ساختارهای تکرار

حلقه for

- معمولاً در برنامه نویسی حالتی پیش می‌آید که میخواهیم کاری را به تعداد دفعات مشخص انجام دهیم

- از حلقه for معمولاً برای تکرار یک عملیات به تعداد مشخص استفاده میشود.

ساختار حلقه for

- ساختار حلقه for به شکل زیر میباشد:

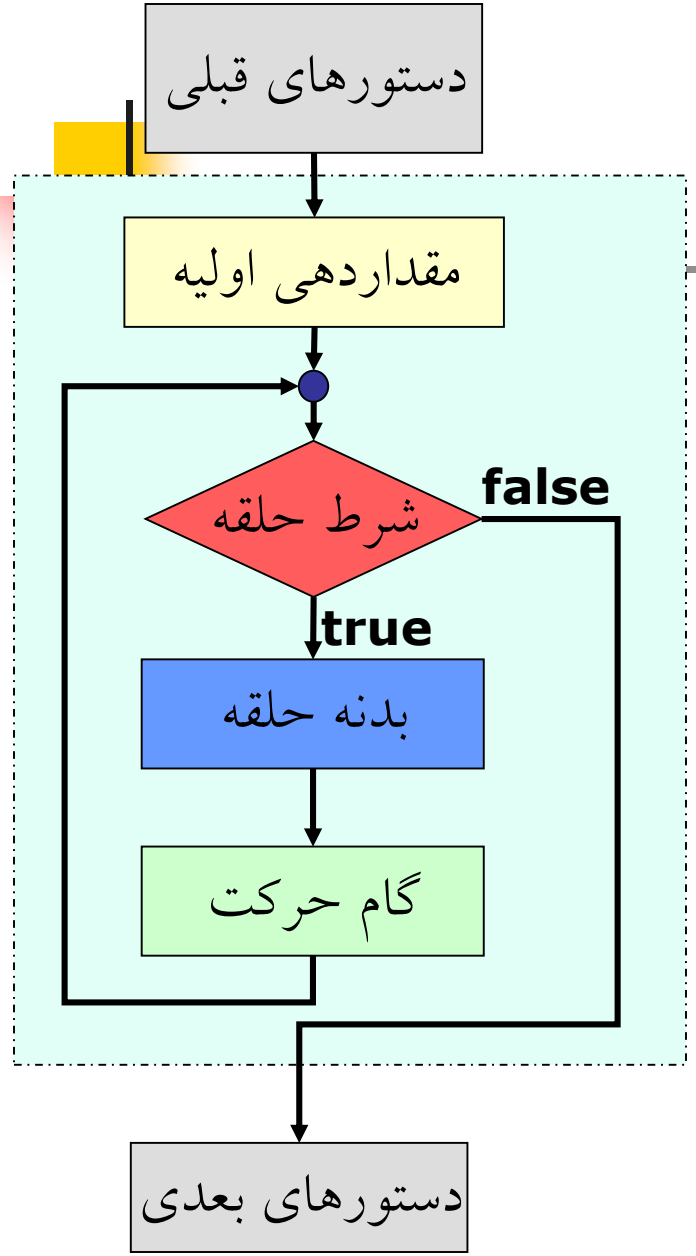
for (<مقداردهی اولیه> ; <شرط تکرار> ; <گام حرکت>)

{

<دستورات حلقه>

}

فلوچارت مربوط به حلقه for



for (<مقداردهی اولیه> ; <شرط تکرار> ; <گام حرکت>)

{

<دستورات حلقه>

}

مثال:

برنامه ای بنویسید که ۱۰ عدد از کاربر گرفته و حاصل جمع را نمایش دهد

```
int main()
{
    int i,num,sum=0;
    for ( i=1 ; i <= 10 ; i++)
    {
        printf( "\nEnter number %d: " , i);
        scanf( "%d" , &num);
        sum += num;
    }
    printf( "\nsum is: %d" , sum);
    getch();
    return 0;
}
```

- در عناصر شروع و گام حرکتی مربوط به حلقه میتوان چند دستور نوشت

```
int main()
{
    int i,num,sum;
    for ( i=1 , sum=0 ; i <= 10 ; i++)
    {
        printf( "\nEnter number %d: " , i);
        scanf( "%d" , &num);
        sum += num;
    }
    printf( "\nsum is: %d" , sum);
    getch();
    return 0;
}
```

مثال: برنامه ای بنویسید که کاراکترهای کد اسکی از ۳۲ تا ۱۰۰ را در خروجی نمایش دهد

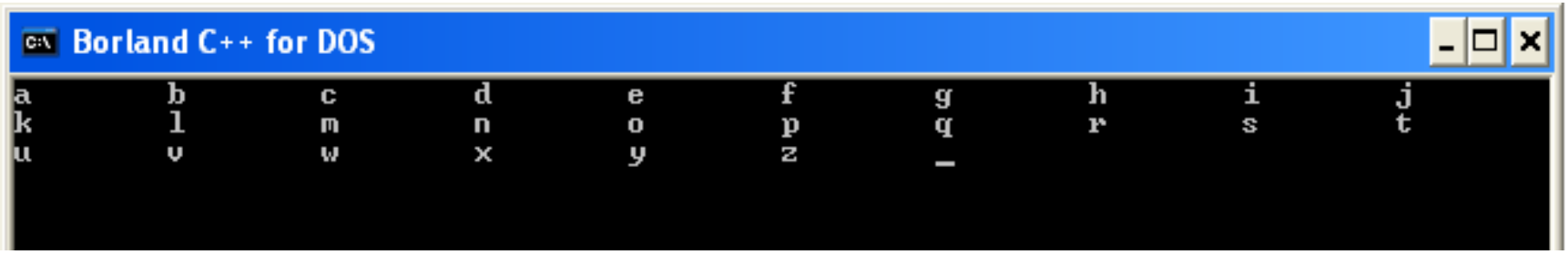
```
int main()
{
    char i;
    for ( i=32 ; i <= 100 ; i++)
    {
        printf( "%3d = %c \t " , i , i);
    }
    getch();
    return 0;
}
```

Borland C++ for DOS

```
 32 =          33 = !        34 = "        35 = #        36 = $
 37 = %        38 = &        39 = '        40 = (        41 = )
 42 = *        43 = +        44 = ,        45 = -        46 = .
 47 = /        48 = 0        49 = 1        50 = 2        51 = 3
 52 = 4        53 = 5        54 = 6        55 = 7        56 = 8
 57 = 9        58 = :        59 = ;        60 = <        61 = =
 62 = >        63 = ?        64 = @        65 = A        66 = B
 67 = C        68 = D        69 = E        70 = F        71 = G
 72 = H        73 = I        74 = J        75 = K        76 = L
 77 = M        78 = N        79 = O        80 = P        81 = Q
 82 = R        83 = S        84 = T        85 = U        86 = V
 87 = W        88 = X        89 = Y        90 = Z        91 = [
 92 = \        93 = ]        94 = ^        95 = _        96 = `
 97 = a        98 = b        99 = c       100 = d
```

مثال: برنامه ای بنویسید که کاراکترهای بین ‘a’ و ‘z’ را چاپ کند

```
int main()
{
    char i;
    for ( i=‘a’ ; i <= ‘z’; i++)
    {
        printf( "%c \t " , i );
    }
    getch();
    return 0;
}
```



مثال: برنامه ای بنویسید با استفاده از حلقه for جدول ضرب ۱ تا ۱۰ را نمایش دهد

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

مثال: برنامه ای بنویسید با استفاده از حلقه for جدول ضرب ۱ تا ۱۰ را نمایش دهد

```
int main()
{
    int row , col;
    for ( row=1 ; row <= 10 ; row++)
    {
        for (col=1 ; col <= 10 ; col++)
        {
            printf("%5d", row*col);
        }
        printf("\n");
    }
    getch();
    return 0;
}
```

Borland C++ for DOS

```
    1    2    3    4    5    6    7    8    9   10
    2    4    6    8   10   12   14   16   18   20
    3    6    9   12   15   18   21   24   27   30
    4    8   12   16   20   24   28   32   36   40
    5   10   15   20   25   30   35   40   45   50
    6   12   18   24   30   36   42   48   54   60
    7   14   21   28   35   42   49   56   63   70
    8   16   24   32   40   48   56   64   72   80
    9   18   27   36   45   54   63   72   81   90
   10   20   30   40   50   60   70   80   90  100
```

مثال: برنامه ای بنویسید که یک عدد را گرفته و تمامی اعداد زوج کوچکتر از آنرا چاپ کند

```
int main()
{
    int n , i;
    printf(" Enter a number:");
    scanf("%d" , &n );
    for ( i=2 ; i < n ; i+=2)
    {
        printf("\n%d", i);
    }
    getch();
    return 0;
}
```

```
Enter a number:  7

2
4
6
```

حلقه while

- معمولاً در برنامه نویسی حالتی پیش می‌آید که میخواهیم تا زمانی که یک شرط برقرار است برنامه یک عملیات خاص را تکرار کند

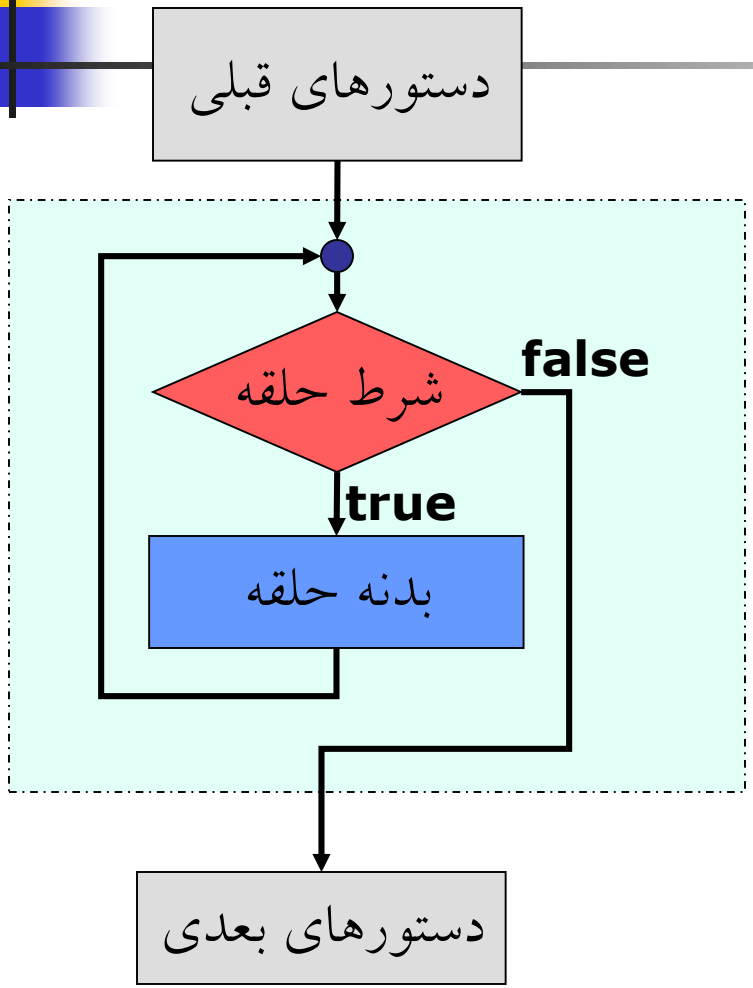
- معمولاً تعداد تکرارها در هنگام استفاده از حلقه while مشخص نمیباشد.

ساختار حلقه while

- ساختار حلقه while به شکل زیر میباشد:

```
while (<شرط حلقه>)
{
<دستورات حلقه>
}
```

فلوچارت مربوط به حلقه while



نکته مهم مربوط به حلقه while

- بدنه حلقه باید حاوی دستوراتی باشد که شرط حلقه را تغییر میدهند.
- در غیر این صورت، حلقه هیچوقت تمام نمی شود.

مثال:

- برنامه ای بنویسید که یک خط متن از کاربر بگیرد. و تعداد کاراکترها و کلمات را در خروجی نمایش دهد.
- تا زمانی که کاربر کلید Enter را نزده است یعنی کد کاراکتر ورودی مخالف ۱۳ ویا '\r' است عملیات گرفتن کاراکتر را ادامه میدهد.

```
int main()
{
    char ch;
    int words = 0 , characters = 0;
    while ( (ch = getche() ) != '\r' )
    {
        characters ++;
        if ( ch == 32 )
            words ++;
    }
    words ++;
    printf( " \nWords = %d  " , words );
    printf( " \nCharacters = %d  " , characters );
    getch();
    return 0;
}
```

```
I am a software programmer↵
Words = 5
Characters = 26
```

مثال:

- برنامه ای بنویسید که یک کاراکتر از کاربر گرفته و کد اسکی آنرا نمایش دهد تا زمانی که کاراکتر Escapeفشرده شود.

- حل:
  - کد اسکی مربوط به کلید Escape عدد ۲۷ است

```
int main()
{
    char ch;
    clrscr ();
    printf( " \nEnter a character:  " );
    while ( (ch = getche() ) != 27 )
    {
        printf( " \nCharacter = %c  " , ch);
        printf( " \nCode = %d  " , ch);
        printf( " \nEnter new character:  " );
    }
    return 0;
}
```

```
Enter a Character:  a
Character = a
Code = 97
Enter new character:  c
Character = c
Code = 99
Enter new character:  1
Character = 1
Code = 49
Enter new character:  ←
```

مثال:

- برنامه ای بنویسید که یک عدد از کاربر گرفته و فاکتوریل آنرا در خروجی نمایش دهد.

- $n! = n * (n-1) * (n-2) * \ldots * 2 * 1$
- $0! = 1$

با استفاده از حلقه for

```
int main()
{
   int i , n , factorial=1;
   clrscr ();
   printf( " \nEnter a number>=0:  " );
   scanf( "%d" , &n);
   for (i = 1  ; i <= n ; i ++ )
   {
         factorial *= i;
   }
   printf( " \n%d!=%d  " , n , factorial);
   getch();
   return 0;
}
```

```
Enter a number>0:  5

5!=120
```

با استفاده از حلقه while

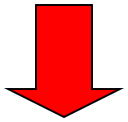
```
int main()
{
    int i , n , factorial;
    clrscr ();
    printf( " \nEnter a number>=0:  " );
    scanf( "%d" , &n);
    factorial = 1 ;
    i = 1 ;
    while ( i <= n )
    {
        factorial *= i++;
    }
    printf( " \n%d!=%d  " , n , factorial);
    getch();
    return 0;
}
```

```
Enter a number>0:  5

5!=120
```

تبدیل حلقه for به حلقه while

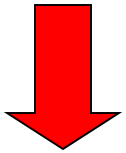
```
for ( <Expr 1> ; <Expr 2> ; <Expr 3> )
{
        <Expr 4>
}
```



```
<Expr 1>
while ( <Expr 2> )
{
        <Expr 4>
        <Expr 3>
}
```

تبدیل حلقه while به حلقه for

```
while ( <Expr 1> )
{
        <Expr 2>
}
```



```
for ( ; <Expr 1> ; )
{
        <Expr 2>
}
```

For OR while

- حلقه های for و while قابل تبدیل به هم هستند
- استفاده از حلقه for:
  - اگر تعداد دفعات تکرار یک حلقه معلوم باشد
  - و اگر بخواهیم دنباله ای از اعداد تولید کنیم بهتر است از حلقه for استفاده کنیم
- استفاده از حلقه while:
  - در حالتی که خاتمه یافتن حلقه باید توسط ورودی تعیین شود بهتر است از حلقه while استفاده شود

حلقه do-while

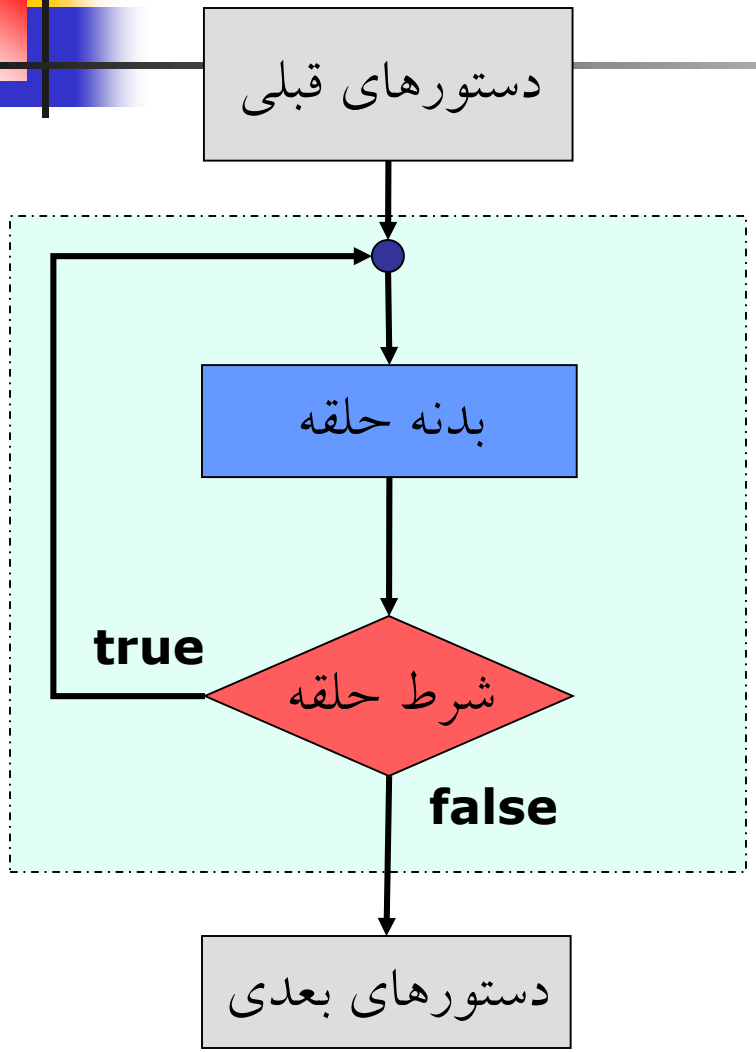
- ساختار کنترلی do while نیز برای پیاده سازی تکرار استفاده می گردد.
- در این حالت شرط خاتمه حلقه در آخر آن قرار دارد.
- **لذا، بدنه حلقه حداقل یکبار اجرا می شود.**

ساختار حلقه do-while

- ساختار حلقه do-while به شکل زیر میباشد:

```
do
{
<دستورات حلقه>
} while (<شرط خاتمه>);
```

فلوچارت مربوط به حلقه do-while



do example

```
int i=1;
do
   printf( "i is %d \n" , i++);
while (i <= 10);
```

دستورات break و continue

- این دو دستور در هر یک از حلقه های سه گانه معرفی شده میتوانند مورد استفاده قرار گیرند
- به محض اینکه دستور break اجراشد، کنترل برنامه از حلقه خارج میشود.
- زمانی که دستور continue در بدنه حلقه اجرا میشود، با گذر از دستوراتی که اجرا نشده اند، کنترل برنامه به ابتدای حلقه منتقل میشود.

دستور break

```
while(<شرایط>)
{
    Some commands
    break;
    Additional commands
}
Application commands
```

دستور continue

```
while(<شرایط>)
{
    Some commands
    continue;
    Additional commands
}
Application commands
```

مثال

- برنامه ای بنویسید که کاراکتری را از کاربر بگیرد و عملیات زیر را با توجه به کاراکتر دریافتی انجام دهد
- ‘a’: پیام Hello نمایش دهد
- ‘b’: پیام I am a programmer نمایش دهد
- ‘c’: به ابتدای حلقه رفته و کاراکتر دیگری بگیرد
- ‘x’: از برنامه خارج شود

```
int main()
{
    char ch;
    clrscr ();
    do
    {
        printf( " \n\nStart of loop  " );
        printf( " \nEnter a character:  " );
        ch = getche();
        if ( ch == 'a' )
            printf( " \nHello  " );
        else if ( ch == 'b' )
            printf( " \nI am a programmer  " );
        else if ( ch == 'c' )
            continue;
        else if ( ch == 'x' )
            break;
        printf( " \nEnd of loop  " );
    } while ( 1 );
     printf( " \nEnd of program  " );
    return 0;
}
```

```
Start of loop
Enter a character:  a
Hello
End of loop

Start of loop
Enter a character:  b
I am a programmer
End of loop

Start of loop
Enter a character:  r
End of loop

Start of loop
Enter a character:  c

Start of loop
Enter a character:  x
End of program
```

تمرین

- برنامه ای بنویسید که یک کلید از کاربر بگیرد. اگر این کلید از کلیدهای جهت نما بود موقعیت cursor روی صفحه نمایش را در جهت اشاره شده حرکت دهد و در موقعیت (0,0) از صفحه نمایش موقعیت فعلی cursor را بنویسد.
- راهنمایی:
  - مکان نما را با استفاده از دستور gotoxy جابجا کنید
  - دو متغیر برای نگهداری موقعیت cursor تعریف نموده و با توجه به کلید فشرده شده موقعیت را جابجا نمایید
  - قبل از رفتن به موقعیت جدید به خانه (0,0) رفته و موقعیت را چاپ کنید و بعد از آن به موقعیت جدید انتقال مکان دهید.