دانشگاه آزاد اسلامی

جزوهٔ درس برنامه سازی (زبان C++)

delbarir@yahoo.com

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	۴ بایت یا ۳۲ بیت	۰ تا ۴۲۹۴۹۶۷۲۹۵
long	۴ بایت یا ۳۲ بیت	۲۱۴۷۴۸۳۶۴۷ تا ۲۱۴۷۴۸۳۶۴۸-
float	۴ بایت یا ۳۲ بیت	دقت ۷ رقم
double	۸ بایت یا ۶۴ 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 برای یک خانه از حافظه است که محتویاتش ممکن است تغییر کند . یک متغیر با حروف **A-Z** یا **a-z** یا **_** (آندرلاین) شروع می شود و شامل آندرلاین ، حروف **A-Z** و **a-z** و ارقام **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 : **--** (ماینس ماینس)

```
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 مورد استفاده قرار گرفته و سپس عملگرها بر روی عملوند عمل می کنند (افزایش و یا کاهش عملوند) ولی اگر در سمت چپ عملوند قرار گیرند ، 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 :

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

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
( )
! ~ ++ -- *(محتوی) &(آدرس) sizeof
% / (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** بریزیم 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 اجرا خواهد شد . **main** تابع اصلی 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 نوع متغیر را که در خروجی نمایش داده می شود را مشخص می کنند.

نکته !) کاراکترهای کنترلی با \ (بک اسلش) شروع می شوند و باعث انتقال مکان نما به یک سطر و یا ستون خاص می شوند.

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

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

جدول کاراکترهای فرمت

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

جدول کاراکترهای کنترلی

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

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 شود . 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 بـر ۷ بخـش پـذیر هسـتند را چاپ کند (نکته : برای محاسبه میانگین همواره به یک متغیر برای نگهداری حاصلجمع و یک متغیـر بـرای تعـداد نیـاز 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 عدد 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;      حلقه محاسبه مغلوب
               a /= 10;
           }

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

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 $10 \le a \le 20$ 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 .

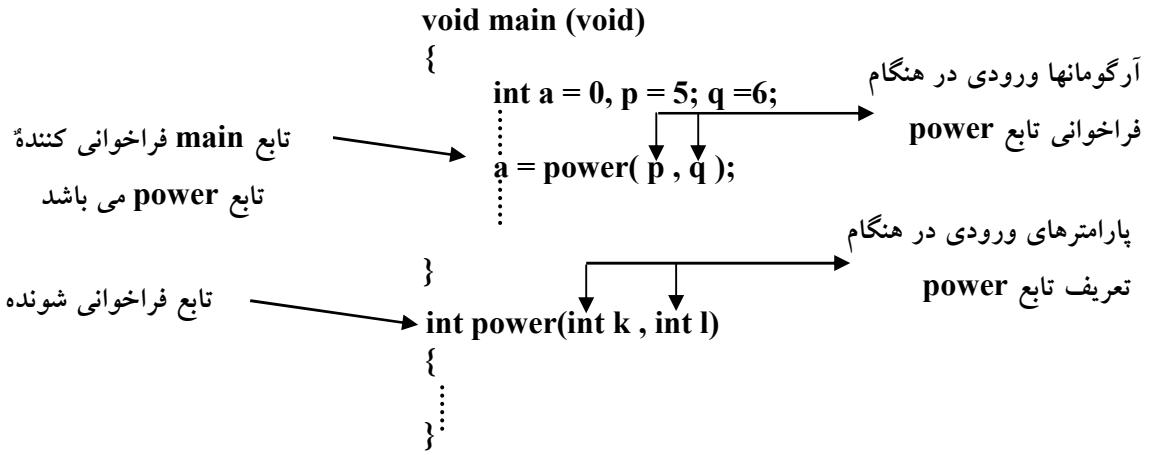
مثال) برنامه ای بنویسید که ۲ عدد **a,p** را از ورودی خوانده و **a** را به توان **p** برساند این برنامه از یک تابع استفاده کرده و **a,p** را به آن تابع ارسال کرده و بعد از محاسبه توان نتیجه را به تابع اصلی بر می گرداند. برای بازگرداندن نتیجه توان به تابع **main** باید از دستور **return** بصورت زیر استفاده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

١) 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( );
}
```

٣- برنامه ای بنویسید که میانگین اعداد اول کوچکتر از ١٠٠ 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 **n << 1** = 2^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 ١ ، **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 ٠ 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 **? , for , if , % , &** 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 های ۲۰ ریالی و ۳۰ ریالی 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 و b 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 : **const** <نوع متغیر> < نام متغیر> =مقدار

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

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

#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 :

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

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 حاقظهٔ مصرفی یک آرایهٔ **n** بعدی :

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

sizeof (int)*۲*۵=۲۰ = (۱)

(۲) = sizeof (int)*۲*۳*۵=٦۰

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};

a[0]	a[1]	a[2]	a[3]	a[4]
٦	۳	٤	۷	٦

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

a[0]	a[1]	a[2]	a[3]	a[4]
۱	۳	٤	۰	۰

۳) روش سوم int a [5] = {0};

در این روش تمام خانه ها صفر می شوند

a[0]	a[1]	a[2]	a[3]	a[4]
۰	۰	۰	۰	۰

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

a[0][0]	a[0][1]	a[0][2]	a[1][0]	a[1][1]	a[1][2]
۳	۱	۲	۲	۱	٤

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 به آرایه ها ما فقط می توانیم بعد 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 ) ;
}

void  sort (   1) int *k
               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
```