله ی 13 :برنامه ای بنویسید که دنباله ی زیر را تا عدد 9999 چاپ کند :
10 30 50 70 90 ...

```
#include <iostream.h>
int main()
   {
      long int a;

      a=10;
      while (a<9999){
      cout<<a<<"    ";
      a+=20;
      }

return 0;
   }
```

مسئله ی 14 : برنامه ای بنویسید که تا تعدادی عدد مثبت را گرفته و مجموع آنها را حساب کند . (شرط پایان کار ، وارد کردن عدد صفر از طرف کاربر خواهد بود) .

```
#include <iostream.h>
int main()
   {
      long int s,a;
      cout<<"enter your numbers :  "<<endl;
      cin>>a;
      s=a;
      while (a>0){
      cin>>a;
      s+=a;
      }
      cout<<"Sum is  "<<s;


return 0;
   }
```

ساختار حلقه ی do while :

```
do{
 دستورات برنامه
}while( شرط اجرای کد حلقه )
```

فرق اساس حلقه ی do while با while در این است که در حلقه ی فوق ، دستورات برای یک بار بدون توجه به شرط برنامه اجرا می شوند .

برای مثال مسئله 14 را یکبار دیگر با حلقه ی do while می نویسیم :

```
#include <iostream.h>
int main()
   {
      int s,a;
      cout<<"enter your numbers :  "<<endl;
      s=0;

      do{
      cin>>a;
      s=a+s;
      } while (a>0);

      cout<<"Sum is  "<<s;


return 0;
   }
```

مسئله ی 15 : برنامه ای بنویسید که تعدادی عدد مثبت را از ورودی گرفته و بزرگترین و کوچکترین آنها را تعیین کند . (شرط پایان کار وارد کردن عدد صفر است)

```
#include <iostream.h>
int main()
   {
      int s,max,min,a;
      cout<<"enter your numbers :  "<<endl;
      cin>>a;
      max=a; min=a;
      while (a>0){
            if (a>max)
            max=a;
            if (a<min)
            min=a;
      cin>>a;
      }

      cout<<"max is  "<<max;
      cout<<"min is  "<<min;

return 0;
   }
```

تمرینات :

1- (مسئله ی 16): برنامه ای بنویسید که دستور فاکتوریل را انجام دهد . یعنی اینکه عددی را از ورودی گرفته و فاکتوریل آنرا را حساب کند .
راهنمایی :

0!=1 و a!= 1*2*3*...*a

2- (مسئله ی 17):برنامه ای بنویسید که 10 عدد از ورودی گرفته و بزرگترین آنها را مشخص کند .

3- (مسئله ی 18): برنامه ای بنویسید که عددی را گرفته و مشخص کند که آیا عدد اول است یا نه .

4- (مسئله ی 19): برنامه ای بنویسید که عددی را از ورودی گرفته و مشخص کند که عدد گرفته شده تام است یا نه .
(راهنمایی : عدد تام عددی است که مجموع مقسوم علیه های کوچکتر از خودش ، برابر خودش باشد)
5- (مسئله ی 20): برنامه ای بنویسید که اعداد اول 1 تا 5000 را چاپ کند .
6- (مسئله ی 21):برنامه ای بنویسید که عددی را گرفته و مقسوم علیه های آنرا چاپ کند . (می توانید برنامه را گسترش دهید تا تعداد و مجموع مقسوم علیه ها را هم چاپ کند .)
7- (مسئله ی 22):برنامه ای بنویسید که مجموع مضارب 5 را بین 1 و 100 چاپ کند .
8- (مسئله ی 23):برنامه ای بنویسید که دو عدد را از ورودی بگیرد و اعداد بین آنها را چاپ کند .
9- (مسئله ی 24):برنامه ای بنویسید که تعداد مضارب 7 و 5 را در بازه ی بین 1 تا 10000 چاپ کند .
(توضیح : اعداد مورد نظر هم باید مضرب 7 باشد و هم مضرب 5)
10-(مسئله ی 25):برنامه ای بنویسید که 10 عدد را گرفته و میانگین و حاصلجمع آنها ر ا به ما بدهد .

11- (مسئله ی 26):دنباله ی فیبوناچی به صورت زیر است :

1,1,2,3,5,8,13,21...

این دنباله به صورت زیر تعریف می شود :

$F_1 = 1$	$F_2 = 1$	$F_n = F_{n-1} + F_{n-2} \qquad n \geq 3$

برنامه ای بنویسید که با دریافت مقدار صحیح n ، مقدار F_n را برای دنباله ی فیبوناچی چاپ کند .

12- (مسئله ی 27) : برنامه ای بنویسید که تعدادی عدد گرفته و مجموع مربعات آن را حساب کند .

پایان قسمت سوم!

نویسنده : دانیال خشابی
ویرایش و صحت مطالب : نوید مردوخ روحانی

www.mrh.ir
www.majidonline.com

کپی رایت :: مرداد 1385
ارائه ی این مطلب فقط با ذکر منبع و دو سایت بالا مجاز است !

بنام خدا

سری های آموزشی آشنایی با زبان برنامه نویسی C++

قسمت چهارم : کار با توابع 1

توابع کتابخانه ای

ویرایش : 1

جواب تمرینات شماره ی قبل :

تمرین 1 :

```
#include <iostream.h>
int main()
   {
      int i,n,a;
      long int p;
      cout<<"structur is p=n! .  enter n : "<<endl;
      cin>>n;
      p=1;
      i=1;
      if(n==0)
      cout<<"  0!  is  1  ";
      else
      {
             while (i<=n)
             {
             p=p*i;
             i++;
             }

      cout<<n<<"!  is  "<<p;
      }

return 0;
   }
```

تمرین 2 :

```
#include <iostream.h>
int main()
   {
      int i,max,n,a;
      cout<<"enter  10 numbers  : "<<endl;
      cin>>n;
      max=n;
      i=1;
             while (i<10)
             {
                    if (n>max)
                    max=n;
             cin>>n;
             i++;
             }

      cout<<max<<"  is max number  ";
return 0;
   }
```

تمرین 3 :

```
#include <iostream.h>
int main()
   {
      int i,p,mod,n,a;
      cout<<"enter  number  : "<<endl;
      cin>>n;
      i=1;
      p=0;
             while (i<=n)
             {
             mod=n%i;
                    if (mod==0)
                    p=p+1;
             i++;
             }
      if (p==2)
      cout<<"  aval ast ! ";
      else
cout<<"  aval nist ! ";
return 0;
   }
```

تمرین 4 :

```
#include <iostream.h>
int main()
   {
      int i,p,mod,n;
      cout<<"enter  number  : "<<endl;
      cin>>n;
      i=1;
      p=0;
             while (i<n)
             {
             mod=n%i;
                    if (mod==0)
                    p=p+i;
             i++;
             }
      if (p==n)
      cout<<" taam ast ! ";
      else
      cout<<"  taam nist ! ";
return 0;
   }
```

تمرین 5 :

```
#include <iostream.h>
int main()
   {
      int i,p,mod,n,a;


       while (n<=50)
       {
             while (i<=n)
             {
             mod=n%i;
                 if (mod==0)
                 p=p+1;
             i++;
             }
      if (p==2)
        cout<<n<<endl;
      n++;
      i=1;
      p=0;

      }
return 0;
   }
```

تمرین 6 :

فقط مقسوم علیه ها :

```
#include <iostream.h> //tedad e magsoom aleyh ha !
int main()
   {
      int i,p,mod,n,a;
      cout<<"enter  number  : "<<endl;
      cin>>n;
      i=1;
             while (i<=n)
             {
             mod=n%i;
                   if (mod==0)
                   cout<<i<<" - ";
             i++;
             }

             cout<<endl;
return 0;
   }
```

مقسوم علیه ها ، تعداد و مجموع آنها :

```
#include <iostream.h>
int main()
   {

      int i,mod,n,a;
      cout<<"enter  number  : "<<endl;
      cin>>n;
      i=1;
      int s=0; //majmo
      int p=0; //tedad
      cout<<"Magsoom aleyh ha are : " ;

             while (i<=n)
             {
             mod=n%i;
                    if (mod==0)
                    {
                    cout<<i<<" - ";
                    p++;
                    s+=i;
                    }
             i++;
             }
 cout<<endl<<"majmo magsoom aleyh ha : "<<s<<endl;
 cout<<"tedad e magsoom aleyh ha : " <<p<<endl;
             cout<<endl;

return 0;
   }
```

تمرین 7 : مجموع مضارب 5 بین 1 و 100 :

```
#include <iostream.h>
int main()
   {
      int i,p,mod,n,s,a;
      cout<<"enter  number  : "<<endl;
      cin>>n;
      i=1;
      p=0;
      s=0;
             while (i<=n)
             {
             mod=n%i;
                    if (mod==0)
                    {
                    cout<<i<<" - ";
             s=s+i;
                    p++;
                    }
                    i++;
             }

             cout<<endl<<"tedad :  "<<p<<endl;
             cout<<"majmo : "<<s<<endl;
return 0;
   }
```

تمرین 8 : دو عدد از ورودی گرفته و اعداد بین آنها را چاپ کند:

```
#include <iostream.h>
int main()
      {

      int i,a,b;
      cout<<"enter the first number number : ";
      cin>>a;
      cout<< "enter the second number : " ;
      cin>>b;
      if (b>a)
        {
        i=b;
        b=a;
        a=i;
        }
      i=b+1;
      cout<<endl<<"Adad e bein : " ;

             while (i<a)
             {
             cout<<i<<" , ";
             i++;
             }
      return 0;
      }
```

تمرین 9 :

```
#include <iostream.h>
int main()
   {
      int i,n;
      i=1;
      int p=0; //tedad
             while (i<=10000)
             {
                    if ((i%7==0)&&(i%5 ==0))
                    p++;

             i++;
             }
             cout<<"tedad : " <<p<<endl;

return 0;
   }
```

تمرین 10 :

```
#include <iostream.h>
int main()
   {
      int i,n;
      i=1;
      int s=0; //majmo
             while (i<=10)
             {
                  cin>>n;
                  s+=n;
                  i++;
             }
             cout<<"majmo : " <<s<<endl;
             cout<<"miangin : " <<(s/10)<<endl;

return 0;
   }
```

تمرین 11: دنباله ی اعداد فیبونچی :

```
#include <iostream.h>
int main()
   {
      int f1,f2,f,i,n;
      i=1;
      f1=1;
      f2=1;
      f=1;
        cin>>n;
             while ((i<n-1)&&(n>2))
             {
                  f=f2+f1;
                  f1=f2;
                  f2=f;
                  i++;
             }
             cout<<"N="<<f;

return 0;
   }
```

تمرین 12 : مجموع مربعات تعداد ی عدد را در خروجی چاپ کند :

```
#include <iostream.h>
int main()
   {
      int s=0,i,n,a;
      i=1;
      cout<<"enter tedad : ";
      cin>>n;
      cout<<endl<<"enter numbers : ";
             while (i<=n)
             {
                  cin>>a;
                  s+=a*a;
                  i++;
             }
             cout<<"Majmo morabaat :  S="<<s;
return 0;
   }
```

توابع و کلاس ها :
در این قسمت مبحث تابع را بیشتر فرا خواهید گرفت بطوریکه بوسیله ی توابعی که معرفی خواهیم ، خوهید توانست قدرت برنامه نویسی خود را فوق العاده افزایش خواهید داد .

تابع : دستور یا مجموعه ای از دستورات است که می تواند بصورت پیش فرض و یا بصورت نوشته شده توسط خود کاربر باشند ، که عملیات خاصی را بر روی متغیری که دریافت می کنند ، انجام می دهند .

همانطور که گفته شد توابع دو گروهند :
1- توابعی پیش ساخته و از قبل به همراه کمپایلر C++ ارائه شده اند که به آنها توابع کتابخانه ای می گویند.مثل تابع cos (گرفتن کسینوس یک زاویه)
2- توابعی که کاربر بر حسب ضرورت و نیاز آنها را می نویسد و در برنامه ی خود استفاده می کند .

البته در مورد استفاده از توابع کتابخانه ای باید توجه کنیم که هر یک از توابع ، هریک در فایل مخصوصی جای داده شده اند که همانطور هم که در قسمت اول گفته شد به آنها فایل های سر آیند می گوییم که دارای پسوند h. هستند . در این فایل ها ، تمام اطلاعات لازم برای اجرای یک تابع خاص وجود دارد . به عنوان مثال برای استفاده از توابع ورودی و خروجی (cout , cin) باید سرفایل iostream.h را فراخوانی کنیم . یعنی همانطور که قبلا گفتیم در بالای کد خود ، به این شکل می نویسیم :

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

توابع کتابخانه ای :

توابع ریاضی : در جدول زیر لیستی از توابع ریاضی را مشاهده می فرمایید .
هر یک از توابع زیر یک یا چند مقدار را در داخل پرانتز می گیرند و بعد از عملیات خاصی آنرا به صورت مقداری خروجی می دهند .
مثلا : نمونه ای از کابرد تابع قدر مطلق را در زیر می بینید . به نحوه ی کار با آن توجه کنید :

```
int b=-9;
cout<<fabs(b); //will print 9 in output
```

شما می توانید بر اساس خلاقیت خود یک تابع را بصورت های گوناگونی بکار ببرید :

```
cout<<fabs(-9)<<endl<<; //will print 9 in output
```

یک نصیحت !
همیشه به یاد داشته باشید که در برنامه نویسی ، بهترین معلم شما خودتان هستید . تا وقتی که خودتان آستین بالانزده و خلاقیت نکرده اید امکان ندارد جایی برسید . سعی کنید همواره کنجکاو باشید. همه چیز را امتحان کنید تا بتوانید چیز های جدیدی را بدست آورید یا اینکه شک شبهه های خود را بر طرف کنید . در برنامه نویسی ، تجربه ، یک امتیاز بزرگ است . حتما لازم نیست ، هرکاری و آزمایشی که می کنید به خاطر بسپارید ، فقط آزمایش کنید و درس بگیرید ، خود به خود در آینده تاثیر خودش را می گذارد.
پس هیچ وقت از آزمایش کردن نترسید !

می توانید هریک از توابع زیر را هریک در برنامه ای مقدار دهی کرده و نتیجه ی هرکدام را چاپ کرده و در خروجی ببینید .

تابع	توضیح	سرفایل مربوط	مثال
fabs(a)	قدر مطلق عدد را محاسبه می کند.	math.h	fabs(2) is 2 fabs(-2) is 2
Acos(a)	آرک کسینوس عدد را حساب می کند.	math.h	Acos(1) is 0
Asin(a)	آرک سینوس عدد را حساب می کند.	math.h	Asin(1) is 90
Atan(a)	آرک تانژانت عدد را حساب می کنند.	math.h	Atan(1) is 45
Ceil(a)	کوچکترین عدد بزرگتر یا مساوی عدد را حساب می کند .	math.h	Ceil(1.2) is 2 Ceil(-1.2) is -1
Cos(a)	کسینوس عدد را حساب می کند.	math.h	Cos(0) is 1
Exp(a)	برای محاسبه ی توانی از عدد e (عدد نپر) : e^x	math.h	Exp(1.0) is 2.71828
Floor(a)	برای محاسبه ی بزرگترین عدد صحیح کوچکتر مساوی عدد	math.h	Floor(4.9) is 4
Log(a)	برای محاسبه ی لگاریتم اعداد با پایه ی عدد نپر (e)	math.h	
Log10(a)	برای محاسبه ی لگاریتم اعداد با پایه ی 10	math.h	
Pow(a,b)	برای به توان رساندن اعداد	math.h	Pow(2,5) is 32
Sin(a)	برای محاسبه سینوس اعداد	math.h	Sin( ) is
Sqrt(a)	برای محاسبه جذر اعداد	math.h	Sqrt(16) is 4
Tan(a)	برای محاسبه تانژانت اعداد	math.h	Tan( ) is
Fmod(a,b)	برای محاسبه باقیمانده ی عدد a بر b	math.h	Fmod(6,5) is 1

چند تابع بدرد بخور دیگر :

تابع	توضیح	سرفایل مربوط	مثال
Rand()	این تابع یک عدد تصادفی از 0 تا 32767 را چاپ خواهد کرد . این تابع هیچ مقداری برای ورودی نمی گیرید. برای بدست آوردن اعداد کاملا تصادفی از دستور radomize; قبل از این تابع استفاده می کنیم .	Stdlib.h	Rand(); is a number between 0 and 32767
Random(a)	از این تابع برای تولید یک عدد تصادفی از 0 تا a-1 استفاده می کنیم .	Stdlib.h	Random(8) Is a number 0,1,2,...,7
Getch(a)	می توان برای قرار دادن یک مقدار یک کاراکتری در یک متغیر استفاده کرد . زمانی که کامپایلر به این تابع می رسد منتظر وارد کردن یک کاراکتر از طرف کاربر می شود . به محض وارد شدن کاراکتر ، کامپایلر کار را ادامه می دهد .	Conio.h	در مثال های بعدی کاربرد هایی خواهیم دید .
Gotoxy(x,y)	برای قرار گرفتن نشانگر صفحه در مختصات نقطه ی مورد نظر	Conio.h	Gotoxy(13,40) نقطه از صفحه است .
Strlen(" ...")	برای شمارش تعداد کاراکتر های متن وارد شده	String.h	Strlen("majid online") is 12

**توجه : در زبان برنامه نویسی c++ بعضی از توابع ممکن است در ابتدا مقدار نگیرند یا خالی بگیرند ، در اینصورت حتما باید پارانتز ها را نوشت . مثل تابع main()که همیشه دو پرانتز را می نویسیم . در ادامه به چنین توابعی بیشتر خواهیم خورد .

تمرین : برنامه ای بنویسید که سه عدد را به عنوان اضلاع مثلث از ورودی بگیرد . با فرض اینکه عدد سوم وتر مثلث است ، به ما بگوید آیا تشکیل یک مثلث قائم الزاویه می دهد یا نه ؟

```
#include <iostream.h>
int main()
   {

      int a,b,c;
      cout<<"enter three sizes : "<<endl;
      cin>>b>>c>>a;
        if(sqrt(pow(b,2)+pow(c,2))==a)
             cout<< " YES !! " ;
        else
             cout<<"  NO  !! ";

return 0;
   }
```

تمرین : برنامه ای بنویسید که سه مقدار به عنوان ضرایب معادله ی درجه ی دوم گرفته و نتایج آن را در سه حالت ممکن در خروجی چاپ کند :

1- دو جواب
2- 1 جواب
3- بدون جواب

```
#include <iostream.h>
int main()
   {
  int  delta,a,b,c,x1,x2;

      cout<<" Enter your  numbers(a,b,c)  as  these structur :
ax^2+bx+c "<<endl;
      cin>>a>>b>>c;
      delta=b*b-4*a*c;
      if (delta>=0)
      {
      cout<<delta<<endl;
            if (delta==0)
                  {
                  cout<<"just 1 root !"<<endl;
                  x1=-b/2*a;
                  cout<<"The Only root is :"<<x1;
                  }
            else
                  {
                  cout<<"2 roots!"<<endl;
                  x1=(-b+sqrt(delta))/2*a;
                  x1=(-b-sqrt(delta))/2*a;
                  cout<<"first root : "<<x1<<endl;
                  cout<<"second root : "<<x2<<endl;
                  }
      }
      else
            cout<<"error ... No root!!!"<<endl;


return 0;
   }
```

پایان قسمت چهارم!

نویسنده : دانیال خشابی
ویرایش و صحت مطالب : نوید مردوخ روحانی

www.mrh.ir
www.majidonline.com

کپی رایت :: شهریور 1385
ارائه ی این مطلب فقط با ذکر منبع و دو سایت بالا مجاز است !

بنام خدا

سری های آموزشی آشنایی با زبان برنامه نویسی C++

قسمت پنجم : کار با توابع 2

تابع نویسی

ویرایش : 1

یک مرور کلی :

تابع : دستورر یا مجموعه ای از دستورات است که می تواند بصورت پیش فرض و یا بصورت نوشته شده توسط خود کاربر باشند ، که عملیات خاصی را بر روی متغیری که دریافت می کنند ، انجام می دهند .

همانطور که گفته شد توابع دو گروهند :
1- توابعی پیش ساخته و از قبل به همراه کمپایلر C++ ارائه شده اند که به آنها توابع کتابخانه ای می گویند . مثل تابع cos
2- توابعی که کاربر بر حسب ضرورت و نیاز آنها را می نویسد و در برنامه ی خود استفاده می کند .

به طور کلی تابع از سه قسمت تشکیل شده است :
1- مقدار دهی اولیه برای ورود به تابع
2- مجموعه ای از دستور العمل ها که روی مقدار ورودی انجام می شوند .
3- مقدار خروجی بعد از انجام عملیات روی مقدار ورودی

با توجه به سه قسمت بالا ، هر تابع را به صورت زیر ، در خارج از تابع main() تعریف خواهیم کرد :

```
Return-type function-name ( parameter-type parameter-Name , ...)
  {
     statements;
     return value;
  }
```

به این مثال ساده توجه کنید :
می خواهیم برنامه ای بنویسیم که در آن تابعی را تعریف کنیم که عددی را به عنوان ورودی بگیرد و مربع آن را به ما بازگرداند :
1- مثل روال گذشته ابتدا می نویسیم :

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

2- برای مثال اسم تابعمان را sqr در نظر می گیریم و کار آن را تعریف می کنیم :

```
int sqr(int x)
   {
     return  x*x;
   }
```

3- متن اصلی برنامه :

```
int main()
   {
   int a;

   cin>>a;
   cout << sqr(a) << endl;

return 0;
   }
```

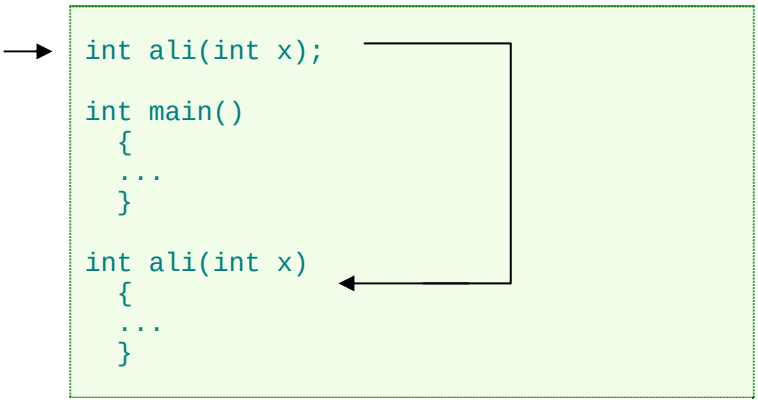
برنامه ی ما حالا کامل است :

```
#include <iostream.h>

int sqr(int x)
   {
     return  x*x;
   }

int main()
   {
   int a;

   cin>>a;
   cout << sqr(a) << endl;

return 0;
   }
```

یکبار دیگر به برنامه ی بالا برگردید ! می خواهیم نکات و ریزه کاری های آن را بررسی کنیم ! (پس خوب بخوانید !):

-کامپایلر برنامه را به این صورت می خواند :
... ابتدا در قسمت بالا تابع را شناسایی می کند و به ترتیب به پایین می آید ، تا اینکه به تابع اصلی می رسد و در نهایت آنرا اجرا می کند .
و چون در بالا ، شناخته است که تابعی با نام sqr معرفی کرده ایم ، هر جا که در متن برنامه نام تابع را ببیند ، سریعا برگشته و از روی دستورات تابع ، عملیات را انجام می دهد . در واقع ساختار استفاده ی ما ، از تابع ، در حالت کلی بدین صورت است :

اما در صورتی که بخواهیم ، ابتدا تابع اصلی برنامه را بنویسیم و بعد سایر توابع را ، ابتدا باید قبل از تابع `main()` یک اشاره ی کوچک به آن تابع دشته باشیم . بدین صورت :



```
int ali(int x);

int main()
  {
  ...
  }

int ali(int x)
  {
  ...
  }
```

یعنی قبل از اینکه کامپایلر به تابع `main()` برسد ، به آن اطلاع می دهیم که تابعی با این نام و مشخصات داریم ، اگر فراخوانی شد ، دستورات آن در زیر تابع اصلی است . توجه کنید که همیشه بعد از اشاره به تابعی حتما ; فراموش نشود .

- به ساختار تعریف تابع sqr نگاه کنید . در اولین قسمت قبل از نام تابع نوع مقدار خروجی نوشته می شود . یعنی اینکه ما که در بالا int نوشته ایم ، بدین معنا است که مقداری که تابع آن را بازگشت خواهد داد ، یک عدد و از نوع int می باشد. در واقع می توان این قسمت را بر حسب نیاز به مقدار بازگشتی از انواع دیگری مثل long int , char و یا ... تعیین کنید .

- در ادامه خواهیم دید یک نوع از تابع ها هستند که هیچ مقداری را بر نخواهند گرداند . این نوع از تابع ها از نوع void هستند . در صورتی که تابعی را از نوع void تعریف کنیم دیگر نیازی به نوشتن return نیست . البته می توان آنرا نوشت و جلوی آن را خالی گذاشت :

```
void ali(int x)
  {
  Statement;
  }
```

یا :

```
void ali(int x)
  {
  Statement;
  Return ;
  }
```

در مورد توابعی که هیچ مقداری بر نمی گردانند ، بیشتر بررسی خواهیم کرد .

- در انتخاب نام تابع (مثلاsqr) سعی کنید نام تابع با کاری که می کند ، همخوانی داشته باشد . چون در برنامه های چند صد خطی ، دیگر فرصتی برای اتلاف وقت برای بازخوانی کد ها یا حفظ کردن نام ها و ... نخواهید داشت . همچنین دقت کنید نامی که در قسمت معرفی تابع معرفی می کنیم ، در بدنه ی اصلی برنامه ، تابع را با همان نام فراخوانی می کنیم .

- در هنگام معرفی تابع در داخل پرانتز ، نوع و تعداد مقدار هایی که تابع به عنوان ورودی خواهد گرفت را معرفی می کنیم . مثلا در این تابع که نوشتیم ، تابع یک مقدار عددی از نوع int می گیرد .

- ممکن است در بعضی برنامه ها ، نیاز باشد که یک تابع ، چندین مقدار را دریافت کند . در اینصورت به عنوان مثال می توانیم بنویسیم :

```
int ali(int x, int y , int z)
  {
  ...
  }
```

که در اینصورت در برنامه ی اصلی می توانستیم تابع را به این صورت فراخوانی کنیم :

```
Cin>>a>>b>>c;
Cout<< ali(a,b,c)<<endl;
```

- یک نکته ی جالب : (default value) : اگر برای مثال ، من ، در معرفی تابعی بدین صورت بنویسم :

```
int ali(int x, int y , int z=2)
  {
  ...
  }
```

من در این حالت به z یک مقدار پیش فرض نسبت داده ام . یعنی اینکه ، در صورتی که در هنگام فرخوانی تابع در متن اصلی برنامه ، مقداری را به عنوان عدد سوم به تابع بفرستم ، که مثل حالت معمولی در z قرار خواهد گرفت . وگرنه اگر هیچ مقداری به آن نفرستم ، مقدار z برابر با 2 خواهد بود . به یاد داشته باشید که همیشه مقدار اولیه متغیر ها در هنگام معرفی آنها همیشه در سمت راست ترین قسمت نوشته می شوند . یعنی اینکه حالت معرفی زیر اشتباه است :

```
int ali(int x, , int z=2, int y)
  {
  ...
  }
```

مقدار نتیجه ای که تابع به عنوان خروجی پس خواهد داد ، با استفاده از دستور return باز گردانده می شود . مثلا در مثال قبلی داریم :

```
return  x*x;
```

یعنی اینکه پس از فراخوانی و اجرای دستورات تابع مقدار توان 2 ی x را به عنوان خروجی بازمیگرداند . اگر می خواستیم که خروجی یک تابع همواره 2 باشد :

```
return  2;
```

و یا...

مثال 2 : برنامه ای که مربع اعداد 1 تا 10 را در خروجی چاپ کند :
این برنامه هم مشابه مثال قبلیست :
جواب:

```
#include <iostream.h>

int sqr(int x)
   {
     return  x*x;
   }

int main()
   {
     for (int i=1; i<=10; i++)
        {
        cout << sqr(i) << endl;
        }
    return 0;
   }
```

یا به این صورت :

```
#include <iostream.h>

int sqr(int x);

int main()
      {
      for (int i=1; i<=10; i++)
        {
        cout << sqr(i) << endl;
        }
    return 0;
      }

int sqr(int x)
      {
      return  x*x;
      }
```

مثال 3 : با استفاده از یک تابع ، برنامه ای بنویسید که اعداد 1 تا 100 را در خروجی چاپ کند :
راهنمایی : تابعی خواهیم نوشت که عدد را بگیرد و توسط مقدار برگردانده شده نشان دهد که اول یا نه . اگر اول بود 1 برگرداند اگر نه 0 را برگرداند .

```
#include <iostream.h>
int aval(int x)
 {
 int w=0;
 for(int i=1;i<=x;i++)
       {
        if (x%i==0)
              w++;
       }
 if(w==2)
      return 1;
 else
      return 0;
 }

 int main()
  {
  int a;
  for(int i=1;i<=100; i++)
       {
       if(aval(i)==1)
            cout<<" adad e aval = " << i<<endl;
       }
  return 0;
  }
```

به این شکل هم می توانستیم بنویسیم :

```
  #include <iostream.h>
  int aval(int x);
  int main()
  {
  int a;
  for(int i=1;i<=100; i++)
       {
       if(aval(i)==1)
            cout<<" adad e aval = " << i<<endl;
       }
  return 0;
  }

 int aval(int x)
 {
 int w=0;
 for(int i=1;i<=x;i++)
       {
        if (x%i==0)
              w++;
       }
 if(w==2)
      return 1;
 else
      return 0;
 }
```

مثال : برنامه ای بنویسید که سه عدد را از کاربر گرفته و آنها را به تابعی فرستاده و بزرگترین آنها را پیدا و چاپ کند .

```
#include <iostream.h>

int max ( int x, int y, int z)
  {
    int max=x;
    if(max<y) max=y;
    if(max<z) max=z;
    return max;
  }

int main()
  {
    int a,b,c;
    cout<<"Enter three numbers: " <<endl;
    cin>>a>>b>>c;
    cout<<"The max number is = " <<max(a,b,c)<<endl;
    return 0;
  }
```

یا به این صورت :

```
#include <iostream.h>

int max ( int x, int y, int z);

int main()
  {
    int a,b,c;
    cout<<"Enter three numbers: " <<endl;
    cin>>a>>b>>c;
    cout<<"The max number is = " <<max(a,b,c)<<endl;
    return 0;
  }

int max ( int x, int y, int z)
  {
    int max=x;
    if(max<y) max=y;
    if(max<z) max=z;
    return max;
  }
```

یک نمونه استفاده از توابعی که هیچ مقداری را برنمی گردانند :
برنامه ای که مقداری را می گیرد و مشخص می کند در چه محدوده ای از صفر قرار دارد .

```
#include <iostream.h>
void fun (int x)
  {
  if(x<0)
       cout<<"The number is lower then zero ! " <<endl;
  else
       cout<<"The number is higher then zero ! (Or equal with ) " <<endl;
  return ;
  }
int main()
  {
  int a;
  cout<<"Enter a number: " ;
  cin>>a;
  fun(a);
  return 0;
  }
```

خوب به ساختار استفاده ی تابع بالا توجه کرده و آن را به خاطر بسپارید . با فراخوانی تابع و اجرای آن مانند این است که عینا دستورات آن را در همان مکان اجرا می کنیم .

ممکن است در جایی حتی لازم نباشد تابع مقداری را بگیرد ! پس :
مثال : برنامه ای که با استفاده از تابعی در صفحه چاپ کند : MajidOnline.com First Persian Graphic and Web design Resource

```
#include <iostream.h>
void fun ()
  {
  cout<<"***MajidOnline.com "<<endl
      <<"First Persian Graphic and Web design Resource "
<<endl;
  return ;
  }
int main()
  {
  fun();
  return 0;
  }
```

یک نکته ی مهم :
انواع شیوه های عمومی فراخوانی داده ها توسط یک تابع :
1- pass by refrence
2- pass by value

برای متوجه شدن مفهوم ایندو شیوه به برنامه ی زیر توجه فرمایید :

```
#include <iostream.h>

int f1( int a )
        {
        return a *= a;
        }

void f2( int &b )
        {
        b*= b;
        }

int main()
        {
        int x = 2, z = 4;

        cout << "x = " << x << " before passByValue\n"
                << "Value returned by passByValue: "
                << f1( x ) << endl
                << "x = " << x << " after passByValue\n" << endl;
        //********
        cout << "z = " << z << " before passByReference" << endl;
        f2( z );
        cout << "z = " << z << " after passByReference" << endl;

        return 0;
        }
```

در شیوه ی pass by value می خواهیم با استفاده از تابع f1 مقداری را مربع کنیم . در نهایت خواهید دید که بعد از اجرای تابع ، هیچ تاثیری در مقدار اولیه ی تابع نخواهد داشت .
اما اگر به قسمت pass by reference و تابع f2 توجه کنید ، خواهد دید از یک علامت & (آمپر سند) بین اسم متغیر ارسالی و نوع آن استفاده کرده ایم . این یعنی اینکه هر تغییر که روی مقدار ارسالی ما در داخل تابع صورت گرفت ، آنرا بر روی آن ذخیره کن . در واقع در حالت pass by reference مقدار متغیر اولیه ارسالی به تابع تغییر کرده و در آن ذخیره خواهد شد .

درواقع در خروجی خواهیم داشت :

```
x = 2  before passByValue
Value returned by passByValue: 4
x = 2  after passByValue

z = 4  before passByReference
z = 16  after passByReference
```

می توانید تغییرات زیادی روی برنامه بالا انجام دهید تا به شیوه ی کار هر کدام پی ببرید . مثلا :
1- یکبار علامت آمپر سند را بردارید .
2- یکبار سعی کنید نوع توابع را تغییر دهید .
3- ...

برای آشنایی بیشتر با کاربرد علامت & به برنامه ی ساده ی زیر توجه کنید :
مثال : برنامه ای بنویسید که عددی را گرفته و مقدار همان را 10 برابر کند :

```
#include <iostream.h>
void jj(int &a)
  {
  a*=10;
  }
int main()
  {
  int x;
  cout<<"Enter a number plz ! : ";
  cin>>x;
  jj(x);
  cout<<"result is : " <<x;
  return 0;
  }
```

با قرار دادن علامت آمپرسند ، با تغییر مقدار a ، این تغییرات در همانجا ذخیره می شود.
می توانید یکبار هم & را بردارید و نتیجه آنرا ببینید . در اینصورت ، مقدار ورودی هیچ تغییری نخواهد کرد.
فقط یادتان نرود که بین نام متغیر و نوع آن (که & در میان آنها قرار خواهد گرفت) فاصله را رعایت کنید .

آشنایی با یک نوع داده : static int :
Static int یک نوع داده است که تقریبا کار آن ، شبیه عمل نوع داده ی int به همراه & است .
برای درک این مطلب به برنامه ی زیر توجه بفرمایید:

```
#include <iostream.h>
void printme()
  {
  static int i=1;
  cout<<i++<<"  ";
  }
int main()
  {
  for(int k=0; k<10; k++)
       printme();
  return 0;
  }
```

حالا خودتان با توجه به خروجی می توانید نحوه ی کار این نوع داده ر ا بگویید ؟
در خروجی این برنامه خواهیم داشت :

```
 1   2   3   4   5   6   7   8   9   10
```

حال در برنامه static int را تبدیل کنید به int تبدیل کنید . در خروجی خواهیم داشت :

```
1   1   1   1   1   1   1   1   1   1
```

حال int را دوباره به حالت اولیه برگردانید و این بار مقدار دهی اولیه ی آن را پاک کنید . یعنی به این صورت : `static int i;`
در خروجی خواهیم داشت :

```
0   1   2   3   4   5   6   7   8   9
```

از این آخری می توانیم نتیجه بگیریم که متغیر static int برعکس متغیر های دیگر (که اگر به آنها مقدار اولیه ندهیم ، مقدار آنها نامعلوم خواهد بود) دارای مقدار اولیه ی صفر است .

آشنایی به توابع بازگشتی :
یک روش استفاده از تابع هستند که در واقع می توانیم با این سبک تابع نویسی ، بعضی از مسائل خاص را به روش ساده تری حل بکنیم .

*** مثلا دنباله ی زیر را در نظر بگیرید :

1 , 3 , 6 , 10 , 15 , 21 , ...

اگر شماره ی جمله را n در نظر گرفته و اولین عدد را 1 بگیریم ، خواهیم داشت : f(1)=1 , f(n) = f(n-1) + n
می خواهیم برنامه ای به روش بازگشتی بنویسیم که شماره ی عدد از دنباله ی بالا را گرفته و خود عدد را چاپ کند :
(خوب به طرز حل این مسئله به روش بازگشتی توجه بکنید)
- در روش بازگشتی از یک فرمول عمومی به صورت f(n) که بر حسب f(n-1) و یا .. . خواهد بود استفاده خواهیم کرد . بصورتی که اگر نتیجه ی تابع f(n) مورد نظر باشد ، کامپایلر ابتدا شروع به محاسبه ی f(n-1) می کند . بعد چون داریم : f(n-1) = f(n-2) + n-1 برای محاسبه ی f(n-1) وارد محاسبه ی f(n-2) می شود و ... و این اعمال تا جایی ادامه پیدا می کند که کامپایلر به f(1)=1 برسد که مقداریسیت ثابت و مشخص . بعد از رسیدن به این مقدار معلوم دوباره همان را آمده را برمی گردد تا جایی که به f(n) برسد .
پس از این داستان بالا ، دو مورد زیر را حتما به خاطر بسپارید :
1- نحوه ی کار توابع بازگشتی
2- حتما به خاطر داشته باشید که حد معین و مجازی برای تابع بازگشتی خود تعیین کنید . مثلا ما در مثال بالا f(1)=1 را تعریف کردیم . اگر تابع بازگشتی خود را محدود نکنید ، کامپایلر تا منفی بینهایت خواهد رفت (و در نتیجه هیچ بازگشتی در کار خواهد بود)

اما خود برنامه :

```
#include <iostream.h>
int f ( int x)
  {
  if (x==1)
       return 1;
  else
       return f(x-1)+x;
  }
int main ()
  {
  int a;
  cout<<"enter Number of figure which u want : " ;
  cin>>a;
  cout<<"f("<<a<<")="<<f(a)<<endl;
  return 0;
  }
```

*** بزرگترین فایده ی توابع بازگشتی سادگی کار آن ، و بزرگترین اشکال آن ، سرعت پایین آن است .

مثال : برنامه ای بنویسید که دنباله ی فیبوناچی را با استفاده از تابع بازگشتی محاسبه کند.

1 , 1 , 2 , 3 , 5 , 8 , ...

```
#include <iostream.h>
int f ( int x)
  {
  if ((x==1)||(x==2))
       return 1;
  else
       return f(x-1)+f(x-2);
  }
int main ()
  {
  int a;
  cout<<"enter Number of figure which u want : " ;
  cin>>a;
  cout<<"f("<<a<<")="<<f(a)<<endl;
  return 0;
  }
```

یک نکته : به برنامه ی زیر توجه کنید :

```
#include <iostream.h>
int main()
   {
   int a=3;
   cout<<a++<<"  "<<a++<<"  "<<a++<<"  "<<a++;
   return 0;
   }
```

فکر می کنید نتیجه ی آن چه باشد ؟!

```
6          5          4          3
```

اگر یه خورده به نتیجه ی آن نگاه کنید شاید عجیب و تازه به نظرتان برسد .
پس این مورد را به خاطر داشته باشید زبان c++ بر عکس زبان های دیگه ای مثل پاسکال و ... محتوای داخل دستور خروجی را ازراست به چپ ، پردازش می کند .
مشابه این مورد را می توان در جاهای دیگه ای یافت :

```
#include <iostream.h>
void  vahid(int a , int b)
  {
  cout<<"a="<<a<<endl;
  cout<<"b="<<b<<endl;
  return ;
  }
int main()
  {
  int a=3;
  vahid(a,a++);
  return 0;
  }
```

اگر توجه کرده باشید ، مقادیر از راست به چپ به تابع ارسال می شوند .

تمرینات:
1- برنامه ای که فاکتوریل عددی را با استفاده از تابع بازگشتی محاسبه کند .
2- برنامه ای که مقدار n رااز ورودی گرفته . s(n) را بصورت بازگشتی ، حساب کند :

$$S_n = \sum_{i=1}^{n} i^2 \qquad \begin{cases} S_n = S_{n-1} + n^2 \\ S_1 = 1 \end{cases}$$

3- تابعی را بنویسید که یک مقدار ورودی (آرگومان) بگیرد و مقدار مقلوب آن را به عنوان خروجی ، به برنامه ی اصلی را برگرداند .

پایان قسمت پنجم!

نویسنده : دانیال خشابی
ویرایش و صحت مطالب : نوید مردوخ روحانی

www.mrh.ir
www.majidonline.com

کپی رایت :: شهریور 1385
ارائه ی این مطلب فقط با ذکر منبع و دو سایت بالا مجاز است !

بنام خدا

سری های آموزشی آشنایی با زبان برنامه نویسی C++

قسمت ششم : آرایه ها

ویرایش : 1

یادآوری از قسمت 1 :

آرایه های یک بعدی : به طور کلی آرایه ها نوعی پیشرفته تر از متغیر ها هستند که می توانند چندین مقدار را در خود ذخیره کنند .

ساختار کلی آرایه ها به صورت زیر است :

```
array type      array name [number of elements] ;
```

مثلا برای آرایه ی یک بعدی A که دارای n خانه است خواهیم داشت :

A[0]	A[1]	A[3]	...	A[n-1]

نکته ی 1 : در زبان C++ شماره ی خانه ی آرایه ها از 0 شروع می شود تا n-1 .

مثال :مثلا در زیر آرایه ی Doste دارای 5 عنصر است و همچنین از نوع کاراکتری است :

```
int main()
   {
   char Doste[5];
   return 0;
   }
```

Doste[0] Doste[1] Doste[2] Doste[3] Doste[4]

برای مقدار دهی به یک آرایه یک برنامه ی ساده و معمولا تکراری در برنامه ها خواهیم داشت . پس به آن خوب توجه کنید :

```
#include <iostream.h>
int main()
   {
   int a[4];
   cout<<"enter content of 4 elements : "<<endl ;
       for(int i=0; i<4; i++)
            {
            cout<<"enter content of a["<<i<<"]"<<endl;
            cin>>a[i];
            }
   return 0;
   }
```

همچنین برای چاپ آن هم برنامه ای شبیه آن خواهیم داشت :

```
#include <iostream.h>
   int main()
   {
      int a[4];
      cout<<"enter content of 4 elements : "<<endl ;
       for(int i=0; i<4; i++)
            {
            cout<<"enter content of a["<<i<<"]"<<endl;
            cin>>a[i];
            }
       for(i=0; i<4; i++)
            cout<<"content of a["<<i<<"]="<<a[i]<<endl;
   return 0;
   }
```

نکته 2 : در هنگام معرفی یک آرایه مثلا آرایه ی a مقداری که به آن به عنوان تعداد خانه ها می دهیم ، باید مقداری ثابت باشد .
یعنی اگر مانند زیر بنویسیم ، اشتباه است و برنامه خطا خواهد داد :

```
int b=4;
int a[b];
```

چون که متغیر مانند b ، ممکن است در طول برنامه مقدارش عوض شود .

بنابرنین تنها دو راه وجود دارد :
1- اینکه مانند برنامه های قبل عدد بنویسیم . مثلا :

```
int a[4];
```

2- استفاده از نوع داده ی const int که مقدار آن همواره ی مقداری است ، ثابت . مثلا :

```
const int b=4;
int a[b];
```

شاید در بعضی از برنامه ها این مشکل بوجود آید که ما نمی دانیم تعداد خانه های یک آرایه باید به چه تعداد باشد ؟
در اینصورت بهترین راه حل دادن مقداری بزرگ در برابر مبحث مورد نظر است . مثلا عدد 512 در این زمینه بسیار پر کاربرد است .

در c++ این امکان وجود دارد که بتوانید خانه های آرایه ای را از قبل در برنامه مقدار دهی کنید . مثلا برای یک آرایه یک بعدی 5 خانه ای می خواهیم مقداری دهی پیش فرض انجام دهیم :

```
int a[5]={24,33,49,55,63};
```

همچنین می توانیم این کار را در قسمت کد برنامه نیز انجام دهیم :

```
#include <iostream.h>
int main()
   {
      int a[5]={24,33,49,55,63};

       for(int i=0; i<4; i++)
            cout<<"content of a["<<i<<"]="<<a[i]<<endl;
   return 0;
   }
```

مثال : برنامه ای بنویسید که مقادیر یک آرایه ی یک بعدی 5 خانه ای را از ورودی گرفته و مجموع مقادیر خانه ها را چاپ کند .

```
#include <iostream.h>
 int main()
       {
       int a[5],sum=0;
            for(int i=0; i<5; i++)
              {
              cout<<"a["<<i<<"]=";
              cin>>a[i];
              sum=sum+a[i];
              }
       cout<<"The sum is " <<sum;
       return 0;
       }
```

- استفاده از آرایه به عنوان آرگومان یک تابع :
آرایه ها را نیز همچون سایر نوع داده ها می توان به یک تابع ارسال کرد. برای اینکار ابتدا باید تابع را بگونه ای تعریف کنیم که یک پارامتر از نوع آرایه را دریافت کند.
ساختار استاندارد استفاده از آرایه به عنوان آرگومان یک تابع به صورت زیر می باشد : (در اینجا مثال برای آرایه ای از نوع int می باشد)
(خوب به آن توجه کنید و آنرا به خاطر بسپارید !)

```
#include <iostream.h>
Void fun(int array[])
  {
  ...
  }


int main()
   {
   int a[10];

   ...

   fun(a);

   ...

   return 0;
   }
```

برای ارسال آرایه موردنظر کافی است که تنها نام آرایه را بدون کروشه استفاده نماییم.(البته لازم می شود اندازه واقعی آرایه را نیز بعدا بعنوان دومین آرگومان به تابع ارسال کنیم.)

توجه کنید که بر عکس انواع داده های دیگر ، هر عملی که به صورت فوق در برنامه به عنوان آرگومان یک تابع استفاده شود ، هر تغییری که در آن توسط تابع رخ دهد در آن ذخیره خواهد شد . (می توانید این مورد را با برنامه های شماره ی قبل در مورد توابع بررسی کنید. این مبحث به تفاوت های بین مقادیر عادی و مقادیر ارجاعی مربوط می شود ؛ که بعدا مفصلا در مورد آن بحث خواهد شد)

بنابرین ، در صورتی که بخواهید آرایه ای را به تابعی طوری بفرستید که در آن هیچ تغییری امکان پذیر نباشد ، آن را از نوع const int تعریف کنید . در اینصورت ، کامپایلر اجازه ی هیچ گونه تغییری را در آن نخواهد داد .

مثال : برنامه ای بنویسید که مقادیر یک آرایه ی یک بعدی 5 خانه ای را از ورودی گرفته و به تابعی ارسال کند . تابع مجموع مقادیر خانه ها را حساب کرده و مقدار آن را به برنامه اصلی بازخواهد گرداند و بعد ما مقادیر آن را در برنامه اصلی چاپ خواهیم کرد.

```
#include <iostream.h>
int fun(int a[])
  {
  int sum=0;
       for(int i=0; i<5; i++)
            {
            sum+=a[i];
            }
  return sum;
  }

 int main()
       {
       int a[5];
            for(int i=0; i<5; i++)
              {
              cout<<"a["<<i<<"]=";
              cin>>a[i];
              }
       cout<<"The sum is " <<fun(a);
       return 0;
       }
```

مثال : برنامه ای بنویسید که اشتراک دو مجموعه را در خروجی چاپ کند :

```
#include <iostream.h>
int main()
   {
  int a[5]={5,4,3,2,1}
        ,b[3]={6,3,1};
  cout<<"Union part is :    " ;
  for(int i=0; i<5; i++)
       {
       for(int k=0; k<3; k++)
            {
             if (a[i]==b[k])
                  cout<<a[i]<<"   ";
            }
       }
   return 0;
   }
```

***نکته بسیار مهم :خروجی یک تابع نمی تواند یک آرایه باشد. برای بازگرداندن یک آرایه از تابع، باید آن را بصورت یک پارامتر خروجی به تابع ارسال نمود.

آرایه های چند بعدی :

آرایه های چند بعدی ، نوع پیشرفته تری از آرایه ها هستند که می توانند اطلاعات بیشتری را در خود ذخیره کنند . و در عوض نیز کار های پیشرفته تری را انجام دهند .

ساختار کلی معرفی این آرایه ها به اینصورت است :

```
type  array-name  [element-size 1][ element-size  2]  ...  [ element-size n] ;
```

بعنوان مثال، اعلان زير يك آرايه دوبعدي را معرفي مي نمايد:

```
int A[2][5] ;
```

این آرایه ی دوبعدی را می توانیم به صورت یک مستطیل بر روی سطح (دوبعد) نشان دهیم :

A[0][0]	A[0][1]	A[0][2]	A[0][3]	A[0][4]
A[1][0]	A[1][1]	A[1][2]	A[1][3]	A[1][4]

که در مستطیل بالا ، آدرس هر خانه را هم می بینید .

در واقع در یک آرایه ی دو بعدی داریم :

A[a][b] : a=تعداد ستون , b= تعداد سطر

توجه : در زبان C بر خلاف زبان های دیگر از جمله پاسکال و بیسیک ، مقدار اول ستون و مقدار دوم سطر می باشد.

همچنین آرایه ی سه بعدی هم به همین صورت :

A[a][b][c] : a=تعداد عرض , b= تعداد ستون , c=تعداد سطر

که شکلی به صورت یک مکعب مستطیل خواهد شد .

حال آیا می توانید حدس بزنید آرایه ی چهار بعدی به چه شکل خواهد بود ؟!! (اگر فهمیدید به من هم بگویید !)

نکته : مانند آرایه های یک بعدی ، اندیس سطرها و ستونها هر دو از 0 آغاز مي گردند.

همچنین مانند آرایه های یک بعدی برای مقدار اولیه دادن به آرایه های دو و سه بعدی به اینصورت عمل خواهیم کرد :

مثلا :

```
int A[3][4] = { {12, 5, 3, 8} , {-3, 7, -9, 2}, {4, 22, 18, 6} };
```

که همان این شکل است :

12	5	3	8
-3	7	-9	2
4	22	18	6

یا مثلا برای آرایه ی سه بعدی خواهیم داشت :

```
int A[2][3][4] = { { {12, 5, 3, 8} , {-3, 7, -9, 2}, {4, 22, 18, 6} } , { {8, 1, -3, 4} , {-2, 8, 11, 21} , {7, 3, -15, -8} } };
```

ارسال آرایه های چند بعدی به توابع :

ارسال آرایه های چند بعدی تقریبا شبیه ارسال آرایه های یک بعد است .

فقط یک نکته را به خاطر داشته باشید که :

شاید تصور کنید که برای تعریف یک آرایه دوبعدی بعنوان پارامتری از یک تابع، تنها قرار دادن دو علامت [] کافی است و نیازی به ذکر ابعاد آن نیست. اما اینگونه نیست، بلکه برنامه نویس باید تعداد ستونهای آرایه دوبعدی را صریحا مشخص نماید، اما نیازی به تعیین تعداد ردیفهای آن و ... نیست. بعنوان مثال فرض کنید تابعی مانند test داریم که بعنوان ورودی یک آرایه دو بعدی و تعدادی پارامتر دیگر دریافت می کند.

مثلا تعریف تابع بصورت زیر اشتباه است:

```
Int  test(int A[ ][ ]...)
 {
...
 }
```

اما کد زیر تعریفی درست است :

```
Int  test(int A[ ][4]...)
 {
...
 }
```

مسئله : برنامه ای بنویسید که حاصل جمع دوماتریس زیر را در خروجی چاپ کند .

-3	-7
4	34
45	7

و

-4	8
53	-17
4	4

```
#include <iostream.h>
int main()
   {
  int a[3][2]={{-3,-7},{4,34},{45,7}}
        ,b[3][2]={{-4,8},{53,-17},{4,4}}
        ,c[3][2];

       for(int i=0; i<3; i++)
            {
            for(int k=0; k<2; k++)
                  {
                  c[i][k]=a[i][k]+b[i][k];
                  cout<<c[i][k]<<"   ";
                  }
            cout<<endl;
            }
   return 0;
   }
```

می توانید برنامه ی جمع ماتریس ها را در حالت کلی برای دو ماتریس m*n خانه ای بازنویسی کنید .

مثال : برنامه ای بنویسید که حاصلضرب دو ماتریس زیر را بدست آورد :

-3	-7
4	34
45	7

×

-4	23	13	8
53	11	-1	-17

```
#include <iostream.h>
int main()
   {
  int a[3][2]={{-3,-7},{3,34},{45,7}}
        ,b[2][4]={{-4,23,13,8},{53,11,-1,-17}}
        ,c[3][4];
  int s;
  for (int i=0; i<3; i++)
             {
             for(int j=0; j<4; j++)
                    {
                    s=0;
                    for(int k=0; k<2; k++)
                           s+=a[i][k]*b[k][j];
                    c[i][j]=s;
                    }
             }
  for(i=0; i<3; i++)
        {
        for(int j=0; j<4; j++)
             cout<<c[i][j]<<"      " ;
        cout<<endl;
        }
   return 0;
   }
```

*** می توانید این برنامه را برای ضرب یک ماتریس m*k در ماتریس k*n که در نهایت ماتریسی بصورت m*n خواهد شد بازنویسی کنید .

تمرینات :

1- برنامه ای بنویسید که دترمینان یک ماتریس n*n را محاسبه کند .
2- برنامه ای که طول و مقادیر یک آرایه ی یک بعدی را گرفته و سپس از ورودی یک مقدار بگیرد و در خروجی چاپ کند که آیا این مقدار در آرایه ای که از ورودی گرفته وجود دارد یا نه ؟!

پایان قسمت ششم!

نویسنده : دانیال خشابی
ویرایش و صحت مطالب : نوید مردوخ روحانی

www.mrh.ir
www.majidonline.com

کپی رایت :: مهر 1385
ارائه ی این مطلب فقط با ذکر منبع و دو سایت بالا مجاز است !

بنام خدا

سری های آموزشی آشنایی با زبان برنامه نویسی C++

قسمت هفتم : کار با رشته های کاراکتری

ویرایش : 1

جواب مثال های شماره ی قبل :
1- دترمینان یک ماتریس n*n :

```
#include <iostream.h>
#include <conio.h>
int calc(int [],int dim);
void revmatrix( int [],int dim);
void main()
       {
            int matrix[1000];
            int dim,temp;
            double leftsum,rightsum;
       cout<<"          PLEASE ENTER MATRIX DIMANTION : ";
       cin>>dim;
       cout<<"\n\n\n";
       for( int i = 0;i<( dim*dim );i ++ )
              {
              cout<<"ENTER ELEMAN : ";
              cin>>temp;
              matrix[i] = temp;
              clrscr();
              cout<<"         PLEASE ENTER MATRIX DIMANTION : "<<dim;
              cout<<"\n\n\n";
       }//for i

       if ( dim > 2 )
              {
                  leftsum = calc( matrix , dim );
                  cout<<"LEFTSUM of the matrix = "<< leftsum <<"\n\n";
                  revmatrix( matrix , dim );
                  rightsum = calc( matrix ,dim );
                  cout<<"RIGHTSUM of the matrix = "<< rightsum <<"\n\n\n\n\n\n" ;
                  cout<<"      ( DETERMINAN OF THE MATRIX = "<< leftsum - rightsum<<" )";
              }

       else
             cout<<"     ( DETERMINAN OF THE MATRIX = "<<(matrix[0] * matrix[3] - matrix[1]
* matrix[2])<<" )";

       getch();
}//end main

/////////////////calc function/////////

int calc( int matrix[ ], int dim )
       {
       int sum = 0, bul, x = 1;
       for( int l = 0; l<( dim*dim );l += ( dim+1 ) )//ghotr asli
       x *= matrix[l];
       sum = x;
       x = 1;
       for( int c = 1;c<dim;c ++ )
              {
              bul=c;
              for( int m = 0;m<dim;m ++ )
                        {
                        if( ( bul+1 )%dim != 0 )
                                    {
                                        x *= matrix[bul];
```

```
                                        bul +=( dim + 1 );
                                    }

                            else
                                    {
                                        x = x * matrix[bul];
                                        bul += 1;
                                     }

                }//for m
                sum += x;
                x = 1;
        }//for c

        return sum;
}

////////////////////revmatriv determinan

void revmatrix(int matrix[ ],int dim)
        {
        int end,temp,counter;
             for( int t = dim-1;t <= dim*dim;t = t + dim )


                {
                end=t;
                counter = end -( dim-1 );
                while( end > counter )
                          {
                          temp = matrix[end];
                          matrix[end] = matrix[counter];
                          matrix[counter] = temp;
                          ++ counter;
                          -- end;
                          }

                }
        }
```

2- برنامه ای که طول و مقادیر یک آرایه ی یک بعدی را گرفته و سپس از ورودی یک مقدار بگیرد و در خروجی چاپ کند که آیا این مقدار در آرایه ای که از ورودی گرفته وجود دارد یا نه ؟!

جواب :

```
#include <iostream.h>
int main()
   {
   int i,j;

   double a[512],b;
   cout<<"How many content do you want to enter ?:";
   cin >>i;
   cout<<"Enter "<<i<<" contents :";
    for(j=0;j<i;j++)
    cin>> a[j];
   cout<<"Enter a keyword to search in your array : ";
   cin >> b;
```

```
      for(j=0;j<i;j++)
      {
        if (a[j]==b)
        {
         break;
        }
      }
       if  (j>=i)
       cout << "not found" << endl;
       else
       cout <<”Hurray !! your searched content is in this row :
“<< j << endl;

   return 0;
   }
```

رشته ها :
رشته نوع جدیدی از داده نیست ! بلکه نوعی از آرایه هاست که بصورت کاراکتری تعریف می شود . در داخل رشته های کاراکتری می توان اسامی افراد ، عبارات و ... را بصورت کاراکتر به کاراکتر نگه داری کرد .
علامت انتهای هر رشته ی کاراکتری \0 می باشد که برابر با مقدار پوچ یا NULL می باشد .
مثلا : عبارت hello در داخل یک آرایه از نوع کاراکتری که رشته می نامیم بصورت زیر نمایش داده می شود :

h	e	l	l	o	\0

در واقع کاراکتر \0 در بسیاری از مسائل می تواند با عناون پایان یک رشته ی کاراکتری به ما کمک کند .

برای گرفتن مقدار یک رشته ی کاراکتری هم می توان آن را مانند مقدار یک متغیر عدد بصورت کامل گرفت یا هم می توان بصورت کاراکتر به کاراکتر گرفت .
(البته فقط در زمان تعریف متغیر رشته ای می توان از روش اول استفاده کرد)

```
Char ali[26]=”WWW.MRH.IR”;

Char aida[26]={ ‘m’ , ‘a’  , ‘j’ , ‘i’ , ‘d’ }
```

که در واقع می توانستید این دوحالت را هم با cin و هم با یک حلقه هم انجام دهید .

برای شروع یک مثال حل می کنیم .
برنامه ای بنویسید که یک رشته را گرفته و طول رشته را چاپ کند :

```
#include <iostream.h>
int main()
   {
   char a[64];
   int i,c;
   cout<<"Enter a strings:"<<endl;
   cin>>a;
      for(int i=0; a[i]!='\0'; i++);
   cout<<"Size of string : " <<i<<endl;
   return 0;
   }
```

اگر بخواهیم تابعی بنویسیم که همین کار بالا را انجام دهد یعنی رشته را گرفته و طول آنرا برای ما چاپ کند ، بصورت زیر می شود .:

```
#include <iostream.h>
void size(char a[],char b[])
{
 int i;
   for(i=0; a[i]!='\0'; i++);
 cout<<"Size of string : " <<i<<endl;
}
int main()
{
 char a[64];
 int i,c;
 cout<<"Enter a strings:"<<endl;
 cin>>a;
 size(a);
return 0;
}
```

مثال 3: برنامه ای بنویسید که دو مقدار برای رشته را گرفته و دومی را در اولی کپی کند :

```
#include <iostream.h>
int main()
   {
   char a[64],b[64];
   cout<<"Enter 2 strings:"<<endl;
   cin>>a;
   cin>>b;
   int i=0;
    do{
      a[i]=b[i];
      }while(b[i++]!='\0');
   a[i]='\0';

   return 0;
   }
```

برای تمرین بهتر و آشنایی بیشتر تمرینات آخر این فصل را انجام دهید .
تقریبا اکثر تمرین هایی که در آخر فصل این اموزش آمده اند ، دارای تابع هایی در کتابخانه ی c++ هستند .
این توابع کتابخانه ای ، که مربوط به کار با رشته ها هستند در کتابخانه ی string.h تعریف می شوند .
این نکته را با خاطر بسپارید ، که متغیر های رشته ای در این زبان ، با دیگر زبان های برنامه نویسی فرق بسیار دارد ، که به مرور به آن می پردازیم.

تابع	توضیح	سرفایل مربوط
Strlen(a)	طول یک رشته را به ما میدهد .	String.h
Strcpy(a,b)	رشته ای را در دیگر کپی می کند .	String.h
Strncpy(a,b,n)	رشته ای را در دیگری حداکثر به تعداد n کاراکتر کپی می کند .	String.h
Strrev(a)	مقدار رشته ای را برعکس می کند .	String.h
Strcat(a,b)	دو رشته را گرفته و آنها را به هم متصل می کند .	String.h
Strncat(a,b,n)	دو رشته را به هم متصل می کند . بطوریکه رشته ی حاصل حداکثر n کاراکتر داشته باشد .	String.h
Strcmp(a,b)	دو رشته را مقایشه می کند و اگر اولی بزرگتر بود 1 برمی گرداند ، اگر دومی بزرگتر بود ، 1- برمی گرداند و اگر برابر بودند 0 برمی گرداند .	String.h
Strncmp(a,b,n)	N کاراکتر اول از دو رشته را با هم مقایسه می کند و نتیجه را مانند بالا بر می گرداند .	String.h

تمرینات :

1- مثال 3 را تبدیل به تابع بکنید . یعنی اینکه تابعی بنویسید که دو رشته را گرفته و دومی را در اولی کپی کند .
2- تابعی بنویسید که دو رشته را بگیرد و اگر اولی بزرگتر بود ، 1 برگرداند . اگر دومی بزرگتر بود ، 0 برگرداند و اگر برابر بودند 1- برگرداند .
3- برنامه ای بنویسید که دو رشته را گرفته ، طول هرکدام را چاپ ، مانند تمرین بالا آنها را مقایسه کرده و در نهایت رشته ی دوم را در اولی قرار دهد .
4- برنامه ای بنویسید که تمرین قبل را روی n کاراکتر اول رشته ها انجام دهد. (n از ورودی گرفته خواهد شد)
5- برنامه ای بنویسید که با استفاده از یک تابع دو رشته را گرفته و آنها را به همدیگر متصل کند .
6- برنامه ای بنویسید که با استفاده از تابعی دو رشته را به هم متصل کرده و n رشته از آنها را در نتیجه قرار دهد . (n از ورودی گرفته خواهد شد)
7- برنامه ای بنویسید که با استفاده از تابعی یک مقدار رشته ای را برعکس کند .

پایان قسمت هفتم!

نویسنده : دانیال خشابی
ویرایش و صحت مطالب : نوید مردوخ روحانی

www.mrh.ir
www.majidonline.com

کپی رایت :: مهر 1385
ارائه ی این مطلب فقط با ذکر منبع و دو سایت بالا مجاز است !