بسم الله الرحمن الرحیم

# مبانی کامپیوتر و برنامه سازی

# در C

Www.iepnu.ir

# زبان برنامه سازی C :

1) **سطح بالا :** به زبان های محاوره ای انسان نزدیک است. مثل Basic , Pascal
2) **سطح میانی :** هم ویژگی های زبان سطح بالا را دارد و هم سطح پایین مثل C
3) **سطح پایین :** به زبان ماشین نزدیک است مثل Assembly

## برنامه نویسی به زبان C

### ساختار کلی برنامه به زبان C :

↕ Header file (فایل های سرآمد)

↕ معرفی ثابتها , متغییرهای سراسری, زیر برنامه ها

```
main()
{
    معرفی متغییر های محلی ;
    ↕
    ↕ دستورات ;

}
```


**مثال** : برنامه ای بنویسید که حاصل جمع دو عدد 10 و 20 را چاپ کند.

```
#include <stdio.h>          (فایل سرآمد)
main()
```

```
{
   int  a,b,c ;            (متغییر های محلی)
    a = 10 ;
    b = 20 ;
    c = a + b ;
   printf (" %d",c) ;
 }
```

### ❖ فایل های سرآمد (Header files ) :

اگر در برنامه ای از توابع آماده استفاده کنیم باید در قسمت فایل های سرآمد آن ها را معرفی کنیم بدین معنی که هر گروه از توابع آماده دارای نام معینی می باشند که در این قسمت آن نام ها را می نویسیم. شکل کلی استفاده از فایل های سرآمد به این گونه است :

#include <فایل سرآمد>

توابع ورودی خروجی معمولاً در فایل سرآمد stdio.h قرار دارند .

### ❖ معرفی :

ثابتها , متغییر های سراسری زیر برنامه های :

در برنامه سازی ساخت یافته یک برنامه بزرگ را به مجموعه ای از زیر برنامه ها می شکنند و زیر برنامه ها را باید در قسمت معرفی , پیاده سازی کنیم .

**توجه :** هر برنامه ممکن است دارای زیر برنامه های متعددی باشد ولی حتماً دارای یک زیر برنامه اصلی است که با ()main آن را مشخص می کنند.

اگر بخواهیم از مقادیری درا داخل یک زیر برنامه استفاده کنیم آن ها را در همان زیر برنامه تعریف می کنیم و اصطلاحاً به آن ها متغییر های محلی ( local ) می گویند.

اگر بخواهیم از مقادیری در تمامی توابع استفاده کنیم باید آن ها را به صورت سراسری معرفی کنیم.

### ❖ شناسه ها :

اسامی زیر برنامه ها , متغییر ها , ثابتها , ساختارها , و ... که برای شناسایی به کار می روند را گویند. تعریف شناسه ها در هر زبان برنامه سازی قواعد خاص خود را دارد.

1) متغییر ( variable ) : مکان هایي از حافظه اصلی می باشند که دارای یک نام بوده و محتویات آن درطول برنامه قابل تغییر است .

**شکل کلی تعریف متغییرها :**

نوع متغییر نام متغییر (ها) ;

**مثال :**

int a , b ; (عدد صحیح )

float c ; (عدد اعشاری )



در هر زبان برنامه سازی , متناسب با نیازها و توانایي های آن انواع مختلفی از متغییر ها استفاده می شوند . در زبان مهمترین آنها عبارتند از :

| نوع متغییر | | مقدار حافظه | دامنه | اختصار |
|---|---|---|---|---|
| عدد صحیح | int | byte 2 | -32000 – +32000 | i,d |
| عدد اعشاری | float | byte 4 | -72000 – +72000 | f |
| کاراکتر | char | byte 1 | - 127 – 127 | c |
| | double float | 64 | $10^{-38}$ – $10^{38}$ | |
| | unsigned char | 8 | 0 – 255 | |
| | singed char | 8 | - 127 – 127 | |
| | int | 32 تا 16 | -32767 – 32767 | |
| | unsinged int | 32 تا 16 | 0 – 65535 | |
| | singed int | 32 تا 16 | -32767 – 32767 | |
| | short int | 16 | -32767 – 32767 | |
| | unsigned short int | 16 | 0 – 65535 | |
| | long int | 32 | 2447483647 تا منفی | |
| | long double | 80 | $10^{-4932}$ – $10^{4932}$ | |

**فایل های سرآمد :**

| | |
|---|---|
| <owl.h> | <window.h> |
| <dos.h> | <stdlib.h> |
| <math.h> | <msystem.h> |
| <stdio.h> | <bios.h> |
| <aclock.h> | <bitmap.h> |
| <string.h> | <dloc.h> |

❖ **نکته :**

**در زبان C :**

1- Error (خطا) : اگر در برنامه متغییری را تعریف نکنیم و در طول برنامه از آن استفاده کنیم. (تا آن رابرطرف نکنیم نمی توانیم به کار ادامه دهیم. )

2- Warning (اخطار) : اگر در برنامه متغییری را تعریف کنیم و از آن استفاده نکنیم . ( برنامه قابل ادامه دادن است.)

**ثابت ها** (constant) : مکان هایی از حافظه اصلی هستند که دارای یک نام بوده و مقدارشان در طول برنامه تغییر نمی کند.

* مزیت استفاده از ثابت علاوه بر افزایش خوانایی برنامه راحت تر شدن تغییرات آن می باشد . مثلاً اگر در اول برنامه pi = 3.14 را تعریف کنیم و در کل برنامه از pi استفاده کنیم , اگر بعد از اتمام برنامه مثلاً باید از 3/1417

استفاده می کردیم می توانیم فقط همان pi را که در اول برنامه تعیین کردیم تغییر دهیم که خود به خود باقی pi ها نیز مقدارشان تغییر می کند.

**شکل کلی تعریف ثابت :**

نوع ثابت  نام ثابت = مقدار ثابت **;**

**مثال** :  const  float  pi = 3.14 **;**

✓ **دستورات زبان C** :

**1- دستورات ورودی / خروجی:**

1.1. دستور ورودی یا خواندن که شکل کلی دستور به شکل زیر است:

scanf (" فرمت خواندن  " , متغییر های ورودی) **;**

**مثال**

```
scanf  (" %i%i" , &a&b) ;
scanf  ("%d %f " , &a &b) ;
```

1.2. دستور خروجی یا نوشتن که شکل کلی دستور به شکل زیر است :

printf (" فرمت نوشتن یا عبارت ", یا متغییر ها  یا عبارت ریاضی) **;**

**مثال**

```
printf (" %d " , a ) ;
printf (" the sum is : %d " , a + b  );
```

خروجی دستور آخر به صورت زیر می باشد :  the sum is : 20

**مثال :** برنامه ای بنویسید که شعاع دایره ای را از ورودی خوانده محیط و مساحت آن را چاپ کند:

```
# include < stdio.h >
main ()
{
   const   float    pi = 3.14 ;
   float    r,p,s ;
```

```
   printf (" please enter reduce :  \n ") ;
   scanf ( " %f " , & r ) ;
   p = 2*pi * r ;
   s = pi * r *r ;
   printf (" The S = %f , The P = % f " , s,p) ;
 }
```

خروجی این برنامه به شکل زیر است :

```
please enter reduce :      2
The S = 12.56 , The P = 12.56
```

**جدول عملگرها درون printf :**

| کاراکتر | عملی که باید انجام شود |
|---|---|
| \ f | موجب انتقال کنترل به صفحه جدید می شود |
| \ n | موجب انتقال کنترل به خط جدید می شود. |
| \ t | انتقال به 8 محل بعدی در صفحه نمایش |
| \ " | چاپ (") |
| \ , | چاپ (,) |
| \ 0 | رشته تهی |
| \\ | back slash |
| \ u | انتقال کنترل به 8 سطر بعد |
| \ N | ثابت های مبنای 8 |

| کاراکتر اختصاری | نوع اطلاعاتی که باید به خروجی برود |
|---|---|
| % c | یک کاراکتر |
| % d (or) i | اعداد صحیح دهدهی مثب |
| % e (or) E | نمایش علمی عدد همراه با حروف e و E |
| % f (or) g | اعداد اعشاری ممیز شناور |
| % 0 | اعداد مبنای 8 مثبت |

| | |
|---|---|
| % s | رشته ای از کاراکتر ها |
| % u | اعداد صحیح بدون علامت |
| % x | اعداد مبنای 16 مثبت با حروف کوچک |
| % X | اعداد مبنای 16 مثبت با حروف بزرگ |
| % p | pointer (اشاره گر) |
| % % | علامت % |
| % n | موجب می شود تا تعداد کاراکترهایی که تا قبل از این کاراکتر به خروجی منتقل شده اند شمارش شده در پارامتر متناظر با آن قرار گیرد |

## ✓ عملگرهای محاسباتي :

| عملگر | نام | مثال |
|---|---|---|
| - | تفریق و منهای یکانی | x-y یا -x |
| + | جمع | y + x |
| * | ضرب | x*y |
| / | تقسیم | x/y |
| % | باقی مانده تقسیم | x%y |
| -- | کاهش | --x |
| ++ | افزایش | y++ |

✓ **عملگرهای منطقی به ترتیب تقدم :**

| عملگر | نام | مثال |
|---|---|---|
| ! | نقیض | ! x |
| && | و | x>y && m<p |
| \|\| | یا | x>y \|\| m<p |

✓ **عملگرهای رابطه ای :**

| عملگر | نام | مثال |
|---|---|---|
| > | بزرگتر | x > y |
| < | کوچکتر | x < y |
| >= | بزرگتر مساوی | x >= y |
| <= | کوچکتر مساوی | x <= y |
| == | متساوی | x == y |
| != | نامساوی | x != y |

✓ **عملگرهای ترکیبی :**

| عملگر | نام | مثال | معادل |
|---|---|---|---|
| += | انتساب جمع | x += y | x = x+y |
| -= | انتساب تفریق | x -= y | x = x – y |
| *= | انتساب ضرب | x *= y | x = x*y |
| /= | انتساب تقسیم | x /= y | x = x/y |
| %= | انتساب باقیمانده | x % = y | x = x%y |

✓ **تقدم عملگرهای منطقی و رابطه ای :**

| | |
|---|---|
| بالاترین تقدم<br>↓<br>پایین ترین تقدم | !<br>>>= <<=<br>== !=<br>&&<br>\|\| |

**مثال1 )** برنامه ای بنویسید که مشخصات مثلثی را خوانده ، مساحت و محیط آن را نشان دهد .

```
# incloude <stdio.h>
# incloude <conio.h>
    Main( )
    {
        Float   a,b,c,p,s ;
        Printf (" \n enter a,b,c: ");
        Scanf(" %f,%f,%f ",&a,&b,&c);
        P=a+b+c;
        S=sqrt(p*(p-a)*(p-b)*(p-c));
        Printf(" \n the p=%f  the s=%f ",p,s );
        Getch( );
        Return(0);
      }
```

**مثال2 )** برنامه ای بنویسید که دو متغیر را از ورودی خوانده و محتویات آن را جابجا و چاپ نماید .

```
# incloude <stdio.h>
# incloude <conio.h>
    Main( )
    {
       Int x,y ;
       Printf (" enter 2 numbers : ");
       Scanf("%d,%d ",&x,&y);
       Printf(" \n  x=%d , y=%d " ,x , y);
       X=x+y ;
       Y=x-y ;
        X=x-y ;
       Printf(" \n \n  x=%d , y=%d " , x , y );
       Getch( );
       Return(0);
    {
```


**دستورات کنترلی :**

در یک برنامه دستورات به صورت ترتیبی و پشت سر هم اجرا می شوند, اگر بر اساس نیاز الگوریتم بخواهیم یک یا چند دستور را بر مبنای شرطی انجام دهیم یا ندهیم یا اینکه آن ها را به دفعات تکرار نماییم از دستورات کنترلی استفاده می کنیم.

1- **دستورات شرطی** :

این نوع از دستورات یک یا چند شرط را بررسی نموده بر اساس برقراری آن شرطهای دستوراتی را انجام می دهند و بالعکس .

1-1 – **دستورات شرطی if** : شکل کلی آن به صورت زیر است :

```
if     ((شرط (ها))                    if   (a > 0)
       دستورات  ;                           a= -a ;
```

**2-1 – دستورات شرطی if – else** : شکل کلی آن به صورت زیر است :

```
if   ((شرط(ها))                      if   (a < 0 )
     دستورات  ;                            printf (" negative " ) ;
else                                  else
     دستورات  ;                            printf (" positive " )
```

مثال3 ) برنامه ای بنویسید که سه عدد از ورودی گرفته مشخص کند می تواند اضلاع مثلث باشند یا خیر؟

```
# incloude <stdio.h>
# incloude <conio.h>
    Main( )
    {
       Float a,b,c ;
       Scanf(" %f , %f , %f ",&a,&b,&c );
       If ((a+b>c)&&(b+c>a)&&(a+c>b))
          Printf("yes");
       Else
          Printf("no");
       Getch( );
       Return(0);
     }
```

مثال4 ) برنامه ای بنویسید که یک عدد تک رقمی گرفته (حداکثر تا 3 ) سپس معادل حروفی آن را چاپ کند .

```
# incloude <stdio.h>
# incloude <conio.h>
    Main( )
    {
      Int a ;
      Clrscr ;
      Scanf("%d",&a);
      If(a=1)
          Printf("one");
     Else
     If(a=2)
          Printf("two");
     Else
     If(a=3)
           Printf("tree");
     Else
           Printf("other number");
      Getch( );
      Retun(0);
     }
```

مثال5 ) برنامه ای بنویسید که معادله درجه 2 را حل کند .

```
# incloude <stdio.h>
# incloude <conio.h>
    Main( )
    {
```

```
    Float a,b,c,d ,x1,x2;
    Printf("enter a,b,c:");
    Scanf("%f,%f,%f",&a,&b,&c);
    d=("\n",(b*b)-4*a*c);
    if (d>= 0)
    {
      X1= (-b+ sqrt(d))/2*a;
      X2= (-b-sqrt(d))/2*a;
      Printf("%f,%f ,\n",x1,x2);
     }
     Else
         Printf("no roat");
         Printf("%f,%f ",x1,x2);
  Getch( );
  Return(0);
  }
```

مثال6 ) برنامه ای بنویسید که اعداد 1 تا 3 را از ورودی دریافت کند و معادل حروفی آن را چاپ کند .

```
# incloude <stdio.h>
# incloude <conio.h>
  Main( )
  {
    Int n;
    Scanf("%d ",&n);
    If (n==1)
       Printf ("one");
    Else
    If (n==2)
```

```
        Printf("two");
      Else
      If (n==3)
        Printf("three");
      Else
        Printf("other number");
    Getch( );
     Return(0);
    }
```

**3-1 – دستور switch :** شکل کلی آن به صورت زیر است :

```
switch          متغییر
{
   case  1 :    1(ات)دستور ;
          break ;
   case  2 :    2(ات)دستور ;
        .
        .
   default  :    n (ات)دستور ;
}
```

**نکته**: در مواقعی که روی یک عبارت یا متغییر شرطهای مختلفی بررسی می شود به جای if های تودرتو از دستور switch استفاده می کنیم .

مثال7) مثال برنامه 4 را با استفاده از دستور switch بنویسید .

```
# incloude <stdio.h>
# incloude <conio.h>
   Main( )
   {
     Int n;
     Scanf("%d",&n)
```

```
Switch (n);
{
  Case 1: printf("\n one");
        Break;
  Case 2: printf("\n two");
        Break;
  Case 3: printf ("\n tree");
        Break;
   Default;
  Printf("\n other number");
 }
Getch( );
Return(0);
}
```

مثال8) را با استفاده از دستور6 مثال برنامه switch بنویسید .

```
# incloude <stdio.h>
# incloude <conio.h>
  Main( )
  {
    Int n ;
    Scanf("%d ",&n);
   Switch(n)
   {
     Case 1: printf("\n one");
          Break;
     Case 2: printf("\n two");
          Break;
```

```
        Case 3: printf("\n three");
              Break
             default;          به جایelse
         Printf ("\n other number");
        }
      Getch( );
      Return(0);
   }
```

**2- دستورات ایجاد حلقه :**برای تکرار یک یا چند دستور به کار می رود .

**2-1 – حلقه با تکرار معین :** اگر تعداد دفعات تکرار قبل از حلقه مشخص باشد از دستور زیر استفاده می کنیم :

for (متغییر شمارنده = مقداراولیه ; شرط اتمام حلقه ; گام حرکت )

{دستور ات ;

مثال :

```
for (i =1 ; i <= 3 ; i++ )
     printf (" \n %d " , i);
```

**نکته :** for(;;) یعنی حلقه بي نهایت و هیچ شرطی درونش نیست .

مثال 9 ) برنامه ای بنویسید که n نمره دانشجویی را خوانده و مشخص کند مشروط است یا نه ؟

```
# incloude <stdio.h>
# incloude <conio.h>
   Main( )
   {
     Int n,c;
     Float   m,s,avg;
     Printf("pleas enter mark num ");
     Scanf("%d \n ",&n);
```

```
S=0;
For (c=0;c<n;c+ +)
{
  Printf(" enter any mark :");
  Scanf("%f \n ", &m);
   S+ = m;  ⟶  s =s +m;
 }
   Avg = s/n ;
    If (avg<12)
        Printf("fail");
    Else
    Printf("suc");
  Getch( );
  Return(0);
}
```

مثال 10) برنامه ای بنویسید که فاکتوریل تعدادی از اعداد را چاپ کند . ( در آخر برنامه : reakb + ctrl استفاده می شود )

```
# include<stdio.h>
# include<conio.h>
Int main ( )
{
  Long  int fact ;
  Inti , num ;
  Clrscr ( );
  For (  ;  ;  )
  {
    Printf(" \n  type a number :");
    Scanf(%d" , &num );
```

```
        Fact=1;
        For (i=1; i<=num ; i++)
         Fact *=I ;
         Printf( " fact is :%d", fact);
      }
  Getch ( );
  Return (0);
}
```

مثال 11) برنامه ای بنویسید که مجموع سری زیر را حساب کند .

1+1/2+1/4+1/8+….


```
# include <stdio.h>
# include<conio.h>
Int main( )
{
   Int  count;
   Float  sum , x;
    Clrscr ( );
  For ( sum=0 , x=1 , count=1;count<=num; count++,x*
   {
     Sum +=1/x;
      Printf(" sum=%7.4 f , when count =%d \n" ,sum , count );
    }
   Return (0);
}
```

**2-2 – حلقه با تکرار معین ونا معین** : اگر تعداد دفعات تکرار از قبل معین نباشد حلقه های نوع زیر به کار می روند.

**1-2-2 – حلقه while** : شکل کلی آن به صورت زیر است :

```
while   ((شرط(ها))
      {دستور(ات} ;
```

```
i = 1 ;
while  (i <= 3)
  {   printf (" %d " , i) ;
      i ++ ; }
```

مثال 12 ) برنامه ای بنویسید که تعدادی عدد را خوانده و مجموع مربعات آنها را محاسبه کند و به همراه تعدادی عدد به خروجی ببرد .

```
# include<stdio.h>
# include<conio.h>
Main( )
{
  Intx , sum = 0,n = 0;
  Char ans= 'y';
  Clrscr( );
  While ( ans = = "y")
   {
     Printf(" \n enter a number:");
     Scanf("%d",&x);
Sum+ = x*x;
     N++ ;
     Printf("\n do you want to cuntiniue?(y/n):");
    Ans=getch( );
   } // end of while
```

```
  Printf("\n you entered %d num", n );
  Printf("\n sum of square is : %d", sum);
  Getch( );
  Return(0);
}
```

مثال 13) برنامه ای بنویسید که تعدادی عدد را خوانده و کوچکترین و بزرگترین آنها را محاسبه کند .

```
# include <stdio.h>
# include <conio.h>
Main( )
{
   Int  n , max , min ;
   Clrscr( );
                Printf("enter first number :");
  Scanf("%d \n",&n);
  Max=n;
  While(n!=0)
   {
       Printf("enter any number :");
       Scanf(%d \n ,&n);
     If (n>max)
       Max=n;
     Else
     If (n<min)
       Min=n;
  }
Printf("max=%d,min=%d",max,min);
Getch ( );
```

```
Return(0);
        }
```

**2-2-2 – حلقه do – while :** شکل کلی آن به صورت زیر است :

```
do                                  do
  {دستور(ات) ;                          { printf (" %d" , i )
while ((شرط(ها)) ;                          i++ ; }
                                    while ( i <= 3) ;
```

**نکته :** در do – while شرط , انتهای کار تست می شود بنابراین دستورات داخل حلقه حداقل یکبار انجام می شود.

مثال 14 ) برنا مه ای بنویسید که تعدادی عدد را خوانده و زوجها و فرد های آن را بشما رد .( شرط اتمام وارد کردن عدد صفر میبا شد ) .

```
# include <htdio.h>
# include <conio.h>
Main( )
{
   Int   no,oc,oc,c;
   Clrscr( )
   Oc=0,c=0;                    =     do
Printf("enter first no:");            {
Scanf("%d\n",&no);                     printf(" enter any no:");
  While (no!=0)                         scanf("%d \n" , &no);
   {                                    c++;
     C++;                                if((no %2)==0)
      If ((no%2)==0)                       oc++;
      Oc++;                            }
      Printf("enter any no:");              while(no!=0)
Scanf(" %d  \n ", &no);                oc--;
   }                                      c--;
```

```
    Oc = c - oc;
    Printf ("the oc= %d,the oc= %d  \n ,oc,oc);
  Getch( );
  Return(0);
}
```

## 2-3 – انتقال ها ی بدون شرط :

1-2-3 – break : خروج از بلاک موجود .

2-2-3 – continue : برگشت به بلاک موجود .

3-2-3 – go to : انتقال به برچسب معین شده .

## *چند نکته در برنامه نویسی به زبان C :

i. در زبان c توان نداریم یعنی : $2 *2 = 2^2$

ii. در آخر if , (;) نمی گذاریم .

iii. همه ی else ها if دارند ولی همه if ها else ندارند.

iv. clrscr() تابعی است برای پاک کردن اطلاعات قبلی بر روی صفحه مانیتور.

**مثال :** برنامه ای بنویسید که n عدد خوانده و بزرگترین آن را محاسبه کند .

```
#include < stdio.h >
#include < conio.h >
خروجی
main ()
 {
    int   n , max ;
    clrscr() ;
    printf ( " enter first num  \n ") ;
    scanf ( " %d " , & n ) ;
    max = n ;
    while ( n != 0 )
```

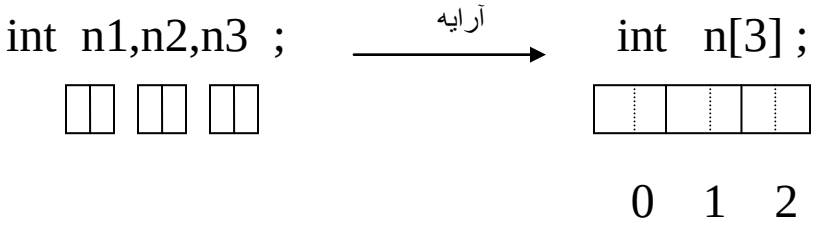
| n | max |
|---|---|
| 2 | 2 |
| 5 | 5 |
| 0 | 5 |

5

```
    {
        printf ( " enter any num \n " ) ;
        scanf ( " %d" , &n ) ;
   if ( n > max )
         max = n ;
     }
   printf ( " max = %d " , max );
   getch () ;
}
```

**آرایه ها و ماتریس :**

**آرایه** : مجموعه ای از عناصر هم نوع که به صورت پشت هم در حافظه ذخیره شده ولی دارای نام مشترکی می باشند , عناصر آن ها را با استفاده از شماره خانه (index) از یکدیگر متمایز می کنند .




**شکل کلی تعریف آرایه :**

نوع عناصر آرایه  نام آرایه [ تعداد عناصر] ;

**مثال :**  int  n [3] ;

شکل کلی دسترسی به عناصر آرایه :

نام آریه [ شاخص] ;

n [1] : = 27


❖ **نکته** : شاخص های آرایه در زبان C از صفر شروع می شوند .

**مثال15)** برنامه ای بنویسید که 10 نمره دانشجویی را گرفته و میانگین نمرات و خود نمرات را نمایش دهد.(حل توسط آرایه انجام شود .)

```
# include <stdio.h>
# include <conio.h>
Main( )
{
  Float  mark [10];
  Float  s , avg ;
  Int  I ;
 For(i=0 ; i<=10 ; i++)
  {
    Printf("enter mark %d : " , i+1);
    Scanf(" %f  \n " , & mark[i] );
    S+=mark[i];
  }// end of for
Avg= S /10;
Printf(" the avg = %5.2f ", avg );
For (I=0 ; I<=10 ; i++);
Prntf("the mark %d = %f " \n , i+1, mark [i] );
Getch( );
Return(0);
}
```

**مثال 16)** برنامه ای بنویسید که یک عضو را در یک آرایه ی n عضوی جست و جو کند .

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

```
Main ( )
{
  Const int n=10;
   Int no[n];
   Int x,I,f=0;
   For (i=0;i<n;i++);
       Scanf("%d \n|",&n0[i]);
   Printf("enter any no for serch:");
   Scanf("%d \n "&x);
   For (i=0;i<n;i++);
   If (x==no[i]);
   {
     If=1;
    }
     Break;
     If(f==1)
          Printf("find");
     Else
          Prinff("not found");

  Getch( );
  Return (0);
}
```

**مثال17 )** برنامه ای بنویسید که دو عدد بسیار یزرگ را با هم جمع کند.

```
# include <stdio.h>
# include <conio.h>
Main ( )
```

```
{
  Const int n=10;
   Int a[n],b[n],s[n];
   Int d,cary=10;
   For (i=0;i<10;i++)
   {
      Printf("enter %d digit no 1",i);
      Scanf ("%d \n ",&a[i]);
    }
    For (i=0;i<10;i++)
Printf("enter %d digit no2",i);
Scanf("%d \n",&b[n]);
For(i=9;i>=0;i--)
{
D=a[i]+b[i]+c;
If(d>=10)
{
S[i]=d-10;
C=1;
}
Else
{
S[i]=d;
C=0;
}
}
Printf("%d",c);
For(i=0;i<10;i++)
```

```
Printf("%d",s[i]);
Getch( );
Return( 0);
}
```

مثال18) برنامه ای بنویسید که عناصردو آرایه n عضوی را مشخص کند با هم برابرند یا نه ؟

```
# include <stdio.h>
# include <conio.h>
Main( )
{
  Const  int n ;
  Float  a[i],b[i];
  Intc     I,c;
  For(i=0;i<n;i++)
    {
      Printf("\n enter a[i],b[i]");
      Scanf("%f %f ",& a[i],& b[i] );
      If (a[i] = = b[i])
         C+ +;
     }
    If (c = = n)
       Printf(" \n yes ");
    Else
       Printf(" \n  No ");
       Getch( );
        Return (0) ;
}
```

مثال19 ) برنامه ای بنویسید که سری فیبوناچی را تولید و در یک آرایه نگهداری کند . ...و8 و 5 و3 و 2 و1 و1

```
# include <stdio.h>
# include <conio.h>
Main ( )
{
  Const
  Int n=20;
 Int m[n] , I;
 M[0]=1;
 M[1]=1
 Clrscr( );
 Printf("%d jomle az fibo :\n = = = = = = = = = \n\n",n);
 Scanf ("%d \n %d \n ", m[0],m[1]);
 For ( i=1; I,n ;i++)
 {
   M[i] =m[i-2]+m[i-1];
   Printf("%d \n ", m[i]);
 }
   Getch( );
   Return 0;
 }
```

مثال20 ) برنامه ای بنویسید که یک آرایه n عنصری را خوانده ومیانگین آنها و انحراف از میانگین را محاسبه کند .

```
# include <stdio.h>
# include <conio.h>
Main( )
{
   Const        int n ;
   Float  a[n],avg,s=0 ;
   Int I;
  Clrscr( );
  For (i=0;i<n;i++)
     Scanf ("%f",&a[i]);
       For (i=0;i<n,i++)
          S+ = a[i];
  Avg=s/n;
  For (i=0;i<n;i++)
     Printf("\n the contrast %f of avg is : a[i] – avg);
Getch( );
Return(0);
}
```

مثال21 ) برنامه ای بنویسید که 5 عدد را از ورودی خوانده سپس انها را به ترتیب معکوس در ارایه ی دیگری قرار داده و نتیجه را به خروجی ببرد .

```
# include <stdio.h>
# include <conio.h>
Main ( )
```

```
{
 Int x[5],y[5],I,j;
Clrscr ( );
For (i=0;i<5;i++)
Printf("enter number %d:",i);
Scanf( "%d",&x[i]);
}
J=0;
Printf(" num in inverse:\n");
For (i=4;i>=0;i--)
Y[j]=x[i];
Printf("%3d",y[j]);
J++;
}
Printf("\n  press okey to coniniue:");
Getch ( );
Return (0);
}
```

**تمرین :**

برنامه ای بنویسید که عناصر آرایه را از انتها به یک آرایه ی دیگر منتقل کند (برعکس شود).

**ماتریس :** آرایه دو بعدی را اصطلاحاً ماتریس گویند . که شکل کلی تعریف آن به صورت زیر می باشد :

نام ماتریس [تعدادسطر][تعدادستون] ;

مثال :

```
Int  m[2][3];
```

مثال22) - برنامه ای بنویسید که عناصر یک ماتریس 3*3 را از ورودی گرفته سپس مجموع عناصر قطر اصلی را محاسبه کند .

```
# include <stdio.h>
# include <conio.h>
Main ( )
{
   Int   m[3] [3];
   Int s=0,I,j;
   For(i=0;i<3;i++)
      For(j=0;j<3;j++)
         Scanf("%d \n",&m[i][j] );
   For (i=0;i<3;i++)
      For(j=0;j<3;j++)
         If (i==j)
           S+=m[i][j];
  Printf("sum=%d \n",s);
  Getch( );
   Return(0) ;
}
```

**مثال23** ) برنامه ای بنویسید که یک عدد از ورودی گرفته و تعداد دفعات تکرار آن را در یک ماتریس m×nچاپ کند

.

```
# include <stdio.h>
# include <conio.h>
Main ( )
{
  Const
   Int   n=4,m=3;
   Int  mat [n][m];
   Int   I,j,x,c=0;
```

```
  For(i=0;i<n;i++)
      Scanf("\n %d", &x);
      For(i=0;j<m;j++)
         For(j=0;j<m;j++)
           If (x=mat[i][j])
              C++;
   Printf("%d",c);
  Getch( );
   Return(0);
 }
```

مثال24 ) برنامه ای بنویسید که حاصلضرب دو ماتریس n×m را پیاده سازی کند .

```
# include <stdio.h>
# include <conio.h>
Main ( )
{
  Const
  Int n;
  Float n[i][j],m[i][j],p[i][j],s;
  Int m,I,j;
  For (i=0;i<n;i++)
      For(j=0;,j<n;j++)
       {
         Printf("enter n[%d][%d],m[%d][%d]",I,j,I,g);
         Scanf("%F%F",&n[i][j],&m[i][j]);
       }
  For (h=0;i<n;i++)
    For (j=0;j<n;j++)
```

```
        {
        S+=n[i][j]*m[m][j];
        Printf("the p[%d][%d] is : %f",I,j,s);
       }
  Getch( );
   Retutn (0) ;
}
```

مثال25) برنامه ای بنویسید که عناصر یک ماتریس n×m را گرفته و تعداد اعداد زوج و فرد آن ها را مشخص کند .

```
# include <stdio.h>
# include <conio.h>
Main ( )
{
Const  int n,m ;
Int   m[n],[m],I,j ,OC,OC ;
For (i=0 ; i<n ; i++);
For (j=0 ; j<n ; j++);
{
Printf("\n enter no[%d][%d]:",I,j);
Scanf("%d",&m[i][j]);
}
For (i=0;i<n;i++)
    For (j=0;j<n;j++)
      If((m[i][j]%2)= =0)
       Oc++;
       Oc=n*m-oc;
Printf("the odd counter is :%d and the even counter is 5D " , OC ,OC );
Getch ( );
Return(0);
```

}

**تمرین :**

برنامه ای بنویسید که دو عدد را از ورودی خوانده و در یک ماتریس n×m اولی را با دومی جابجا کند.

**رشته (string) :**

آرایه ای از کاراکترها را رشته گویند. char name[10] ;

نکته : برای خواندن رشته از صفحه کلید :

scanf ( " % s " , name ) ; یا gets (name) ;

برای چاپ کردن رشته :

printf ( " % s " , name ) ; یا puts (name) ;

نکته : برای خواندن رشته ها اگر حروف کمتر از جای خالی بود از علامت \0 در انتهای رشته استفاده می کنیم .

**زیر برنامه ها ( sub program ) :**

اساس برنامه سازی ساخت یافته زیر برنامه ها می باشند .

یک برنامه را ابتدا تجزیه و تحلیل کرده و در صورت نیاز آن را به زیر برنامه های کوچکتری تقسیم می کنیم , سپس زیر برنامه ها را فرخوانی می کنیم .

**مراحل استفاده**:

**1- تعریف و پیاده سازی :** برای این مرحله باید توابع را با کلیه عملیات مربوط به آن پیاده سازی کنیم . یعنی همانند یک برنامه کلیه متغییر ها , دستورات , و نیز ورودی ها و خروجی ها را تعیین می کنیم .

**2- فرا خوانی** (call) : صدا کردن یک زیر برنامه( تا زمانی که فرا خوانی نگردد اجرا نخواهد شد) .

ـ توجه : در هر بنامه یک زیر برنامه اصلی و در صورت نیاز چندین زیر برنامه فرعی خواهیم داشت که اجرای عملیات از زیر برنامه اصلی شروع خواهد شد .

**شکل کلی تعریف و پیاده سازی زیربرنامه :**

(اسامی و نوع پارامتر ورودی) نام زیربرنامه نوع خروجی زیر برنامه

```
{
    تعریف متغییرهای محلی ;
        دستورات ;
    return متغییر خروجی ;
}
```

**شکل کلی فراخوانی زیربرنامه :**

نام زیربرنامه (اسامی پارامترهای ورودی) ;

ـ توجــه : تعداد پارامترهای مجازی با تعداد پارامترهای واقعی باید یکسان باشد .

ـ در زبان C به زیر برنامه ها تابع گویند .

ـ اگر یک تابع خروجی نداشته باشد نوع آن را void تعریف می کنیم .

مثال26 ) برنامه ای بنویسید که دوعدد ازورودی گرفته با استفاده از زیر برنامه ها توان آن ها را محاسبه و چاپ کند.
(توان :عدد اول به توان عدد دوم )

```
# include <stdio.h>
# include <conio.h>
  Int power (int a ,int b);
{
Int i,p;
P=1;
For (i=1;i<=b;i++);
P=p*a;
Return p;
}
Int main ( )
{
 Int n1,n2,n3;
Scanf("%d %d \n ",&n1,&n2);
N3=power(n1,n2);
Printf( "% \n ",n3);
Return 0;
}
```

مثال27 ) برنامه ای بنویسید که حاصل عبارت زیر را محاسبه کند.

**C = $n1^{n2} / n1 - n2^{n1} / n2$**

```
# include <stdio.h>
# include <conio.h>
Int power(inta,intb )
{
  Int I,p;
  P=1;
  For (i=1;i<=n;i++);
  P=p*a;
 Return p;
}
Int main( )
{
   Int  n1,n2;
   Float  n3,x,y ;
  Scanf("%d %d \n ", &n1,&n2);
  X=poer(n1,n2)/n1;
  Y=power(n2,n1)/n2;
  N3=x-y;
  Printf("%f ", n3);
  Return(0);
}
```

مثال28 ) برنامه ای بنویسید که حاصل عبارت روبرو را محاسبه نماید .

$C = n1^{n2} / n1 - n2^{n1} / n2$

```
# include <stdio.h>
# include <conio.h>
Int power(inta,intb )
{
  Int I,p;
  P=1;
  For (i=1;i<=b;i++);
  P=p*a;
 Return p;
}
Int fact (inta)
  {
     Int I,p;
     F=1;
     For (i=a;i>=1;i--)
     F = f * I;
     Return  f ;
  }
 Int main( )
{
   Int a,b ;
```

```
   Float c ;
   Clrscr ( );
   Scanf(" %d %d \n ",&a,&b);
   If (a<0 || b<0)
 {
  Printf(invalid);
  Exit 0;
 }
C= ( power (a,b)/fact( c ) ) – ( power (b,a)/ fact( c ) );
Printf("%f \n ",c);
Return (0);
}
```


مثال29 ) برنامه مثال بالا را با استفاده ازvoid بنویسید .

```
# include <stdio.h>
 # include <conio.h>
   Void power (int a,int b)
       {
         Int I,p;
         P=1;
         For( i=1;i<=b;i++)
             P=p*a;
         Printf( "%d",p );
        }
     Int main( )
       {
         Int a,b;
         Scanf("%d %d \n ",&a,&b);
```

```
    Power(a,b);
    Return 0;
}
```

**انواع متغییر ها از نظر اعتبار :**

**1- محلی(local) :**این نوع از متغییر ها را در داخل زیر برنامه ها تعریف می کنند و فقط در همان زیر برنامه ها اعتبار دارند .

**2- سراسری** (global) : این نوع از متغییرها را در ابتدای برنامه و قبل از همه توابع تعریف و پیاده سازی می کنند که امکان استفاده از آن ها در کلیه زیر برنامه ها وجود دارد.

استفاده از متغییر های سراسری :

دارای اثرات جانبی می باشد و به همین دلیل تا حد امکان از آن ها استفاده نمی کنیم .

اثرات جانبی بدین معنی است که اگر مقدار متغییر سراسری به نادرستی تغییر کند بخش های بعدی برنامه در صورت استفاده از آن نتایجش اشتباه خواهد بود .

**زیربرنامه های بازگشتی (recursive) :**

زیر برنامه هایی را گویند که خود را فراخوانی می کنند.

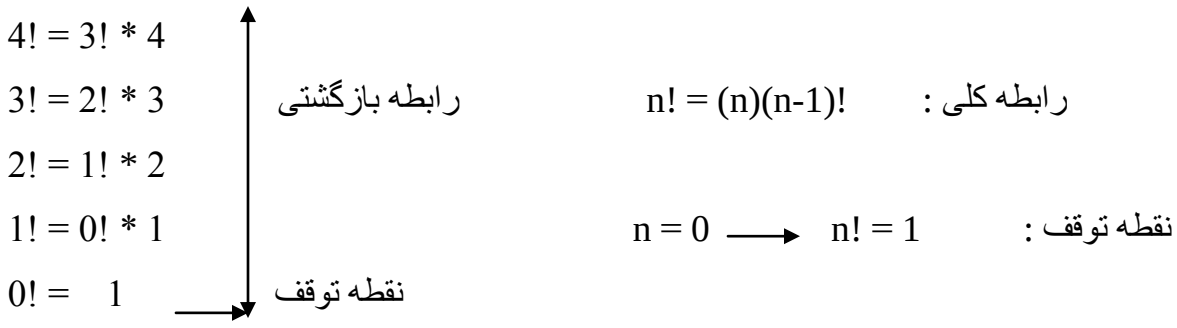
مسایلی را به صورت بازگشتی پیاده سازی می کنند که یا ماهیت بازگشتی داشته باشتند مانند فاکتوریل و فیبوناچی یا از ساختمان داده های غیر خطی استفاده می نماید.

ـ برای نوشتن یک تابع بازگشتی ابتدا مسأله را بامثال های عددی بررسی می کنیم سپس دو رابطه زیر را روی آن ها تعیین می کنیم .

1- **رابطه بازگشتی** : رابطه ای که طبق آن زیر برنامه خود را فراخوانی می کند .

2- **نقطه توقف :** مرحله که فراخوانی در آن توقف می کند .

مثال : بدست آوردن فاکتوریل عدد n :



پس از بدست آوردن دو رابطه مورد نظر کافیست با استفاده از دستورif – else زیر برنامه را پیاده سازی کنیم .

```
int   fact (int n)
 {
    if (n == 0)
        return 1 ;
    else
         return   n*fact(n-1) ;
  }
```

مثال : فرض کنید صرفاً امکان جمع و تفریق با یک وجود دارد زیر برنامه بازگشتی بنویسید که دو عدد صحیح مثبت را با یکدیگر جمع کند :

فرض : 5+3 = ?

s (5,3) = 1 + s (5 , 3-1)

s (5,2) = 1 + s (5 , 2-1)

s (5,1) = 1 + s (5, 1-1)

s (5,0) =1 : نقطه توقف

```
int  s (int  n , int  m );
 {
   if  ( m==0)
         return  m ;
    else
         return (1+ s(n , m-1)) ;
  }
```

**توجـه** : اگر زیر برنامه ها را بعد از main بنویسیم error می دهد و باید پیش از تعریف بیاوریم.

مثال30) برنامه ای به صورت بازگشتی بنویسید که سری فیبوناچی را چاپ کند .

```
# include <stdio.h>
# include <conio.h>
 Int fibo(int);
Int main ( )
{
  Int n,I ;
  Printf("enter any no \n");
  Scanf( "%d",&n);
  For (i=0;i<n;i++)
```

```
 Printf("\n %d , fibo(i));
 Getch ( );
 Return 0 ;
}
Int fibo(int n);
  {
    If (n<=2)
    Return(1);
    Return(fibo(n-1)+fibo(n-2));
  }
```

**ساختمان یا رکورد (structure) :**

مجموعه ای از داده ها می باشند که نوع آنها الزاماً یکسان نبوده و دارای نام مشترکی می باشند .

برای پیاده سازی نرم افزارهای کاربردی ابتدا باید موجودیت های سیستم مورد نظر را شناسایی کنیم , سپس هر کدام از موجودیت ها را با استفاده از یک ساختمان پیاده سازی می کنیم .

به اشخاص , اشیا , و پدیده های موجود در هر سیستم عملیاتی اصطلاحاً موجودیت گویند . مانند : اعضا و کتاب در سیستم کتابخانه .

شکل کلی تعریف نوع رکوردی :

```
struct   نام نوع ساختار {
                          نوع      نام فیلد 1 ;
                          ↕
                          نوع      نام فیلدn ;
                         } ;
```

مثال :

```
struct   book {
              char   title[30] ;
              char   writer[10] ;
```

```
        int     id ;
         }
```

* پس از تعریف نوع ساختار باید متغییر هایی از آن نوع را تعریف نماییم .

الف )* تعریف متغییر ها پس از معرفی نوع ساختار. مثلاً در مثال بالا بعد از { :

```
} b1,b2,b[10] ;
```

ب ) تعریف در بخش متغییر ها .

```
struct   book   b1,b2,b[10] ;
```

شکل کلی دسترسی به فیلدهای ساختاری :

; نام متغییر . نام متغییر ساختاری

```
b1 . title = "abc" ;
```

| | b1 | b2 |
|---|---|---|
| title | abc | |
| writer | | |
| id | | |

**نکات :**

1- اگر بخواهیم همزمان با تعریف متغییر های ساختاری , به ان ها مقدار بدهیم به صورت زیر عمل می کنیم .

```
srtuct   book   b3 = {"xyz","abc",554};
```

2- اگر متغییرهای ساختاری هم نوع داشته باشیم می توانیم آن ها را به هم انتساب دهیم .

```
b1 = b2 ;              b2 = b[10] ;
```

مثال : برنامه ای بنویسید که اطلاعات 10 کتاب را از ورودی گرفته سپس شماره کتابی را خوانده و مشخصات کامل آن را چاپ کند.

```
#include <stdio.h>
#include <conio.h>
#define   n   10
int main()
{
  struct  book{
```

```
                char   title[15] ;
                char   writer[10] ;
                  int      id ;
                }b1 , b[n] ;
  int   x,i,j ;
  for (i=0 ; i<n ; i++ )
     {
       printf (" enter title\writer\id : \n") ;
       scanf ("%s%s%d",b[i].title , b[i].writer , &b[i].id ) ;
      }
 printf (" enter num for search \n") ;
 scanf ("%d" , &x);

  for (i=0 ; i < n ; i++ )
      if (x = = b[i].id)
          {j=i;
            break ;
           }
printf (" the title and writer is: %s%s " ,b[i].title,b[i].writer) ;
getch();
return 0 ;
}
```

مثال : برنامه ای بنویسید که در یک سیستم کتاب فروشی کلیه کتاب هایی که قیمت آن ها زیر x ریال می باشد را گزارش کند :

```
#include < stdio.h >
#include < conio.h >
int main ()
```

```
{
  const   int    n=10 ;
  struct   book {
                    char    title[15] ;
                    char    writer[15];
                     int     id , price ;
                  } b[n] ;
 int   x , i ,j ;
for (i=0 ; i < n ; i++ )
  scanf ("%s%s%d%d" , b[i].title , b[i].writer , &b[i].id , &b[i].price ) ;
printf (" enter price for search \n") ;
scanf ("%d",&x) ;
  for (i=0 ; i < n ; i++ )
       if (b[i].price <= x)
            printf (" %d" , b[i].id) ;
getch() ;
return 0 ;
   }
```

**اشاره گر ها(pointer) :**

ویژگی متغییر : 1- نام  2- ادرس  3- محتویات  4- نوع

```
int  x ;                              int   x ;
x = 55 ;                              int  *p ;
printf ("%d",x) ;  : 55               x = 55 ;
p = &x ;                    آدرس p را در x بریز
printf ("%d" , x) ;   : 55
                  یا
printf ("%p",*p ) ;    : 55
```

**اشاره گر** : متغییری است که در آن آدرس متغییر دیگر را می توان نگهداری کرد .

**شکل کلی تعریف اشاره گر :**

```
Type   *var name;
int     *p ;                                    یعنیp  اشاره گر است
float    *q ;
char     * r ;
```

**نکات :**

1- در نام رشته و آرایه , آدرس شروع رشته و آرایه ذخیره می شود . از این رو می توانیم با استفاده از اشاره گر هابه آن ها دسترسی یابیم .

2- در پیاده سازی زیر برنامه ها هنگام فراخوانی می توان از دو روش استفاده نمود .

## فراخوانی توابع:

الف- فراخوانی توسط مقدار (call by value ) :

در این روش مقدار پارامتر حقیقی در پارامتر مجازی کپی می شود و پس از فراخوانی پارامترهای حقیقی و مجازی مستقل از هم عمل می کنند .

ب – فراخوانی توسط ارجاع یا آدرس ( call by refrence ) :

در این روش آدرس پارامترهای حقیقي در پارامتر های مجازی کپی می شود یا به عبارت دیگر پارامترهای مجازی اشاره گر به پارامتر های حقیقی می باشند , از این رو هرگونه تغییر در پارامترهای حقیقی عیناً در پارامترهای مجازی تأثیر گذاشته و بالعکس .

## پرونده یا فایل (File ) :

درکلیه برنامه هایی که قبلاً پیاده سازی می کردیم داده ها درون متغییر ها ذخیره می شوند و متغییر ها در حافظه اصلی (Ram ) نگهداری می شوند و پس از خروج از هر برنامه محتویات متغییر ها از بین می رود که برای رفع این مشکل از file استفاده می کنیم .

**فایل :** مجموعه ای از داده های مرتبط به هم را گویند که در حافظه جانبی ذخیره شده و یک نام به آن مجموعه نسبت داده می شود .

**انواع فایل ازنظر دسترسی :**

**1- فایلهای با دسترسی مستقیم یا تصادفی :** در این نوع فایل ها نحوه دسترسی به داده ها مستقل از ترتیب ذخیره سازی آن ها می باشد و می توان به هرکدام از داده ها به طور مستقیم دسترسی داشت . مثلاًدر **فایل** دیکشنری برای دیدن کلمه ای که با B شروع می شود دیگر کلماتی را که با حرف A شروع می شود را نمی بیند.

2- **فایلهای با دسترسی ترتیبی** : در این نوع از فایل ها دسترسی به داده ها به همان ترتیبی است که ذخیره شده است یعنی برای دسترسی به داده n ام باید از n-1 داده عبور کنیم . مثلاً اگر بخواهیم دقیقه دوم یک موزیک را گوش کنیم باید از دقیقه اول آن عبور کنیم .

**انواع فایل از نظر محتویات :**

1- **فایل های متنی (text) :** در این نوع از فایل ها داده ها به همان شکلی که نوشته می شوند ذخیره می گردند .

**2- فایل های دودویی یا باینری :** در این نوع از فایل ها داده ها به صورت دودویی تبدیل شده و ذخیره می گردند .

برای استفاده از فایل ها در هر زبان برنامه سازی مجموعه ای از عملیات باید انجام پذیرد که عبارتند از :

**1- تعریف متغییر فایلی :** هر فایلی دارای دو اسم می باشد که یکی نام کامل در سیستم عامل و دیگری نام مستعار در زبان برنامه نویسی که به این نام مستعار در اصطلاح متغییر فایل گویند .

**شکل کلی تعریف متغیر فایلی :** (در قسمت متغیر ها تعریف می شود.)

FILE (حروف باید بزرگ باشد.) * (نام متغیر فایلی (نام داخلی) ;

مثال : FILE *f1 ;

2- **باز کردن فایل :** باید نام خارجی و نام داخلی فایل را به یکدیگر نگاشت نموده و نیز نوع فایل را مشخص کنیم , همچنین تعیین کنیم که فایل را برای چه عملی می خواهیم باز کنیم که کل این کار توسط دستور fopen انجام می پذیرد.

**شکل کلی دستور** :

; ( " نوع فایل و حالت باز کردن " ," مسیر و نام خارجی فایل " ) fopen = نام متغییر فایلی

**حالتهای مختلف باز کردن فایل :**

1- باز کردن فایل متنی برای خواندن read text r (rt)

2- باز کردن فایل متنی برای نوشتن write text w (wt)

3- باز کردن فایل متنی برای اضافه کردن append text a (at)

4- باز کردن فایل دودویی برای خواندن rb (rbt)

5- باز کردن فایل دودویی برای نوشتن wb (wbt)

6- باز کردن فایل دودویی برای اضافه کردن ab (abt)

7- فایل متنی را هم برای خواندن و هم برای نوشتن باز مي کند . r+

8- فایل متنی را هم برای نوشتن و هم برای خواندن باز می کند . w+

9- فایل متنی را برای اضافه کردن و خواندن و نوشتن باز می کند . a+

10- فایل دودویی را برای خواندن و هم نوشتن باز می کند . r+b

11- فایی دودویی را برای نوشتن و هم خواندن باز می کند . w+b

12- فایل دودویی را برای اضافه کردن و خواندن و نوشتن باز می کند. a + b

**مثال :** f1= fopen ("D: example .dat" , "w") ;

توجه:دستور باز کردن فایل f1 را نوشتیم اگر f1 موجود باشد اطلاعات جدید را می ریزد درون f1 که اطلاعاتی از قبل درونش بود و اگر f1 موجود نباشد f1 جدیدی را می سازد.

***نکته :**

پس از آنکه فایل مورد نظر باز شد باید بررسی کنیم که این کار موفقیت آمیز بوده یا نه . برای امتحان کردن اینکه f1 باز شده است یا نه دستور زیر را در برنامه می نویسیم :

```
if (f1== NULL)
        {
          printf (" file not opened ") ;
           exit (0) ;
          }
```

**3- نوشتن در فایل یا خواندن در آن :**

متناسب با محتویات (جنس , نوع )  اده می کنیم متفاوت است.

الف – نوشتن و خواندن کاراکتر : شکل کلی دستور به شکل زیر است

نوشتن : putc (نام متغییر , نام داخلی فایل ) ;

این دستور موجب می شود محتویات ch درون فایلf1 ذخیره گردد  مثال : putc (ch , f1) ;

خواندن: نام متغیر = getc(نام داخلی فایل ) ;

این دستور موجب می شود که یک کاراکتر f1 خوانده شده ودر متغییر ch ذخیره گردد  مثال : ch = getc(f1) ;

از فایل

ب- نوشتن و خواندن رشته : شکل کلی دستور به صورت زیر است :

نوشتن: fputs ( نام متغییر رشته ای , نام داخلی فایل ) ;

مثال : fputs (s1 , f1 ) ;

خواندن : fgets (نام متغییر رشته ای , طول رشته , نام داخلی فایل ) ;

مثال : fgets (s1 , 10 , f1 ) ;


## 4- بستن فایل :

پس از آنکه عملیات مورد نظر بر روی فایل به اتمام رسید آن را می بندیم . که شکل کلی دستور به شکل زیر است:

fclose (نام داخلی فایل) ;

**دلایل بستن فایل :**

- ✓ تعداد فایل هایی که یک سیستم عامل همزمان می تواند باز کند محدود است .
- ✓ افزودن علامت مخصوص به انتهای فایل , فایلی که می بندیم به ویژه فایل متنی یک علامتی( مثل( .)) انتهای متن می آید که اگر دفعه بعدی فایل را باز کردیم برای پیدا کردن انتهای متن دنبال آن علامت بگردیم .
- ✓ سرعت دسترسی ما به حافظه جانبی بسیار پایین است

*** نکته :**

برای تست انتهای فایل از تابع feof استفاده می کنیم .

* خواندن و نوشتن structure در فایل :

نوشتن: fwrite (نام متغییر ساختاری , sizeof( نوع متغییر ساختاری ), تعداد, نام داخلی فایل ) ;

خواندن:  ; ( نام داخلی فایل, تعداد, (نوع متغییر ساختاری )sizeof , نام متغییر ساختاری) fread

مثال 31) برنامه ای بنویسید که مشخصا ت پرسنل یک شرکت را گرفته سپس آنها را در فایلی نگهداری کند سپس از آن لیست حقوق کارمندان را محاسبه و چاپ کند .

```
# include <stdio.h>  ⟶  # include <conio.h>
# include <stdlib.h>  ⟶  # include <string.h>
int main( )
{
File *f;
Char numstr[10];
Int I,salary;
Struct  em
        {
           Char name[10];
            Int hp;
            Int h;
        }
Emp;
Clrscr( );
F=fopen("employ.dat" ,"wb+");
If (!f)
{
Printf("can not open fill");
Exit 0;
}
Printf("name hour pay , hour");
```

**S = 0**
**G = 0**

```
        e(y)
{
Gets(em.name);  =  scanf("%d",&emp,name);
If(!emp.name);
Break;
Gets(num str );
Emp.hp=atio(numstr);
Gets(num str );
Emp.h=atio(num str);
Fwrite(&emp,sizeof(structem),I,f);
{
  Rewrute(f);
   Clrscr ( );
 Puts(name salary):
FREAD(&emp,size of (struct em),I,f);
While (!feof(f))
{
Puts(ep.name);
Printf(%d \n memp.hp*emp.h);
Fread(&emp , size of (structem), I,f );
}
Close (f);
Getch( );
Return 0;
}
```