برنامه سازی پیشرفته

مدرس:

مرتضی نوذری

نام درس :

برنامه سازی پیشرفته

تعداد واحد درسی :

۳ واحد

فهرست مطالب

فصل اول

مقدمات C++

فهرست مطالب فصل اول

تاریخچه مختصر C++

این زبان در اوائل دهه ۱۹۸۰ توسط Bjarne stroustrup در آزمایشگاه بل طراحی شده. این زبان عملاً توسعه یافته زبان برنامه نویسی C می باشد که امکان نوشتن برنامه‌های ساخت یافته شئ گرا را می‌دهد.

الگوریتم

تعریف:

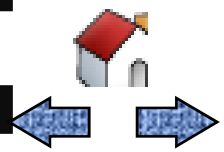
مجموعه محدودی از **دستورالعمل هاست** که اگر دنبال و اجرا شود **هدف خاصی** را دنبال می کند.بعلاوه موارد زیر در هر الگوریتم قابل بررسی است:

- **ورودی:** الگوریتم **هیچ یا چند کمیت** را به عنوان ورودی دریافت کند.
- **خروجی:**الگوریتم بایستی **حداقل یک کمیت** به عنوان خروجی داشته باشد.
- **قطعیت:**هر دستورالعمل باید **واضح و خالی از هر نوع ابهامی** باشد.
- **محدودیت:**الگوریتم باید بعد از طی مراحلی محدود **خاتمه** یابد.
- **کارآیی:**هر دستورالعمل علاوه بر قطعیت باید انجام پذیر باشد. هر دستورالعمل با استفاده از کاغذ و به صورت دستی اجرا شود.

قانون نامگذاری شناسه‌ها

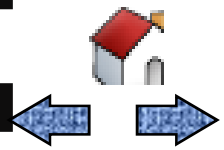
۱) حروف کوچک و بزرگ در نامگذاری شناسه‌ها متفاوت می‌باشند.

بنابراین Xy ، XY ، xY ، xy چهار شناسه متفاوت از نظر C++ می‌باشد.



قانون نامگذاری شناسه‌ها

۲) در نامگذاری شناسه‌ها از حروف الفباء، ارقام وزیر خط (underscore) استفاده می‌شود و حداکثر طول شناسه ۳۱ می‌باشد و شناسه بایستی با یک رقم شروع نگردد.

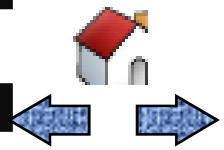


قانون نامگذاری شناسه‌ها

۳) برای نامگذاری شناسه‌ها از کلمات کلیدی نبایستی استفاده نمود. در زیر بعضی از کلمات کلیدی داده شده است.

And	Sizeof	then	xor	Template
Float	False	Friend	While	continue
extern	Private	Switch	Default	Const
delete	typedef	if	this	Virtual

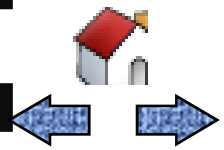
لیست کامل کلمات کلیدی



متغیرها

متغیر، مکانی در حافظه اصلی کامپیوتر می‌باشد که در آنجا یک مقدار را می‌توان ذخیره و در برنامه از آن استفاده نمود. قانون نامگذاری متغیرها همان قانون نامگذاری شناسه‌ها می‌باشد.

در اسلاید بعد به انواع داده ها اشاره می شود.



متغیرها

- در زبان C++ چهار نوع داده اصلی وجود دارد که عبارتند از:
- **char**:

این نوع داده برای ذخیره داده های کاراکتری مانند ‘a’,’2’ و بازه قابل قبول آن ۱۲۸- تا ۱۲۷ می باشد.

- **int**:

این نوع داده برای ذخیره اعداد صحیح می باشد. قابل قبول بین ۳۲۷۶۸- تا ۳۲۷۶۷ می باشد.

- **float**:

این نوع داده برای ذخیره اعداد اعشار به کار می رود و دقت آن تا ۷ رقم اعشار است.

- **double**:

این نوع داده برای ذخیره سازی اعداد اعشاری بزرگ به کار می رود و دقت آن از float بیشتر است.

متغیرها

- signed (علامت دار)
- unsigned (بدون علامت)
- short (کوتاه)
- long (بلند)

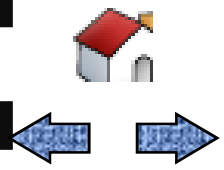
نوع int هر چهار کلمه فوق را می تواند استفاده کند

نوع char می تواند با signed و unsigned به کار رود

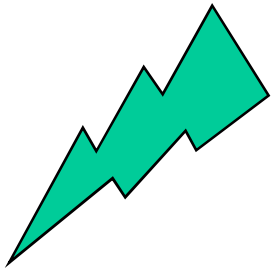
نوع double با long به کار می رود.

انواع داده ها

نوع داده	مقادیر	حافظه لازم
int	-32768 تا 32767	۲ بایت
unsigned int	0 تا 65535	۲بایت
long int	-2147483648 تا 2147483647	۴ بایت
unsigned long int	0 تا 4294967295	۴ بایت
char	یک کارکتر	۱بایت
unsigned char	-128 تا 127	۱ بایت
float	1.2e-38 تا 3.4e38	۴ بایت
double	2.2e-308 تا 1.8e308	۸ بایت



اعلان متغیرها



قبل از آنکه در برنامه به متغیرها مقــداری تخصــیص داده شود و از آنها استفاده گردد بایستی آنها را در برنامه اعلان نمود.

فرمت کلی:

نام متغیر; نوع داده

در اسلاید بعد مثال هایی از اعلان متغیر ذکر شده است.

چند مثال از اعلان متغیر ها :

- ✓ برای اعلان متغیرx از نوع int :

```
int   x;
```

- ✓ برای اعلان متغیرهای p و q را از نوع float که هر کدام چهار بایت از حافظه را اشغال می‌کنند :

```
float  p , q  ;
```

- ✓ برای اعلان متغیر next از نوع کرکتر که می‌توان یکی از ۲۵۶ کرکتر را به آن تخصیص داد و یک بایت را اشغال می‌کند.

```
char next ;
```

تخصیص مقادیر به متغیرها

با استفاده از عملگر = می‌توان به متغیرها مقدار اولیه تخصیص نمود.

در اسلاید بعد مثال هایی از اعلان متغیر ذکر شده است.

مثال :

- در دستورالعمل `int x=26;`
  X را از نوع int با مقدار اولیه 26 اعلان نموده .

- در دستورالعمل `long a=67000 , b=260;`
  متغیرهای b و a را از نوع long int تعریف نموده با مقادیر بترتیب 260 و 67000.

کرکترهای مخصوص

کامپیلر C++ بعضی از کرکترهای مخصوص که در برنامه می‌توان از آنها برای فرمت بندی استفاده کرد را تشخیص می‌دهد. تعدادی از این کرکترهای مخصوص به همراه کاربرد آنها در اسلاید بعد آورده شده است .

کرکترهای مخصوص

\n	Newline
\t	Tab
\b	Backspace
\a	Beep sound
\”	Double quote
\’	Single quote
\0	Null character
\?	Question mark
\\	Back slash

بعنوان مثال از کرکتر \a می‌توان برای ایجاد صدای beep استفاده نمود.

```
char        x = '\a' ;
```

رشته‌ها

رشته یا string عبارتست از دنباله‌ای از کرکترها که بین " " قرار داده می‌شود. در حافظه کامپیوتر انتهای رشته‌ها بوسیله \0 ختم می‌گردد.

در اسلاید بعد به دو مثال دقت نمایید.

مثال ۱ :

"BOOK STORE" یک رشته ده کرکتری مـی‌باشـد که با توجه به کرکتر 0\ که به انتهای آن در حافظه اضافه می شود جمعــاً یازده بایت را اشغال می‌کند.

مثال ۲ :

دقت نمایید که "W" یک رشته می‌باشد که دو بایت از حافظه را اشغال می‌کند در حالیکه 'W' یک کرکتر می‌باشد که یک بایت از حافظه را اشغال می‌نماید.

نمایش مقادیر داده‌ها

برای نمایش داده‌ها بر روی صفحه مانتور از cout کـه بـدنبال آن عملگـر درج یعنی >> قید شده باشد استفاده می‌گردد. بایستی توجه داشـت کـه دوکرکتر > پشت سر هم توسط C++ بصورت یک کرکتر تلقی می‌گردد.

مثال :

- ✓ برای نمایش پیغام good morning بر روی صفحه نمایش :

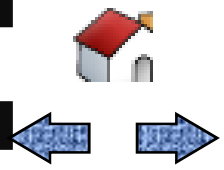
```
cout << "good morning";
```

- ✓ برای نمایش مقدار متغیر x بر روی صفحه نمایش :

```
cout << x ;
```

دریافت مقادیر متغیرها

به منظور دریافت مقادیر برای متغیرها در ضمن اجرای برنامه از صفحه کلید، از cin
که بدنبال آن عملگر استخراج یعنی >> قید شده باشد می‌توان استفاده نمود.



مثال :

```
int x;
cout << "Enter a number:" ;
cin >> x;
```

عملگر انتساب

عملگر انتساب = می‌باشد که باعث می‌گردد مقدار عبارت در طرف راست این عملگر ارزیابی شده و در متغیر طرف چپ آن قرار گیرد.

مثال :

```
x=a+b;
x=35 ;
x=y=z=26 ;
```

از عملگرهای انتساب چندگانه نیز می‌توان استفاده نمود. که مقدار سه متغیر z و y و x برابر با 26 میشود.

عملگرهای محاسباتی

در **C++** پنج عملگر محاسباتی وجود دارد که عبارتند از :

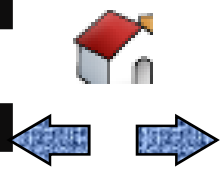
+	جمع
–	تفریق
*	ضرب
/	تقسیم
٪	باقیمانده

این عملگرها دو تائی می‌باشند زیرا روی دو عملوند عمل می‌نمایند. از طرف دیگر عملگرهای + و – رامی‌توان بعنوان عملگرهای یکتائی نیز در نظر گرفت.

مثال ۱ :

در حالتی که هر دو عملوند عملگرهای ٪ ، / ، * ، + ، – از نوع صحیح باشد نتیجه عمل از نوع صحیح می‌باشد.

عبارت	نتیجه
5 + 2	7
5 * 2	10
5 – 2	3
5 % 2	1
5 / 2	2



مثال ۲ :

در صورتیکه حداقل یکی از عملوندهای عملگرهای / ، * ، − ، + از نوع اعشاری باشد نتیجه عمل از نوع اعشاری می‌باشد.

عبارت	نتیجه
5. 0 +2	7. 0
5* 2. 0	10.0
5. 0 / 2	2.5
5. 0 -2	3. 0
5. 0 / 2. 0	2.5

عملگرهای افزایش و کاهش

در C++ ، افزایش یک واحد به مقدار یک متغیر از نوع صحیح را افزایش و بطور مشابه کاهش یک واحد از مقدار یک متغیر از نوع صحیح را کاهش می‌نامند..

عملگرهای افزایش و کاهش

عملگر کاهش را با – – و عملگر افزایش را بـــا ++ نمـــایش مـــی‌دهنـــد. چـــون عملگرهـــای ++ و – – فقـــط روی یـــک عملوند اثر دارند این دو عملگر نیز جـــزء عملگرهای یکتائی می‌باشند.

مثال :

سه دستور العمل :

```
++x;
x++;
x=x+1;
```

معادل می‌باشند و بطریق مشابه سه دستورالعمل زیر نیز معادل می باشند.

```
- - y ;
y=y-1;
y- - ;
```

مثال :

```
int x,y;
x=10;
y=++x;
```

یک واحد به x اضافه می شود و حاصل در y قرار می گیرد لذا y برابر ۱۱ خواهد شد

```
int x,y;
x=10;
y=x++;
```

مقدار x در y قرار می گیرد و سپس یک واحد به x اضافه می شود لذا مقدار y برابر با ۱۰ ولی مقدار x برابر با 11 خواهد شد

مثال

```
int x,y,m;
x=10;
y=15;
m=++x+y++;
```

```
x=11;
y=16;
m=26;
```

از عملگرهای ++ و -- می‌توان بدو صورت پیشوندی و پسوندی استفاده نمود.
در دستورالعمل‌های پیچیده عملگر پیشوندی قبل از انتساب ارزیابی میشود و عملگر پسوندی بعد از انتساب ارزیابی می‌شود.

مثال :

```
int x=5;
y=++x * 2;
```

پس از اجرای دستورالعملهای فوق :

```
y=12
```

```
int  x=5;
y=x++ * 2;
```

پس از اجرای دستورالعملهای فوق :

```
y=12
```

عملگر sizeof

Sizeof ازعملگرهای یکتائی می باشد و مشخص کننده تعداد بایت هائی است که یک نوع داده اشغال می‌کند.

```
sizeof  نام متغیر;
sizeof  (نوع داده);
```

مثال :

```
int   x;
cout << sizeof x ;
```

مقدار ۲ نمایش داده می‌شود .

```
cout << sizeof(float) ;
```

مقدار ۴ نمایش داده می شود.s

عملگرهای جایگزینی محاسباتی

برای ساده‌تر نوشتن عبارتها در C++ ، می‌توان از عملگرهای جایگزینی محاسبـــــاتی استفاده نمود.

+= -= *= /= %=

اولویت عملگرها

ارزیابی مقدار یک عبارت ریاضی براساس جدول اولویت عملگرها انجام می‌گردد. در ذیل جدول اولویت عملگرها براساس بترتیب از بیشترین اولویت به کمترین اولویت داده شده است.

چپ به راست	پرانتزها	( )
راست به چپ	عملگرهای یکتایی	sizeof ++ -- + -
چپ به راست	عملگرهای ضرب و تقسیم و باقیمانده	* / %
چپ به راست	عملگرهای جمع و تفریق	+ –
چپ به راست	عملگرهای درج و استخراج	>> <<
راست به چپ	عملگرهای جایگزینی و انتساب	%= /= *= -= += =

مثال ۱ :

(5+2) *(6+2*2)/2

با توجه به جدول اولویت عملگرها داریم که

7 *(6+2*2)/2
7*(6+4)/2
7* 10 /2
70 /2
35

مثال ۲ :

```
int a=6 , b=2, c=8, d=12;
d=a++ * b/c ++;
cout << d << c << b << a;
```

خروجی :

توضیحات (Comments)

توضیحات در برنامه باعث خوانائی بیشتر و درک بهتر برنامه میشود. بنابراین توصیه بر آن است که حتی الامکان در برنامه‌ها از توضیحات استفاده نمائیم. در C++، توضیحات بدو صورت انجام می‌گیرد که در اسلایدهای بعد به آن اشاره شده است.

توضیحات (Comments)

الف: این نوع توضیح بوسیله // انجام می‌شود. که کامپیوتر هر چیــزی را که بعد از // قرار داده شود تا انتهای آن خط اغماض می‌نماید.

مثال :

```
c=a+b;//c is equal to sum of a  and b
```

ب:توضیح نوع دوم با /* شروع شده و به */ ختم می‌شود و هر چیزی که بین /* و */ قرار گیرد اغماض می‌نماید .

مثال :

```
/ * this is a program
to calcufate sum of
n integer numbers */
```

توابع کتابخانه

زبان **C++** مجهز به تعدادی توابع کتابخانه می‌باشد. بعنوان مثال تعدادی توابع کتابخانه برای عملیات ورودی و خروجی وجود دارند.

معمولاً توابع کتابخانه مشابه ، بصورت برنامه‌های هدف (برنامه ترجمه شده بزبان ماشین) در قالب فایلهای کتابخانه دسته بندی و مورد استفاده قرا رمی‌گیرند.

این فایلها را فایلهای **header** می‌نامند و دارای پسوند **h.** می‌باشند.

نحوه استفاده از توابع کتابخانه ای

برای استفاده از توابع کتابخانه خاصی بایستـــــی نام فایل header آنرا در ابتدای برنامه در دستور #include قرار دهیم.

#include < اسم فایل header >

تابع	نوع	شرح	فایل هیدر
abs(i)	*int*	قدرمطلق i	stdlib.h
cos(d)	*double*	کسینوس d	math.h
exp(d)	*double*	e^x	math.h
log(d)	*double*	$\log_e$ d	math.h
log10(d)	*double*	Log_{10} d	math.h
sin(d)	*double*	سینوس d	math.h
sqrt(d)	*double*	جذر d	math.h
strlen(s)	*int*	تعداد کرکترهای رشته s	string.h
tan(d)	*double*	تانژانت d	math.h
toascii(c)	*int*	کداسکی کرکتر c	stdlib.h
tolower(c)	*int*	تبدیل به حروف کوچک	stdlib.h
toupper(c)	*int*	تبدیل به حرف بزرگ	stdlib.h

برنامه در C++

اکنون باتوجه به مطالب گفته شده قادر خواهیم بود که تعدادی برنامه ساده و کوچک به زبان C++ بنویسیم. برای نوشتن برنامه بایستی دستورالعملها را در تابع ( )main قرار دهیم و برای اینکار می‌توان به یکی از دو طریقی که در اسلایدهای بعد آمده است ، عمل نمود.

روش اول :

```
#include     <       >
int main( )
{
۱ دستورالعمل ;
۲ دستورالعمل ;
.
.
.
n  دستورالعمل;
return 0 ;
}
```

روش دوم :

```
#include     <       >
void main( )
{
۱ دستورالعمل ;
۲ دستورالعمل ;
.
.
.
n  دستورالعمل;
}
```

نکات–۲

- **error: به خطاهاي برنامه نويسي error مي گويند.**
- **انواع خطاها در برنامه نويسي:**
  - **خطاهای نحوی (خطاهای زمان کامپایل):**

این خطاها در اثر رعایت نکردن قواعد دستورات زبان بوجود می آیند و در همان ابتدا توسط کامپایلر به برنامه نویس اعلام می گردد. برنامه نویس باید این خطا را رفع کرده و سپس برنامه را مجددا کامپایل نماید. لذا معمولا این قبیل خطاها خطر کمتری را در بردارند.

ادامه

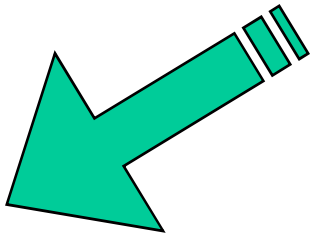
- **خطاهای منطقی (خطاهای زمان اجرا):**

این دسته خطاها در اثر اشتباه برنامه نویس در طراح الگوریتم درست برای برنامه و یا گاهی در اثر درنظر نگرفتن بعضی شرایط خاص در برنامه ایجاد می شوند.

- **خطاهای مهلک:** در این دسته خطاها کامپیوتر بلافاصله اجرای برنامه را متوقف کرده و خطا را به کاربر گزارش می کند. مثال معروف این خطاها خطای تقسیم بر صفر می باشد.

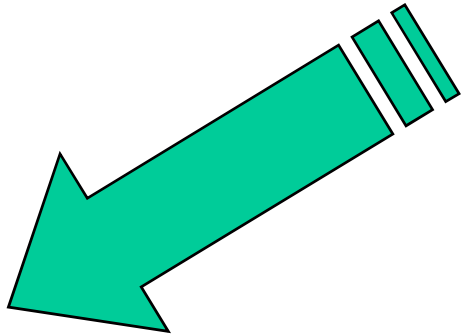
- **خطاهای غیرمهلک:** در این دسته خطا اجرای برنامه ادامه می یابد ولی برنامه نتایج اشتباه تولید می نماید. بعنوان مثال ممکن است دراثر وجود یک خطای منطقی در یک برنامه حقوق و دستمزد حقوق کارمندان اشتباه محاسبه شود و تا مدتها نیز کسی متوجه این خطا نشود!

برنامه ای که پیغام C++ is an object oriented language را روی صفحه مانیتور نمایش می دهد.

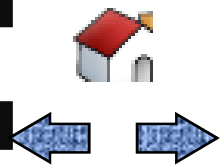


```
#include    <iostream.h>
int main( )
{
cout <<"C++ is an object oriented language \n" ;
return 0 ;
}
```

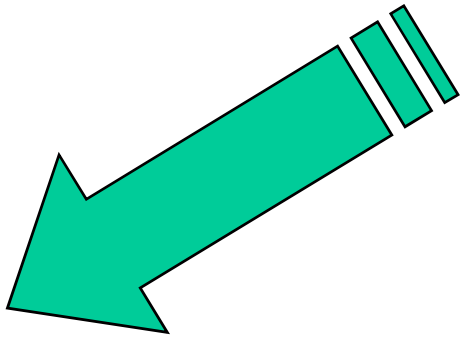
برنامه زیر یک حرف انگلیسی کوچک را گرفته به حرف بزرگ تبدیل می‌نماید.



```
#include    <iostream.h>
#include    <stdlib. h>
int main( )
{
 char c1 , c2;
cout << "Enter a lowercase letter:"
cin >> c1;
c2 = toupper(c1);
cout << c2 << endl;
return 0; }
```



دو عدد از نوع اعشاری را گرفته مجموع و حاصلضرب آنها را محاسبه و نمایش می‌دهد.



```
#include    <iostream.h>
int main( )
{
float        x,y,s,p ;
cin >> x >> y ;
s= x+y ;
p=x*y;
cout << s <<endl << p;
return 0 ;
}
```

فصل دوم

ساختارهای تصمیم گیری و تکرار

فهرست مطالب فصل دوم

عملگرهای رابطه ای

از این عملگرها برای تعیین اینکه آیا دو عدد با هم معادلند یا یکی از دیگری بزرگتر یا کوچکتر می‌باشد استفاده می‌گردد. عملگرهای رابطه‌ای عبارتند از:

مساوي	= =
مخالف	! =
بزرگتر	>
بزرگتر يا مساوي	> =
كوچكتر	<
كوچكتر يا مساوي	<=

عملگر شرطی

شکل کلی عملگر شرطی بصورت زیر می‌باشد:

```
expression _ test ? expression _ true : expression _ false
```

عملگر شرطی تنها عملگری در C++ می‌باشد که دارای سه عملوند می‌باشد.

مثال ۱ :

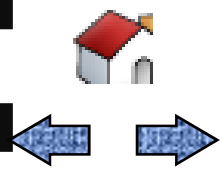
```
int x=10,y=20,b;
b=(x>y) ? x : y ;
```

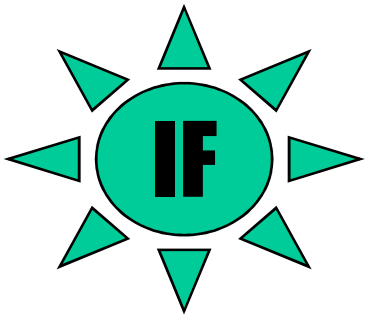
این دو دستور العمل باعث میشوند که ماکزیمم مقادیر **y** و **x** در **b** قرار بگیرد.

مثال ۲ :

```
x>=10 ? cout << "passed" : cout << "failed" ;
```

اگر مقدار **x** بزرگتر یا مساوی ده باشد رشتهٔ **passed** در غیر اینصورت رشته **failed** نمایش داده میشود.





دستورالعمل شرطی

توسط این دستور شرطی را تست نموده و بسته به آنکه شرط درست یا غلـط باشـد عکـس‌العمـل خاصی را نشان دهیم.

```
if(   عبارت )
{
        دستورالعمل 1 ;
              .
        دستورالعمل n;
}
 else
{
        دستورالعمل 1 ;
              .
        دستورالعمل n;
 }
```

مثال ١ :

```
if(x != y)
{
cout << x ;
++ x ;
}
else
        {
cout << y ;
- - y ;
         }
```

مثال ۲:

برنامه زیر یک عدد اعشاری را از ورودی گرفته جذر آنرا محاسبه می‌نماید.

```
#include    <iostream.h>
#include    <math . h>
int main( )
{
float  x,s;
cin >> x ;
if( x < 0 )
cout << " x is negative" << endl ;
else
{
s = sqrt(x) ;
cout << s << endl ;
                    }
return 0;
}
```

عملگر کاما

تعدادی عبارت را می‌توان با کاما بهـم متصـل نمـود و تشکیل یک عبارت پیچیده‌تری را داد. این عبارتها بـه ترتیب از چپ به راست ارزیابی شـده و مقـدار عبـارت معادل عبارت n می‌باشد.

(عبارت n , ... , عبارت 3 , عبارت 2 , عبارت 1)

مثال :

اگر داشته باشیم int a=2 , b=4 , c=5 ; عبارت زیر را در نظر بگیرید:

(++ a , a+b, ++ c, c+b)

مقدار عبارت برابر است با b+c که معادل 10 می‌باشد.

عملگرهای منطقی

با استفاده از عملگرهای منطقی می‌توان شرطهای ترکیبی در برنامه ایجاد نمود.
عملگرهای منطقی عبارتست از :

AND

OR

NOT

که در **C++** به ترتیب بصورت زیر نشان داده میشود.

&&

||

!

جدول درستی سه عملگر شرطی

And

a	b	a && b
true	true	True
true	false	False
false	true	False
false	false	False

Or

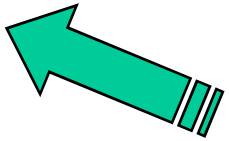
a	b	a || b
true	true	True
true	false	True
false	true	True
false	false	False

Not

a	!a
true	False
false	True

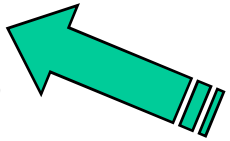
چند مثال :

```
if ((x= = 5) ||(y != 0))
    cout << x << endl ;
```



اگر x برابر با 5 یا y مخالف صفر باشد مقدار x نمایش داده شود .

```
if(x)
  x = 0 ;
```



اگر مقدار x مخالف صفر باشد، آنگاه x برابر با صفر شود .

برنامه زیر طول سه پاره‌خط را از ورودی گرفته مشخص می‌نماید که آیا تشکیل یک مثلث میدهد یا خیر؟

```
#include      < iostream.h >
int main( )
{
float  a, b, c;
cout << "Enter three real numbers" << endl ;
cin >> a >> b >> c; //
if(( a < b + c) &&(b < a+c) &&(c < a+b))
cout << "It is a triangle" ;
else
cout << "Not a triangle" ;
return  0 ;
}
```

دستورالعمل For

از دستور العمل for برای تکرار دستورالعملها استفاده میشود. شکل کلی دستور for بصورت زیر می‌باشد:

```
for (عبارت 1 ; عبارت 2 ; عبارت 3)
{
 دستورالعمل 1 ;
 دستورالعمل 2 ;
 .
 .
 .
 دستورالعمل n;
}
```

برنامه زیر عدد صحیح و مثبت n را از ورودی گرفته فاکتوریل آنرا محاسبه ونمایش می‌دهد.

```
#include        <iostream.h>
int      main( )
{
int  n, i ;
long fact = 1 ;
cout << "Enter a positive integer number";
cin >> n;
for( i=1; i<=n; ++i)
     fact *= i;
 cout << fact << endl;
return 0 ;
}
```

برنامه زیر مجموع اعداد صحیح و متوالی بین ۱ تا n را محاسبه نموده و نمایش می‌دهد.

```
#include        <iostream.h>
int     main( )
{
int n, i=1 ;
long s = 0 ;
cin >> n ;
for(; i<=n; i++)
      s += i;
cout << s ;
return  0 ; }
```

برنامه زیر ارقام 0 تا 9 را نمایش می‌دهد.

```
#include    <iostream.h>
int main( )
{
int  j=0 ;
for( ; j <= 9 ; )
      cout << j++ << endl;
return 0 ;
}
```

تمرین های فصل دوم

۱. برنامه ای بنویسید که دو عدد را از کاربر دریافت کرده و مجموع، تفاضل، حاصل ضرب، خارج قسمت و باقیمانده را به صورت مناسب نمایش دهد؟
۲. برنامه ای بنویسید که دو عدد را از کاربر دریافت کرده و جابه جا نماید؟
۳. برنامه ای بنویسید که یک عدد سه رقمی را از کاربر دریافت کرده و برعکس نماید، مانند ۳۶۵ را از کاربر بگیرد و ۵۶۳ را در خروجی نمایش دهد.
۴. برنامه ای بنویسید که شعاع را از کاربر گرفته و مساحت و محیط دایره را محاسبه نماید؟
۵. برنامه ای بنویسید که عددی از کاربر دریافت نماید و مشخص نماید "زوج" است یا خیر؟ اگر زوج باشد پیغام دهد "Zooj" وگرنه پیغام دهد "Fard"؟
۶. برنامه ای بنویسید که دو عدد از کاربر دریافت کرده و اعداد فرد بین آن دو عدد را نمایش دهد؟
۷. برنامه ای بنویسید که عددی را از کاربر گرفته و سری فیبوناچی را تا آن عدد نمایش دهد؟
۸. برنامه ای بنویسید که سن کاربر را گرفته و روز و ماه و سال را به کاربر به شکل مناسب نمایش دهد؟

فصل سوم

سایر ساختارهای تکرار

فهرست مطالب فصل سوم

دستورالعمل while

از این دستور العمل مانند دستورالعمل for برای تکرار یک دستورالعمل ساده یا ترکیبی استفاده می‌گردد. شکل کلی این دستور العمل بصورت زیر می‌باشد.

```
while( شرط)
{
 دستورالعمل ۱ ;
 دستورالعمل ۲ ;
    .
    .
 دستورالعمل n ;
}
```

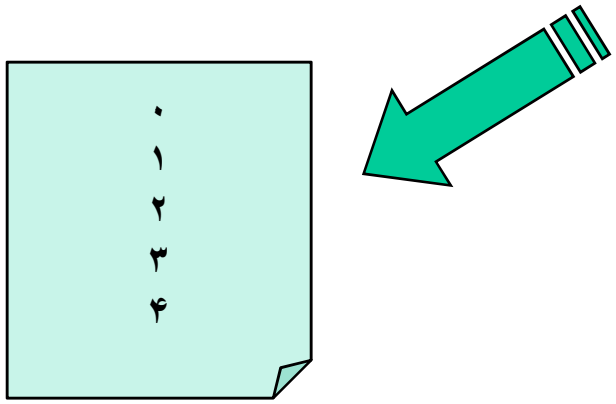
تفاوت دستورهای while و for

دستورالعمل for زمانی استفاده میشود که تعداد دفعات تکرار از قبل مشخص و معین باشد. در صورتیکه تعداد دفعات تکرار مشخص نباشد بایستی از دستورالعمل while استفاده نمود.

مثال :

```
int x=0
while(x<5)
cout << x ++<< endl;
```

با اجرای قطعه برنامه فوق مقادیر زیر نمایش داده میشود :



برنامهٔ فوق n مقدار از نوع اعشاری را گرفته میانگین آنها را محاسبه و در متغیر avg قرار می‌دهد.

```
#inclode   <iostream.h>
int main( )
{
int count = 0 , n;
float  x, sum = 0 , avg ;
cin >> n ; /*  تعداد مقادیر ورودی n*/
while(count < n){
cin >> x ;
sum += x ;
++ count ; }
avg = sum / n ;
cout << avg << endl;
return 0 ; }
```

دستورالعمل do while

این دستور العمل نیز برای تکرار یک دستورالعمل ساده یا ترکیبی استفاده می‌شود. شکل کلی این دستورالعمل بصورت زیر می‌باشد.

```
do
{
 دستورالعمل ۱ ;
 دستورالعمل ۲ ;
    .
    .
 دستورالعمل n ;
} while( شرط);
```

تفاوت دستورهای while وdo while

در دستورالعمل while ابتدا مقدار شرط ارزیابی شده اما در دستورالعمل do while ابتدا دستورالعمل اجرا شده سپس مقدار شرط ارزیابی می‌گردد. بنابراین دستورالعمل do while حداقل یک بار انجام میشود .

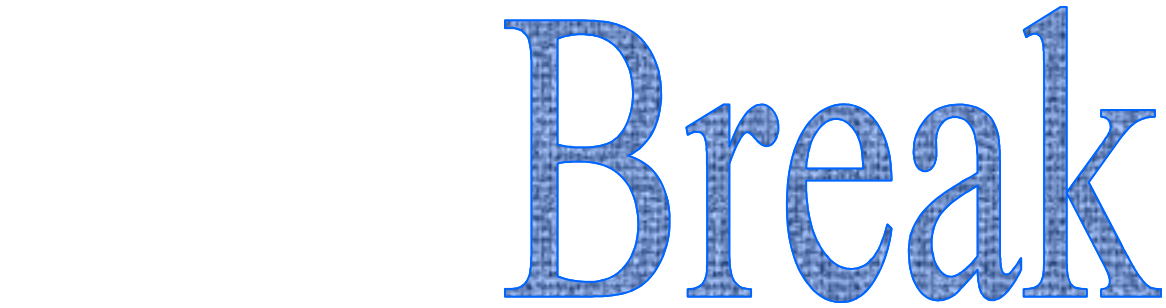
مثال :

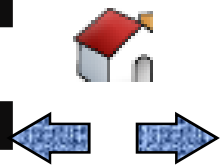
```
#include   <iostream.h>
int main( )
{
int count = 0;
do
cout << count ++<<endl ;
while(count <= 9);
return 0 ; }
```

ارقام 0 تا 9 را روی ده خط نمایش می‌دهد

دستورالعمل break

این دستورالعمل باعث توقف دستورالعملهای تکرار(for , while ,do while) شده و کنترل به خارج از این دستورالعملها منتقل می‌نماید.





مثال ١ :

```
#include   <iostream.h>

int main( )
{
float x, s=0.0 ;
cin >> x ;
while(x <= 1000.0) {
if(x < 0.0){
cout << "Error-Negative Value" ;
break;
}
s += x ;
cin >> x ;}
cout << s << endl ;
return 0 ; }
```

مثال ٢:

```
#include   <iostream.h>
int main( )
{
int count = 0 ;
while( 1 )
{
count ++ ;
if(count > 10 )
break ;
}
cout << "counter :   " << count << "\n";
return 0 ;
}
```

مثال ٣:

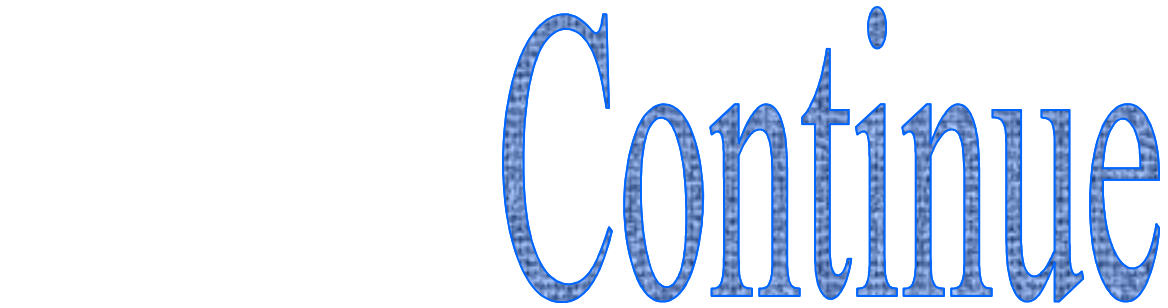
```
#include   <iostream.h>
void  main( )
{
int count;
float x, sum = 0;
cin >> x ;
for(count = 1; x < 1000 . 0; ++ count )
{
cin >> x ;
if(x < 0.0) {
cout << "Error – Negative value " <<endl;
break ;
}
sum += x ; }
cout << sum << '\n' ; }
```

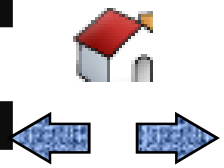
مثال ٤:

```
#include   <iostream.h>
int main( )
{
float x , sum = 0.0 ;
do
{
cin >> x ;
if(x < 0.0)
     {
cout << "Error – Negative Value" << endl ;
break ;
     }
sum += x ;
} while(x <= 1000.0);
cout << sum << endl ;
return 0 ; }
```

دستورالعمل continue

از دستورالعمل continue می‌توان در دستورالعملهای تکرار do while ، while ، for استفاده نمود. این دستورالعمل باعث می‌شود که کنترل بابتدای دستورالعملهای تکرار منتقل گردد.





مثال ١:

```
#include   <iostream.h>
int main( )
{
float  x, sum = 0.0 ;
Do
 {
cin >> x ;
if(x < 0 . 0)
        {
cout  << "Error" << endl ;
continue ;
        }
sum += x ;
} while(x <= 1000.0 );
cout << sum ;
return 0 ; }
```

مثال ۲:

```
#include   <iostream.h>
int main( )
{
int n , navg = 0 ;
float  x, avg, sum = 0 ;
cin >> n ; / * عبارت از تعداد اعداد ورودی n * /
for(int count = 1 ; count <=n; ++ count )
{
cin >> x ;
if(x < 0 ) continue ;
sum += x ;
++ navg ;
}
avg = sum / navg;
cout << avg << endl ;
return 0 ;
 }
```

دستورالعمل switch

همانطور که می دانید از دستورالعمل شرطی(if else) می توان بصورت تودرتو استفاده نمود ولی از طرفی اگر عمق استفادهٔ تو در تو از این دستورالعمل زیاد گردد، درک آنها مشکل میشود . برای حل این مشکل C++ ، دستورالعمل switch که عملاً یک دستورالعمل چند انتخابی می‌باشد را ارائه نموده است.

switch

case

شكل كلي دستور العمل Switch

```
switch(عبارت)
{
case     valueone : statement;
                          break;
case      valuetwo: statement;
                         break;
:

case       valuen : statement;
                        break;
default: statement ;
}
```

مثال ١ :

```
#include   <iostream.h>
void main( )
{
unsigned int n ;
cin >> n;
switch(n)
{
     case   0:
              cout << "ZERO" << endl ;
              break;
     case   1:
              cout << "one" << endl ;
              break ;
     case 2:
              cout << "two" << endl ;
              break;
     default :
              cout << "default" << endl;
}       / * end of switch statement * /
}
```

مثال ٢ :

```
#include   <iostream.h>
void main( )
{
unsigned  int  n;
cin >> n ;
switch(n) {
case 0 :
case 1:
case 2:
        cout << "Less Than Three" << endl;
         break;
case 3:
         cout << "Equal To Three" << endl ;
          break;
default:
         cout << "Greater Than Three" << endl;
              }
}
```

در قطعه برنامه ذیل ازتابع cin.get() و دستور switch استفاده شده است.

```
char        x;
x = cin.get( );
switch(x)  {
case  ' r ' :
case  ' R ' :
            cout << "RED" << "\n" ;
            break ;
case  ' b ' :
case  ' B ' :
             cout << "BLUE" << endl ;
             break ;
case  ' y ' :
case  ' Y ' :
            cout <<  "YELLOW" << endl;
  }
```

جدول اولویت عملگرها

چپ به راست	( )
راست به چپ	Static_cast < >( ) ++ -- + - sizeof
چپ به راست	* / %
چپ به راست	+ -
چپ به راست	<< >>
چپ به راست	< <= > >=
چپ به راست	= = !=
راست به چپ	? :
راست به چپ	= += -= *= /= %=
چپ به راست	,

تمرین های فصل سوم

۱. برنامه ای بنویسید که عدد صحیحی را گرفته (While)
- همه مقسوم علیه هایش را چاپ کند
- تعداد مقسوم علیه هایش را نشان دهد
- با استفاده از شمارش مقسوم علیه ها نشان دهد عدد اول است یا خیر؟

---

۲. برنامه ای بنویسید که عددی را از ورودی گرفته(While)
- رقم های زوج موجود در آن عدد را نمایش دهد
- حاصل جمع رقم های موجود را جمع کرده و خروجی نمایش دهد

---

۳. برنامه ای بنویسید که با استفاده از دستور break، دو عدد ورودی گرفته شود و بزرگترین مقسوم علیه مشترک را نمایش دهد.(Break,While)

۴.برنامه ای بنویسید که عدد ۳ رقمی را دریافت کرده و حروفی آن را چاپ نماید (Switch)

فصل چهارم

آرایه ها

فهرست مطالب فصل چهارم

آرایه یک بعدی

آرایه یک فضای پیوسته از حافظه اصلی کامپیوتر می‌باشد که می‌تواند چندین مقدار را در خود جای دهد.

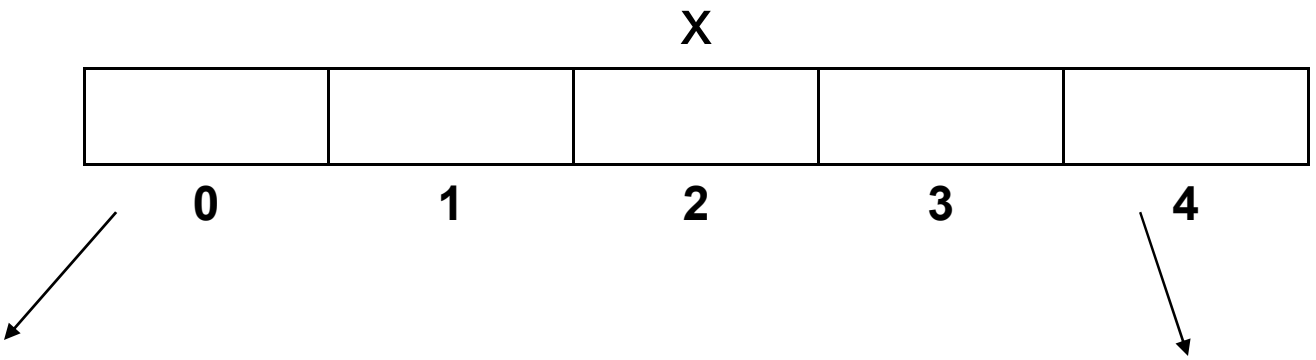
- کلیه عناصر یک آرایه از یک نوع می‌باشند.
- عناصر آرایه بوسیله اندیس آنها مشخص می‌شوند.
- در C++ ، اندیس آرایه از صفر شروع می‌شود.

کاربرد آرایه ها

آرایه‌ها در برنامه‌نویسی در مواردی کاربرد دارند که بخواهیم اطلاعات و داده‌ها را در طول اجرای برنامه حفظ نمائیم.

آرایه یک بعدی از نوع int

```
int   x[5] ;
```



اولین عنصر x[0]

پنجمین عنصر x[4]

تخصیص مقادیر اولیه به عناصر آرایه :

```
int  x[5]= {4, 2, 5, 17, 30};
```

X

4	2	5	17	30
0	1	2	3	4

دریافت مقادیر عناصر آرایه :

```
int  x[5];
for(int    i=0; i<=4; ++i)
cin >> x[ i ] ;
```

نمایش مقادیر عناصر آرایه :

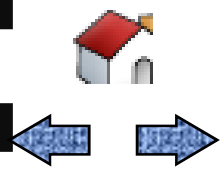
```
for(int   i=0; i<5; ++i) cout << x[ i ] ;
```

اگر تعداد مقادیر اولیه کمتر از تعداد عضوهای آرایه باشد عضوهای باقیمانده بطور اتوماتیک، مقدار اولیه صفر می‌گیرند.

```
int   x[5] = {12, 5, 7};
```

X

12	5	7	0	0
0	1	2	3	4



بایستی توجه داشت که آرایه‌ها به صورت ضمنی مقدار اولیه صفر نمی‌گیرند. برنامه نویس باید به عضو اول آرایه، مقدار اولیه صفر تخصیص دهد تا عضوهای باقی‌مانده بطور اتوماتیک، مقدار اولیه صفر بگیرند.

```
int   x[5] = {0} ;
```

X

0	0	0	0	0
0	1	2	3	4

دستور زیر یک آرایه یک بعدی شش عنصری از نوع float ایجاد می‌نماید.

```
float x[  ] = {2.4, 6.3, -17.1, 14.2, 5.9, 16.5} ;
```

X

2.4	6.3	-17.1	14.2	5.9	16.5
0	1	2	3	4	5

برنامه ذیل 100 عدد اعشاری و مثبت را گرفته تشکیل یک آرایه میدهد سپس مجموع عناصر آرایه را مشخص نموده نمایش می‌دهد.

```
#include  <iostream.h>
#include   <iomanip.h>
int main( )
{
const int arrsize = 100 ;
float x[ arrsize], tot = 0.0 ;
for(int   j=0; j<arrsize; j++)
cin >> x[ j ];
for(j=0; j<arrsize; j++)
cout << setiosflags(ios::fixed|ios :: showpoint ) << setw(12) <<
     setprecision(2) << x[ j ] << endl;
for(j=0; j<arrsize; j++)
tot += x[ j ] ;
cout << tot ;
return 0 ;
}
```

برنامه ذیل 20 عدد اعشاری را گرفته تشکیل یک آرایه داده سپس کوچکترین عنصر آرایه را مشخص و نمایش می‌دهد.

```
#include  <iostream.h>
#include   <conio.h>
int main( )
{
float   x[20], s;
int  j ;
clrscr( ) ;
for(j=0; j<20 ; ++j) cin >> x[ j ];
s = x[0 ] ;
for(j=1; j<20; ++j)
if(x[ j] <s) s = x[ j ];
cout << s << endl;
return 0;
}
```

برنامه زیر 100 عدد اعشاری را گرفته بروش حبابی (Bubble sort) بصورت صعودی مرتب می‌نماید.

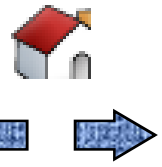
```
#include  <iostream.h>
#include  <conio.h>
int main ( )
{
float  x[100] , temp;
int   i,j ;
clrscr(  );
for(i=0; i<100; ++i)
      cin >> x[i ];
for(i=0; i<99; i++)
      for(j=i+1 ; j<100; j++)
                        if(x[ j ] < x[i ]
                        {
                                    temp = x[ j ] ;
                                    x[ j ] = x[ i ];
                                    x[ i ] = temp ;
                        }
for(i=0; i<=99; i++)
      cout << x[ i ] << endl;
return 0 ;
```

آرایه‌های دوبعدی (ماتریس‌ها)

ماتریسها بوسیله آرایه‌های دوبعدی در کامپیوترنمایش داده میشوند.

```
int a[3][4];
```

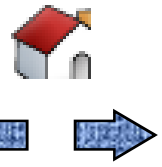
	ستون 0	ستون 1	ستون 2	ستون 3
سطر 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
سطر 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
سطر 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]



تخصیص مقادیر اولیه به عناصر آرایه :

```
int a[3][4]={ {1,2,3,4}, {5,6,7,8}, {9,10,11,12} } ;
```

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12



نكته ١ :

```
int   a[3][4]= { {1}, {2,3} , {4,5,6} } ;
```

	0	1	2	3
0	1	0	0	0
1	2	3	0	0
2	4	5	6	0

نكته ٢ :

```
int   a[3][4]= {1, 2, 3, 4,5 } ;
```

	0	1	2	3
0	1	2	3	4
1	5	0	0	0
2	0	0	0	0

نکته ۳ :

در یک آرایهٔ دواندیسی، هر سطر، در حقیقت آرایه‌ای یک اندیسی است. در اعلان آرایه‌های دواندیسی ذکر تعداد ستونها الزامی است.

```
int   a[  ][4]={1,2,3,4,5};
```

	0	1	2	3
0	1	2	3	4
1	5	0	0	0

برنامه زیر یک ماتریس 3*4 را گرفته مجموع عناصر آن را مشخص نموده و نمایش می‌دهد.

```
#include  <iostream.h>
#include  <conio.h>
int main( )
{
float   x[3][4], total= 0.0;
int   i, j ;
// generate matrix x.
for(i=0; i<3; ++i)
for (j=0; j<4; j++)
cin >> x[ i ][ j ];
// calculate the sum of elements.
for(i=0; i<3; ++i)
for(j=0; j<4; j++)
total + = x [ i ][ j ];
cout << "total =  " << total << endl;
return 0 ;
}
```

تمرین های فصل چهارم

۱. برنامه ای بنویسید که ۲۰ عدد اعشاری دریافت کرده
   - چند تای این اعداد از اولین عدد وارد شده بزرگتر است و چاپ کند؟
   - چند تای این اعداد از آخرین عدد وارد شده کوچکتر است و چاپ کند؟

---

۲. برنامه ای بنویسید که ۲۰ عدد اعشاری دریافت کرده
   - چند تای این اعداد از میانگین اعداد کوچکتر و چند تا بزرگتر و چاپ کند؟

---

۳. برنامه ای بنویسید که ۲۰ عدد اعشاری دریافت کرده
   - اعداد تکراری و تعداد آن اعداد را به شکل مناسب نمایش دهد

---

۴. برنامه ای بنویسید که یک ماتریس ۴*۳ دریافت کرده
   - مجموع هر سطر را چاپ نماید؟
   - مجموع هر ستون را چاپ نماید؟
   - بزرگترین عنصر هر سطر را نمایش دهد؟
   - کوچکترین عنصر هر ستون را نمایش دهد؟

فصل پنجم

توابع

فهرست مطالب فصل پنجم

تعریف توابع

استفاده از توابع در برنامه‌ها به برنامه‌نویس این امکان را می‌دهد که بتواند برنامه‌های خود را به صورت قطعه قطعه برنامه بنویسد. تا کنون کلیه برنامه‌هائی که نوشته‌ایم فقط از تابع ( )main استفاده نموده‌ایم .

شکل کلی توابع بصورت زیر می‌باشند :

نوع مقدار برگشتی

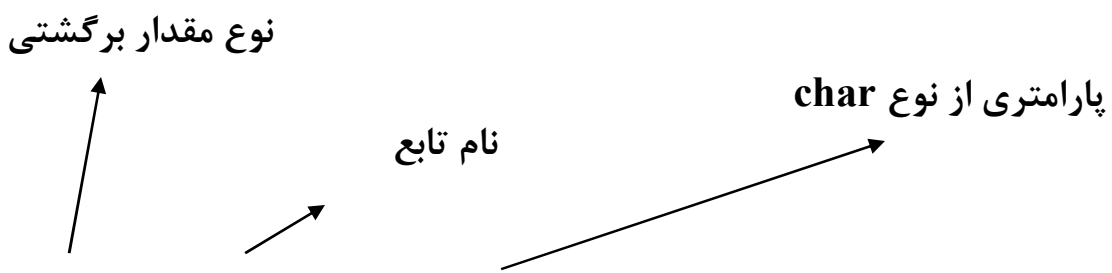
لیست پارامتر ها جهت انتقال اطلاعات از تابع احضار کننده به تابع فراخوانده شده

```
return-value-type   function-name (parameter-list)
{

declaration and statements


}
```

نام تابع

تعریف اعلان‌های تابع و دستورالعملهای اجرائی

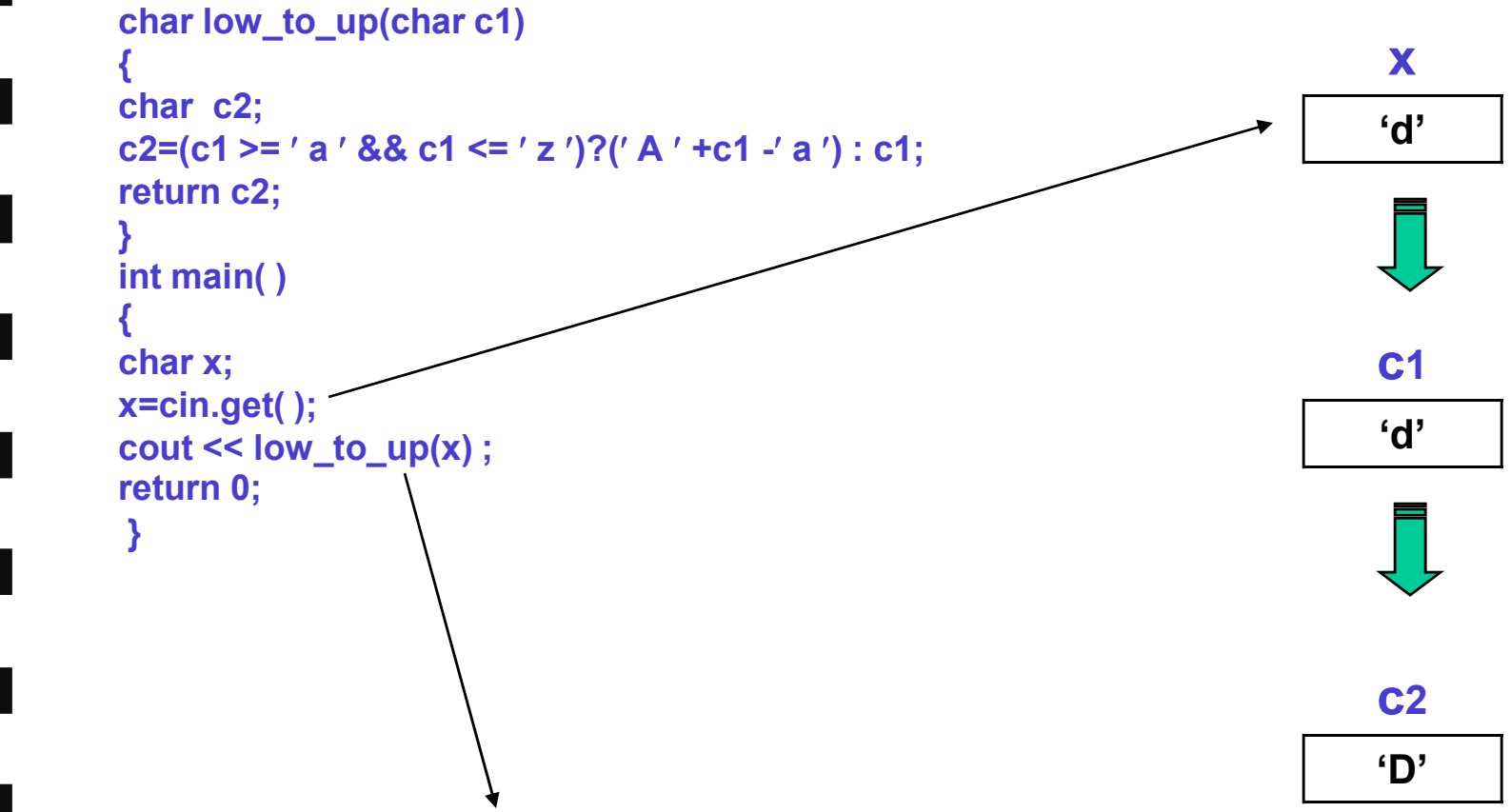
تابع زیر یک حرف کوچک را به بزرگ تبدیل می‌نماید.




```
char   low_to_up (char c1)
{
char c2;
c2 = (c1>= ′ a ′ && c1<= ′ z ′)?(′ A ′ + c1- ′ a ′): c1;
return (c2) ;
}
```

برنامه کامل که از تابع قبل جهت تبدیل یک حرف کوچک به بزرگ استفاده می‌نماید.

```
#include  <iostream.h>
char low_to_up(char c1)
{
char  c2;
c2=(c1 >= ' a ' && c1 <= ' z ')?(' A ' +c1 -' a ') : c1;
return c2;
}
int main( )
{
char x;
x=cin.get( );
cout << low_to_up(x) ;
return 0;
 }
```



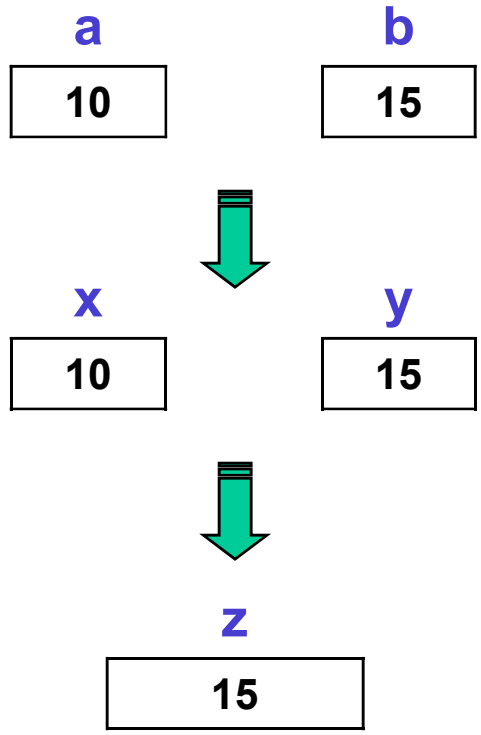
آرگومان

تابع maximum دو مقدار صحیح را گرفته بزرگترین آنها را برمیگرداند.

```
int   maximum(int x, int y)
{
int z ;
z=(x >= y)? x : y;
return z;
}
```

برنامه کامل که از تابع maximum جهت یافتن ماکزیمم دو مقدار صحیح استفاده می نماید.

```
#include  <iostream.h>
int   maximum(int x , int y)
{
int z ;
z=(x > y)? x : y ;
return z;
}
int main( )
{
int a, b ;
cin >> a >> b ;
cout << maximum(a,b);
return 0;
}
```



آرگومانهای تابع maximum a, b

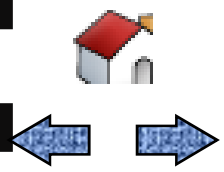
نکته ۱ :

اسامی پارامترها و آرگومانهای یک تابع
می‌توانند همنام باشند.

برنامه زیر یک مقدار مثبت را گرفته فاکتوریل آنرا محاسبه نموده نمایش می‌دهد.

$$x!=1*2*3*4*...*(x-1)*x$$

```
#include    <iostream.h>
long int factorial(int n)
{
long int prod=1;
if(n>1)
for(int  i=2; i<=n; ++i)
prod *=i;
return(prod);
}
int main( )
{
int n;
cin >> n ;
cout << factorial(n) ;
return 0 ;
}
```



n در main

3

n در factorial

3

i در factorial

2,3,4

prod در factorial

6

نکته ۲ :

وقتی در تابعی، تابع دیگر احضار می‌گردد
بایستی تعریف تابع احضار شونده قبل از
تعریف تابع احضار کننده در برنامه ظاهر گردد.

نکته ۳ :

اگر بخواهیم در برنامه‌ها ابتدا تابع **main** ظاهر گردد بایستی **prototype** تابع یعنی پیش نمونهٔ تابع که شامل نام تابع، نوع مقدار برگشتی تابع، تعداد پارامترهائی را که تابع انتظار دریافت آنرا دارد و انواع پارامترها و ترتیب قرارگرفتن این پارامترها را به اطلاع کامپیلر برساند.

در اسلاید بعد مثالی در این زمینه آورده شده است.

```
#include    <iostream.h>
#include   <conio.h>
long int factorial(int); // function prototype
int main( )
{
int n;
cout << "Enter a positive integer" << endl;
cin >> n;
cout << factorial(n) << endl;
return 0 ;
}
long int factorial(int n)
{
long int prod = 1;
if(n>1)
for(int i=2; i<=n; ++i)
prod *= i;
return(prod);
}
```

نکته ۶ :

در صورتی که تابع مقداری بر نگرداند نوع مقدار برگشتی تابع را **void** اعلان می‌کنیم. و در صورتیکه تابع مقداری را دریافت نکند بجای **parameter- list** از **void** یا ( ) استفاده میگردد.

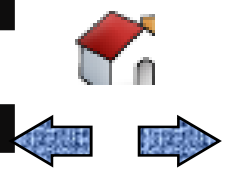
در اسلاید بعد مثالی در این زمینه آورده شده است.

```
#include    <iostream.h>
#include   <conio.h>
void maximum(int , int) ;
int main( )
{ int x, y;
clrscr( )
cin >> x >> y;
maximum(x,y);
return 0;
}
void maximum(int x, int y)
{
int z ;
z=(x>=y) ? x : y ;
cout << "max value \n" << z<< endl;
return ;
}
```

تابع مقداری بر نمی گرداند.

احضار بوسیله مقدار (Call By Value)

```
#include    <iostream.h>
int modify(int)
int main( )
{
int a=20;
cout << a << endl;
modify(a) ;
cout << a << endl;
return 0 ;
}
int modify(int a)
{
a *= 2;
cout << a << endl;
return ;
}
```



a در main

20

a در modify

20

a در modify

40

خروجی برنامه :

20

40

20

احضار بوسیله مقدار (Call By Value)

```
#include    <iostream.h>
int modify(int)
int main( )
{
int a=20;
cout << a << endl;
modify(a) ;
cout << a << endl;
return 0 ;
}
int modify(int a)
{
a *= 2;
cout << a << endl;
return ;
}
```

a در main

20

a در modify

20

a در modify

40

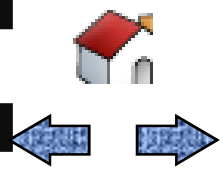
در این نوع احضار تابع حافظه‌های مورد استفاده آرگومانها و پارامترها از هم متمایزند و هرگونه تغییر در پارامترها باعث تغییر در آرگومانهای متناظر نمی‌گردد.

نکته ۵ :

هر زمان که نوع مقدار برگشتی تابع **int** می‌باشد نیازی به ذکر آن نیست و همچنین نیازی به تعریف پیش نمونه تابع نمی‌باشد.

تابع بازگشتی (recursive functions)

توابع بازگشتی یا recursive توابعی هستند که
وقتی احضار شوند باعث می‌شوند که خود را احضار نمایند.



نحوه محاسبه فاکتوریل از طریق تابع بازگشتی

$n != 1*2*3*\ldots*(n-1)*n$

$f(n) = n !$

$$f(n) = \begin{cases} 1 & \text{اگر } n=0 \\ n * f(n-1) & \text{در غیر اینصورت} \end{cases}$$

$n!= 1* 2* 3* \ldots *(n-2) *(n-1) *n$

$n!=(n-1)! *n$

در اسلاید بعد تابع بازگشتی مورد نظر پیاده سازی شده است.

تابع بازگشتی محاسبه فاکتوریل

```
#include    <iostream.h>
long int factorial(int) ;
int main( )
{
     int n ;
     cout << " n= " ;
     cin >> n ;
     cout << endl << " factorial = " << factorial(n) << endl;
     return 0 ;
}
long int factorial(int n)
{
     if(n<=1)
          return(1);
     else
          return(n *factorical(n-1) ) ;
}
```

نحوه محاسبه n امین مقدار دنباله فیبناکی از طریق تابع بازگشتی

دنباله فیبناکی : 0 , 1, 1, 2, 3, 5, 8, 13, 21 , 34, …

$$fib(n)=\text{جمله } n \text{ ام}=\begin{cases} 0 & \text{اگر } n=1 \\ 1 & \text{اگر } n=2 \\ fib(n-1)+fib(n-2) & \text{در غیر اینصورت} \end{cases}$$

در اسلاید بعد تابع بازگشتی مورد نظر پیاده سازی شده است.

برنامه‌زیر n امین مقدار دنبالهٔ فیبناکی (fibonacci) را مشخص و نمایش می‌دهد.

```
#include    <iostream.h>
long int fib(long int); // forward declaration
int main( )
{
      long int r ;
      int n ;
      cout << " Enter an integer value " << endl ;
      cin >> n ;
      r = fib(n) ;
      cout << r << endl ;
      return 0 ;
}
long int fib(long int n)
{
      if(n = = 1  ||   n= = 2)
            return 1 ;
      else
            return(fib(n-1) + fib(n-2) ) ;
}
```

برنامه زیر یک خط متن انگلیسی را گرفته آنرا وارون نموده نمایش می‌دهد.

```
#include    <iostream.h>
void reverse(void) ; // forward declaration
int main( )
{
     reverse( ) ;
     return 0 ;
}
void reverse(void)
// read  a line of characters and reverse it
{
     char c ;
     if(( c=cin.get( )) != ′ \n ′)
               reverse( );
     cout << c ;
     return ;
}
```

نکته :

استفاده از آرایه‌ها بعنوان پارامتر تابع مجاز است.

در اسلاید بعد به یک مثال توجه نمایید.

در برنامه زیر تابع modify آرایه a را بعنوان پارامتر می‌گیرد.

```
#include    <iostream.h>
void  modify(int [  ] ); // forward declaration
int main( )
{
int  a[5] ;
for(int j=0; j<=4; ++j)
a[ j ] = j+1 ;
modify(a) ;
for(j=0; j<5; ++j)
cout << a[ j ] << endl ;
return 0 ;
}
void modify(int a[  ]) // function definition
{
for(int j=0; j<5; ++j)
a[ j ] += 2 ;
for(j=0; j<5; ++j)
cout << a[ j ] << endl ;
return ;
}
```

خروجی :

```
1
2
3
4
5

3
4
5
6
7
```

نکته :

در صورتیکه آرایه بیش از یک بعد داشته باشد بعدهای دوم به بعد بایستی در تعریف تابع و پیش نمونه تابع ذکر گردد.

در اسلاید بعد به یک مثال توجه نمایید.

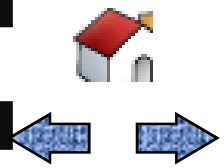
```
#include    <iostream.h>
void printarr(int [   ][ 3 ]);
int main( )
{
      int  arrl [2][3] = { {1,2,3}, {4,5,6} };
      arr2 [2][3]= {1,2,3,4,5};
      arr3 [2][3]={ {1,2}, {4} };
      printarr(arr1);
      cout << endl ;
      printarr(arr2);
      cout << endl ;
      printarr(arr3);
      return 0 ;
}
void printarr(int a[  ][3] )
{
      for(int i=0; i<2; i++)
      {
            for(int j=0; j<3; j++)
                  cout << a[ i ][ j ] << ' ';
                  cout << endl ;
      }
}
```

خروجی :

```
1     2     3
4     5     6
1     2     3
4     5     0
1     2     0
4     0     0
```

توابع درون خطی (inline)

کلمه inline بدین معنی است که به کامپیلر دستور می‌دهد که یک کپی از دستورالعلمهای تابع در همان جا (در زمان مقتضی) تولید نماید تا از احضار تابع ممانعت بعمل آورد.



اشکال توابع inline

بجای داشتن تنها یک کپی از تابع ، چند کپی از دستورالعملهای تابع در برنامه اضافه می‌شود که باعث بزرگ شدن اندازه یا طول برنامه می‌شود.
بنابراین از **inline** برای توابع کوچک استفاده می‌گردد.

مثالی از توابع درون خطی

```
#include    <iostream.h>
inline float cube(const float s) {return s*s*s; }
int main( )
{
float side ;
cin >> side ;
cout << side << cube(side) << endl ;
return 0 ;
}
```

انتقال پارامترها از طریق ارجاع

تاکنون وقتی تابعی را احضار می‌کردیم یک کپی از مقادیر آرگومانها درپارامترهای متناظر قرار می‌گرفت . این روش احضار بوسیله مقدار یا **call by value** نامیده شد.
در انتقال پارامترها از طریق ارجاع در حقیقت حافظه مربوط به آرگومانها و پارامترهای متناظر بصورت اشتراکی مورد استفاده قرار می‌گیرد. این روش **call by reference** نامیده می‌شود .

انتقال پارامترها از طریق ارجاع

در این روش پارامترهائی که از طریق **call by reference** عمل می‌نمایند در پیش نمونه تابع قبل از نام چنین پارامترهائی از **&** استفاده می‌شود.واضح است که در تعریف تابع نیز بهمین طریق عمل می‌شود.

```
#include    <iostream.h>
int vfunct(int); // for
void rfunct (int &) ;
int main( )
{
      int x=5, y=10;
      cout << x << endl << vfunct(x) << endl << x << endl ;
      cout  << y << endl ;
      rfunct(y) ;
      cout << y << endl ;
      return 0 ;
}
int vfunct(int a)
{
      return a *= a ;
}
void rfunct(int &b)
{
      b *= b ;
}
```

مثال :

x	y
5	10

خروجی :

5
25
5
۱۰

مقدار آرگومان x تغییر نمی کند.

x	y b
5	10 100

ادامه خروجی :

100

نکته :

وقتی پارامتری بصورت call by reference اعلان می‌گردد این بدان معنی است که با تغییر مقدار این پارامتر در تابع احضار شده مقدار آرگومان متناظر نیز تغییر می‌نماید.

برنامه‌زیر با استفاده از fswap دو مقدار اعشاری را مبادله می‌نماید.

```
#include    <iostream.h>
void fswap(float & , float & );
int main( )
{
      float a=5.2, b=4.3;
      cout << a << endl << b ;
      fswap( a , b) ;
      cout << a << endl << b ;
      return 0 ;
}
void fswap(float &x , float & y)
{
      float t;
      t = x ;
      x = y ;
      y = t ;
}
```

کلاس‌های حافظه (storage classes)

متغیرها بدو طریق متمایز مشخص می‌شوند یکی بوسیله نوع (type) آنها و دیگری بوسیله کلاس حافظه آنها. نوع متغیر قبلاً اشاره شده بعنوان مثال int ، float ، double ، . . . ولی کلاس حافظهٔ یک متغیر در مورد طول عمر و وسعت و دامنهٔ متغیر بحث می‌نماید.

در اسلاید بعد به انواع کلاس حافظه می پردازیم.

بطور کلی کلاس حافظه متغیرها به چهار دستهٔ تقسیم می‌گردد :

۱. automatic
۲. static
۳. external
۴. register

automatic

متغیرهای **automatic** در درون یک تابع تعریف می‌شوند و در تابعی که اعلان می‌شود بصورت متغیرهای محلی برای آن تابع می‌باشند. حافظه تخصیص داده شده به متغیرهای **automatic** پس از اتمام اجرای تابع از بین می رود بعبارت دیگر وسعت و دامنهٔ متغیرهای از نوع **automatic** تابعی می‌باشد که متغیر در آن اعلان گردیده است.

static

متغیرهای **static** نیز در درون توابع تعریف میشوند و از نظر وسعت و دامنه شبیه متغیرهای **automatic** هستند ولی در خاتمهٔ اجرای تابع، حافظه وابسته به این نوع متغیرهـــــا از بین نمی‌رود بلکه برای فراخوانی بعدی تابع باقی می‌ماند.

در اسلاید بعد به یک مثال از کاربرد این نوع کلاس حافظه می پردازیم.

مثال :

```
#include    <iostream.h>
// program to calculate successive fibonacci numbers
long int fib(int) ;
int main( )
{
int n ;
cout << " how many fibonacci numbers?" ;
cin >> n ;
cout << endl ;
for(int j=1; j<=n; ++j )
cout << j << "    " << fib(j) << endl ;
return 0 ;
}
long int fib(int count)
{
static long int t1 = 1, t2=1;
long int  t ;
t =(count <3) ?1 : t1 + t2 ;
t2 = t1 ;
t1 = t ;
return(t) ;
}
```

بایستی توجه داشت که اگر در توابع به متغیرهای از نوع static مقدار اولیه تخصیص ندهیم مقدار صفر بصورت اتوماتیک برای آنها در نظر گرفته می‌شود.

external

متغیرهای از نوع **external** متغیرهائی هستند که در بیرون از توابع اعلان میشوند و وسعت و دامنه فعالیت آنها کلیه توابعی می‌باشد که در زیر دستور اعلان متغیر قرار دارد.

در اسلاید بعد به یک مثال از کاربرد این نوع کلاس حافظه می پردازیم.

مثال :

```
#include    <iostream.h>
int w; // external variable
functa(int x, int y)
{
cout << w ;
w = x + y ;
cout << endl << w << enld;
return x%y ;
}
int main( )
{
int a, b, c, d;
cin >> a >> b ;
c=functa(a, b) ;
d=functa(w, b+1);
cout << endl << c << endl << d << endl << w ;
return 0 ;
}
```

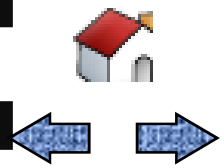
بایستی توجه داشت که اگر در توابع به متغیرهای از نوع external مقدار اولیه تخصیص ندهیم مقدار صفر بصورت اتوماتیک برای آنها در نظر گرفته می‌شود.

register

وقتی متغیری از نوع **register** اعلان می‌شود از کامپیوتر عملاً درخواست می‌شود که به جای حافظه از یکی از رجیسترهای موجود استفاده نماید.

کاربرد کلاس register

معمولاً از نوع رجیستر برای شاخص‌های دستور تکرار و یا اندیسهای آرایه‌ها استفاده می‌شود. بایستی توجه داشت که متغیرهای از نوع رجیستر قابل استفاده دردستور cin نمی‌باشند



سربارگذاری توابع (function overloading)

در C++ این امکان وجود دارد که دریک برنامه بتوانیم از چند توابع هم نام استفاده نمائیم مشروط بر این که پارامترهای این توابع متفاوت باشند. (از نظر تعداد پارامتر و یا نوع پارامترها و ترتیب آنها)

مثال :

```
#include    <iostream.h>
float addf(float , int);
int addf(int , int);
int main( )
{
int a=5, b=10 ;
float d=14.75 ;
cout << addf(a , b) << endl;
cout << addf(d ,  b) << endl;
return 0 ;
}
int addf(int x, int y)
{
return x+y ;
}
float addf(float x, int y)
{
return x+y ;
}
```

تمرین های فصل پنجم

۱. برنامه ای بنویسید که **m,n** را خوانده و حاصل زیر را محاسبه نماید:

((a+b)^a (2a+b)^b) / (3a+2b)^2

۲. برنامه ای بنویسید که **m,n,p** را خوانده و حاصل زیر را محاسبه نماید:

(m^n . n! . n^p) / ((m+p)^m . p!)

۳. برنامه ای بنویسید که عملیات تقسیم را با عملگرهای + و – انجام دهد؟

۴.برنامه ای بنویسید که عدد **N** را گرفته و معکوس تا آن عدد را نشان دهد؟

6,5,4,3,2,1

۵. برنامه ای بنویسید که **R,N** را بخواند و عبارت زیر را محاسبه نماید (فاکتوریل به صورت بازگشتی حل شود)

N!/N!(R-N)!

تابع strcmpi(s1, s2)

رشته‌های s1 و s2 را با هم مقایسه نموده (بدون توجه به حروف کوچک و بزرگ) اگر رشته s1 برابر با رشته s2 باشد مقدار صفر و اگر رشته s1 کوچکتر از رشته s2 باشد یک مقدار منفی در غیر اینصورت یک مقدار مثبت بر می‌گرداند.

تابع strcmp(s1, s2)

رشته‌های s1 و s2 را با هم مقایسه نموده اگر s1 برابر با s2 باشد مقدار صفر و اگر رشته s1 کوچکتر از رشته s2 باشد یک مقدار منفی در غیر اینصورت یک مقدار مثبت برمی‌گرداند.

تابع strncmp(s1, s2,n)

حداکثر n کرکتر از رشتهٔ s1 را با n کرکتر از رشتهٔ s2 مقایسه نموده در صورتیکه s1 کوچکتر از s2 باشد یک مقدار منفی، اگر s1 مساوی با s2 باشد مقدار صفر در غیر اینصورت یک مقدار مثبت برمیگرداند.

تابع strcat(s1, s2)

دو رشتهٔ s1 و s2 را بعنوان آرگومان گرفته رشتهٔ s2 را به انتهای رشتهٔ s1 اضافه می‌نماید. کرکتر اول رشتهٔ s2 روی کرکتر پایانی '\0' رشتهٔ s1 نوشته می‌شود ونهایتاً رشتهٔ s1 را برمیگرداند.

تابع strncat(s1, s2,n)

دو رشته s1 و s2 ومقدار صحیح و مثبت n را بعنوان آرگومان گرفته، حداکثر n کرکتر از رشتهٔ s2 را در انتهای رشتهٔ s1 کپی می‌نماید. اولین کرکتر رشته s2 روی کرکتر پایانی '\0' رشتهٔ s1 می‌نویسد ونهایتاً مقدار رشتهٔ s1 را برمیگرداند.

تابع strlen(s)

رشتهٔ s را بعنوان آرگومان گرفته طول رشته را مشخص می‌نماید.

تابع strlen(s)

رشتهٔ s را بعنوان آرگومان گرفته طول رشته را مشخص می‌نماید.

تابع strcpy(s1,s2)

دو رشتهٔ s1 و s2 را بعنوان آرگومان گرفته رشتهٔ s2 را در رشتهٔ s1 کپی می‌نماید و نهایتاً مقدار رشتهٔ s1 را بر می‌گرداند.

تابع strncpy(s1, s2,n)

دو رشتهٔ s2 , s1 و مقدار صحیح و مثبت n را بعنوان آرگومان گرفته، حداکثر n کرکتر را از رشتهٔ s2 در رشتهٔ s1 کپی نموده، نهایتاً مقدار رشتهٔ s1 را برمیگرداند.

نکته مهم

برای استفاده از توابع مربوط به رشته‌ها بایستی حتماً در ابتدا برنامه #include <string.h> را قرار دهیم.

مثال :

```
#include      <iostream.h>
#include      <string.h>
#include      <conio.h>
int main( )
{
char *s1= ″happy birthday″;
char *s2= ″happy holidays ″;
clrscr( );
cout << strcmp(s1, s2) << endl;
cout << strncmp(s1, s2, 7) << endl ;
return 0;
 }
```

مثال :

```
#include      <iostream.h>
#include      <string.h>
#include      <conio.h>
int main( )
{
char *s = ″computer″;
clrscr( );
cout << strlen(s);
return 0; }
```

فصل ششم

برنامه نویس شی گرا

تعریف شی گرایی

برنامه نویسی شئ گرا یا oop یک روش جدید برنامه نویسی می‌باشد که در آن از ویژگی ساختیافته همراه با چند ویژگی‌های قوی جدید استفاده می‌شود. زبان برنامه نویسی C++ امکان استفاده از oop را به راحتی فراهم می‌نماید.

تعریف شی گرایی

در oop ، بطور کلی مساله به تعدادی زیرگروه قطعات مربوط بهم شکسته می‌شود که برای هر زیر گروه code و data تهیه شده و نهایتاً این زیرگروه‌ها تبدیل به unit ها یا واحدهائی می‌شود که objects (یا اشیاء) نامیده میشوند.

نکته مهم :

تمام زبانهای برنامه نویسی شی‌گرا دارای سه خصوصیت مشترک زیر می‌باشند :

الف: encapsulation (محصورسازی)

ب: polymorphism (چندریختی)

ج: inheritance (ارث بری)

محصورسازی (Encapsulation)

محصورسازی مکانیزمی است که **code** و **data** را بهم وصل نموده و هر دوی آنها را از استفاده‌های غیرمجاز مصون نگه می‌دارد. شی یک مؤلفه منطقی است که **data** و **code** را محصور نموده و **code** باعث دستکاری و پردازش **data** می‌شود.

polymorphism (چند ریختی)

چند ریختی مشخصه‌ای است که بیک وسیله امکان میدهد که باتعدادی
از سیستمها یا عمیات یا دستگاهها، مورد استفاده قرار گیرد.

inheritance (ارث بری)

ارث بری فرآیندی است که بوسیله آن یک شی (object) می‌تواند خاصیت‌های شی دیگری را دارا شود.

پایان

keywords and alternative tokens.

asm	enum	protected	typedef
auto	explicit	public	typeid
bool	extern	register	typename
break	false	reinterpret_cast	union
case	float	return	unsigned
catch	for	short	using
char	friend	signed	virtual
class	goto	sizeof	void
const	if	static	volatile
const_cast	inline	static_cast	wchar_t
continue	int	struct	while
default	long	switch	xor
delete	mutable	template	xor_eq
do	namespace	this	or_eq
double	new	throw	not
dynamic_cast	operator	true	bitand
else	private	try	and_eq
And -- or	bitor	not_eq	compl