دانشگاه آزاد اسلامی

واحد دزفول

جزوهٔ درس برنامه سازی پیشرفته (زبان C & C++)

زبان **C** در سال ۱۹۷۲ توسط دنیس ریچی ، طراحی شد. زبان **C** تکامل یافته زبان **BCPL** که طراح آن مارتین ریچاردز است ، می باشد زبان **BCPL** نیز از زبان **B** مشتق شده است که طراح آن کن تامپسون می باشد.

زبانهای برنامه سازی به سه دسته عمده تقسیم می شوند :

۱- زبانهای سطح بالا مانند **Pascal** و **Basic** و **Cobol** و **Fortran** ...

۲- زبانهای میانه مانند **C , C++** و **FORTH**...

۳- زبانهای سطح پایین مانند اسمبلی و زبان ماشین و ...

سطح میانه بودن این زبان به این معنی است که این زبان امکانات و قدرت زبانهای سطح پایین را دارد و همچنین عناصر زبان های سطح بالا را نیز پشتیبانی می کند . زبان **C** دارای قابلیت های حمل یا **Portability** می باشد . یعنی برنامه نوشته شده با این زبان بر روی کامپیوترهای **Apple** و کامپوتر های سازگار با **IBM** بدون تغییر کد منبع قابل کمپایل می باشد . **C** برای نوشتن برنامه های سیستمی به کار می رود .

برنامه های سیستمی عبارتند از :

۱- سیستم عامل ۲- کامپایلر ۳- مفسر ٤- ویرایشگر ٥- برنامه های مدیریت بانک اطلاعاتی ٦- اسمبلر

-- در زبان **C** بین حروف بزرگ و کوچک اختلاف وجود دارد .

-- هر خط شامل ۲۵۵ کاراکتر می باشد .

-- انتهای هر خط **;** می گذاریم .

-- زبان **C** دارای ۳۲ کلمهٔ کلیدی می باشد .

-- حجم کم برنامه های اجرایی که به زبان **C** نوشته می شوند ، باعث افزایش سرعت اجرای آنها می شود .

انواع داده

در زبان **C** ، ٥ نوع داده اصلی وجود دارد :

نوع داده	صحیح	اعشاری کوتاه	اعشاری بلند	کاراکتر	پوچ
در زبان **C**	**int**	**float**	**double**	**char**	**void**
در زبان پاسکال	**integer**	**real**	**---**	**char**	**---**

نوع های داده اصلی (بجز **void**) می تواند با عباراتی مانند **signed** (علامت دار) ، **unsigned** (بدون علامت) ، **long** (بلند) ، **short** (کوتاه) ترکیب شده و نوع های دیگری را بوجود آورند.

در زبان **C** سه نوع ثابت وجود دارد :

۱–ثابت عددی ۲–ثابت کاراکتری ۳–ثابت رشته ای

ثابت عددی: ثابتهای عددی شامل اعداد صحیح و اعشاری می باشد. بطور کلی اعداد صحیح می توانند به سه روش در زبان برنامه نویسی نوشته شوند:

۱–اعداد ده دهی (**decimal**)

۲–اعداد در مبنای هشت : اگر قبل از عدد صفرقرار دهیم نشان دهنده عدد در مبنای هشت **(oct)** می باشد.

۳–اعداد در مبنای ۱۶ : اگر قبل از عدد مورد نظر عبارت **0x** را اضافه کنیم نشان دهنده عدد در مبنای ۱۶ می باشد (اصطلاحاً ان را **Hex** می نامند).

ثابت های کاراکتری:

در زبان **C** تمامی کاراکتر ها به عنوان ثابت کاراکتری در نظر گرفته می شود. علاوه بر آن نیز می شود کد یک کاراکتر را به عنوان یک ثابت کاراکتری در نظر گرفت.

'A' که معادل کد اسکی ۶۵ می باشد.

ثابت رشته ای:

در زبان **C** عبارتهایی که در بین گیومه قرار می گیرند(" ")رشته محسوب می شوند.

نوع داده	طول به بیت	رنج
unsigned char	۱ بایت با ۸ بیت	۲۵۵ تا ۰
signed char یا **char**	۱ بایت با ۸ بیت	۱۲۷ تا ۱۲۸–
unsigned int	۲ بایت یا ۱۶بیت	۶۵۵۳۵ تا ۰
signed int یا **int**	۲ بایت یا ۱۶بیت	۳۲۷۶۷ تا ۳۲۷۶۸–
short int	۲ بایت یا ۱۶بیت	۳۲۷۶۷ تا ۳۲۷۶۸–
unsigned long	٤ بایت یا ۳۲ بیت	٤۲۹٤۹٦۷۲۹۵ تا ۰
long	٤ بایت یا ۳۲ بیت	۲۱٤۷٤۸۳٦٤۷ تا ۲۱٤۷٤۸۳٦٤۸-
float	٤ بایت یا ۳۲ بیت	دقت ۷ رقم
double	۸ بایت یا ٦٤ بیت	دقت ۱۵ رقم
long double	۱۰ بایت یا ۸۰ بیت	دقت ۱۹ رقم

اگر قبل از نوع ، علامت دار یا بدون علامت بودن آن مشخص نگردد ، کامپایلر بطور پیش فرض نوع را علامت دار در نظر می گیرد.

متغیر چیست :

متغیر نامی یرای یک خانه از حافظه است که محتویاتش ممکن است تغییر کند . یک متغیر با حروف **A-Z** یا **a-z** یا **_** (آنـــدرلاین) شـــروع مـــی شـــود و شـــامل آنـــدرلاین ، حـــروف **A-Z** و **a-z** و ارقـــام **0-9** می باشد .

نحوه تعریف :

نوع متغیر **[** اسامی متغیرها **];**

```
int  i ;              char c,e,r ;
int  k, j ;           float  f ;
double  d, w ;
```

نحوه مقداردهی به متغیرها :

۱- در هنگام تعریف متغیر :

```
int a = 12, b =  15;          double h=1.25676;
char d='A';یا char d=65       char c='b',d='h';
```

۲- در خلال برنامه :

```
K = a+b;
K = 10;
```

در زبان **C** پس از تعریف متغیر یک مقدار تصادفی در آن قرار میگیرد. لذا در صورتی که کاربر متغیر را مقدار دهی اولیه نکند یک مقدار اشتباه در برنامه ظاهر می شود و ممکن است سبب تولید جواب نادرست میشود.

عملگرها به چند دسته تقسیم می شوند :

۱- عملگرهای محاسباتی + و – و / و * و ٪ (**Mod**)

یک واحد به متغیر اضافه می کند **: ++ (** پلاس پلاس **)**

یک واحد از متغیر کم می کند **: --** (ماینـاس ماینـاس **)**

```
int i=10;                     i++➔ i=i+1
i++;     /* عدد ۱۱ می شود */   i-- ➔ i=i-1
```

نکته !) اگر عملگر++ و یا -- در سمت راست عملوند قرار بگیرد ، مقدار فعلی عملوند مـورد اسـتفاده قـرار گرفتـه و سپس عملگرها بر روی عملوند عمل می کنند (افزایش و یا کاهش عملوند) ولی اگر در سمت چپ عملوند قرار گیرنـد ، ابتدا بر روی عملوند عمل می کنند (افزایش و یا کاهش عملوند) و سپس مقدار آنها مورد استفاده قرار می گیرد.

در مثال زیر ابتدا مقدار **i** به درون **C** ریخته شده و سپس به **i** یک واحد اضافه می شود.

```
i =  10;
C = i++;
C = 10;
```

مقدار فعلی مورد استفاده قرار می گیرد

i = 11; سپس یک واحد به آن اضافه می شود

در مثال زیر ابتدا مقدار **i** یک واحد اضافه می شود و سپس به درون **C** ریخته می شود.

C= ++ i;

ابتدا افزایش می یابد **i = 11**

سپس مورد استفاده قرار می گیرد **C = 11**

```
int  i=10 , j=7 , c=0;
c= i++ + ++j;
c= 18
j=  8
i= 11
```

اگر ++ در سمت چپ قرار بگیرد در همان لحظه به مقدار آن اضافه می شود ولی اگر در سمت راست باشد از مقادیر قبلی استفاده شده و در خط بعد یک واحد اضافه می شود .

```
int   a=10 ,  b=11 , c=12 ,i=11;
```

تمرین !)

c = ++c + ++a + b++ ; **c=13 b=12 a=11 C=35**

c = ++a + ++a; **a=12 C=24**

c = ++a + ++a + ++a + i++ + ++i; **a=12 i=12 C=48**

```
int c = 4, i = 2, e = 4;
++c = c++ + ++i + e++ + ++c; ?
```

۱- عملگرهای رابطه ای :

==	!=	>=	>	<	<=
مساوی یا برابر	**مخالف**	**بزرگتر مساوی**	**بزرگتر**	**کوچکتر**	**کوچکتر مساوی**

C=10یعنی مقدار ۱۰ رادر C قرار می دهد (انتساب)

C==10یعنی اگر مقدار C برابر ۱۰ باشدآنگاه ... (شرط)

۲- عملگرهای منطقی :

And ===➔ && و

OR ===➔ || یا

NOT ===➔ ! نقیض

Example:

if (a==10 || b<15) یا **(if a==10 && b<15)**

int a=0 if (!a)

در زبان **C** هر عددی غیر از صفر **True** حساب میشود. مثلا در مثال بالا مخالف مقدار **a** که صفر(**False**) است یک(**True**) قرار می گیرد.

۳- عملگرهای محاسباتی رابطه ای :

+= -= *= /= %=

a -= c; معادل با **a = a-c;**

a += 5; معادل با **a = a+5;**

a *= 6; معادل با **a = a*6;**

a /= b; معادل با **a = a/b;**

a %= k; معادل با **a = a%k;**

۳- عملگرهای بیتی جهت دستکاری بیتها :

&	بیتی **And**	**C = a & b;**
|	بیتی **Or**	**C = a | b;**
~	بیتی **Not**	**C = ~a;**
^	بیتی **Xor**	**C = a^b;**
<<	شیفت به چپ	**C = a << 2**
>>	شیفت به راست	**C= a >> 1**

نکته !) در هر بار شیفت به چپ عدد شیفت داده شده در ۲ ضرب می شود .

نکته !) در هر بار شیفت به راست عدد شیفت داده شده بر ۲ تقسیم می شود .

int a = 15 , b = 10 , c = 0;

c = a & b; حاصل بصورت زیر محاسبه می شود

	بایت پر ارزش								بایت کم ارزش							
بیت	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
a	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**
b	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**
c	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**

c = 10

c = a | b

	بایت پر ارزش								بایت کم ارزش							
بیت	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
a	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**
b	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**
c	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**

c = 15;

c = b << 2;

	بایت پر ارزش								بایت کم ارزش							
بیت	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
b	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
c	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0

c=2*2*10=40

c=a^b

	بایت پر ارزش								بایت کم ارزش							
بیت	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
a	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
b	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
c	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

برای محاسبه **a^b** هر گاه دو داده دارای یک ارزش بودند **False** (صفر) و اگردارای ارزش یکسانی نبودند **True** (یک) قرار می دهیم.

۵. عملگر **?** با تست یک شرط , عبارتی را به یک متغییر نسبت می دهد. نحوه استفاده **:**

۱) انتقال به یک متغیر

متغیر = **exp1? exp2 : exp3 ;**

۲) اجرای دستورات

exp1? مجموعه دستورات۱ **:** مجموعه دستورات ۲**;**

اگر حاصل شرط **exp1** درست **(TRUE)** باشد ، **exp2** یا مجموعه دستورات ۱ اتفاق می افتد در غیر اینصـورت **exp3** یا مجموعه دستورات ۲ رخ خواهد داد.

int a=5 , b =10 , c=0 ?

C = (a < b) ? 8 : 6; حالت انتقال به یک متغیر

خروجی **C=8**

(a < b) ? a++,b-- : a--, b++; حالت بدون انتقال

عملگر **?** معادل دستورات زیر است :

```
if exp1 = true  then
        متغیر = exp2;
else  exp1 = false
        متغیر = exp3;
```

٦. عملگر **sizeof** :

نکته !) این عملگر ، عملگر زمان ترجمه می باشد . (**Compile**)

نکته !) این عملگر طول یک نوع را به ما می دهد .

متغیر = **sizeof (data type);**

مورد اول

```
A = sizeof (long);
A = 4

A =sizeof (char);
A =1
```

نکته !) خروجی **sizeof** همیشه **int** است .

مورد دوم

```
متغیر = sizeof (متغیر)
 float f;

A = sizeof (f);
A = 4
```

این مقدار دهی به این دلیل می باشد که این زبان قابل حمل است. مثلا مقدار **k** در این مثال **k=sizeof (int)** تحت **dos** دو بایت است .ولی تحت ویندوز چهار بایت می شود. چون ویندوز یک سیستم عامل ۳۲ بیتی است و **dos** یک سیستم عامل ۱۶بیتی است.

تقدم عملگرها :

بالاترین تقدم
( )
sizeof (آدرس)& (محتوی)* -- ++ ~ !
% / (ضرب) *
+ -
<< >>
< <= > >=
!= ==
&
^
|
&&
||
?
= += -= *= /= %=

تبدیل انواع (**type casting**) : قاعده کلی این است که انواعی با طول کوچکتر به نوع هایی با طول بزرگتر تبدیل شوند. اگر نوع های با طول بزرگتر را به نوع های با طول کوچکتر تبدیل کنیم باعث از دست رفتن اطلاعات می شود.

```
charc = 'A' ;
int a;
a=(int) c;
a = 65;
```

حا فرض کنید که بخواهیم **int** را به **char** تبدیل کنیم : فرض کنیم **a = 1090** باشد و آن را بخواهیم به درون **c** بریـزیم چون **char** یک بایت و **int** دو بایت است باعث از دست رفتن بایت پر ارزش **a** می شود. (ناحیه هاشور خورده)

c = a;

	بایت پر ارزش								بایت کم ارزش							
بیت	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
a	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**
c	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**

c = 66 یا **c = 'B'**

تمرین:

مناسب ترین نوع داده را بیابید.

```
int a,b;
float f,c;
double e;
k=e+f*c+(a+b)          k=?
```

حل:

a+b=int f*c=flaot f*c+(a+b)=float+int=float e+f*c+(a+b)=double+float=double
k=double

شروع برنامه نویسی با زبان **C** :

تابع **main** نقطهٔ ورود به هر برنامه می باشد و اولین خطی است که در برنامه اجـرا خواهـد شـد . **main** تـابع اصـلی برنامه می باشد .

```
main
{

دستورات

}
```

تابع خروجی : **printf** (فرمت دار) :

printf (" عبارت رشته ای " , ...);

نکته !) عبارت رشته ای می تواند کاراکترهای فرمت ، کاراکترهای کنترلی و یا یک متن و یا ترکیبی از آنها می باشد.

نکته !) کاراکترهای فرمت با ٪ شروع می شوند و نـوع متغیـر را کـه در خروجـی نمـایش داده مـی شـود را مشـخص می کنند.

نکته !) کاراکترهای کنترلی با \ (بک اسلش) شروع می شوند و باعث انتقال مکان نما به یک سـطر و یـا سـتون خـاص می شوند.

مثال **printf(" this is the map ");** ← معمولی

دارای کاراکتر فُرمت ← **printf(" %d ", a);** **int a=10;**) مثال

جدول کاراکترهای فرمت

عملکرد	کاراکتر فرمت
برای چاپ اعداد صحیح مثبت و منفی	**%d** **%i**
برای چاپ عدد صحیح طولانی **(long)**	**%ld**
برای چاپ کاراکتر	**%c**
برای چاپ اعداد ممیز شناور (اعشاری)	**%f** **%g** **%G**
برای چاپ اعداد در مبنای اکتال **(8)Octal**	**%o**
برای چاپ اعداد در مبنای هگزا**(16)Hex**	**%x** **%X**
برای چاپ رشته	**%s**
برای نمایش اعداد در مبنای علمی	**%e**
برای نمایش اعداد بدون علامت	**%u**
برای چاپ اشاره گر	**%p**

جدول کاراکترهای کنترلی

عملکرد	کاراکتر کنترلی
برای انتقال مکان نما در خروجی به خط بعدی	**\n**
برای انتقال مکان نما به اندازه یک**Tab**	**\t**
Null انتهای رشته را مشخص می کند .	**\0**
برای چاپ کوتیشن	\"
برای چاپ تک کوتیشن	\'
برای چاپ بک اِسلَش	\\
انتقال مکان نما به ۸ خط بعد	**\v**
انتقال مکان نما به یک کاراکتر قبل	**\b**
انتقال مکان نما به صفحه بعد	**\f**

مثال)

```
int a=10;
float f=2.25;
char c='A' ;
```

```
printf(" %d  %f %c ",a ,f ,c);  ➔  10 2.25 A

printf(" a=%d\n    f=%f\n     C=%c " , a,f,c );
➔ a=10
   b=2.25
   c=A
printf("\" this is the Avrage of Student \" = %d",a);
 ➔ " this is the Avrage of Student " = 10
printf("a=%d \n c=%c",a,c);
➔  a=10
    c=A
```

برای رفتن به ۱۰خط بعد**printf("\v\n\n")**

در این زبان عباراتی که بین گیومه(" ") قرار می گیرند رشته محسوب می شوند.

در زبان **c++** برای چاپ یک متغیر یا یک عبارت از **cout** استفاده می کنیم. برای استفاده از این دستور از فایل سرباره **#include <iostream.h>** استفاده می کنیم.

نحوه استفاده ار **cout** :

cout << عبارت اول **<<** عبارت دوم **<<** عبارت سوم **<< ... <<** عبارت آخر**;**

هر یک از عبارت ها می تواند یا اسم یک متغیر و یا ترکیبی از کاراکترهای کنترلی و متن باشند. استفاده از کاراکترهای فرمت بی معنی می باشد.

```
cout<<"\"in the name of God\"";

➔    "in the name  of God"
```

در زبان**c++** به جای **"\n"**می توان **endl** را نیز بکار برد.

```
cout<<"a=" << a << endl <<"c=" << c;
    ➔a=10
      c=A
```

تابع **scanf** :

برای خواندن متغیرها از ورودی ، از این تابع استفاده می شود. در هنگام استفاده از **scanf** بجای نام متغیرها باید آدرس متغیرها بصورت زیر تعیین می شود.

scanf ("کاراکترهای فرمت"**,** آدرس متغیراول , آدرس متغیر دوم , **...);**

آدرس متغیر **a** ⟵ **&a** **int a;** اسم متغیر + عملگر **&**

فرمت	عملکرد
%c	برای خواندن یک کاراکتر
%d	برای خواندن یک عدد صحیح
%f	برای خواندن یک عدد اعشار
%s	برای خواندن خواندن یک رشته
%ld	خواندن یک متغیر**long**

scanf("کاراکترهای فرمت**"**, آدرس متغیرها **);**

مثال ۱) **int a;**

scanf(" %d" , &a);

مثال ۲) **char c;**

scanf(" %c " , &c);

مثال ۳) **int a,b;**

float f;
char c;
scanf("%d%d%f%c",&a,&b,&f,&c);

در**C++** می توان از کلاس **cin** به جای **scanf** استفاده کرد. که احتیاجی به آدرس دهی ندارد. **cin** درفایل سرباره **#include<iostream.h>** قرار دارد.

```
int a,b;      float f ;    char c;
cin >> a >> b >> f >> c;
```

نحوهٔ تعیین طول میدان :

طول میدان در اعداد اعشاری بوسیله **w.d** مشخص می شود ، که **w** طول میدان و **d** تعداد اعشار را مشخص می کند. در تعیین میدان به همه فرمتها اگر طول میدان از عدد کوچکتر باشد طول میدان در نظر گرفته نمی شود.

مثال) **float k=1.223;**

printf(" %8.2f " , k);

```
float f=22.2546738,
print f("%3.3f",f)
➔22.254
```

نکته !) چاپ اعداد از سمت راست صورت می گیرد .

اگر بخواهیم اعداد از سمت چپ چاپ شوند ، بعد از علامت ٪ و قبل از طول میدان یک علامت منفی (-) قرار می دهیم .

printf(" % -8.2f" , k);

اگر w.d در مورد اعداد صحیح به کار برده شود **w** حداقل طول میدان و **d** تعیین کننده حداکثر طول میدان می باشد .

```
int a=1000;
printf (" %8d " , a );        ➔   a =    1000
```

چهار تا فضای خالی نسبت به شروع خط داریم

printf (" a=% -8d " , a); ➔ **a=1000** از ابتدای خط شروع به چاپ می کند

تابع **scanf** و **printf** در فایل سربارهٔ **stdio.h** قرار دارند و برای استفاده از این توابع باید با دستور پیش پردازندهٔ **#include** در ابتدای سورس برنامه آنها را به برنامه ضمیمه کنیم. مثال : **#include <stdio.h>**

تابع **getch( )** :

برای خواندن یک کاراکتر از ورودی به کار می رود . کاراکتر خوانده شده بر روی صفحهٔ نمایش نشان داده نمی شـود . از این تابع معمولاً به عنوان آخرین خط برنامه استفاده می کنند تا برنامه منتظر ورود یک کلید باشد و بتوانیم خروجی را مشاهد کنیم.

مثال : **char c; c = getch();**

تابع **putch( )** : برای چاپ یک کاراکتر بر روی خروجی به کار برده می شود .

```
char c = 65;
putch (c); ➔A
putch('A'); ➔A
```

نکته : توابع **getch() , putch()** در فایل سربارهٔ **conio.h** قرار دارند.

نکته) برای نوشتن توضیحات **(comment)** درون یک برنامه از دو روش زیر استفاده می شود. توجه شود که توضیحات در هنگام کامپایل برنامه ترجمه نمی شوند و فقط جهت خوانایی برنامه بکار می روند.

۱) از // برای توضیحات کردن یک خط خاص توضیحات //

۲) از /* توضیحات */ برای توضیحات کردن چند خط

```
/* ----------------------------------
   ----------------------------------
   ---------------------------------- */
```

مثال ۱) برنامه ای بنویسید که یک عدد را از ورودی خوانده و آن را به توان ۲ برساند و در خروجی چاپ کند ؟

```
#include  <stdio.h>
#include  <conio.h>

void main  (void)
{   int a;
    scanf(" %d" , &a);
    printf(" power is = %d " , a*a);
    getch( );
}//end main
```

مثال) برنامه ای بنویسید که یک عدد صحیح و یک کارکتر از ورودی خوانده و اگر عدد با کارکتر برابر بود مقدار ۱۰ و اگر نبود مقدار ۲۰ را چاپ کند .

C++

```
#include <iostream.h>
#include <conio.h>
void   main   (void)
{
   int a,b;
   char c;
   cin>>a>>b;
   (a==b)? cout<<10 : cout<< 20;
 }
```

C

```
#include  <stdio.h>
#include  <conio.h>
void main (void)
{
   int a,b=0;
   char c;
   scanf("%d%c",&a,&c);
   b=(a==c)?10:20;
   print f ("%d ", b);
   getch ( );
}//end main
```

ساختارهای تکرار :

for : ساختار حلقه **for** به صورت زیر می باشد انواع حلقهٔ **for**

حلقه یک دستوری

for (مقداردهی های اولیه **;** شروط حلقه **;** افزایش و یا کاهش)

    ; دستور

حلقه چند دستوری

for (مقداردهی های اولیه **;** شروط حلقه **;** افزایش و یا کاهش)

{

    ; مجموعه دستورات

}

۱) **for (i=0; i<10; i++)**

۲) **for(j=10; j>-1; j--)**

۳) **for(i=0,j=10; i<10 && j>-1; i += 2, j=j*2)**

٤) **for (;;)** یک حلقهٔ بی نهایت

نکته :) برای شکستن یک ساختار تکرار (حلقه) از دستور **break;** استفاده می کنیم : مثال

for (;;)

if (شرط برقرار بود)

break;

مثال) برنامه ای بنویسید که میانگین عناصر یک جدول ضرب ۱۰*۱۰ را که هم بر۵ و هم بر ۷ بخش پذیر هستند را چاپ کند (نکته : برای محاسبه میانگین همواره به یک متغیر برای نگهداری حاصلجمع و یک متغیر برای تعداد نیاز داریم). تابع **clrscr()** برای پاک کردن صفحه نمایش به کار می رود و در سرباره **conio.h** قرار دارد .

```
#include <iostream.h>
#include <conio.h>
void main (void)
{
    int t = 0, sum = 0;
    clrscr();
    for (int i=1;  i<=10;  i++)
        for (int j=1; j<=10;  j++)
            if (((i*j % 5) == 0) && ((i*j % 7) == 0))
                {
                    sum += i*j;
                    t++;
                }
   cout << "Avg = " << sum/t << endl;
   getch( );
} // end main
```

مثال) برنامه ای بنویسید که ۱۰ عدد از ورودی خوانده و مغلوب آنها را به ترتیب چاپ کند ؟ مثال مغلوب عـدد ۲۵۸ عدد ۸۵۲ می باشد.

```
#include  <iostream.h>
#include  <conio.h>
void  main (void)
{
      int a, i;
      clrscr();
      for(i=1; i<=10; i++)
      {
          cin >> a;
          for (;a > 0;)
           {
                cout >> a % 10;
                a = a / 10;
           }
          cout << endl;
      }
     getch ( );
}  //end main
```

مثال) برنامه ای بنویسید که تا موقعیکه یک عدد متقارن را از ورودی نگیرد از ورودی عدد بگیرد و تعداد دفعات ورود عدد را بشمارد و آن را چاپ کرده و سپس خارج شود (عدد متقارن با مغلوبش برابر است)؟

```
#include  <iostream.h>
#include  <conio.h>
void  main (void)
{
       int t=0, a, k, n;
       for (;;)
       {
          cin >> a;
          n = a; // چون می خواهیم دستکاری کنیم پس یک کپی بر می داریم
          t++;  k=0;
          for (;a > 0;)
          {
              k = k*10 + a%10;
              a /= 10;
          }

          (n == k) ? cout << t, break : 0;
      }
      getch( )
}//end main
```

حلقه محاسبه مغلوب

مثال) برنامه ای بنویسید که خروجی زیر را چاپ کند ؟

```
    ١
   ١٢١
  ١٢٣٢١
 ١٢٣٤٣٢١
١٢٣٤٥٤٣٢١
 ١٢٣٤٣٢١
  ١٢٣٢١
   ١٢١
    ١
```

```
#include<stdio.h>
#include<conio.h>
viod  main (void)
{
   int i, j;
   clrscr( );
   for (i=1; i<=5; i++)
   {
      for (j=1; j<=5-i; j++)
          printf("  ");

      for (int j=1; j<=i ; j++)
          printf ("%2d",j);

      for(j=i-1;j>0;j--)
         printf("%2d",j);

      printf("\n\n");
   }

   for (i=4; i>0 ; i--)
   {
       for (j=1; j<=5-i; j++)
          printf("  ");

       for (j=1  j<=i ; j++)
          printf ("%2d" , j);

       for (j=i-1 ; j>0 ; j --)
          printf ("%2d" ; j);

      printf ("\n\n");
    }
    getch( );
}//end main
```

مثال) برنامه ای بنویسید که x , n را از ورودی خوانده و سری زیر را تا n جمله حساب کند ؟

$$sum = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} + ...$$

```
#include <iostream.h>
#include <conio.h>
void  main (void)
{
    float x, sum, sum1;
    int n, i, k;
    i=1; k=0;
    clrscr ( );
    cout << "please enter the x radian :” ;
    cin >> x;
    cout << “please enter the n :" ;
    cin >> n
    sum = sum1 = x ;
    for (i=1; i<n; i++)
    {
       k=(2*i)+1;
        sum1 = sum1* ( ( (x*x)* (-1) ) / (k*(k-1)) );
        sum += sum1;
     }
     cout << sum;
     getch( );
} // end main
```

ساختار تکرار while

while (شرط یا شروط)

; دستور

while (شرط یا شروط)

{

; مجموعه دستورات

}

برنامه ای بنویسید که تا موقعی که کلید f فشرده نشود به طور متوالی از ورودی کارکتر دریافت کند و آنها را بشمارد .

```
#include <iostream.h>
#include <conio.h>
void  main (void)
{
        char c;
        int i=0;
        while ((c=getch( )) ! ='f')
                i++;
        cout << i;
        getch( );
}//end main
```

مثال)برنامه ای که عددی را از ورودی خوانده و آن را به صورت باینری نمایش دهد.

```
#include <iostream.h>
#include <conio.h>
void   main   ( void)
{
    int a;
    int i=15 , z;
    clrscr();
    cin >> n;
    while   (i > -1);
    {
        z =1 << i;
       (i & z )?  cout <<"1" :  cout << "0";
       i--;
    }
    getch( );
}
```

ساختار تکرار do –while :

```
do
 {
; دستورات
 }    while (شرط یا شروط) ;
```

نکته : در حلقه **do–while** درآخر شروط حتما (;) قرار داده شود . این حلقه بر خلاف حلقه **while** حداقل یکبار اجرا شده و سپس شرط تست می شود.

برنامه ای بنویسید یک عدد را از ورودی خوانده و مجموع فاکتوریل ارقام آن را چاپ کند ؟

```
#include <iostream.h>
#include <conio.h>
void  main (void)
{
   int a, t, i, sum = 0;
   cin >> a;
   do
   {
       t =  a%10;
       f = 1;
       for (i=1; i<=t; i++)   } حلقه محاسبه فاکتوریل
           f *= i;
       sum += f;
       a = (int) a/10;
    } while(a>0);
    cout << sum;
    getch( );
}
```

ساختارهای تصمیم :

۱) ساختار تصمیم **if - else**

if (شرط یا شروط)

; دستور

else

; دستور

---

if (شرط یا شروط)

{

; دستورات

}

else

{

; دستورات

}

برنامه ای بنویسید که **a** را از ورودی خوانده اگر **a** بین ۱۰و۲۰ بود $10 \le a \le 20$ **a** به توان ۲ را چاپ کند و در غیر این صورت a^3 را چاپ کند ؟

```
#include <stdio.h>
void  main (void)
{        int a;
         scanf ("%d",&a);
         if (a>=10&& a=<20)
            printf("%d",a*a);
         else
            printf("%d",a*a*a);
}
```

(a>=10&& a<=20)? printf ("%d",a*a): printf("%d", a*a*a);

برنامه ای بنویسید سه عدد را از ورودی دریافت کرده و آن ها را به صورت مرتب ودر خروجی نمایش دهد.

```
#include <iostream.h>
#include <conio.h>
void   main   (void)
{
        int  a, b, c, temp;
        cin >> a >> b >> c;
        if  (a<b)
            { temp=a;   a=b;   b=temp; }
        if  (a<c)
            { temp=a;   a=c;   c=temp; }
        if (b<c)
           { temp=b;    b=c;   c=temp; }
       cout << a << b << c;
}
```

در زبان c اگر مقدار شرط درست باشد دستور بعد از if اجرا می شود و اگر درست نبود چیزی را چاپ نمی کند مثلا:

```
int a=-1
if (++a)
   cout << "ok";
```

چون صفر می شود و صفر یک مقدار غلط است پس چیزی را چاپ نمی کند.

۲) ساختار else if : برنامه ای بنویسید که به طور مکرر از ورودی کارکتر خوانده در صورتی که کارکتر وارد شده a باشد کارکتر **b** را چاپ کند اگر کارکتر **b** باشد **c** را چاپ کند و اگر **c** باشد **d** را چاپ کند و اگر **f** باشد از برنامه خارج شود .

```
#include <iostream.h>
void  main (void)
{
      char c;
      while (1)
      {
          c = getch ( );
          if (c == 'a')
             putch('b')
          else if (c == 'b')
             putch('b');
          elsee if (c == 'c')
             putch('d');
          else if (c == 'f')
             break;
     }
}//end main
```

۳) ساختار تصمیم **switch – case**

```
switch( متغیر)
{
case مقدار اول :
      دستورات;
break;
case مقدار دوم :
      دستورات ;
break;
⋮
case ....مقدار :
      دستورات ;
break;
default :
    دستورات ;    اگر هیچکدام از case ها اجرا نشود این دستورات اجرا می شوند :
break;
} //end switch
```

برنامه ای بنویسید که ۲ عدد **a,b** ویک عملگر محاسباتی / *- + را از ورودی خوانده و عمل مناسب را با توجه به عملگر وارد شده انجام دهد ؟

```
#include <iostream.h>
#include <conio.h>
void  main (void)
{
        int a,b;
        char c;
        cin >> a >> b >> c;
        switch (c)
        {
        case '+' :
                cout << "a+b =" << a+b;
                break;
        case '-' :
                cout << "a-b =" << a-b;
                break;
        case '*' :
                cout << "a*b =" << a*b;
                break;
        case '/' :
                cout << "a/b =" << a/b;
                break;
        }
        getch( );
}//end case
```

برنامه ای بنویسید که **n** را از ورودی خوانده و **n** جمله از سری فیبوناچی را چاپ کند ؟

```
#include <iostream.h>
#include <conio.h>
void  main (void)
{
        int a, b, c, i, n;
        a = b = 1;
        clrscr();
        cout << "enter a number :";
        cin >> n;
        cout << a <<"  " << b;
        for (i=3; i<=n; i++);
        {
                c=a+b;
                cout <<"  " << c;
                a=b;
                b=c;
        }
        getch( );
} // end main
```

توابع : در اکثر زبانهای برنامه نویسی برنامه ها به بخشهای مختلفی تقسیم می شوند که به این بخشها زیر برنامه گفته می شود.

زیر برنامه ها به ۲ دسته کلی تقسیم می شوند

۱- زیر برنامه زیرروال **procedure** : که دارای چندین خروجی هستند
۲- زیربرنامه تابع **function** : که حداکثر فقط یک خروجی بر می گردانند

در زبان **c** ما فقط زیر برنامه تابع داریم (توابع فقط دارای یک خروجی می باشند)

ساختار یک تابع به صورت زیر است :

< نوع خروجی تابع > < اسم تابع > (پارامترهای ورودی)

```
{
      بدنه تابع
}
```

نوع خروجی تابع می تواند شامل یکی از ۵ نوع اصلی زبان **c** باشد، مانند :

void - double – float – char - int

اسم تابع از قواعد نام گذاری متغیرها تبعیت می کند . اگر تعداد پارامترهای ورودی بیشتر از یکی باشد با عملگر ویرگول **(,)** از هم جدا می شوند .

*) اگر نوع تابع مشخص نگردد کامپایلر زبان **c** بصورت پیش فرض نوع صحیح (**int**) به آن تابع اختصاص می دهد.

انواع توابع :

۱- توابعی که هیچ مقداری را بر نمی گرداند. نوع خروجی آنها **void** می باشد.

مثال) برنامه ای بنویسید که یک عدد را از ورودی خوانده و آن را به یک تابع فاکتوریل ارسال کرده و فاکتوریل آن را آن تابع چاپ کند ؟

```
#include <iostream.h>
#include <conio.h>
void  main (void)
{
    void  fact (int k);
    int a;
    cin >> a;
    fact (a);
    getch( );
} //end main

void  fact (int k)
{   int i , f=1;
    for (i=1; i<=k; i++)
        f *= i;
    cout << f;
} // end fact
```

امضای یا الگوی تابع fact

اگر تابع تعریف شده در زیر تابع main نوشته شود باید از امضا یا الگوی تابع استفاده کنیم. امضای هر تابع به اولین خط تعریف هر تابع گفته می شود که در انتهای آن ; قرار گرفته باشد .اگر تابع تعریف شده در بالای تابع main قرار گیرد نیاز به تعریف امضای تابع نمی باشد . نحوهٔ تعریف امضای تابع :

< نوع خروجی> <نام تابع> (پارامترهای ورودی تابع) ;

نکته : درون یک تابع نمی توان تابع دیگر را تعریف کرد .

۲- توابعی که یک مقدار را بر می گردانند.

نکته : برای برگرداندن یک مقدار باید از دستور **return** استفاده کرد .

نکته : مقداری که برگرداننده می شود باید با نوع داده خروجی یکی باشد .

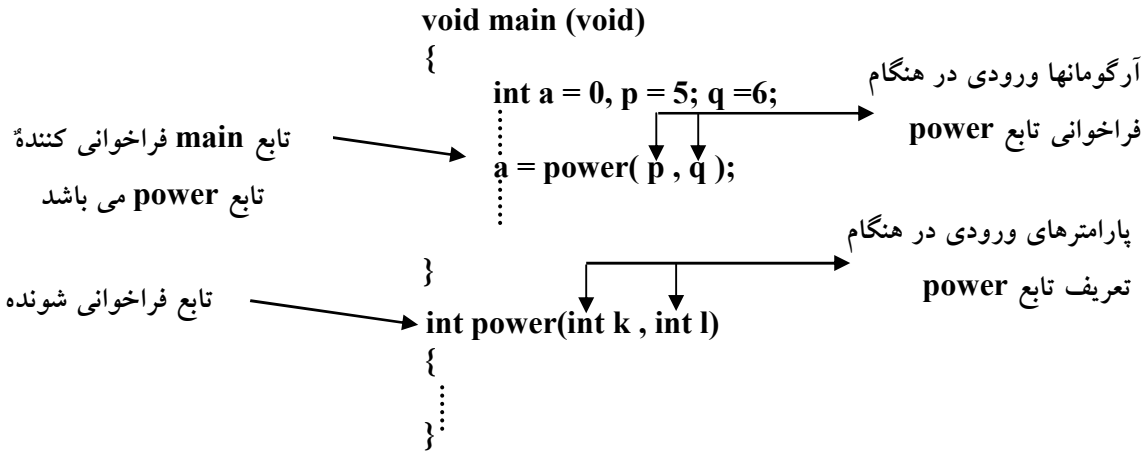
مثال) برنامه ای بنویسید که ۲ عدد **a,p** را از ورودی خوانده و **a** را به توان **p** برساند این برنامه از یک تابع استفاده کرده و **a,p** را به آن تابع ارسال کرده و بعد از محاسبه توان نتیجه را به تابع اصلی بر می گرداند. برای بازگرداندن نتیجه توان به تابع **main** باید از دستور **return** بصورت زیر استفاده کرد ؟

```
#include <iostream.h>
void  main (void)
{
    int power(int k , int l);
    int  a, p, z;
    cin >> a >> p;
    z = power(a, p);
    cout << "a ^p = " << z;
}
int power (int k , int l)
{
    int  i, n=1
    for (i=1; i<=l ; i++)
        n=n*k;
    return (n);
}
```

نحوه فراخوانی توابع :

۱- فراخوانی توسط ارزش **call by value** ۲- فراخوانی توسط ارجاع **call by reference**

در فراخوانی توسط ارزش مقدار آرگومان تابع در پارامتر متناظر آن کپی می شود ، لذا هر گونه تغییری در پارامترها هیچ گونه تاثیری در آرگومان ها نخواهد داشت . اما در فراخوانی توسط ارجاع ، آدرس متغیر به جای مقدار متغیر به درون یک تابع ارسال می شود و هر گونه تغییر در پارامترهای تابع فراخوانی شونده باعث تغییر آرگومان ها در تابع فراخوانی کننده خواهد شد . در مثال زیر فراخوانی توسط مقدار صورت گرفته شده است و هرگونه تغییر در پارامترهای **k , l** هیچگونه تأثیری بر مقادیر **p , q** نخواهد داشت.



در حل تمرینات فرض کنید که تابع **fact** و **power** از قبل موجود هستند.

مثال) برنامه ای بنویسید که x , n را از ورودی خوانده و سری زیر را تا n جمله حساب کند ؟

$$sum = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} + ...$$

```
#include <iostream.h>
#include <conio.h>
void  main (void)
{
   float x, sum, sum1;
   int n, i, k;
   clrscr ( );
   cout << "please enter the x radian :" ;
   cin >> x;
   cout << "please enter the n :" ;
   cin >> n
   sum = x;
   k = 3;
   for (i=1; i<n; i++)
   {
      sum1 = power(x, k) / fact (k);
      sum += sum1;
      k += 2;
   }
   cout << sum;
   getch();
}
```

۱) برنامه ای بنویسید که یک عدد را از ورودی خوانده و مجموع فاکتوریل ارقام را با استفاده از تابع فاکتوریل محاسبه کند ؟

```
#include <iostream.h>
#include <conio.h>
void  main (void)
{
        int  fact (int n);
        int  n , s=0;
        cin >> n;
        while (n>0)
        {
                s += fact(n%10);
                n = n/10;
        }
        cout << s;
        getch( );
}
```

۳- برنامه ای بنویسید که میانگین اعداد اول کوچکتر از ۱۰۰ را چاپ کند ؟

این برنامه شامل تابع **prime** که مشخص کنندهٔ اول بودن عدد است ، می باشد.

```
int  main (void)
{
     void  prime(int n);
     int  i, t =0, s = 0;
     clrscr( );
     for (i=2; i<100; i++)
          if (prime (i) == 0)
          {
             s += i;
             t++;
          }
     cout << s/t;
}
int  prime (int n)
{
         int  j , t=0;
         for (j=2; j<=(int)n/2; j++)
                  if (n%j==0)
                           t++;
        return t;
}// end prime
```

٤- برنامه ای بنویسید که n را از ورودی خوانده و زیرمجموعه های یک مجموعهٔ n عضوی را بصورت زیر چاپ کند.

n = 3 **{ } , {A} , {B} , {C} , {A,B} , {A,C} , {B,C} , {A,B,C}**

توجه کنید که یک مجموعهٔ **n** عضوی 2^n زیر مجموعه دارد. برای محاسبهٔ 2^n از عملگر شیفت به چپ استفاده می کنیم ، پس داریم 2^n = **1 << n**. در حل این مسئله از عملگرهای بیتی مانند **&** (**And** بیتی) نیز استفاده شده است.

در این روش حل 2^n از (**1 -** 2^n **تا 0**) فرض شده است. عدد 0 چون تمامی بیت های آن 0 می باشند ،پس بیانگر مجموعهٔ تهی می باشد. به جداول زیر توجه کنید. فرض کنیم **n = 2** باشد : پس 4 زیر مجموعه از **0** تا **3** می باشند.

نکتهٔ مهم : هر بیتی که 1 باشد ، حرف متناظرش را که در سطر بالای آن نوشته شده است را چاپ می کنیم.

	بایت پر ارزش								بایت کم ارزش							
بیت	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
حرف	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

حروف مرتبط با هربیت

زیرمجموعهٔ {}

زیر مجموعهٔ {A}

زیر مجموعهٔ {B}

زیر مجموعهٔ {A,B}

سؤال : الف) آیا نیاز است که هر ١٦ بیت **n** (از **0** تا **15**) تست شوند تا هر جا که **1** بود حرف متناظرش چاپ شود ؟

ب) چگونه تشخیص دهیم که آیا یک بیت **0** و یا **1** می باشد ؟

جواب الف) خیر. اگر مجموعهٔ n عضوی باشد ما باید n بیت اول را تست کنیم. ب)برای تشخیص اینکه آیا بیتی 0 است و یا 1 ؟ باید بصورت زیر عمل کنیم. فرض کنیم متغیری که به عنوان شمارنده از 0 تا **1 -** 2^n تعریف کرده ایم **i**

باشد. پس در هر بار چرخش حلقه باید n بیت از i تست شود که آیا 1 است و یا 0 ؟ برای تست کردن بصورت زیر عمل می کنیم :

فرض کنیم **i=1** باشد و **n=3** : اگر بخواهیم بیتی با ارزش مکانی 0 را تست کنیم باید با ۱ ، **AND** کنیم اگر حاصل بزرگتر از 0 شد پس بیت 1 است و باید حرف متناظرش چاپ شود. برای تست بیتی با ارزش مکانی 1 باید با 2 ، **AND** کنیم. بیتی با ارزش مکانی 2 باید با 4 ، **AND** شود. اگر دقت کنید هر چه ارزش مکانی بیشتر می شود باید با توان بیشتری از 2 ، **AND** شود . فرض کنیم اگر بخواهیم بیت j ام را تست کنیم باید با 2^j ، **AND** شود.

	بایت پر ارزش								بایت کم ارزش							
بیت	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
حرف	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
i = 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
&	&	&	&	&	&	&	&	&	&	&	&	&	&	&	&	&
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
نتیجه> 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

```
#include <iostram.h>
#include <conio.h>
void  main (void)
{
        int j, n, i, l, k, z;
        clrscr( ); cin >> n;
        k = 1<< n;  // 2^n برای محاسبهٔ
        for (i=0; i<=k-1; i++) // 2^n − 1 از ۰ تا
        {       cout << "{";
                l = 0; z = 1;
                for (j=0; j<n;  j++, z=z*2)   // i بار تست کردن  n   z = z*2  ===== z = z << 1
                    if ( (i&z) > 0 )     // تست بیت jام
                    {
                        l++;
                        if (l == 1)
                            printf("%c", 65+j); // اگر اولین عضو یک زیر مجموعه بود نیاز به ویرگول ندارد
                        else
                            printf(",%c", j);// اگر اولین عضو نبود باید قبل از چاپ عضو بعدی یک ویرگول قرار دهیم
                    }
                if (i < k-1)
                   cout <<"} , "; // بعد از اکولاد بستهٔ هر زیر مجموعه باید یک ویرگول قرار دهیم بجز زیر مجموعهٔ آخر
                else
                   cout << "}";   // اگر آخرین زیر مجموعه بود نیازی به چاپ ویرگول بعد از اکولاد نمی باشد
        }
        getch();
}
```

راه حل دوم مسئله (مختص به خردمندان)

```
void  main (void)
{
        int j, n, i, l, k;
        clrscr( );
        cin >> n;
        k=1<< n;
        for (i=0; i<=k-1; i++)
        {
              cout << "{";
               l = 0;
               for (j=0; j<n; j++)
                     if ((i&(1<<j)
                        (++l == 1) ? printf("%c", 65+j) : printf(",%c", j);
               (i < k-1) ? printf("} , ") : printf("}");
        }
        getch();
}
```

انواع متغیرها :

۱- متغیرهای محلی ۲- متغیرهای سراسری

متغیرهای محلی (Local) : متغیرهایی هستند که در درون یک بلوک تعریف می شوند و حوزه قلمرو و طول عمر استفاده از آنها از هنگام تعریف تا آخر بلوک که در آن تعریف شده اند ، می باشد .

متغیرهای سراسری (global) : متغیرهایی هستند که در خارج از بلوک و توابع تعریف می شوند و حوزه استفاده از آنها از هنگام تعریف تا انتهای برنامه می باشد . طول عمر این متغیرها **(lifetime)** تا انتهای اجرای این برنامه است.
محل تعریف متغیر سراسری قبل از تابع **main** است . معمولاً از متغیرهای سراسری زمانی استفاده می شود که توابع نیاز به یک متغیر مشترک داشته باشد .

نکته) امضای توابع نیز می تواند بصورت سراسری تعریف شود (مانند مثال زیر).

مثال) برنامه ای بنویسید که کاربرد متغیرهای سراسری را نشان دهد. این برنامه دو عدد **p,a** را از ورودی خوانده و به تابع **power_fact** ارسال می کند در این تابع **a** را به توان **p** رسانده و a! حساب می شود که سپس در تابع اصلی آنها را چاپ می کند. توجه کنید چون در زبان C توابع یک خروجی دارند و ما در این برنامه نیازمند دو خروجی هستیم. پس خروجی اول را با return بر می گردانیم و خروجی دوم را به درون یک متغیر سراسری می ریزیم.

```
#include <iostream.h>
#include <conio.h>
int power_fact (int a, int p);
int f = 1;

void main (void)
{
    int a , p, z;
    cin >> a >> p;
    z = power fact (a, p)
    cout << "a^p" << z << " a! =" << z <<  f;

} // end main

int power_fact (int a,int p)
{
    int k =1, i;
    for (i=1; i<=p; i++)
        k=k*a;
    for (i=1; i<=a; i++)
        f=f*i;
    return k;
}// end power_fact
```

امضای تابع **power_fact** و متغیر **f** سراسری هستند

مثال) برنامه ای بنویسید که **n** را از ورودی بخواند و اعداد بین ۱ تا ۱۰۰ را آنهایی که بر **n** بخش پذیر نیستند را چاپ کند ، مجوز استفاده از **& , % , if , for , ?** را نیز نداریم .

```
#include <iostream.h>
#include <conio.h>
void  main (void)
{
        int i=1, j=0, k=0, n;
        clrscr( );
        cout << "please enter the n = ";
        cin >> n;
        while( i <= (100/n)+1)
        {
                k++;
                j += n;
                while (k<j && k<=100)
                {
                    cout << "   " << k;
                    k++;
                }
                i++;
    }
        getch( );
}//end main
```

مثال) برنامه ای بنویسید که از ورودی یک مقدار پول دریافت کرده و آن را با سکه هـای ۲۰ ریـالی و ۳۰ ریـالی و ۵۰ ریالی خرد کند .

راه حل : اگر بخواهیم از هر سکه حداقل یکی وجود داشته باشد باید متغیرهای حلقه از ۱ شروع شوند.

```
#include <iostream.h>
#include <stdio.h>
#include <conio.h>
void  main (void)
{
        int  k, i, j, n, t=1;
        clrscr( );
        cout << "please enter the N = ";
        cin >> n;
        for (i=1; i<n/20 && t; i++)
            for (j=1; j<n/30&& t; j++)
                for(k=1; k<n/50 && t; k++)
                  if (i*20 + j*30 + k*50 == n)
                  {
                        printf("\n%3d*20+%3d*30+%3d*50==%d",i,j,k,n);
                        t=0;
                  }
        getch( );
}
```

مثال)برنامه ای بنویسید که a وb را از ورودی خوانده و بدون استفاده از عمل ضرب a را به توان b برساند.

```
void   main   (void)
{
    int  i , j, z=0;
    int  s=0, a, b;
    cin >> a >> b;
    for(i=1; i<=b; i++)
    {
       s=0;
       for (j=1; j<=z; j++)
          s += a;
       z=s;
    }
}
```

آرایه ها :

تعریف آرایه : آرایه اسمی برای چند متغیر هم نوع می باشد یا به عبارت دیگر آرایه از چندین کمیت درست شده است که همگی دارای یک نام می باشد . هر یک از این کمیت ها را یک عنصر می گویند برای دسترسی به عناصر آرایه باید اسم آرایه و شماره اندیس آرایه را ذکر کنیم. به آرایه یک متغیر اندیس دار نیز گفته می شود .

نحوه تعریف آرایه :

[تعداد عناصر آرایه] <اسم آرایه> < نوع آرایه>

اسم آرایه از قوانین نام گذاری متغیرها تبعیت می کند . تعداد عناصر آرایه (بعد آرایه) یا باید یک ثابت باشد و یا یک عدد صحیح مثبت بزرگتر از صفر.

نوع آرایه از انواع اصلی در زبان **c** می باشد **char , double , float , int**

int a [5]; char str[15];

طول بعد آرایه (تعداد عناصر) * (نوع آرایه) **sizeof** = مقدار حافظه مصرفی

= sizeof (int) * 5 = 10

در زبان **c** اندیس آرایه از صفر شروع می شود .

int a[5];

a[0]	a[1]	a[2]	a[3]	a[4]

رشته ها آرایه هایی از کارکترها می باشند **char str [20] ;**

برای تعریف یک ثابت , دو روش داریم :

۱- استفاده از **const** به صورت روبرو : مقدار= <نام متغیر> < نوع متغیر> **const**

const int n=15; **int a[n];**

۲- استفاده از دستور پیش پردازنده **#define** بصورت روبرو : مقدار <نام متغیر> **#define**

#define n 15 int a[n]

مثال) برنامه ای بنویسید که ۲۰ عدد را از ورودی خوانده و آنها را در درون یک آرایه ریخته و ماکزیمم و محل قرار گرفتن آنرا چاپ کند.

```
void  main (void)
{
        int a[20], i, max, t;
        for (i=0; i<20; i++)
           cin >> a[i];          ====➔   scanf("%d",&a[i]);
        max=a[0];
        t=0;
        for (i=1; i<20; i++)
           if (max<a[i])  {
              max = a[i];
              t = i;
            }
         cout << "max = " << max << "location = " << t;
        getch( );
}
```

مثال) برنامه ای بنویسید که ده عدد از ورودی دریافت کرده:

۱) میانگین ۲) معکوس آنها را (از انتها به ابتدا) ۳) به صورت صعودی نمایش دهد.

```
void main (void)
{
     int a[10], i, sum=0, temp;
     for (i=0; i<10; i++)
     {
         cin >> a[i];
         sum+=a[i];
     }
    cout<<"average ="<<sum/10;
    for(i=9; i>-1;i--)
      cout << a[i]:
    for (i=0;i<9;i++)
        for (j=i+1;j<10;j++)
            if (a[i]>a[j])
           {
              temp=a[i];
              a[i]=a[j];
              a[j]=temp;
           }
    for (i=0;i<10;i++)
        cout<< a[i];
   getch():
}
```

نکته : در زبان **c** اسم آرایه به آدرس اولیه عنصر آرایه اشاره می کند .

```
char a [4];
```

a[0] ➔ *(a+0) ➔ 'E'	**a[0] = 'E'**
a[1] ➔ *(a+1) ➔ 'T'	**a[1] = 'T'**
a[2] ➔ *(a+2) ➔ 'J'	**a[2] = 'J'**
a[3] ➔ *(a+3) ➔ 'L'	**a[3] = 'L'**

نحوه تعریف آرایه های **n** بعدی :

<نوع آرایه> <اسم آرایه> **[طول بعد اول] [طول بعد دوم] ... [طول بعد n ام]**

۱) تعریف یک آرایه ۲ بعدی از نوع صحیح : **int a[2] [5] ;**

سطر ستون

۲) یک ارایه ۳ بعدی از نوع صحیح : **int k[2][3][5]**

میزان حافظهٔ مصرفی یک آرایهٔ **n** بعدی :

میزان حافظه مصرفی = **sizeof** (نوع آرایه)*طول بعد اول*طول بعد دوم* ... *طول بعد **n** ام

(۱) = **sizeof (int)***۲*۵=۲۰

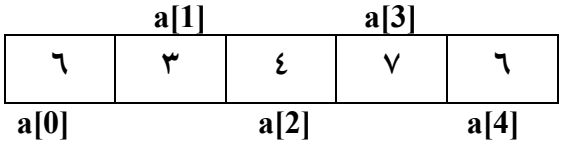
(۲) = sizeof (int)*۲*۳*۵=۶۰

روش های مقدار دهی اولیه به آرایه ها :

نکته مهم در مقداردهی اولیه به آرایه ها این است که فقط در زمان تعریف یک آرایه می توان آن را مقدار دهی اولیه کنیم.

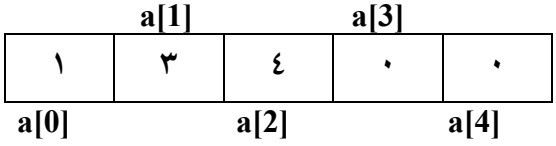
- **روش های مقدار دهی اولیه به آرایه یک بعدی :**

۱) روش اول **int a[5] = {6,3,4,7,6};**



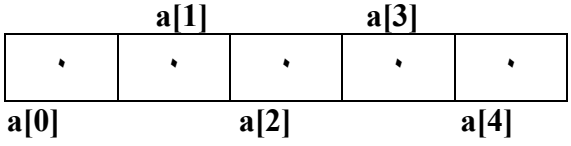
۲) **روش دوم** **int a[5] = {1,3,4};**

در این روش ۳ خانهٔ اول مقدار دهی شده و بقیهٔ خانه ها صفر می شود



۳) **روش سوم** **int a [5] = {0};**

در این روش تمام خانه ها صفر می شوند



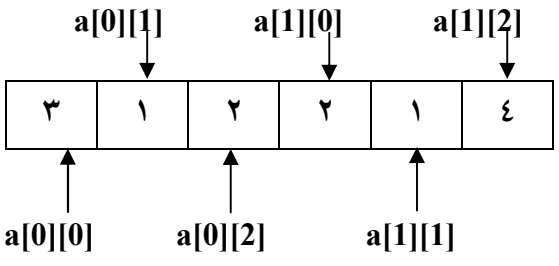
٤) **روش چهارم** **int a[] ={3,6,4,7,8}**

در این روش کامپایلر تعداد عناصر را شمارده و عدد مناسب را بجای بعد اول قرار می دهد

- **روش مقدار دهی اولیه به آرایه دو بعدی :**

۱) روش اول : **int a[2][3] = { {3,1,2} , {2,1,4} }**

سطر اول سطر دوم



۲) روش دوم : **int a[2][3] = {1,3}** باقی عناصر با صفر مقدار دهی می شوند

۱	۳	۰	۰	۰	۰

۳) روش سوم : ننوشتن بعد اول. در زبان **C** در هنگام مقدار دهی اولیه بـه آرایـه هـا مـا فقـط مـی تـوانیم بعـد اول را ننویسیم.

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

٤) روش چهارم : **int a[2][4] = {0};** در این روش تمامی عناصر آرایه صفر می شوند.

- **روش مقدار دهی به آرایه ۳ بعدی :**

int a [2][3][4] = { { {1,2,3,1} , {2,3,5,7} , {5,6,9,2} }, { {2,1,0,7} , {1,9,2,4} , {5,0,0,1} } }

آرایه هاو نحوه استفاده از آنها در فراخوانی توابع :

در زبان **c** ، بین اسم آرایه و اشاره گرها ارتباط تنگاتنگی وجود دارد . آرایه های بصورت روش فراخوانی توسط ارجاع به توابع ارسال می شوند. برای استفاده آرایه ها در فراخوانی توابع ، در توابع فراخوانی کننده فقط اسم آرایه به عنوان آرگومان نوشته می شود. در تعریف توابع پارامترهای ورودی از نوع آرایه می توان به یکی از ۳ فرم زیر باشند .

۱- به عنوان اشاره گر ۲- آرایه ای با طول ثابت ۳- آرایه های با طول نامشخص

```
void   main (void)
{
       int  a[10];
       sort( a ) ;
}
                 1) int *k
void  sort (     2) int  k[10]   )
{                3) int  k[ ]
  .....
}
```

اسم آرایه به عنوان آرگومان در هنگام ارسال به تابع **sort**

مثال) برنامه ای بنویسیدکه یک آرایه ۲۰ عنصری را از ورودی دریافت کرده آنرا به یک تابع مرتب سازی ارسال نموده و سپس بعد از مرتب سازی آن را در تابع اصلی چاپ نماید ؟

```
#include <iostream.h>
#include <conio.h>
const int n=20;
void  boublesort (int k [n]); // امضای تابع
void main (void)
{
        int a [n], i;
        clrscr();
        for (i=0; i<n; i++)
           cin >> a[i];
        bouble_sort (a);
        for (i=0; i<n; i++)
            cout << a[i];
}//end main
void bouble_sort (int k[n])
{
        int i, j, temp;
        for (i=1; i<n; i++)
           for (j=0; j <n-i; j++)
              if (k[j] > k[j+1])
              {
                 temp   = k[j];
                 k[j]    = k[j+1]
                 k[j+1] = temp;
              }
}//bouble sort .
```

برای حالت مرتب سازی نزولی کافیست تغییر زیر را اعمال کنیم:

```
if (k[j] <k[j+1])
```

مثال) برنامه بنویسید که دو عدد **a , b** را از ورودی خوانده بدون کمک متغیر اضافی جای **a , b** را عوض کند ؟

```
void main (void)
{
    int a,b;
    clrscr();
    cin >> a >> b;
    a=a+b;
    b=a-b;
    a=a-b;
    cout << a << b;
    getch( );
}//end main
```

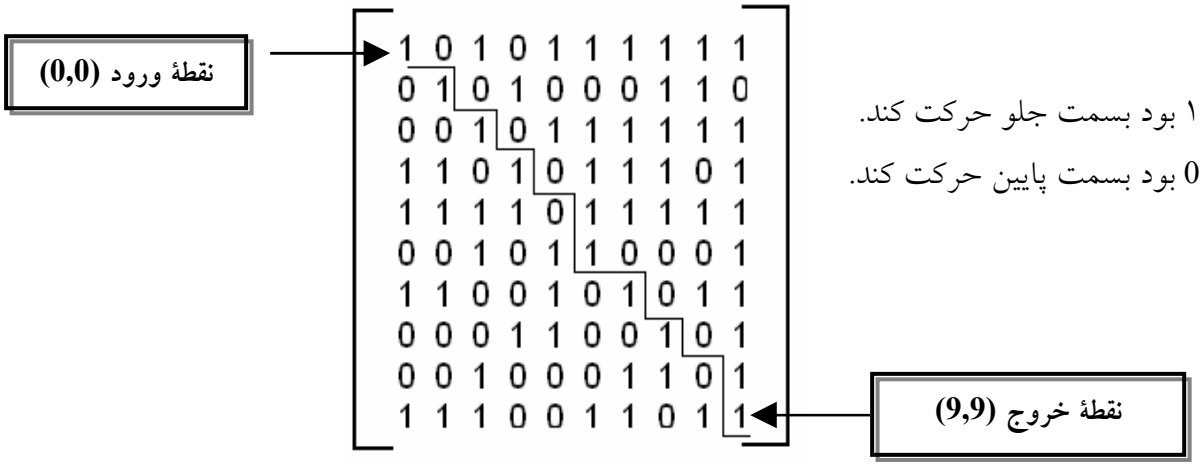
برنامه ای بنویسید که یک عدد را از ورودی خوانده و معادل باینری آن را چاپ کند ؟

```
#include <iostream.h>
#include <conio.h>
void main (void)
{
        int n, a[16], i=0;
        clrscr();
        cout << "please enter the n=";
        cin >> n;
        while (n>0)
        {
                a[i] = n%2;
                n /= 2;
                i++;
        }
        for (int j=i-1; j>=0; j--)
            cout << a[j];
        getch( );
}//end main
```

جواب سوّال قبل با استفاده از **for**

```
#include <iostream.h>
#include <conio.h>
void main (void)
{
        int n, a[16], i=0;
        clrscr( );
        cout << "please enter the n=";
        cin > n;
        for (i=0; n>0; i++, n/=2)
            a[i] = n%2;
        for(int j=i-1; j>=0; j--)
           cout << a[j];
        getch( );
}//end main
```

برنامه ای بنویسید که حرکت یک موش را در یک فضای ۲ بعدی طبق قوانین زیر شبیه سازی کند.



۱) هر کجا که ۱ بود بسمت جلو حرکت کند.

۲) هر کجا که 0 بود بسمت پایین حرکت کند.

```
#include <iostream.h>
#include <conio.h>
const  int n=10;
void main (void)
{
    int a[n][n], i, j, k=0, l=0, z=0;
    clrscr( );
    for(i=0; i<n; i++)
        for(j=0; j<n, j++)
            cin >> a[i][j];        // a[i][j] = rand()%2;      پرکردن ماتریس بصورت تصادفی
    cout << "\n\n\t\t";

    while (z <= (2*n) - 1)
    {
        if (a[l][k])
            k++;
        else
            l++;
        if ( (l>=n-1) || (k>=n-1))
            break;
        z++;
    }
    if (l==n-1 && k==n-1)
        cout << "path find";
    else
        cout << "path not find";
    getch( );
    }//end main
```

نکته : خروجی تابع **rand()** یک عدد صحیح مثبت تصادفی می باشد. برای اینکه خروجی این تابع کمتر از یک مقدار خاصی باشد مثلاً کمتر از ۲ باشد , باید از آن **mod** بگیریم : rand() % 2

رشته ها :

در زبانهای برنامه سازی مختلف رشته ها به عنوان نوع داده **(data type)** می باشد که برای نگهداری اسامی و متن ها بکار می روند . در زبان **c** رشته ها نوع داده نیستند بلکه آرایه ای از کارکترها می باشند که به **NULL** که دارای ارزش عددی صفر است ختم می شود . برای نمایش **NULL** از کارکتر **'\0'** استفاده می شود .

در زبان **c** باید طول رشته یک واحد بیشتر از طول واقعی آن باشد تا بتوانیم کارکتر NULL را در آخر آن قرار دهیم.

char s[10];

اگر محتویات رشته s را به روش هایی که بعداً گفته می شود برابر با **"ali"** قرار دهیم ، این رشته بصورت زیر نمایش داده می شود.

a	l	i	\0	?	?	?	?	?	?
s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]

مثال) برنامه ای بنویسید که یک رشته را از ورودی خوانده و آن را چاپ کند ؟

```
void main (void)
{
    char str [21];
    scanf("%s", str);        //  cin >> str;    C++ معادل در
    printf("%s",str);        //   cout << "s = " << str; C++ معادل در
    getch( );
} //end main
```

نکته : در این مثال چون اسم رشته اشاره گر به اولین عنصر رشته می باشد نیازی به قراردادن **&** (آدرس) در تابع **printf** نمی باشد .

مقدار دهی اولیه به رشته ها

هنگام تعریف رشته ها می توان به آنها مقدار اولیه داد. هنگام مقدار اولیه دادن می توان طول رشته را مشخص نکرد. در اینصورت ، اندازه رشته یک واحد بیش از تعداد کاراکترهایی است که به آن نسبت داده می شود. دو روش برای مقدار اولیه دادن به رشته ها وجود دارد. (۱) رشته در داخل کوتیشن قرار گرفته و به متغیر رشته ای نسبت داده شود (۲) هر یک از کاراکتر های رشته ای به عنوان یک عنصر رشته به آرایه نسبت داده شوند. در روش اول ، کاراکتر **'\0'** به طور خودکار در انتهای رشته قرار می گیرد و لی در روش دوم ، **'\0'** باید توسط برنامه نویس در انتهای رشته قرار داده شود.
مثال :

```
char s1[ ] = "ALLAH"; // روش اول بدون تعیین بعد
char s2[12] = "ALLAH";// روش اول با تعیین بعد
char s3[ ] = {'A', 'L', 'L', 'A', 'H', '\0'};// روش دوم بدون تعیین بعد
char s3[6] = {'A', 'L', 'L', 'A', 'H', '\0'};// روش دوم با تعیین بعد
```

توابع ورودی و خروجی رشته ها :

تابع gets در زبان C و cin در زبان C++ : برای خواندن یک رشته از ورودی به کار می رود .

```
gets (متغیر رشته ای) ;
char s[21] ;
gets (s);    // در زبان C
cin >> s; // در زبان C++
```

نکته : توابع رشته ای درون فایل سرآیند **#include <string.h>** قرار دارند .

سوال : چرا با وجود تابع **scanf** ما باید از تابع **gets** نیز استفاده کنیم !؟

جواب : تابع **scanf** رشته را پیوسته در نظر می گیرد . یعنی اگر در بین یک رشته از کارکتر **space** (فاصله) و یا کارکتر **Tab** استفاده شود از این کارکترها به بعد به عنوان رشته دیگری منظور می شود برای رفع این مشکل می توان از تابع **gets** استفاده کرد .

خواندن رشته با تابع **get** از کلاس **cin**

این تابع عضوی از کلاس **cin** است و یکی از کاربرد های آن برای خواندن رشته ها می باشد. از این تابع بصورت های زیر برای خواندن رشته ها بکار می رود.

1) cin.get(نام رشته , طول رشته);

2) cin.get(نام رشته , طول رشته , 'کاراکتر جدا کننده');

در کاربرد اول ، برای خواندن رشته از ورودی نام رشته و حداکثر طول رشته مشخص می گردد و **"\n"** یا همان کلید **enter** تعیین کننده انتهای رشته می باشد. در کاربرد دوم ، می توانیم با 'کاراکتر جدا کننده' کارارکتری را که پایان جمله را مشخص می کند ، تعیین کنیم. مثال :

```
char s[21];
cin.get(s, 15);
cin.get(s, 15, '.');
```

دستور اول یک رشتهٔ ۲۰ حرفی را تعریف می کند. دستور دوم یک رشته بطول حداکثر ۱۵ کاراکتر خوانده و در **s** قرار می دهد و با رسیدن به کلید **enter** انتهای رشته مشخص می گردد. دستور سوم رشته ای را به طول حداکثر ۱۵ کاراکتر از ورودی می خواند و یا پس از رسیدن به '.' ، خواندن رشته خاتمه می یابد.

تفاوت **cin.get** و **cin** :

در تابع **cin.get** کلید **enter** انتهای رشته را مشخص می کند ، مگر اینکه برنامه نویس کاراکتر دیگری را برای این منظور مشخص کند. در این تابع رشته می تواند حاوی فاصله **(space)** و یا **(Tab)** باشد. در حالی که در دستور **cin** فاصله و **Tab** نیز به عنوان جدا کننده تلقی شده و انتهای رشته را مشخص می کنند.

مثال) برنامه ای بنویسید که یک جمله بطول حداکثر ۵۰ کاراکتر را از ورودی خوانده و تعداد کلمات آنرا بشمارد. انتهای جمله با '.' مشخص می گردد؟

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

void main (void)
{
    char str[51];
    int   i=0, t=0;
    cin.get (str, 50, '.'); // gets(s);
    while (str[i] != '.')
    {
        if (str[i] ==' ')
            t++;
        i++;
    }
    printf("taded  kalamat=%d", ++t);
    getch( );
}// end main
```

تابع puts در زبان C یا کلاس cout در C++ : برای چاپ یک رشته در خروجی بکار می رود .

puts (متغیر رشته ای) **;**

char s[20] = “Ali”; puts(s);

puts("Ali"); یا

cout << “ali”; cout << s;

تابع strcmp : برای مقایسه دو رشته به کار می رود. ساختار تابع بصورت زیر می باشد.

strcmp (S1,S2) = خروجی یک عدد صحیح

خروجی	
خروجی>۰	**S1>S2**
خروجی=۰	**S1=S2**
خروجی<۰	**S1<S2**

آرایه ای از رشته ها

اگر بخواهیم متغیری را تعریف کنیم که چندین رشته را در خود نگه دارد باید آرایه ای از رشته ها تعریف کنیم. برای تعریف آرایه ای از رشته ها بصورت زیر عمل می کنیم.

char name[3][20]; // ۳ رشته ۲۰ حرفی

تعداد رشته — طول هر رشته

برای بازیابی هر یک از رشته ها باید از یک اندیس استفاده کنیم. **name[0]** اولین رشته ، **name[1]** دومین رشته ، **name[3]** سومین رشته می باشد. اگر از دو اندیس استفاده کنیم ، اندیس اول مشخص کننده رشته و اندیس دوم شمارهٔ کاراکتری از رشته را مشخص می کند. به عنوان مثال **name[2][5]** کاراکتر ۵ از رشته سوم را تعیین می کند..

مقدار دهی اولیه به رشته های دو بعدی :

char str[3][25] = { “Mohammad Golshahi”, “Mohammad Khirandiesh”, “Mohammad Mosleh”};

name [0]	**M**	**o**	**h**	**a**	**m**	**m**	**a**	**d**		**G**	**o**	**l**	**s**	**h**	**a**	**h**	**i**	**\0**	**?**	**?**	**?**	**?**	**?**	**?**	**?**
name [1]	**M**	**o**	**h**	**a**	**m**	**m**	**a**	**d**		**K**	**h**	**i**	**e**	**r**	**a**	**n**	**d**	**i**	**e**	**s**	**h**	**\0**	**?**	**?**	**?**
name [2]	**M**	**o**	**h**	**a**	**m**	**m**	**a**	**d**		**M**	**o**	**s**	**l**	**e**	**h**	**\0**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**

name[2][5] = ‘m’
name[2] = “Mohammad Mosleh”

مثال) برنامه ای بنویسید که یک رشته ۱۰ حرفی را از ورودی خوانده و با ۵ رشته داده شدهٔ اولیه مقایسه کند و تشخیص دهد که آیا رشته خوانده شده درون آرایه ای از رشته های اولیه موجود است یا نه؟

```
#include <conio.h>
#include <string.h>
void main (void)
{
    char s[5][10] = {"ali","javad","akbar","hasan","zinol"};
    char s1[10];
    gets[S1];
    for (int i=0; i<5; i++)
        if (strcmp (s[i], s1)==0)
           {
                puts("find string");
                break;
           }
    getch( );
}//end main
```

تابع strlen : طول یک رشته را بر می گرداند .

n = strlen(متغیر رشته ای)

s1 = "javad"
n = strlen(s1) // طول رشته می باشد n=5

تابع strchr : یک کارکتر را در درون یک رشته جستجو کرده و زیررشته ای را از محل اولین وقوع کاراکترمورد نظر تا اخر آن بر می گرداند . دارای ساختار زیر می باشد :

خروجی یک رشته = strchr (رشته , کاراکتر)

s = "in the name of GOD";
char c = 'm';
char *p = strchr(s,c); ➔➔ p = "me of GOD"

تابع strstr : زیر رشته **s2** را درون رشته **s1** جستجو کرده و زیر رشته ای را از محل اولین وقوع زیررشته مورد نظر بر می گرداند .

char *p = strstr (s1, s2);
char s1[ ] ="in the name of god";
char s2[ ]="he";
char *p = strstr(s1, s2);
p= “he name of god”;

تابع strupr : یک رشته را گرفته و تمام حروف کوچک آن را به حروف بزرگ تبدیل می کند .

chars1[6] = "Hasan";
strupr(s1);
در نتیجه s1 = “HASAN “

مثال) بدنه تابع **strupr** را بنویسید.

```
void mystrupr (char s[ ])
{
      int i=0;
      while (s[i])
            {
                if (s[i] >= 97 && s[i] <=123)
                   s[i]=s[i]-32;
                i++;
            }
}
```

تابع strlwr : یک رشته را به عنوان پارامتر ورودی گرفته و حروف بزرگ آن را تبدیل به حروف کوچک می کند .

```
char s1[ ] = "HASAN";
strlwr(s1);
s1 = "hasan";
```

مثال) تابعی بنویسید که عملکرد تابع **strlwr** را پیاده سازی کند.

```
void mystrlwr (char s[ ])
{
      int i=0;
      while (s[i])
            {
                if (s[i] >= 65 && s[i] <=90)
                   s[i]=s[i]+32;
                i++;
            }
}
```

مثال) برنامه ای بنویسید که یک فرمول ریاضی پرانتز گزارش شده را به عنوان یک رشته از ورودی گرفته و همه پرانتزهای باز و بسته را به ترتیب از آخر به اول چاپ کند؟

```
void main (void)
{
    char str[20];
    gets(str);
    int i = strlen (str);
    i--;
    while(i>=0)
    {
        if ( str[i] == ')' || str[i] =='(' )
            putch(str[i]);
        i--;
    }
    getch();
}//end main;
```

تابع strcpy : یک رشته را در رشته دیگر کپی می کند باید مواظب باشیم طول رشته دوم از رشته اول بزرگتر نباشد زیرا باعث سرریز پشته **(stack overflow)** می شود.

strcpy(s1, s2)

```
char s1[10]="ali"
char s2[10];
strcpy (s1,s2); ➔➔ s2="ali"  ➔ strcpy(s2,"ali");
```

نکته : ما نمی توانیم رشته ها را از طریق دستور انتساب '=' به درون یکدیگر کپی کنیم و حتماً باید از تابع **strcpy** استفاده کنیم. فقط در هنگام تعریف رشته (مقداردهی اولیه) است که می توانیم رشته ها را با = مقداردهی کنیم.

```
char s1[20] , s2[20];
  ⋮
s1 = s2; // نادرست. دستور انتساب در مورد رشته ها صادق نمی باشد
strcpy(s1, s2);// باید از این روش استفاده کنیم
```

تابع strncpy : به تعداد **n** کارکتر از رشته اول را به درون رشته دوم کپی می کند و ساختار آن به فرم زیر است .

```
strncpy (s1, s2, n);
```

مثال)برنامه ای بنویسید که ۵ نام را از ورودی بخواند وآن ها رامرتب کرده و چاپ نماید.

```
void main (void)
{
   char s[5][20];
   int i;
   for (i=0;i<5;i++)
      gets (s[i]);
   sort (s);
  for (i=0;i<5;i++)
     puts(s[i]);
}
void sort ( char k [5][20])
{
    int i,j;
    char temp [20];
    for (i=0;i<5;i++)
        for (j=0;j<5-i;j++)
            if (strcmp (k[j] ,k[j+1]) >=0)
            {
                strcpy (temp , k[j]);
                strcpy (k[j],k[j+1]);
                strcpy (k[j+1],temp);
            }
}//end sort
```

strcat : برای الحاق یک رشته به انتهای رشته دیگر به کار می رود؟

```
strcat(s1, s2);  // رشته دوم را به انتهای رشتهٔ اول اضافه می کند
```

- بدنهٔ تابع **strcat** :

```
void strcat(char s1[], char s2[])
{
    l = strlen(s1);
    for (int j=0; j<= strlen(s2); j++)
        s1[l+j] = s2[j];
}
```

- **بدنهٔ تابع strlen**

```
int strlen(char str[])
{
    int i=0;
    while (str[i] != NULL)
        i++;
    return i;
}
```

- **بدنهٔ تابه strcpy**

```
int strcpy (char s1[], char s2[])
{
    int i=0;
    while (s2[i] != NULL)
    {
        s1[i] = s2[i];
        i++;
    }
    s1[i] = NULL;
    return i;
}
```

- **بدنهٔ تابع strncpy**

```
void strncpy(char s1[], char s2[], int  n)
{
    int i;
    for (i=0; i<n; i++)
        s1[i]=s2[i];

    s1[i] = NULL;
}
```

- بدنهٔ تابع strchr

```
char* strchr(char str[] , char c)
{
    int i=0;
    while (str[i] != NULL)
    {
        if (str[i] == c)
            break;
        i++;
    }
        return (str+i);
}
```

- بدنهٔ تابع gets

```
void gets(char str[])
{
    int i=0;
    while(c != 13)      //  '\n'
    {
        str[i] = getche();
        i++;
    }
    str[i] = NULL;
}
```

- بدنهٔ تابع puts

```
void puts(char str[])
{    int i=0;
    while (str[i] != NULL)
    {
        putch(str[i]);
        i++
    }
    printf("\n");
}
```

- بدنه تابع strstr

```
char* strstr (char s1[ ], char s2[])
{
    int i=0, j=0;
    while ( i <= (strlen(s1)-strlen(s2)))
    {
        for(j=0; j<l ; j++)
            if (s1[i+j] !=s2[j])
                break;
        if (j==l)
            break
        i++;
    }
    return (s1+i);
}
```

توابع بازگشتی :

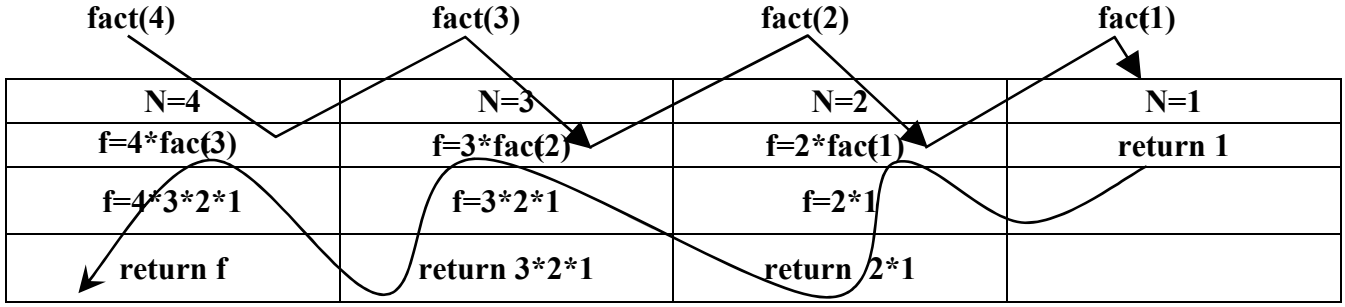
به توابعی گفته می شوند که در بدنه شان خودشان را فراخوانی کنند. اگر مسئله ای دارای ماهیت بازگشتی باشد برای حل آن مسئله از توابع بازگشتی استفاده می کند هر تابع بازگشتی دارای یک شرط پایانی یا یک شرط بازگشت می باشد.

n!=n*(n-1)!
(n-1)!=(n-1)*(n-2)!

اگر تابع بازگشتی شرط نداشته باشد باعث سرریز پشته می شود.

برنامه ای بنویسید که یک عدد را از ورودی خوانده فاکتوریل آن را بوسیله یک تابع بازگشتی محاسبه کند ؟

```
#include <iostream.h>
#include <conio.h>
int fact ( int n );
void main(void)
{
    int n;
    cin >> n;
    cout << fact(n);
    getch( );
}
int fact(int n)
{   int f ;
    if (n==0 || x==1)
       return 1;
    else
       f=n*fact(n-1);
    return f;
}
```



تابعی بنویسید که دو عدد را از ورودی خوانده و بصورت بازگشتی ترکیب آنها را محاسبه کند.

```
int  tarkib  (int n,p)
{
     if (n== p || p==0)
        return 1
     else
       return  tarkib (n-1 ,p) + tarkib (n-1 , p-1 )
}
```

برنامه ای بنویسید که یک عدد را از ورودی خوانده و به روش بازگشتی معادل باینری آن را چاپ کند ؟

```
void main (void)
{
    int n;
    cin >> n;
    binary(n)
    getch( );
}
void binary (int n)
{
    if(n>0)
    {
        binary(n/2)
        cout << n %2;
    }
}
```

binary(13)	binary(6)	binary(3)	binary(1)
n=13	n=6	n=3	n=1
binary(13/2)	binary (6/2)	binary(3/2)	binary (1%2)
cout << 13%2;	cout << 6%2;	cout << 3%2;	cout << 1%2;

برنامه ای بنویسید که ۲ ماتریس را از ورودی خوانده و آنها را به یک تابع ارسال کرده و در آن تابع درهم ضرب کند؟

```
const  int n=5 ,  m=4 , l=3;
void zarb_matrix (int a[n][m], intb[m][l], int c[n][l]);
void main (void)
{   int i, j;
    int a[n][m], b[m][l], c[n][l];
    for(i=0; i<n; i++)
        for(j=0; j<m; j++)
            cin >> a[i][j];
    for(i=0; i<m;i++)
        for(j=0; j<l; j++)
            cin >> b[i][j];
    zarb_matrix(a,b,c);
    for(i=0; i<n; i++)
    {
        for(j=0; j<l; j++)
            printf("%4d",c[l][j]);
        printf("\n");
    }
}
void zarb_matrix (int a[n][m], intb[m][l], int c[n][l])
{   int i, j, k;
    for (i=0; i<n; i++)
        for (j=0; j<m; j++)
            for (k=0; k<l; k++)
                c[i][k]=c[i][k]+a[i][k] * b[k][j];
}
```

اشاره گرها (pointer) :

اشاره گر چیست ؟ اشاره گرها به متغیری گفته می شود که آدرس یک متغیر یا یک تابع و یا مکانی از حافظه را در خود نگه دارد . اشاره گرها در زبان c دارای انواع مختلف می باشند مثل اشاره گر از نوع **int** برای نگه داری آدرس یک متغیر از نوع **int** به کار می رود و اشاره گر از نوع کارکتر آدرس یک متغیر از نوع کارکتر را در خود نگه داری می کند . اگر یک اشاره گر از نوع **int** را بخواهیم به آدرس یک متغیر از نوع کارکتر مقداردهی کنیم کامپایلر به برنامه نویس هیچ خطای را اعلام نمی کند ، ولی این امر یقیناً در نتیجه اجرای برنامه اثر مطلوب خواهد گذاشت .

چرا از اشاره گرها استفاده می کنیم ؟

۱- عمل تخصیص حافظه پویا را امکان پذیر می سازد .

۲- موجب بهبود کارایی بسیاری از توابع می شود .

۳- کار با رشته ها و آرایه ها را آسان تر می سازد .

٤- فراخوانی با ارجاع در توابع از طریق اشاره گرها امکان پذیر می شود .

نحوه تعریف اشاره گر :

اسم اشاره گر* نوع اشاره گر

توضیحات	تعریف اشاره گر
اشاره گری از نوع int	**int *ptr;**
اشاره گری از نوع char	**char *ch;**
اشاره گری از نوع float	**float *f;**
اشاره گری از نوع double	**double *d;**
اشاره گری از نوع void	**void *p;**

همانطور که گفته شد اشاره گر خود نیز یک متغیر می باشد ، پس مکانی را از حافظه اصلی اشغال می کند ؟ به نظر شما میزان حافظهٔ مصرفی اشاره گرهایی از انواع مختلف با هم برابرند ؟ بله چون اشاره گر آدرس را در خود ذخیره می کند و اندازهٔ آدرس ها در یک سیستم کامپیوتری با هم برابرند پس حافظه مصرفی اشاره گر ها نیز با هم برابر است. مثلاً میزان حافظه ای را که یک متغیر اشاره گر از نوع int مصرف می کند با میزان حافظه ای که یک متغیراشاره گر از نوع float و یا هر نوعی مصرف می کند با هم برابرند.

سوال : طول یک متغیر از نوع اشاره گر چند بایت است ؟ بسته به نوع پلتفرم سیستم عامل و سخت افزار طول یک اشاره گر در MS_DOS ,۲ بایت در ویندوز ۹۸ ، ۲۰۰۰ ، XP ، ٤ بایت و در ویندوز XP Longhorn 64 ، ۸ بایت می باشد

دو عملگر مورد استفاده در اشاره گرها بصورت زیر می باشند .

۱- عملگر آدرس & : یک عملگر یکانی است که آدرس عملوند خود را تعیین می کند .

۲- عملگر محتوا * : یک عملگر یکانی است که محتویات عملوند خود را تعیین می کند .

```
int *p, t;
t = 10;
p = &t;
1) printf("address t = %p \n", p);
2) printf("\nt=%d",*p);
```

```
address t = 500
t = 10
```

خروجی

نام متغیر	محتوی	آدرس
؟	؟	٤٩٨
؟	؟	٤٩٩
t	۱۰	۵۰۰
		۵۰۱
⋮	⋮	⋮
p	۵۰۰	٦۰۰
		٦۰۱
⋮	⋮	⋮

اعمال بر روی اشاره گرها

a. عمل انتساب

```
int x = 1024,*p1,*p2;
p1=&x;   آدرس متغیر x را به درون اشاره گر می ریزد
p2=p1;   p1 و p2 هر دو به متغیر x اشاره می کنند
printf("x=%d", *p2);
```

```
x = 1024
```

خروجی

نام متغیر	محتوی	آدرس
x	۱۰۲٤	۵۰۰
		۵۰۱
⋮	⋮	⋮
p1	۵۰۰	٦۰۰
		٦۰۱
p2	۵۰۰	٦۰۲
		٦۰۳
⋮	⋮	⋮

۲-عمل محاسباتی : جمع - تفریق - مقایسه

```
int x=20
int y=50
int *p;
p=&x;
```

نام متغیر	محتوی	آدرس
x	۲۰	۵۰۰
		۵۰۱
y	۵۰	۵۰۲
		۵۰۳
⋮	⋮	⋮
p	۵۰۰	۶۰۰
		۶۰۱
⋮	⋮	⋮

چون اشاره گر از نوع **int** می باشد و آدرس یک متغیر **int** را نگه می دارد و هر متغیر **int** ⟵ **p++; // p += 1;**
دو بایت از حافظه را اشغال می کند پس به جای یک واحد دو واحد به **p** اضافه می شود

1) printf("%d", *p);

۱) محتویات آدرسی که **p** به آن اشاره می کند چاپ می شوند یعنی ۲۰ چاپ می شود.

2) printf("%d", *++p);

۲) اول ++ می شود یعنی ۲ واحد اضافه می شود بعد محتویاتش چاپ می شود و ۵۰ چاپ می شود

3) printf("%d", *p++);

۳) اول محتویاتش چاپ می شود و بعد ۲واحد اضافه می شود و اول ۲۰چاپ می شود و بعد به ۵۰۲ اشاره می کند.

4)printf("%d",(*p)++);

٤) اول ۲۰ را چاپ می کند سپس محتویات خانه ۵۰۰ را یک واحد اضافه می کند یعنی ۲۱ می شود.

۱- انتساب

```
char   x='h' ,  *p1, *p2;
  *p1 = &x;    p2 = p1;
```

۲- اعمال محاسباتی

جمع **p1++ یا p1 += 1 یا p1 = p1 + 1**

تفریق **p1-- یا p1 -= 1 یا p1 = p1 – 1**

مقایسه **p1 == p2، p1 > p2 و . . .**

اشاره گرها و آرگومان توابع :

در روش فراخوانی توسط ارجاع باید در آرگومان تابع به جای اسامی متغیرها آدرس آنها را قرار داد و چون آدرس متغیرها در اشاره گرها ذخیره می شوند پس پارامترهای تابع باید از نوع اشاره گر باشند .

مثال) برنامه ای بنویسید که ۲ عدد **a,b** را از ورودی خوانده به روش فراخوانی توسط ارجاع آنها را به یک تابع ارسال کرده و در آن تابع جای **a,b** را باهم تعویض کند ؟

```
#include <stdio.h>
#include <conio.h>
void swap (int *p, int*2)
void main (void)
{
    int a,b
    cin >> a > b'
    swap(&a,&b);
    cout < "a = " << a << " b = " << b;
    getch( );
}
void swap(int *p, int*q)
{
    int temp;
    temp=*p;        *p=*p + *q;
    *p=*q;          *q=*p - *q;
    *q=temp         *p=*p - *q;
}
```

بدون متغیر کمکی

اشاره گر از نوع void :

اشاره گری از نوع **void** می تواند آدرس متغیری را از هر نوع دیگر در خود جای دهد ولی برای اعمال محاسباتی مانند جمع و تفریق باید نوع اشاره گر قبل از اسم متغیر از نوع **void** ذکر شود .

```
int x=10
int *p;
void *q;
q=&x;
(*(int*)q)++;
p=(int*)q;
```

تخصیص حافظهٔ پویا

گاهی لازم است که برنامه نویس در زمان اجرای یک برنامه از سیستم عامل درخواست حافظه نماید، مثلا برنامه رکوردهای مختلفی از دانشجویان را از ورودی می خواند اگر برنامه نویس در برنامه خود از یک آرایه ثابت استفاده کند، ممکن است طول آرایه کم باشد و کاربر بخواهد رکوردهای بیشتری را ثبت کند و یا برعکس ممکن است طول آرایه خیلی بیشتر از تعداد رکوردهای ثبت شده باشد که در این حالت اتلاف حافظه داریم .

در زبان **C** برای تخصیص حافظـه پویـا (یعنـی در زمـان اجـرا) از تـابع **malloc** کـه در **#include <stdlib.h>** یـا **#include <alloc.h>** قرار دارد ، استفاده می کنیم.

باید توجه داشته باشید که بعد از خروج از یک تابع باید حافظه که با استفاده از تابع **malloc** تخصیص داده شده است را با استفاده از تابع **free** به سیستم بازگردانده شود. متغیر هایی که ما در یک تابع تعریف می کنیم درهنگام خروج از تابع به سیستم بازگردانده می شوند. ولی حافظه ای را که با **malloc** تخصیص می دهیم در انتهـای اجـرای تـابع آزاد نمی شود و باید با تابع **free** آن را آزاد نمود. اگـر حافظـه را آزاد نسـازیم باعـث نشـتی حافظـه یـا ***Memory Leak*** می شود.

الگوی تابع **malloc** بصورت روبرو می باشد : **void* malloc (اندازه به بایت)**

اگر به الگوی تابع **malloc** نگاه کنید متوجه خواهید شد که خروجی آن ***void** می باشد. در هنگـام اسـتفاده از تـابع **malloc** باید نوع مقصد قبل از اسم تابع اعلان شود ، تا کامپایلر خطای را گزارش ندهد. مثلاً فرض کنید می خـواهیم که یک آرایهٔ ۱۰ عنصری از نوع **int** را در زمان اجرا ایجاد کنیم :

```
int *p = (int*) malloc ( 10*sizeof (int) )
```

اعلان نوع قبل از **malloc**

میزان حافظهٔ مصرفی یک آرایه به طول ۱۰ از نوع **int**

سئوال : چرا با وجود اینکه ما می دانیم یک آرایه ۱۰ عنصری از نوع **int** بـه **10*2 = 20** بایـت نیـاز دارد ، از فرمـت **10*sizeof(int)** استفاده می کنیم؟

جواب : برای حفظ خاصیت قابل حمل بودن برنامه. چون ممکن است برنامه در یک پلتفرم ۳۲ بیتی اجرا شود کـه در آن هر نوع **int** ، ٤ بایت از حافظه را مصرف کند ، آنگاه ما به **10*4 = 40** بایت نیاز داریم و مشخص است که برنامـه نتیجهٔ مطلوبی نخواهد داد.

دستوری بنویسسد که یک آرایهٔ ۲۰ عنصری از نوع کاراکتر را در زمان اجرا ایجاد کند ؟

```
char *str = (char*) malloc (20*sizeof(char));
```

برای آزاد سازی حافظهٔ تخصیص یافته از تابع **free** بفرمت زیر استفاده می کنیم.

free (اشاره گر)

```
int *p = (int*) malloc ( 10*sizeof (int) )
char *str = (char*) malloc (20*sizeof(char));
  ⋮
free (p);
free (str);
```

برنامه ای بنویسید که **n** را از ورودی خوانده یک آرایه **n** عنصری ایجاد کرده آرایه **n** عنصری را از ورودی خوانـده و ماکزیمم آن را حساب کند ؟ چون **n** از ورودی خوانده می شود پس دارای مقدار ثابتی نیست که ما بتوانیم یک آرایـهٔ ثابت تعریف کنیم پس نیاز به تخصیص زمان اجرا داریم ، چون مقدار **n** در زمان اجرا خوانده می شود.

```
void main (void)
{
    int n, i, max;
    int *a;
    cin >> n;
    a = (int*) malloc( n*sizeof(int) );  // تخصیص پویای یک آرایه
    for(i=0; i<n, i++)
        cin >> a[i];
    max=a[0];
    for(i=1; i<n, i++)
        if (max<a[i])
            max=a[i];
    free(a); // آزاد سازی حافظه پس از اتمام کار با آرایه
    cout << "max = " << max;
} //end main
```

در زبان C++ برای تخصیص و برگرداندن حافظهٔ پویا به سیستم از عملگـر **delete** , **new** بـه دوفرمـت زیـر اسـتفاده می کنیم :

- تخصیص به اندازه نوع

اشاره گر = **new** <نوع> ;

delete اشاره گر ;

مثال :

```
int *p = new int;      // تخصیص به اندازه نوع
delete  p;             // برگرداندن به سیستم
```

- تخصیص به اندازهٔ آرایه ای از یک نوع

```
اشاره گر = new <نوع> [طول];
delete [] اشاره گر ;
```

مثال :

```
int *p = new int[10]; // تخصیص یک آرایهٔ پویا به طول ۱۰ از نوع صحیح
char *str = new char[20]; // تخصیص یک رشته به طول ۲۰
delete [] p;                // رها سازی
delete [] str;              // رها سازی
```

حل برنامهٔ بالا به زبان **C++** :

```
void main (void)
{
    int n, i, max;
    int *a;
    cin >> n;
    a = new int[n];  // تخصیص پویای یک آرایه
    for(i=0; i<n, i++)
        cin >> a[i];
    max=a[0];
    for(i=1; i<n, i++)
        if (max<a[i])
            max=a[i];
    delete [] a; // آزاد سازی حافظه پس از اتمام کار با آرایه
    cout << "Max = " << max;
} //end main
```

اشاره گر ها و آرایه ها

در زبان **C** بین آرایه ها و اشاره گر ها ارتباط نزدیکی وجود دارد (اشاره گر ها حاوی یک آدرس و اسم آرایه نیز یک آدرس است). اسم آرایه ، آدرس اولین عنصر آرایه را مشخص می کند.

```
int *ptr;
int a[5] = { 10, 20, 30, 40, 50 };
ptr = a;
```

a[0] ➔ *(a+0) ➔ *ptr	a[0] = 10
a[1] ➔ *(a+1) ➔ *(ptr+1)	a[1] = 20
a[2] ➔ *(a+2) ➔ *(ptr+2)	a[2] = 30
a[3] ➔ *(a+3) ➔ *(ptr+3)	a[3] = 40
a[4] ➔ *(a+4) ➔ *(ptr+4)	a[4] = 50

اشاره گرها و رشته ها : چون رشته ها نوعی آرایه از جنس کاراکتر هستند ، بین آنها و اشاره گرها ارتباط نزدیکی وجود دارد :

روش های مقداردهی اولیه به اشاره گری از رشته ها :

```
char *c = “In The Name Of GOD”;
char *str[5] = {“Ali” , “Babak”, “zinol”, “baqher”, “sohrab”};
```

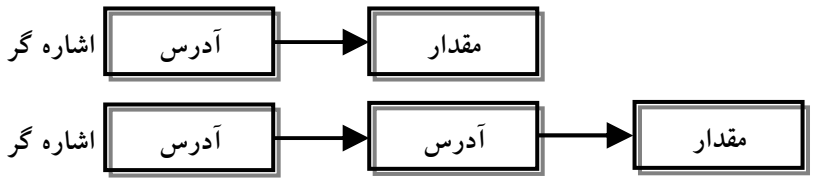
تابعی بنویسید که رشته ای را به عنوان پارامتر ورودی گرفته و تمام حروف کوچک آن را به حروف بزرگ تبدیل کند.

```
void strupr (char  *str)
{
        while (*str)
            if  ( *str >= ‘a’ && *str <= ‘z’)
                *str -= 32;
}//end strupr
```

اشاره گر به اشاره گر : اگر اشاره گری آدرس اشاره گر دیگری را در خود ذخیره کند ، در اینصورت به آن اشاره گر به اشاره گر می گویند. نحوه تعریف اشاره گر به اشاره گر بصورت زیر است.

<نوع> ** اسم متغیر اشاره گر به اشاره گر ;

int **p , *q , x = 10;
q = &x;
p = &q;



نام متغیر	محتوی	آدرس
x	۱۰۲٤	۵۰۰
		۵۰۱
⋮	⋮	⋮
q	۵۰۰	٦۰۰
		٦۰۱
⋮	⋮	⋮
p	٦۰۰	۷۰۰
		۷۰۱
⋮	⋮	⋮

printf(“%d”, x); ==== printf (“%d”, *q); ==== printf(“%d”, **p);

مثال) برای هر مورد خروجی را مشخص کنید؟

نام متغیر	محتوی	آدرس
k	۷۰	۵۰۰
		۵۰۱
⋮	⋮	⋮
p	۵۰۰	٦۰۰
		٦۰۱
⋮	⋮	⋮
q	٦۰۰	۷۰۰
		۷۰۱
⋮	⋮	⋮

```
int k=70;
int *p=&k;
int **q=&p;
```

1)cout<<q;	1)600
2)cout<<*q;	2)500
3)Cout<<**q;	3)70
4)cout<<*p;	4)70
5)cout<<&q;	5)700
6)cout<<*(++q);	6)?
7)cout<<++**q;	7)71
8)cout<<++**q;	8)72

حل (۷) بدون در نظر گرفتن (٦)می باشد.

دستورات پیش پردازنده :

پیش پردازنده نوع مترجم است که دستورات توسعه یافته ای از یک زبان را به دستورات قابل فهم برای کمپایلر همان زمان تبدیل می کند . قبل از اینکه برنامه کمپایل شود پیش پردازنده اجرا می شود و دستورات را که با # شروع شده اند را ترجمه می کند . سپس کمپایلر برنامه را کمپایل می کند. دستورات پیش پردازنده شامل :

#include // **برای ضمیمه کردن یک فایل سرآیند به برنامه استفاده می شود**

#define // **برای تعریف ماکرو**

#undef // **حذف تعریف یک ماکرو**

#ifdef // **اگر یک ماکرو تعریف شده است آنگاه**

#ifndef // **اگر یک ماکرو تعریف نشده است آنگاه**

#if // **اگر شرط برقرار بود آنگاه**

#endif // **انتهای بلوک if**

#else // **در غیر اینصورت**

#elif // **else if**

#error // **ایجاد یک خطا در روند کمپایل یک برنامه**

#pragma // **تغییر رفتار کامپایلر**

ماکرو چیست ؟ **(macro)**:

ماکرو نامی برای یک عبارت است که این عبارت می تواند ترکیبی از حروف رشته ها اعداد و یا توابع باشد برای تعریف ماکرو از دستور پیش پردازنده **#define** استفاده می کنیم .

نحوه تعریف ماکرو :

#define	**اسم ماکرو**	**عبارت**
#define	**a**	**10**
#define	**str**	**"REZA"**
#define	**TRUE**	**1**
#define	**FALSE**	**0**
#define	**Begin**	**{**
#define	**End**	**}**
#define	**prns(x)**	**printf("%s",x)**
#define	**Random(N)**	**rand( ) % (n+1)**

ساختمانها structures :

گاهی لازم است چند عنصر غیر هم نام را تحت عنوان یک نام در حافظه ذخیره کنیم برای انجام این کار از ساختمانها استفاده می کنیم .

نحوه تعریف یک ساختمان :

```
struct نام ساختمان {
                اجزاء ساختمان ;
            };
```

نام ساختمان از قوانین نام گذاری متغیر ها تبعیت می کند. اجزاء ساختمان می توانند متغیرها ، آرایه ها و یا حتی ساختمان های دیگری باشند. مثال :

```
struct student {
                char name[21];
                int id;
                float avg;
            };
```

ساختمان بالا ۲۷ = ٤+۲+۲۱ بایت از حافظه را اشغال می کند

class ها و **struct** ها در زبان c++ عملکرد یکسانی دارند ولی در زبان c , struct فقط شامل عناصر داده ای می باشد.

روشهای تعریف متغیری از جنس ساختمان :

۱) **روش اول : در حین تعریف ساختمان متغیری را نیز از جنس ساختمان را نیز تعریف می کنیم .**

```
struct student {
                char name[21];
                int id;
                float avg;
            } std1, std2;
```

std1 , std2 متغیرهایی از جنس student هستند

۲) **روش دوم : پس از تعریف ساختمان و در خلال برنامه**

```
struct student {
                char name[21];
                int id;
                float avg;
            };
main( )
{
    struct student std1, std2; متغیرهایی از جنس student هستند std1 , std2
}
```

دسترسی به اجزاء ساختمان

اسم جزء . اسم متغیر ساختمانی

```
std1.avg = 10.25;
std.id   = 83223132;
```

مقدار دهی اولیه به یک ساختمان :

```
struct student s1 = {“Ali Ahmadi”, 83223132, 10.25};
struct student s2;
        s2 = s1;
```

انتساب در ساختمانها :

مثال) برنامه ای بنویسید که نام و نام خانوادگی و معدل ۱۰ دانشجو را از ورودی بخواند و مشخصات دانشجوی نمونه را چاپ کند؟

```
struct  student  {
                char name [31];
                char family[31];
                float  avg;
            };

void main(void)
{
    struct  student s, max;
    int i;
    cin >> s.name >> s.family >> s.avg;
    max=s;
    for (i=1;  i<10; i++)
    {
        cin >> s.name >> s.family >> s.avg;
        if (max.avg < s.avg)
            max = s;
    }
    cout  << “name   =”  << max.name  << endl ;
    cout  << “family = ” << max.family << endl;
    cout  << “avg      = " << max.avg << endl;
   getch();
}
```

آرایه ای از ساختمان :

```
struct  student  {
                char  name[31];
                char family[31];
                float   avg;
            };

struct  student  s[10];          // آرایه ای ۱۰ عنصری از ساختمان

strcpy  (s[0].name, "Ali");
strcpy  (s[0].family,"Ahmad");
s[0].avg = 15.25;

cin >> s[1].name >> s[1].family >> s[1].avg;
```

برنامه ای بنویسید که مشخصات ۱۰ دانشجو شامل نام و نام خانوادگی و معدل را از ورودی خوانده و آنها را براساس معدل مرتب کرده و به ترتیب چاپ کند .

```
struct  student  {
                        char name [31];
                        char family [31];
                        float avg;
                  };

void main (void)
 {
    int i;
    struct  student s[10], temp;
    for(i=0; i<10; i++)
        cin >> s[i].name >> s[i].family >> s[i].avg;

    for (i=1; i<10; i++)
       for (j=0; j <10-i; j++)
            if (s[j].avg > s[j+1].avg)
            {
                temp   = s[j];
                s[j]   = s[j+1];
                s[j+1] = temp;
            }
   for (i=0;  i<10;  i++)
       cout << s[i].name << s[i].family << s[i].avg << endl;
}
```

مثال) برنامه ای بنویسید که مشخصات ۲۰ دانشجو را گرفته و معدل دومین دانشجوی نمونه را چاپ نماید.

```
Struct student {
            int id;
            char name[20];
            float avg;
                 };
void main (void)
{
   struct student s, max1 , max2 ;
   for (int i=0; i<20; i++)
   {
       cin >> s.id >> s.nsme >> s.svg;
       if(s.avg > max1.avg)
       {
         max2=max1;
         max1=s;
        }
       else if (s.avg > max2.avg)
            max2 = s;
   }
   cout << max2.id << max2.name << max2.avg;
}
```

مثال) برنامه ای بنویسید که n را از ورودی بخواند یک آرایه n عنصری از ساختمانها ایجاد کرده و ساختمان ها را از ورودی خوانده ومشخصات افرادی که نامشان **zinol** است را چاپ کند.

(struct student موجود است.)

```
void main (void)
{
    int n;
    cin>>n;
    struct student *p= new struct student [n];
    for (int i=0; i<n; i++)
        cin >> p[i].id >> p[i].name >> p[i].avg;
    for (i=0 ; i<n; i++)
        if ( strcmp(p[i].name,"zinol") == 0)
           cout <<p[i].id<<p[i].name<<p[i].avg;
   delete[] p;
   getch();
}
```

تعریف ساختمانها بصورت تودرتو (لانه ای) :

- روش اول

```
struct date {
                int day;
                int month;
                int year;
            };

struct employee {
                    char name[31];
                    struct date dt;
                    int salary;
                    };

struct employee emp;
strcpy (emp.name,"Ali Ahmadi");
emp.dt.day=2
emp.dt.month=8
emp.dt.year=1300;
emp.salary=20;
```

- **روش دوم :**

```
struct employee {
                    char name[21];
                    struct date {
                                    int day;
                                    int month;
                                    int year;
                                } dt;
                    int salary;
                  };
```

اشاره گری از نوع ساختمان : در زبان **c** می توان اشاره گری از نوع ساختمان تعریف نمود .

علت استفاده از اشاره گر ساختمان :

۱- فراخوانی توسط ارجاع را امکان پذیر می سازد .

۲- برای ایجاد لیست های پیوندی مورد نیاز است .

```
Struct student {
                    char name[21];
                    int id;
                    float Avg;
                };
struct student  s = {"Ali", 8225, 14.75};
struct student  *p;
p = &s;
```

برای دسترسی به فیلدهای با استفاده از اشاره گرها :

- روش اول (*p).id = ٤٠٠ فیلدموردنظر.(اسم اشاره گر *)
- روش دوم و مناسبتر **p → id** = ٤٠٠

union ها : در زبان **c** ، **union** محلی از حافظه است که توسط ۲ یا چند متغیر به طور اشتراکی مورد استفاده قرار می گیرد این متغیرها همزمان نمی توانند از این محل استفاده کنند . بلکه هر متغیر می تواند در زمان های متفاوتی این محل را مورد استفاده قرار دهد .

union نام **{**

اجزاء ;

} لیست متغیرها **;** // می توانیم متغیر تعریف کنیم

```
union  test {
            char c;
            int i;
            float f;
            double;
        };
```

نکته : حافظه مصرفی یک union برابر است با طول بزرگترین متغیر آن.

تغییر نام نوع های موجود با استفاده ازعملگر **typedef** :

typedef نوع موجود اسم جدید ;

```
typedef          int              integer;
typedef          unsigned int     unit;
typedef          struct student   STD;
typedef          struct student*  PSTD;
```

تعریف و تغییر نام نوع یک ساختمان در هنگام تعریف ساختمان :

```
typedef   struct   student {
                        char name [21];
                        int id;
                        float avg;

                } std, *pstd;
```

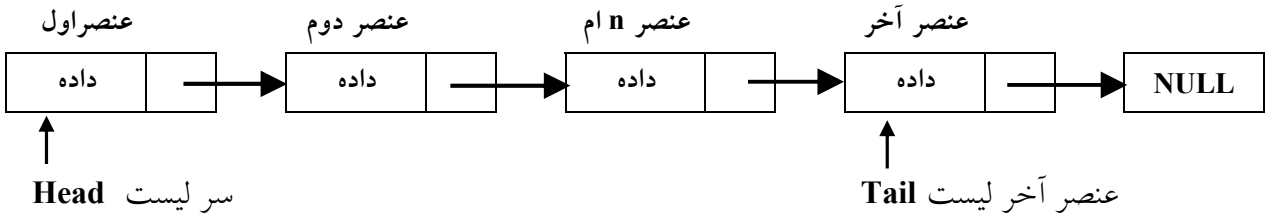
دو نوع جدید از struct student

std k ; متغیری **k** از جنس ساختمان **student**

pstd pk ; اشاره گری **pk** از جنس ساختمان **student**

لیست پیوندی :

Linked list : لیست پیوندی ساختمان داده ای است که در آن هر عنصر آدرس عنصر بعدی را در خود نگه می دارد نحوهٔ دسترسی به عناصر یک لیست پیوندی بصورت ترتیبی می باشد . یعنی اگر بخواهیم به عنصر سوم برسیم ابتدا به عنصر اول سپس به عنصر دوم و در نهایت به عنصر سوم خواهیم رسید. لیست پیوندی ساختمانی است که یکی از اجزایش اشاره گری از همان ساختمان است. به عنصر اول لیست ، سر لیست یا **Head** و به عنصر آخر لیست **Tail** می گویند. اشاره گر عنصر آخر لیست به **NULL** اشاره می کند.




```
typedef struct  student {
                        char name[21];
                        char family[21];
                        int Id;
                        float Avg;
                        struct student * next;// اشاره گری از جنس خود ساختمان
                } std;
```

آدرس عنصر اول = Head

آدرس عنصر دوم = Head–> Next

آدرس عنصر سوم = عنصر دوم–>Next

آدرس شروع لیست را همواره به درون یک اشاره گر به نام **Head** می ریزیم و این **Head** را هیچ وقت دستکاری نمی کنیم .

مثال) برنامه ای بنویسید که مشخصات چندین دانشجو را شامل نام ، نام فامیل ، شماره دانشجویی ، معدل را از ورودی خوانده و یک لیست پیوندی ایجاد کرده و سپس آنها را چاپ کند ؟ (آخرین دانشجو با معدل صفر مشخص می شود).

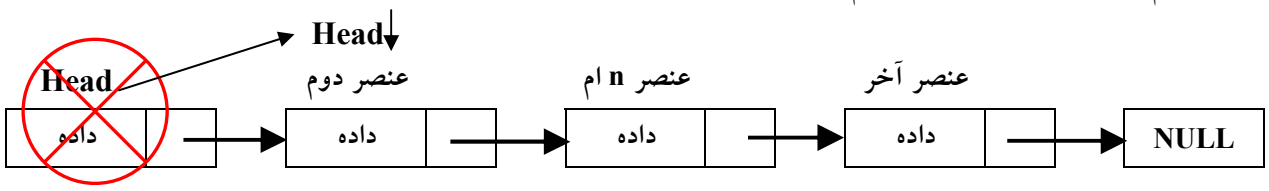
```
typedef struct {
                    char name[21];
                    char family[21];
                    int id;
                    float avg;
                    struct student * next;
                } std;
```

std *Head= NULL ,*Tail=NULL; // اشاره گر سر و آخر لیست معمولاً بصورت سراسری تعریف می شوند

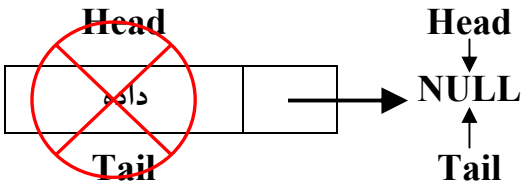
```
void main(void)
{
    std s;

    do {
            cin >> s.name >> s.family >> s.id >> s.avg;
            if (s.avg>0)
                Add toEndlist(s);
        }   while(s.Avg>0);

    Print_list( );
    Delete_All( );
}//end main

void AddtoEndlist(std p)
{
    std *temp = new std;
    *temp=p;

    if (Head == NULL) // یعنی لیستی وجود ندارد و این اولین عنصر است

        Head = Tail =Temp; //لیست تک عنصری اولین و آخرین عنصر آن یکی هستند

    else // در غیر اینصورت آن را به انتهای لیست اضافه کن

    {
        Tail->next=Temp;
        Tail=Temp;
    }
    Tail->next =NULL;  // اشاره گرآخرین عنصر به NULL اشاره می کند

}

void Print_list( )
{
    std *temp = Head;
    while (temp != NULL)
    {
        printf("\n name=%s , family=%s avg=%f Id=%d",temp->name,
                temp->family, temp->avg, temp->id);
        temp = temp->Next
    }
}

void  Delete_All( )
{
    std *Temp;
    while (Head != NULL)
    {
        Temp=Head;
        Head=Head ->Next;
        delete Temp;
    }
    Head = Tail = NULL;
}
```

حذف یک رکورد خاص : در حذف یک رکورد خاص باید به این نکته توجه کنیم که رکورد در کجای لیست قرار دارد. یک رکورد ممکن است در یکی از چهار موقعیت زیر قرار گرفته باشد.

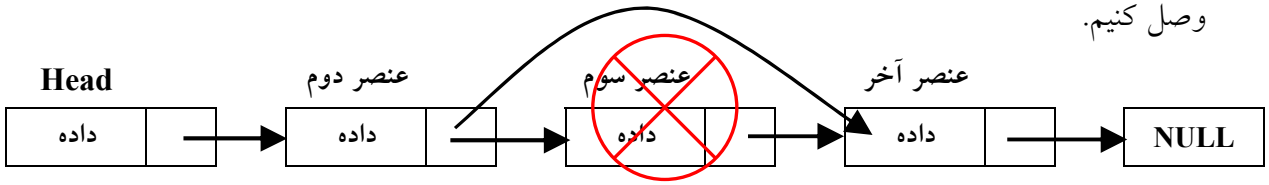
۱- عنصر که می خواهیم حذف کنیم اولین عنصر لیست **(Head)** می باشد. پس باید **Head** را یک عنصر بـه جلـو ببریم ، بعد عنصر را حذف کنیم.



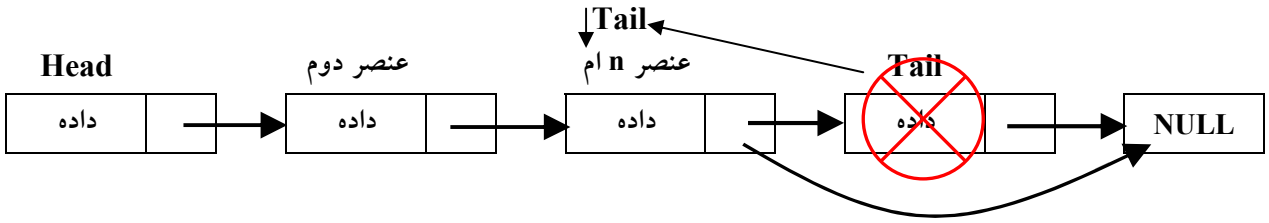
۲- لیست فقط یک عنصر دارد که هـم **(Head)** و هـم **(Tail)** مـی باشـد. در اینصـورت **Head , Tail** هـر دو بـه NULL اشاره می کنند.



۳- یکی از عناصر وسط لیست. در این صورت باید عنصر قبلی را به عنصر بعدی که بعد از عنصر مورد نظر اسـت وصل کنیم.



٤- آخرین عنصر لیست **(Tail)**. در اینصورت باید **Tail** را یک عنصر قبل متصل کنیم.



نکته : برای حذف عنصر خاصی از لیست پیوندی باید آدرس عنصر قبل از آن را داشته باشیم تا اشاره گر عنصر قبلـی را به عنصر بعدی وصل کنیم.

تابعی بنویسید که یک رکورد خاص را به عنوان ورودی گرفته و رکورد متناظر را در لیست پیوندی حذف کند.

```
typedef struct {
                char name[21];
                char family[21];
                int id;
                float avg;
                struct student * next;
        }  std;
```

```
int  DeleteSpecialRecord (std p)
{
    int    ret=0;
    std   *pervious = NULL;
    std   *Temp     = Head;

    if (Head == NULL) // اگرلیستی وجود ندارد
        return ret;
    else
    {
        while (temp != NULL) // تا موقعیکه به انتهای لیست نرسیم
        {
            if (strcmp(temp->family , p.family) == 0) // اگر فامیلی مورد نظر پیدا شد
            {
                    if (temp==Head && temp==Tail) //  حالت ۲ : لیست فقط یک عنصر دارد
                    {
                            Head = Tail = NULL;
                    }
                    else if (temp == Head) // حالت ۱ : اولبن عنصر لیست است
                    {
                        Head = Head->next;
                    }
                    else if  (temp ==Tail) // حالت ٤ : آخرین عنصر لیست
                    {
                            Tail = pervious;
                            pervious->next = NULL;
                    }
                    else // حالت ۳ : عنصر میانی لیست
                    {
                        pervious->Next=temp ->next
                    }
                    delete temp ;// آزاد کردن عنصر مورد نظر
                    ret=1;
                    break; // شکستن حلقه
            }\\end if
            pervious = temp; // نگهداری آدرس عنصر قبلی
            temp = temp->next; // برو به عنصر بعدی
        }\\end while

    }//end else
return ret;

}//end Delete
```

جستجو کردن (find) :

تابعی بنویسید یک فامیلی را به عنوان ورودی گرفته و درون لیست جستجو کند و آدرس عنصری که فامیل آن با فامیل رکورد ورودی مساوی باشد را برگرداند ؟

```
std* find (char *family)
{
    std *temp = Head;

    while(temp!=NULL)
    {
        if (strcmp(family , temp->family) == 0)
            break;

        temp=temp->next;
    }
    return temp;
}//end find
```

تابعی بنویسید که یک رکورد را بعد از رکورد خاص اضافه کند این تابع ۱ رکورد و یک فامیلی را به عنوان ورودی گرفته و رکورد دوم را بعد از رکوردی که فامیلش با فامیلی ورودی برابر است به لیست اضافه می کند ؟

```
int AddAfterSpecialRecord (char *family, std r2)
{
    int ret=0;
     if (Head== NULL)
       return ret;
    else
    {
       std * temp = new std;
      *temp = s;
       std *p=Head;
       while (p != NULL)
       {
           if (strcmp   (name, p->name) == 0)
           {
               temp->next = p->next;
               p->next = temp;
               if (p == Tail)
                  Tail = temp;
               Ret =1;
               break;
           }
           p=p->next;
       }
       if (ret == 0)
          delete temp;
        return ret;
    }
}
```

فایلها :

تاکنون تمام برنامه هایی که نوشته شده ، داده های مورد نیازشان را در متغیرهای معمولی ، آرایه ها و ساختمانها ذخیره می کردند . باتوجه به این که تمام این ساختارها در حافظه **RAM** ذخیره می شوند با قطع جریان برق محتویات آنها پاک می شود در برخی از برنامه ها لازم است که این اطلاعات بصورت دائمی ذخیره می شوند در نتیجه برای رفع این مشکل ساختمان داده جدیدی به نام فایلها که در حافظه ثانویه ذخیره می شوند پدیدار گشتند .

انواع فایلها : ۱- فایلهای متنی ۲- فایلهای باینری

بررسی فایلهای متنی :

در این نوع فایلها داده ها (حروف ، اعداد ، علائم) بصورت رشته ای از کارکترها ذخیره می شوند در این نوع فایلها اساس خواندن و نوشتن کارکتر است . در حالت متنی کارکتر ۲۶ انتهای فایل را مشخص می کند . یا می توان از ماکرو **EOF** استفاده نمود .

نحوه کار با فایلها : ۱- باز نمودن فایل ۲- خواندن و نوشتن و یا اضافه نمودن ۳- بستن فایل

۱- باز نمودن فایل : برای باز نمودن فایل از تابع **fopen** بصورت زیر استفاده می کنیم .

اشاره گر فایل // ; FILE *f p

; ("مدبازکردن" و"مسیر فایل")fp = f open

مد باز کردن : شامل نوع دستیابی [یعنی ورودی (r) یا خروجی (w)] و نوع فایل [یعنی متنی (t) یا باینری (b)] می باشد.

مدبازکردن	توضیحات
wt	**یک فایل متنی در حالت نوشتن (خروجی) باز می کند اگر فایل از قبل موجود باشد آن را overwrite می کند.**
rt	**یک فایل متنی را در حالت خواندنی (ورودی) باز می کند .**
wb	**یک فایل باینری را در حالت نوشتن(خروجی) باز می کند اگر فایل از قبل موجود باشد آن را overwrite می کند.**
rb	**یک فایل باینری را در حالت خواندنی باز می کند .**
w+t	**یک فایل متنی را در حالت خواندنی و نوشتنی باز می کند اگر فایل از قبل موجود باشد آن را overwrite می کند.**
r+t	**یک فایل متنی را در حالت خواندنی و نوشتنی باز می کند اگر فایل از قبل موجود باشد آن را overwrite می کند.**
w+b	**یک فایل باینری را در حالت خواندنی و نوشتنی باز می کند اگر فایل از قبل موجود باشد آن را overwrite می کند**
r+b	**یک فایل باینری را در حالت خواندنی و نوشتنی باز می کند اگر فایل از قبل موجود باشد آن را overwrite می کند**
a+t	**یک فایل متنی را در حالت خواندنی و نوشتنی باز می کند اگر فایل از قبل موجود باشد به انتهای آن اضافه میشود**
a+b	**یک فایل باینری را در حالت خواندنی و نوشتنی باز می کند اگر فایل از قبل موجود باشد به انتهای آن اضافه میشود**

۲- خواندن ونوشتن در یک فایل متنی

خواندن از تابع getc : char getc (اشاره گر فایل);

نوشتن از تابع putc : putc (اشاره گر فایل ,کاراکتر);

۲- خواندن و نوشتن در یک فایل باینری :

- خواندن و نوشتن با فرمت :
  - خواندن **fscanf** : (آدرس متغیرها ، کارکتر فرمت ، اشاره گر فایل) **fscanf**
  - نوشتن **fprintf** : (منغیرها ، کارکترهای فرمت ، اشاره گرفایل) **fprintf**
- خواندن و نوشتن رکورد :
  - خواندن **fread** :(اشاره گر فایل،تعداد بلوک،اندازه به بایت،آدرس بلوک داده) **fread**
  - نوشتن **fwrite** :(اشاره گر فایل، تعداد بلوک،اندازه به بایت،آدرس بلوک داده)**fwrite**

```
int i; char c; float f; FILE* fp;
fscanf(fp,"%d%f%c",&i,&t,&c)
fprint(fp,"%d%f%c",i,f,c);
```

برای تشخیص انتهای یک فایل باینری از تابع **feof** بفرمت زیر استفاده می کنیم.

int feof (اشاره گر فایل);

خروجی تابع **feof** در صورتی که به انتهای فایل رسده باشیم صفر می باشد ، در غیر اینصورت مخالف صفر می باشد.

۳- بستن فایل : برای بستن هر نوع فایلی از تابع **fclose** به فرمت زیر استفاده می کنیم :

fclose (اشاره گر فایل);

سوال : برنامه ای بنویسید که تا زمانی که کلید X فشرده نشده است از ورودی کاراکتر خوانده و آنها را در یک فایل متنی به اسم **ali.txt** ذخیره کند ، در نهایت تعداد کاراکتر های فایل را شمرده و فایل را چاپ کند.

```
void main (void)
{
    int t = 0;   char c;
    FILE *fp;
    fp = fopen("c:\\ali.txt",  "wt");
    do {
            c = getch();
            if (c == 'x' || c == 'X')
                break;
            putc (c, fp);
        }while (1);
    fclose(fp);
    c = 0;
    fp = fopen("c:\\ali.txt",  "rt");
    while(c != EOF)
    {
        c = getc(fp);
        t++;
        putch(c);
    }
    printf("Tadad = %d", t);
}// end main
```

برنامه ای بنویسید که یک رشته را از ورودی بخواند و در فایل c:\\ali.txt جستجو کند و تعداد تکرار رشته خوانده شده را چاپ کند.

```
void main (void)
{
    FILE *fp;
    char str[10], c=0;
    int i, l, t ;
    cin >>str;
    fp = fopen("c:\\ali.txt","rt");
    l=strlen(str);
    i=0;
    while (c != EOF)
   {
       c=getc(fp);
       if (c == str[i])
      {
          i++;
          if (i == l)
          {
            t++;
            i=0;
          }
      }
      else
        i=0;
    }
    cout<< "tedad = "<< t;
    getch();
}
```

برنامه ای بنویسید که مشخصات ۱۰ دانشجو رااز ورودی بخواند و انهارا درون یک فایل باینری با مسیرc:\\test.dat ذخیره کند.

```
void  main  (void)
{
     std s[10];
     int i;
     for (i=0; i<=10; i++)
         cin>> s[i].id >> s[i].name >> s[i].avg;
     file *fp;
     fp = fopen("c:\\test.dat","wb");
     fwrite(s, sizeof(std)*10; 10, fp);
     fclose(fp);
    getch();
}
```

تابعی بنویسید که رکوردهای دانشجویان را از درون فایل باینری استخراج کرده و به لیست پیوندی اضافه کند ؟

```
void  LoadFromFile ( )
{
    std r ;
    FILE *fp = fopen("c:\\test.dat","rb");
    fread(&r, sizeof (std)-sizeof(std*), 1, fp);
    while (feof (fp) == 0)
    {
        AddToEndList (r );
        fread (&r,sizeof(std) - 2, 1, fp);
    }
    fclose(fp);
}
```

تابعی بنویسید که رکوردهای دانشجویان را از لیست پیوندی بـه درون یـک فایـل بـاینری بـه نـام **test.dat** بریزیـد. اشاره گر سر لیست از قبل موجود می باشد **(Head)**.

```
void SaveToFile( )
{
    std *temp = Head;
    FILE *fp=fopen("c:\\test.dat","wb");

    while (temp != NULL)
    {
        fwrite(temp, sizeof(std)-sizeof(std*), 1, fp);
        temp = temp->next;
    }
}
```

تابع window : برای رسم یک پنجره در مد متنی به کار می رود . **window (x1,y1,x2,y2);** در اینجا **(x1, y1)** مختصات گوشه سمت چپ و بالا و **(x2, y2)** مختصات گوشه سمت راست و پایین را مشخص می کند.

تابع **textcolor** : رنگ متن درون یک پنجره را مشخص می کند.

تابع **textbackgroung** : رنگ پیش زمینهٔ یک پنجره را مشخص می کند.

تابع **()clrscr** : باعث پاک شدن یک پنجره می شود.

اگر بخواهیم یک پنجره را در مد متنی رسم کنیم باید دستورات زیر را بترتیب زیر بکار بگیریم :

1) **window(x1, y1, x2, y2);**
2) **textcolor(textColor);**
3) **textbackground(backcolor);**
4) **clrscr();**

برنامه ایجاد یک منو با ۷ آیتم:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>

const int MI  = 7;      // تعداد آیتم های منو
const char MenuItem[MI][10] = {"Add", "Delete", "Update" , "Search",
                               "Show" , "Dos Shell", "Exit" };
const int MH  = (MI * 2) + 4;    // ارتفاع منو
const int MW  = 28;  //پهنای منو
const int X1  = 25;  // نقطهٔ شروع منو
const int Y1  = 3;
const int X2  = X1 + MW;
const int Y2  = Y1 + MH;

void DrawMenu();
void windows(int x1, int y1, int x2, int y2, int textColor, int backcolor);
void DrawMenuWindows();
void DosShell();
void RunMenuItem (int l);
/*************************************************************************/
void main (void)
{
    _setcursortype(_NOCURSOR);
    DrawMenu();
}
/*************************************************************************/
void DrawMenu()
{
    int l = 0; char c;

    DrawMenuWindows();
    while(1)
    {
        windows(X1+5, Y1+3 + 2*l, X2-5, Y1+3 + 2*l, 14, 0);
        printf("    %s",MenuItem[l]);

        c =  getch();

        if( c != 0)
        {
            if (c == 13)
              RunMenuItem(l);
        }
        else
        {
            windows(X1+5, Y1+3 + 2*l, X2-5, Y1+3 + 2*l, 7, 1);
            printf("    %s",MenuItem[l]);
            c = getch();
```

```
            switch (c)
            {
                 case 80:    // Press DownArrow
                    l++;
                    if (l > MI -1 )
                      l = 0;
                    break;


                 case 72:   // Press UpArrow
                    l--;
                    if (l < 0 )
                        l = MI-1;
                    break;
            }
        }
    }
}
/*************************************************************************/
void windows(int x1, int y1, int x2, int y2, int textColor, int backcolor)
{
    window(x1, y1, x2, y2);
    textcolor(textColor);
    textbackground(backcolor);
    clrscr();
}
/*************************************************************************/
void DrawMenuWindows()
{
    windows(1, 1, 80, 25, 7, 0);
    windows(X1, Y1, X2, Y2, 42, 4);
    windows(X1+4, Y1+2, X2-4, Y2-2, 0, 0);
    windows(X1+5, Y1+2, X2-5, Y2-2, 0, 1);

    for(int i=0; i<MI; i++)
    {
        windows(X1+5, Y1+3 + 2*i, X2-5, Y1+3 + 2*i, 7, 1);
        printf("    %s", MenuItem[i]);
    }
}
/*************************************************************************/
void DosShell()
{
     windows(1,1,80,25,7,0);
     system("command.com");
     DrawMenuWindows();
}
/*************************************************************************/
void RunMenuItem (int l)
{
        switch (l)
        {
```

```
case 0:
    //Add();
    break;
case 1:
    //Delete();
    break;
case 2:
    //Update();
    break;
case 3:
    //Search();
    break;
case 4:
    //Show();
    break;
case 5:
    DosShell();
    break;
case 6:
    _setcursortype(_NORMALCURSOR);
    exit(0);
    break;
}
```

کلاس ها و اشیا:

کلاس ها هسته بر نامه نویسی شی گرا **Object Orignted (OO)** در C++ هستند.کلاس مجموعه ای از متغیر ها و توابعی است که بر روی این متغیرها عمل می کنند. همان طوری که نمونه ای از یک نوع از انواع اصلی را متغیر می گویند نمونه ای از کلاس را شی می نامند.

جمع کردن اطلاعات و مسؤلیت های هر نهاد در یک شی را بسته بندی **(Encapsulation)** می گویند. کلاس می تواند حاوی متغیر ها و توابع باشد. متغیر های کلاس را اعضای داده ای و توابع کلاس را توابع عضو و یا متد می گویند.

برای تعریف کلاس در زبان C++ از کلمه کلیدی**class** بصورت زیر استفاده کنیم :

```
class نام کلاس {
اعضای خصوصی کلاس
public:
اعضای عمومی کلاس
private:
اعضای خصوصی کلاس
protected:
اعضای محافظت شده کلاس
} اشیا کلاس;
```

نام گذاری کلاس از قوانین نام گذاری متغیر ها تبعیت می کند.

داده ها و توابعی که بلافاصله پس از نام کلاس می ایند مختص به این کلاس هستند و هیچ تابع یا کلاس دیگری نمی تواند به این اجزای کلاس دسترسی داشته باشد بلکه فقط اجزای همین کلاس می توانند از انها استفاده نمایند به همین دلیل آنها را اجزای اختصاصی و یا خصوصی کلاس می گوییم.

اگر توابع وداده هایی پس از کلمه کلیدی **public** اعلان کنیم. این داده ها وتوابع به صورت عمومی خواهند بود یعنی هر بخش دیگری از برنامه می تواند به آنها دسترسی داشته باشد.

اگر توابع یا داده هایی را پس از کلمه کلیدی **private** اعلان کنیم برای این کلاس اختصاصی خواهند بود.این توابع و داده ها شبیه توابع وداده هایی هستند که بلافاصله پس از نام کلاس ظاهر شده اند.

اگر توابع وداده هایی پس از**protect** تعریف کنیم محافظت شده اند که در آینده مورد بررسی واقع می شوند.

اگر درانتهای تعریف کلاس یعنی پس از آکولاد بسته } اسامی اشیا را ذکر کنیم این اشیا از نوع آن کلاس تعریف می شوند. تعریف اشیایی از کلاس مثل تعریف متغیری از انواع اولیه است.

مثال) کلاسی بنویسید که شعاع دایره ایی را از ورودی خوانده و مساحت آنرا محاسبه نموده و در خروجی چاپ نماید :

```
class  ccircle  {
    int radiuse ;
public:
   void    get radiuse ( );
   void    print ( );
};

void  ccircle:: get radiuse ( )
{
     cout << " Please Enter The Radiuse: " ;
     cin >> radiuse ;
}
void  ccircle:: print ( )
{
      cout << "s ="<< 3.14 * radiuse * radiuse ;
}

void  main  (void)
{
   ccircle  k;
   k .get radiuse ( );
   k .print ( );
   getch ( );
}
```

نکته)

۱.در تعریف یک کلاس نمی توانیم شی از همان کلاس داشته باشیم مگر به صورت اشاره گر.

۲.اعضای داده یک کلاس را نمی توانیم مقدار دهی اولیه کنیم مگر اینکه آن اعضا دارای کلاس حافظه استاتیک (static) باشند.

سازنده ها (constructors) :

سازنده تابعی است هم نام با نام کلاسی که در آن تعریف می شود و در هنگام ایجاد اشیایی از ان کلاس به اعضای داده ای انها مقدار اولیه می دهد. توابع سازنده هنگام تعریف شی به طور خودکار اجرا می شوند. این توابع هیچ مقداری را برنمی گردانندو حتی از نوع void هم نیستند.

نحوه تعریف سازنده ها :

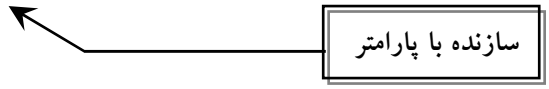
مثال)کلاسی بنویسید که دو متغییر b و a را از ورودی می خواند و دارای توابعی برای محاسبه توان و فاکتوریل می باشد.

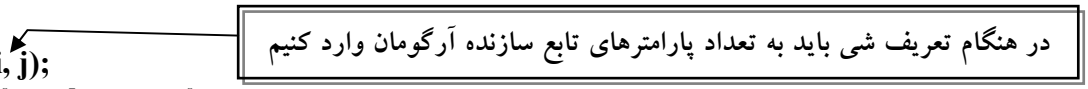
```
class   cpowerfact
{
 private :
        int a;
        int b;
 public:
        cpowerfact();
        void read():
        int  fact();
        int  power();
 private:
        int f;
        int p;
};
cpowerfact::cpowerfact()
{
     a = b = 0;
     f=p=1;
}
void  cpower::read()
{
     cout << "please enter a ,  b  = ";
     cin >> a >> b;
}
int cpowerfact:fact()
{
   if  (a ==1 || a == 0)
        f =  1;
  else
      for (int i=1; i<=a; i++)
          f=f*i;
  return f;
}
int cpowerfact::power()
{
     for (int i=1;i<=b;i++)
         p=p*a;
     return p;
}
void  main  (void)
{
     cpowerfact  h;
     h.read();
     cout << h.fact() << h.power();
}
```

برای سازنده با ارگومان مثلاً برای مثال قبل بدین صورت عمل می کنیم :

اول تابع **read** را پاک کرده و تابع سازنده را بدین صورت اصلاح می کنیم :

```
cpowerfact::cpowerfact(int k, int l)
{
   a = k;
   b = l;
   f = p = 1;
}
void  main (void)
{
  int i,j;
  cin >> i >> j;
  cpowerfact h(i, j);
  cout << h.fact() << h.power();
  getch();
}
```





مخرب ها (destructors) :

یک تابع ویژه ای از کلاس می باشد. مخرب کلاس هم نام با کلاس است وبا کاراکتر مد (~) شروع می شـود. هنگـامی که شی از بین می رود تابع مخرب فراخوانی می شود. و در قسمت **public** تعریف می شود. کاربرد تابع مخرب معمولاً در آزاد سازی حافظه ای است که بصورت پویا اختصاص داده شده است.

برنامه ای بنویسید که ۲۰ عدد را از ورودی خوانده و مجموع فاکتوریل ارقـام اعـداد اول را محاسـبه کنـد (تـابع main حداکثر ۵ خط باشد با این فرض که تابع fact وprime را نیز داریم).

```
class ctest
{
     int a[20];
     int sum;
  public:
     void read();
     int calc();
     ctest()    { sum=0; }
};
int ctest::calc()
{
   int i, n;
   for (i=0; i<20; i++)
        if (prime(i) == 2)
        {
            n=a[i];
            while(n>0)   {
               sum += fact(n%10);
               n=n/10;
            }
        }
   return sum;
}
```

```
void ctest::read()
{
    for (int i=0; i<20; i++)
       cin >> a[i];
}
void   main   (void)
{
     ctest  h;
     h.read();
     cout<< "sum = "<< h.calc();
}
```

مثال) برنامه بنویسیدکه ۵ نام را از ورودی بخواند و آنها را مرتب کرده و چاپ نماید.

```
class csort  {
     char s[5][20];
public:
     csort( )
    {
         for (int i=0;i<5;i++)
              s[i][0]='\0';
    }
    void read ();
    void sort();
    void print();
};
void csort :: read ()
{
     for (int i=0; i<5; i++)
          gets(s[i]);
}
void csort :: print ()
{
      for (int i=0;i<5; i++)
      put(s[i]);
}
void csort :: sort()
{   int i, j ;
    char temp[20];
    for (i=4; i>0; i--)
        for (j=0;j<i-1; j++)
             if (strcmp(s[i] , s[j+1] ) > 0)
             {
                  strcpy (temp , s[j]);
                  strcpy (s[j] , s[j+1]);
                  strcpy (s[j+1] , temp);
             }
}
void main (void)
{    csort k;
     k.read ();
     k.print ();
}
```

کلاسی بنویسید که یک آرایه ۲۰ عنصری را از ورودی دریافت کرده و آنها را مرتب سازی نماید.

```
const int n=20;
class csort  {
        int a[n];
 public:
        csort() { }
        void  read();
        void sort();
        void print();
};
void  csort::read()
{
    for (int i=0; i<n; i++)
         cin >> a[i];
}
void  csort()::print()
{
   for (int i=0; i<n; i++)
        cout << a[i] << endl;
}
void  csort ::sort()
{
     int  temp, i, j;
     for (i=n-1; i>0; i--)
        for (j=0; j<i-1; j++)
           if (a[j] < a[j+1])
           {
              temp=a[j];
              a[j]=a[j+1];
              a[j+1]=temp;
           }
}
void  main  (void)
{
   csort   h;
   h.read();
   h.sort();
   h.print();
}
```

تابع دوست کلاس(friend):

همان طوری که گفته شد توابعی که عضو کلاس نباشند نمی توانند به اعضای اختصاصی آن کلاس دسترسی داشته باشند.اما اگر تابعی دوست کلاس تعریف شود می توانند به تمام اعضای ان کلاس دستیابی داشته باشند. برای اعلان تابع دوست باید الگوی ان را داخل کلاس قراردهیم و کلمه کلیدی friend را قبل از آن ذکر کنیم.

مثال) برنامه ای بنویسید که کاربرد توابع دوست کلاس را نشان دهد. این برنامه n را از ورودی خوانده و یک ماتریس n*n ایجاد کرده و سپس میانگین عناصر روی دو قطر ماتریس را که اول هستندمحاسبه کند.

فرض کنید کلاس ctest موجود باشد که توابع prime و fact را دراختیار ما قرار می دهد. اصول برنامه نویسی شی‌گرا oo را رعایت فرمایید و از نشت حافظه ***(Memory Leak)*** جداً خودداری فرمایید.

```
class  CDynamicArray  {
       int *p;
       int  k;
  public:
      friend  float  avgmatrix();
      CDynamicArray(int  n)
      {
          k = n;
          p = new int[n*n];
      }
      ~CdynamicArray()
      {
          delete [ ]p;
      }
      void  CDynamicArray::read()
      {
         int  i, j;
         for (i=0;i<k; i++)
             for (j=0;j<k; j++)
                  cin >> p[(i*k)+j];
      }
      float  avgmatrix( )
     {
          int i, j, n, sum=0, t=0;
          ctest  c;
          cin  >>  n;
          CDynamicArray d(n);
          d.read();
          for (i=0;i<n;i++)
              for (j=0;j<n;j++)
                  if ((i==j || i+j==n+1)  &&  (c.prime(d.p[i*n+j]==2))
                  {
                      sum+ = d.p[i*n+j];
                      t++;
                  }
          return (sum / t);
      }

      void  main  (void)
     {
         clrscr( );
         cout << " avg = " << avgmatrix( );
     }
```

کلاس ها را نیز می توان دوست کلاس های دیگر تعریف کرد. در این حالت کـلاس دوسـت وتمـام عضـوهای ان بـه اعضای اختصاصی کلاس دیگر دسترسی دارند.

مثال) برنامه ای بنویسید که دو زمان را از ورودی بخواند و آنها را با هم مقاسیه کند. این برنامه شامل کـلاس ctime و کلاس ccompare می باشد{(o.o) و بسته بندی را رعایت کنید}

```
class  ctime  {
        unsigned  int  sec;
        unsigned  int min;
        unsigned  int hour;
  public:
       friend  class   ccompare;
       ctime( )  {  sec = min = hour = 0; }
       void read( ) { cin >> sec >> min >> hour; }
};
class  ccompare  {
   public:
     int cmptime(ctime   t1,ctime  t2)
     {
        return  (t1.hour*3600+t1.time*60+t1.sec)-(t2.hour*3600+t2.time*60+t2 .sec);
     }
void  main  (void)
{
     ctime  t1,t2;
     ccompare  k;
     t1 .read( );
     t2read( );
     int i = k.cmptime(t1,t2);
    if (i>0)
      cout << "t1>t2";
    else if (i<0)
      cout << "t1<t2";
   else
      cout << "t1=t2";
}
```