برنامه نویسی C++

نام درس :

برنامه سازی پیشرفته

دانشگاه آزاد اسلامی کرمان – اکبری فر

تهیه کننده اسلایدها : صالح جوان – علی میر
http://software-zb.ir/

فهرست مطالب

- فصل اول : مقدمات زبان C++
- فصل دوم : ساختار های تصمیم گیری و تکرار
- فصل سوم : سایر ساختار های تکرار
- فصل چهارم : آرایه ها
- فصل پنجم : توابع
- فصل ششم : ساختارها و اشاره گرها

فصل اول

مقدمات C++

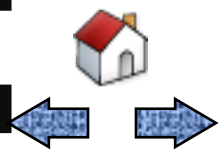
فهرست مطالب فصل اول

۱. قانون نامگذاری شناسه ها
۲. متغیر ها
۳. اعلان متغیر
۴. تخصیص مقادیر به متغیر
۵. داده های از نوع کرکتر
۶. کرکتر های مخصوص
۷. رشته ها
۸. نمایش مقادیر داده ها
۹. دریافت مقادیر
۱۰. عملگر انتساب
۱۱. عملگر های محاسباتی
۱۲. عملگرهای افزایش و کاهش
۱۳. عملگر sizeof
۱۴. عملگرهای جایگزینی محاسباتی
۱۵. اولویت عملگرها
۱۶. توضیحات (Comments)
۱۷. توابع کتابخانه
۱۸. برنامه در C++

قانون نامگذاری شناسه‌ها

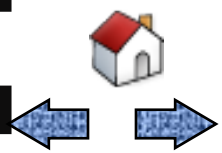
۱) حروف کوچک و بزرگ در نامگذاری شناسه‌ها متفاوت می‌باشند.

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.

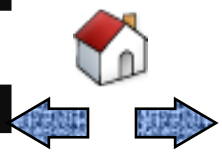


قانون نامگذاری شناسه‌ها

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

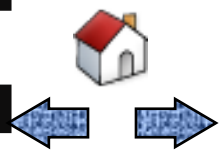
لیست کامل کلمات کلیدی



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.

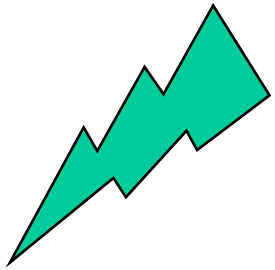
در اسلاید بعد به انواع داده ها اشاره می شود.



انواع داده ها

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

اعلان متغیرها



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

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.

داده‌های از نوع کرکتر

برای نمایش داده‌های از نوع **char** در حافظه کامپیوتر از جدول **ASCII** استفاده می‌شود. جدول اسکی به هر یک از ۲۵۶ کرکتر یک عدد منحصر بفرد بین ۰ تا ۲۵۵ تخصیص می‌دهد.

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 جمعـاً 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- - ;
```

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.

مثال :

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

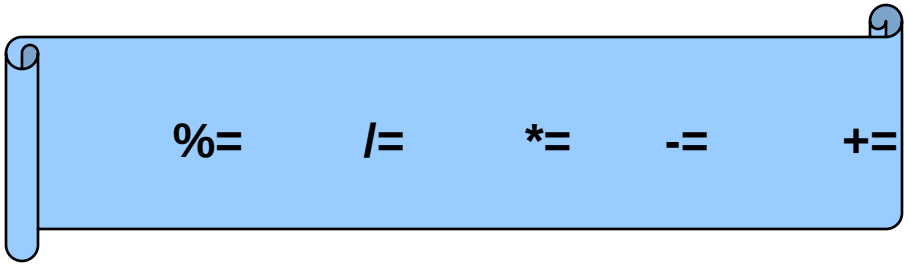
مقدار ۲ نمایش داده می‌شود .

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

مقدار ۴ نمایش داده می شود.

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 ٢ :

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

خروجي :

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*	$\mathrm{Log}_{10} d$	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 کرکترc	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.

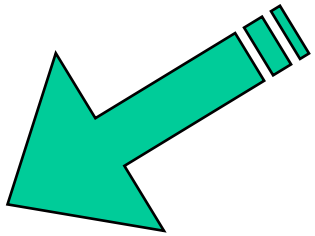
روش اول :

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

روش دوم :

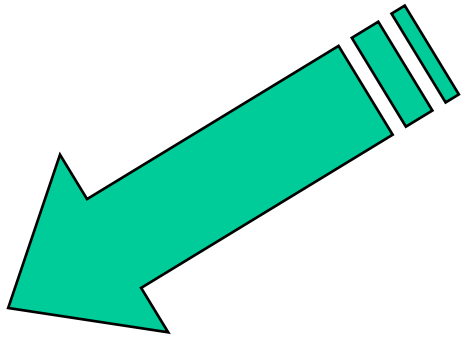
```
#include     <       >
void main( )
{
۱ دستورالعمل ;
۲ دستورالعمل ;
.
.
.
n  دستورالعمل;
}
```

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



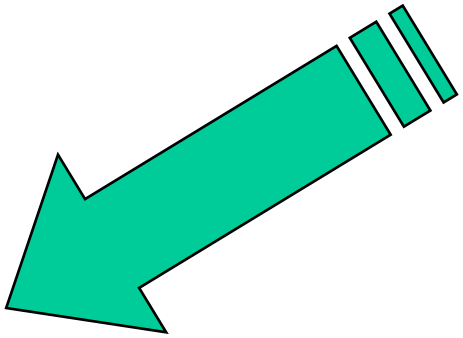
```
#include    <iostream.h>
int main( )
{
cout <<"C++ is an object oriented language \n" ;
return 0 ;
}
```

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

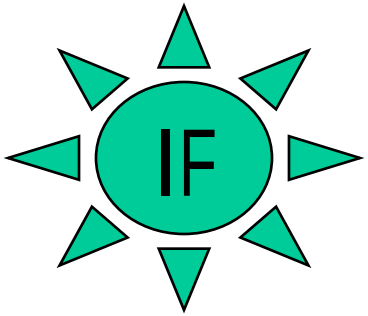
فهرست مطالب فصل دوم

۱. عملگر های رابطه ای
۲. دستورالعمل شرطی
۳. عملگر های منطقی
۴. دستورالعمل For

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 یا مساوي	> =
کوچکتر	<
کوچکتر یا مساوي	<=



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;
}
```

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

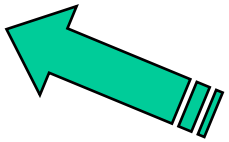
a	b	a || b
true	true	True
true	false	True
false	true	True
false	false	False

Not

a	!a
true	False
false	True

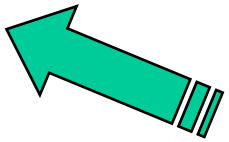
چند مثال :

```
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 ;
}
```

برنامه زیر کلیه اعداد سه رقمی که با ارقام 1 ، 2 ، 3 ایجاد می‌شوند را نمایش می‌دهد.

```
#include    <iostream.h>
int main( )
{
int  i,j,k,n;
for(i=1; i<=3; ++i)
for(j=1; j<=3; ++j)
for(k=1; k<=3; ++k)
{
n=i*100 + j*10+k;
cout << n << '\n' ;
}
return 0 ;
}
```

فصل سوم

سایر ساختارهای تکرار

فهرست مطالب فصل سوم

۱. دستورالعمل while
۲. دستورالعمل do while
۳. دستورالعمل break
۴. دستورالعمل continue
۵. دستورالعمل switch
۶. جدول اولویت عملگرها

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 ;
}
```

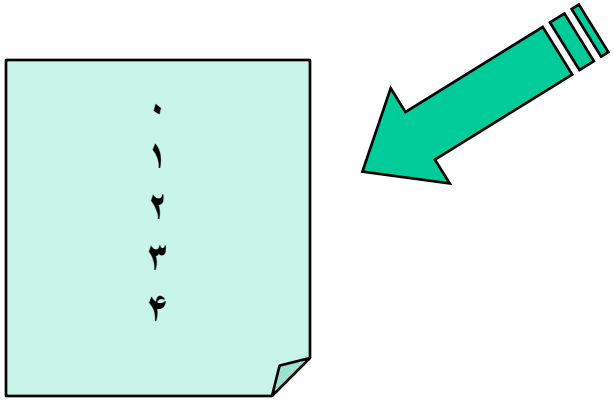
تفاوت دستورهای for وwhile

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 و do 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 for ، while ، do while 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;
              }
}
```

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	,

فصل چهارم

آرایه ها

فهرست مطالب فصل پنجم

۱. آرایه یک بعدی
۲. آرایه دو بعدی (ماتریس ها)

آرایه یک بعدی

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

کلیه عناصر یک آرایه از یک نوع می‌باشند.

عناصر آرایه بوسیله اندیس آنها مشخص می‌شوند.

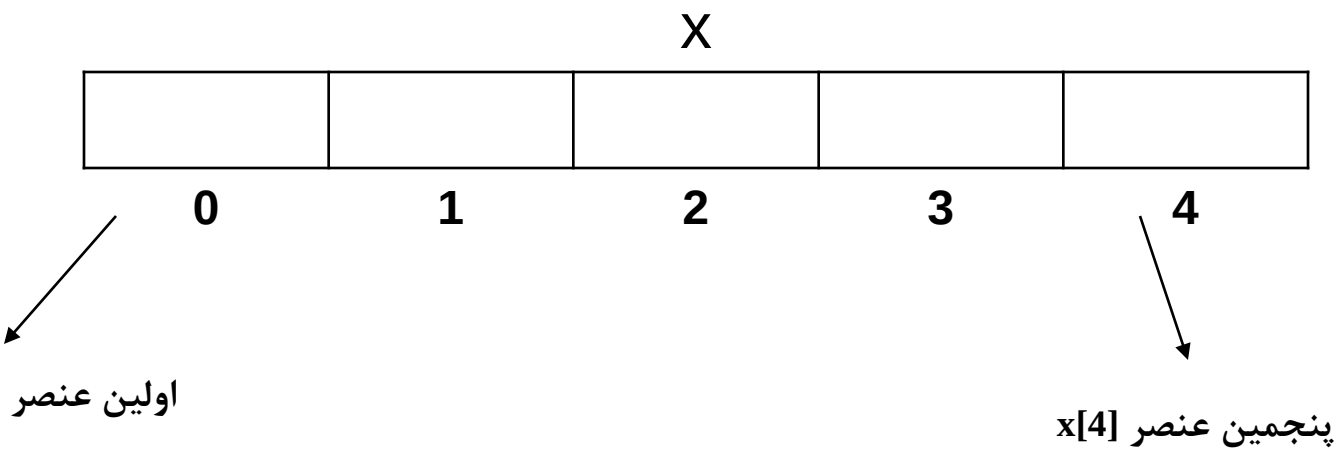
در C++ ، اندیس آرایه از صفر شروع می‌شود.

کاربرد آرایه ها

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

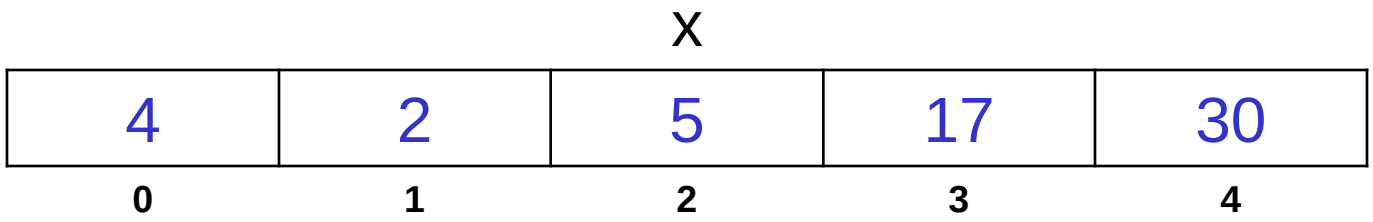
آرایه یک بعدی از نوع int

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



تخصیص مقادیر اولیه به عناصر آرایه :

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



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

نکته ۱ :

```
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

نکته ۲ :

```
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<3; j++)
cin >> x[ i ][ j ];
// calculate the sum of elements.
for(i=0; i<3; ++i)
for(j=0; j<4; j++)
tot + = x [ i ][ j ];
cout << "total =  " << total << endl;
return 0 ;
}
```

فصل پنجم

توابع

فهرست مطالب فصل پنجم

۱. تعریف تابع
۲. تابع بازگشتی

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

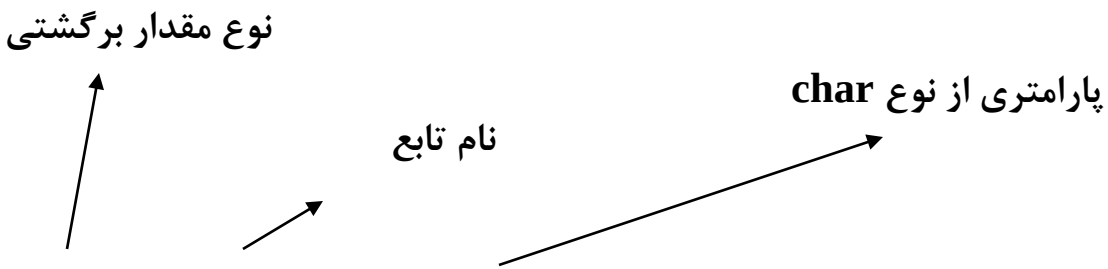
لیست پارامتر ها جهت انتقال اطلاعات از تابع احضار کننده به تابع فراخوانده شده

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

declaration and statements


}
```

نام تابع

تعریف اعلان‌های تابع و دستورالعملهای اجرائی

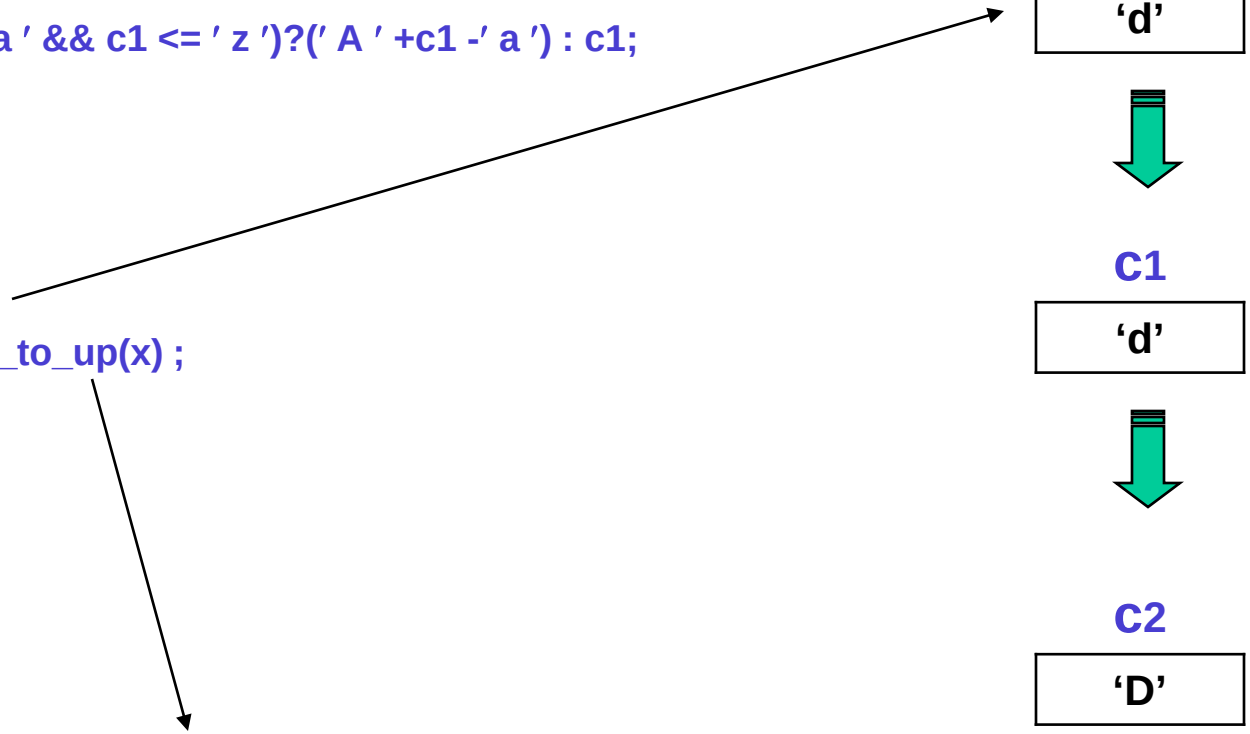
تابع زیر یک حرف کوچک را به بزرگ تبدیل می‌نماید.




```
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;
 }
```



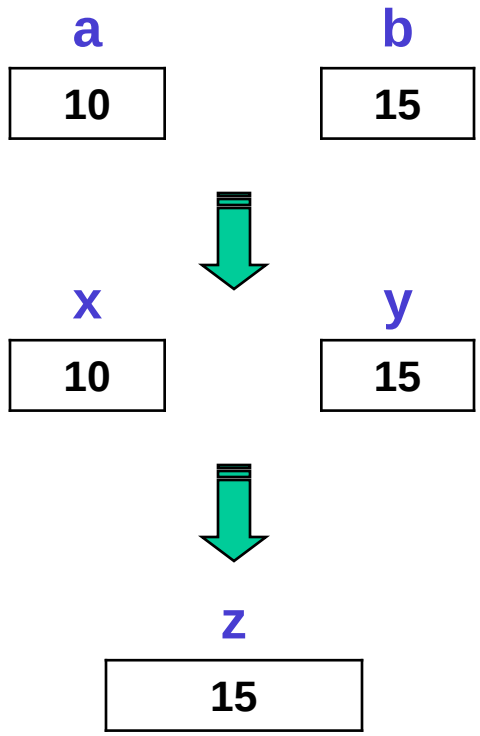
آرگومان

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;
}
```



a, b آرگومانهای تابع maximum

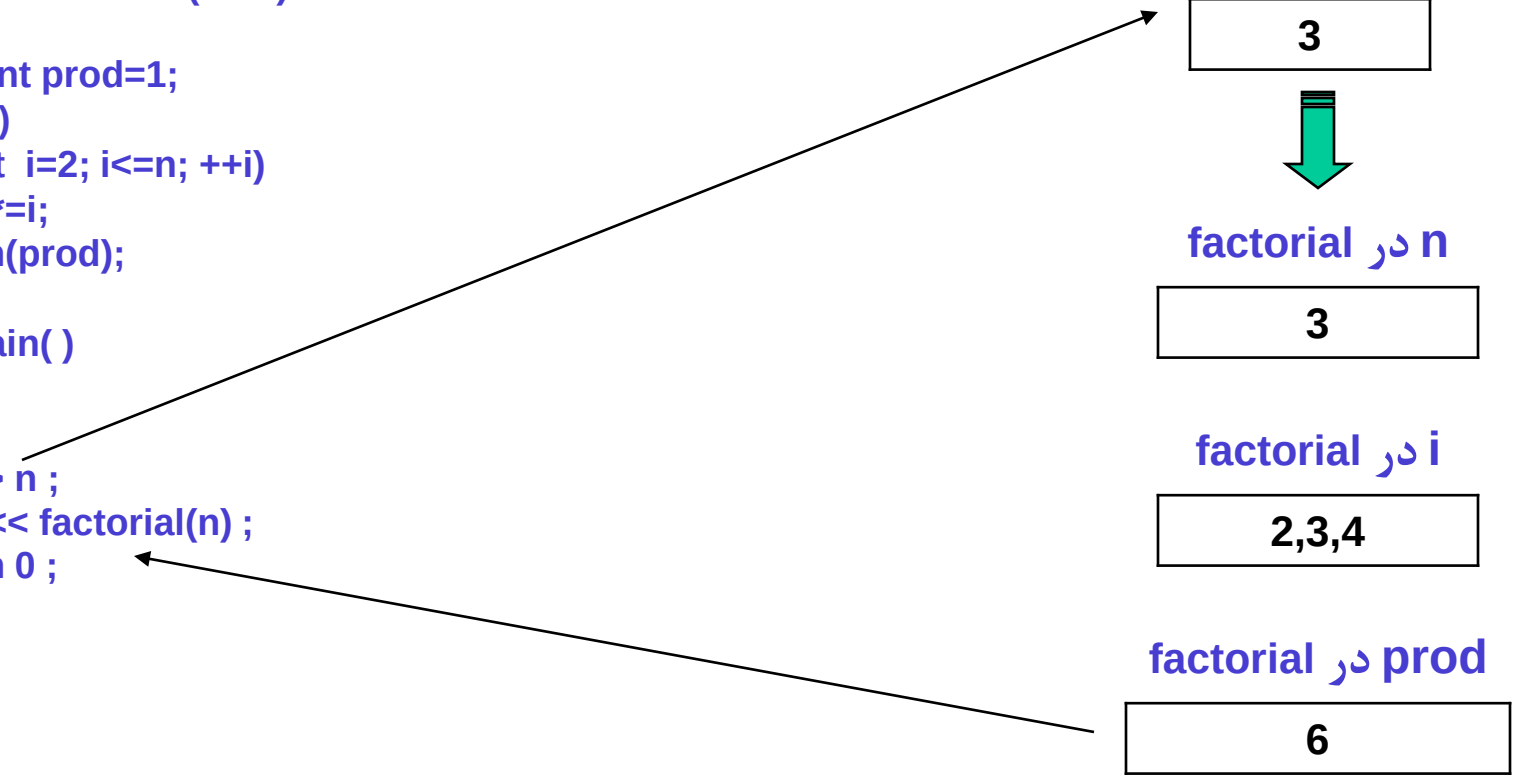
نکته ۱ :

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 ;
}
```



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.

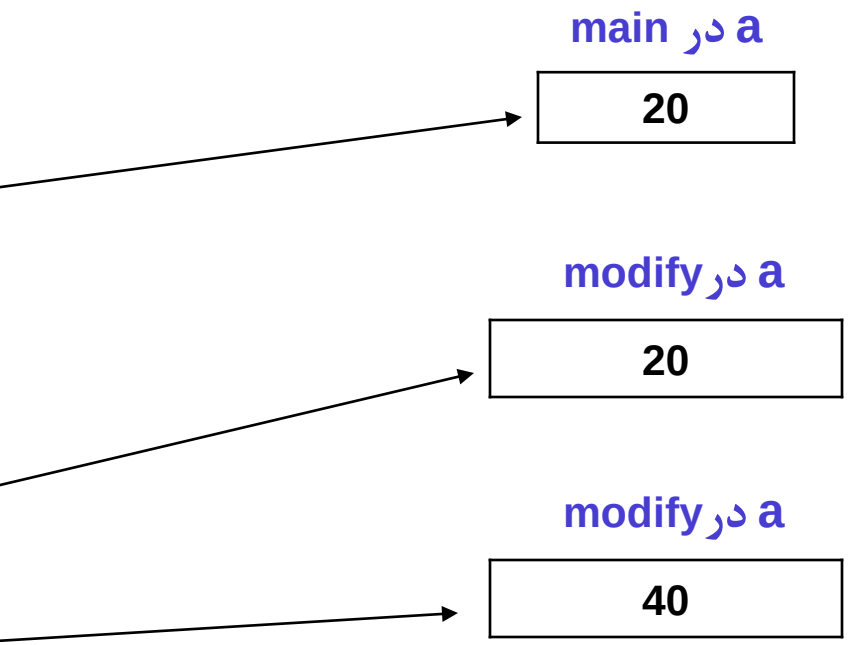
در اسلاید بعد مثالی در این زمینه آورده شده است.

```
#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 ;
}
```



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
```

فصل ششم

ساختار ها و اشاره گرها

فهرست مطالب فصل ششم

۱. ساختارها
۲. اشاره گرها (Pointer)
۳. تعریف آرایه
۴. آرایه های دو بعدی و اشاره گرها

ساختارها

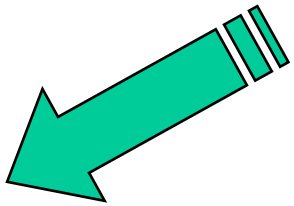
ساختارها شبیه آرایه‌ها بوده بدین صورت که یک نوع داده گروهی است که فضای پیوسته از حافظه اصلی را اشغال می‌نماید. اما عناصر ساختار الزاماً از یک نوع نمی‌باشند بلکه اعضای یک ساختار می‌توانند از نوع‌های مختلفه از قبیل **char** ، **int** ، **float** ، **...** باشند.

تعریف ساختار

نام ساختار

```
struct time
{
int hour ; // 0 – 23
int minute ; // 0 – 59
int second; //
} ;
```

اعضا ساختار



مثال :

```
struct account {
int acc_no ;
char acc_type;
char name[80] ;
float balance ;
};
```

ساختار account دارای چهار عضو می‌باشد.

acc_no	شماره حساب از نوع int
acc_type	نوع حساب از نوع char
name	مشخصات صاحب حساب از نوع رشتهٔ 80 کرکتری
balance	مانده حساب از نوع float

به دو صورت می توان اعلان یک متغیر از نوع ساختار را نمایش داد :

روش اول :

```
struct account {
int acc_no;
char acc_type;
char name[80];
float balance;
} cust1, cust2, cust3;
```

روش دوم :

```
struct account {
int acc_no ;
char acc_type;
char name[80];
float balance;
};
account cust1, cust2, cust3;
```

به ساختارها می‌توان مقدار اولیه نیز تخصیص داد

```
account cust = {4236, 'r', "Nader Naderi" , 7252.5};
```

دسترسی به عناصر یک ساختار

بمنظور دسترسی به عناصر یک ساختار از عملگر • استفاده می‌گردد . عملگر. جزء عملگرهای یکتائی می‌باشد.

مثال :

cust .acc_no = 4236

cust .acc_type = ′r′

cust . name = ″Nader Naderi″

cust . balance = 7252.5

نکته :

عضو یک ساختار خود می‌تواند یک ساختار دیگر باشد.

```
struct date {
int month;
int day;
int year;
};
struct account {
int acc_no ;
char acc_typer;
char name[80];
float balance ;
date lastpay ; };
```

اگرداشته باشیم

```
account x, y ;
```

آنگاه عضو lastpay بوسیله

```
x.lastpay.day
x.lastpay.month
x.lastpay.year
```

مشخص می‌گردد.

نکته :

می‌توان آرایه‌ای تعریف نمود که هر عضو آن یک ساختار باشد و حتی به آنها مقادیر اولیه تخصیص نمود.

```
struct struc1 {
char name[40];
int pay1;
int pay2; } ;
 strucl cust[  ]= {"nader", 3000 , 40000,
                   "sara", 4200, 6000,
                   "susan", 3700, 25000,
                    "saman", 4800 , 2000, };
```

برنامه زیر هر عدد مختلط را بصورت یک ساختار در نظر گرفته، دو عدد مختلط را می‌گیرد و مجموع آنها را مشخص و نمایش می‌دهد.

```
#include   <iostream.h>
int main( )
{
struct complex{
float  a;
float  b; } x, y, z;
cout << "enter 2 complex numbers" << endl ;
cin >> x.a>>x.b ;
cout << endl;
cin >> y.a >> y.b;
z.a = x.a + y.a ;
z.b = x.b + y.b ;
cout << endl << z.a << "   "<< z.b;
return 0 ;
}
```

بطوریکه $i2 = -1$

$x = a + ib$

$y = c + id$

$x+y = (a+c) + i(b+d)$

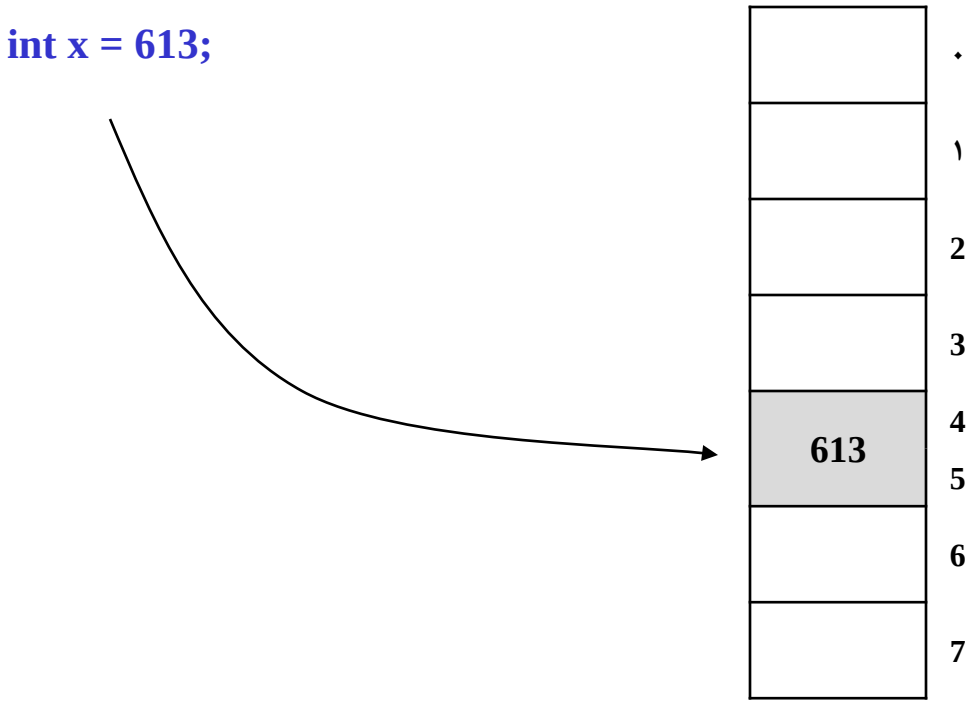
اشاره‌گرها (Pointers)

داده‌هائی که در کامپیوتر در حافظه اصلی ذخیره میشوند
بایت‌های متوالی از حافظه بسته به نوع data اشغال می‌کنند.

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

اشاره‌گرها (Pointers)

با داشتن آدرس داده در حافظهٔ اصلی می‌توان براحتی به آن داده دسترسی پیدا نمود و از طرف دیگر آدرس هر داده در حافظه آدرس بایت شروع آن داده می‌باشد.



نکته :

در کامپیوتر آدرس‌ها معمولاً دو بایت اشغال می‌نمایند. اگر آدرس x را در px قرار دهیم آنگاه می‌گوئیم که px به x اشاره می‌نماید.

آدرس متغیر x را بوسیله &x نشان میدهیم و عملگر & را عملگر آدرس می‌نامند.

```
int  x , *px
px = &x ;
```

مثال :

```
int y , x , *px ;
x = 26 ;
px = &x ;
```

حال اگر دستور العمل ; x += 10 را بدهیم :

حال اگر دستورالعمل ; *px = *px + 7 بدهیم.

آرایه یک بعدی و اشاره گرها

0	26.5	x
1	24.7	
2	5.8	
3	-73.2	
4	69.0	
5	100.5	
6	-13.24	
7	424.3	
8	187.8	
9	358.2	

اولین عنصر آرایه بوسیله x[0] مشخص می‌شود.

آدرس اولین عنصر آرایه بوسیله &x[0] یا بوسیله x مشخص می‌شود.

آدرس i امین عنصر آرایه بوسیله &x[i-1] یا بوسیله x(i-1) مشخص می‌شود.

دو دستورالعمل زیر با هم معادلند .

```
x[ i ] = 82.5 ;
*(x + i) = 82.5 ;
```

از طرف دیگر اگر داشته باشیم

```
float x[10];
float *p;
```

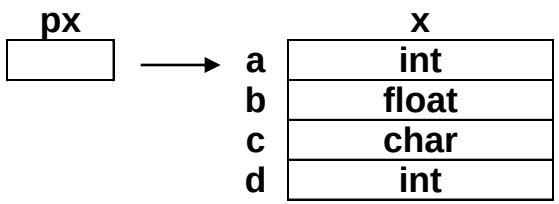
دو دستورالعمل زیر معادلند.

```
p = &x[2] ;
p= x + 2 ;
```

ساختارها و اشاره گرها

می‌توان اشاره‌گری را تعریف نمود که به اولین بایت یک ساختار (struct) اشاره نماید.

```
struct    struc1
{
        int a ;
        float b ;
        char  c;
        int d ;
} x, *px ;

px = &x ;
```



عبارت x.a معادل px–>a معادل (*px).a می‌باشد.

آرایه‌های دوبعدی و اشاره‌گرها

یک آرایه دوبعدی بصورت تعدادی آرایه یک بعدی می‌توان تعریف نمود.
اگر x یک ماتریس 5 سطری و 4 ستونی از نوع اعشاری باشد قبلاً این ماتریس را با

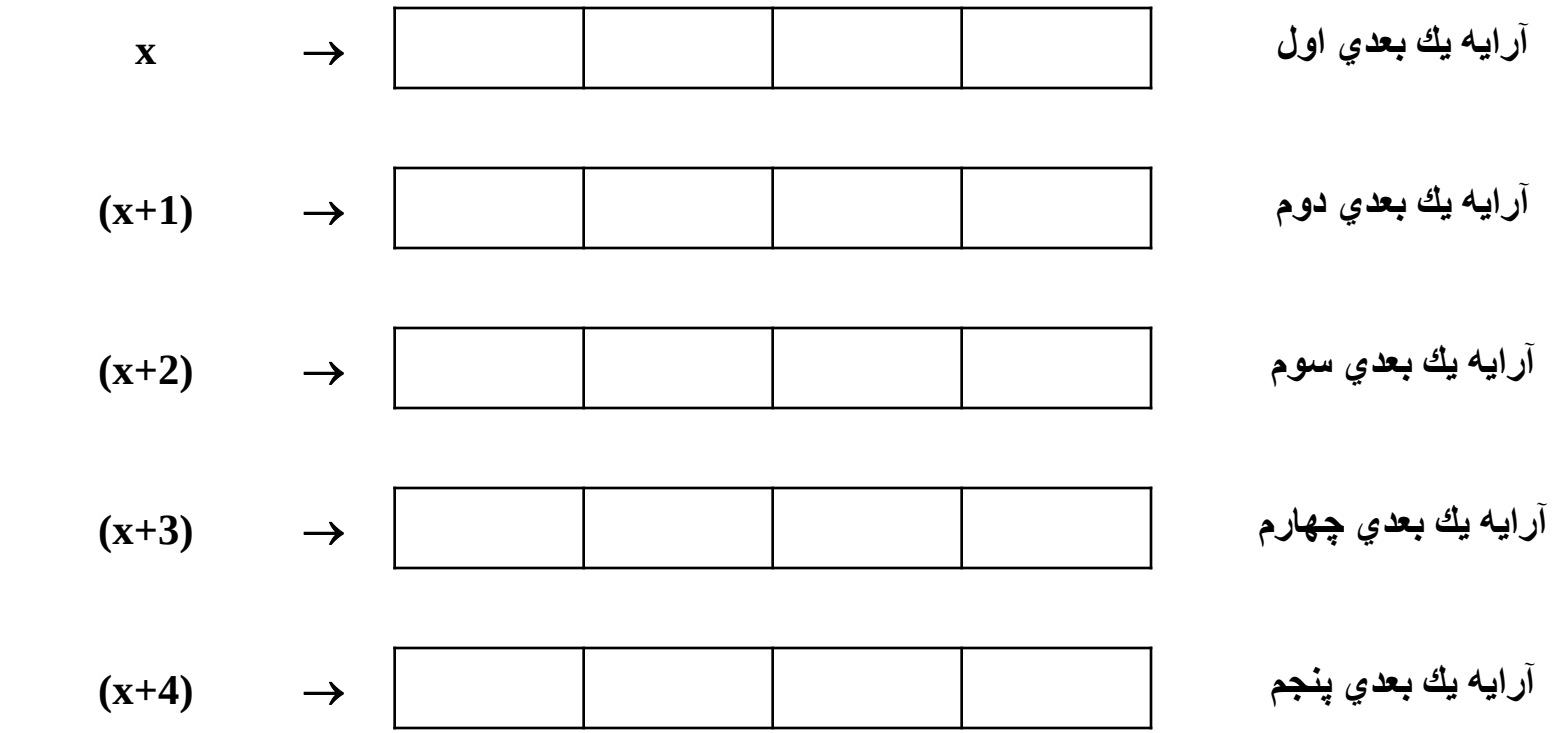
```
float x[5][4];
```

معرفی کردیم. حال با استفاده از اشاره‌گرها بصورت زیر معرفی نمائیم:

```
float  (*x)[4];
```

آرایه‌های دوبعدی و اشاره‌گرها

```
float  (*x)[4];
```



ایجاد شده و مقادیر عناصر آرایه را به چهار طریق نمایش int عنصری از نوع 5در برنامه زیر یک آرایه می‌دهد.

```
#include      <iostream.h>
#include      <conio.h>
int main( )
{
int x[ ]={12, 25, 6, 19, 100};
clrscr( );
int *px=x;
//  نام آرایه بدون اندیس، اشاره به عنصر اول آرایه می‌نماید
for(int i=0; i<=4; i++)
cout << *(x+i) << endl;
//the second method
for(i=0; i<5; i++)
cout << x[ i ] << ′ \n′;
//the third method
for(i=0; i<=4; i++)
cout << px[ i ]<<endl;
//the forth method
for(i=0; i<=4; i++)
cout << *(px+i)<<endl;
return 0; }
```

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