مبانی کامپیوتر و برنامه‌سازی

ساختارهای کنترلی

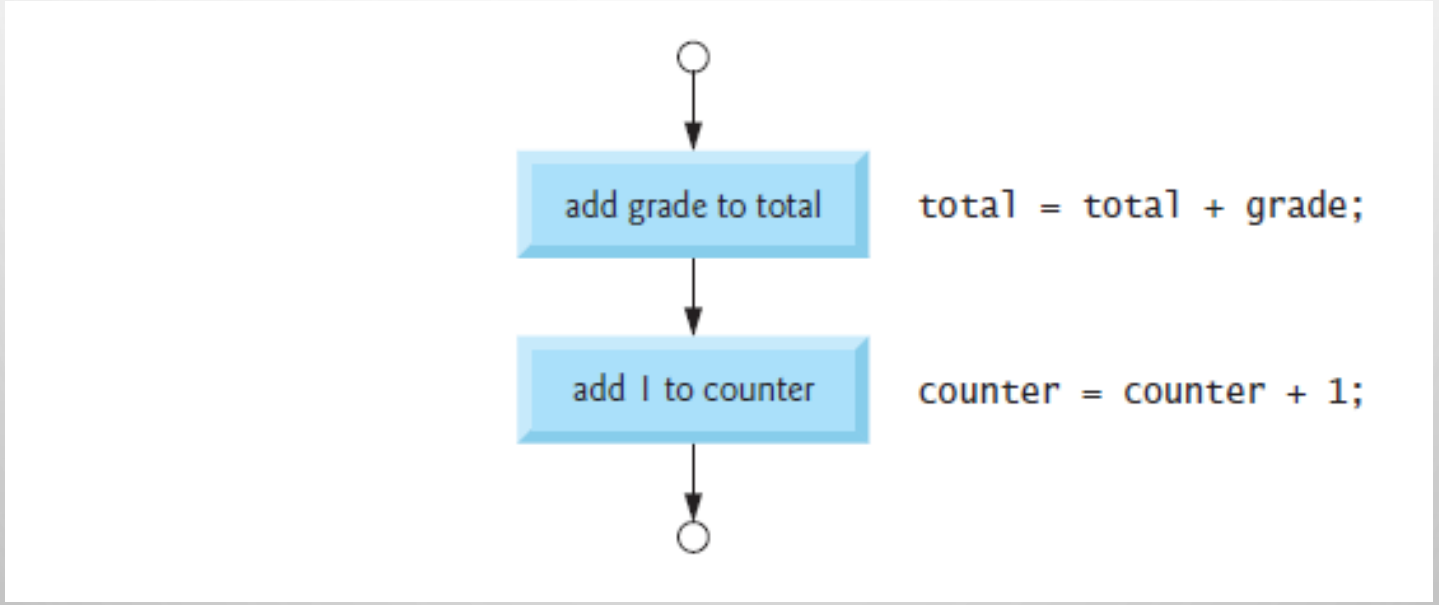
- ساختارهای ترتیبی Sequence Structure
- ساختارهای انتخابی Selection Structure
- ساختارهای تکرار Repetition Structure
- انتقال کنترل
  - به جای دستور بعدی، دستور دیگری اجرا شود: goto
  - به دلیل ایجاد آشفتگی در برنامه، از دستور goto استفاده نمی‌کنیم

ساختارهای کنترلی

- C هفت ساختار کنترلی دارد
  - ترتیبی
  - سه نوع انتخابی
  - سه نوع تکرار
- برنامه C با ترکیب‌های مختلفی از این 7 ساختار، شکل می گیرد

ساختارهای کنترلی: ساختارهای ترتیبی

- در ساختار ترتیبی، دستورات یکی پس از دیگری به ترتیبی که نوشته شده‌اند، اجرا می شوند
- فلوچارت ساختار ترکیبی:



ساختار کنترلی: ساختارهای انتخابی

- زبان C سه نوع ساختار انتخابی دارد
- if: اگر شرط درست باشد، دستوری را اجرا می کند و اگر شرط نادرست باشد از روی دستور رد می شود.
  - تصمیم گیری های ساده Single-selection Statement

- if...else: اگر شرط درست باشد، دستوری را اجرا می کند و اگر نادرست باشد، دستور دیگری را انجام می دهد
  - تصمیم گیری های دوگانه Double-selection Statement
-
- switch: یکی از چند تا دستور، بر اساس مقادیر مختلف یک عبارت، انجام می‌شود
  - تصمیم گیری های چند گانه Multiple-selection Statement

ساختار کنترلی: ساختارهای انتخابی

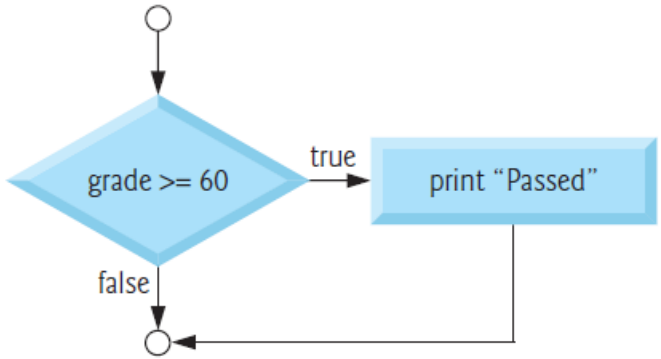


Fig. 3.2 | Flowcharting the single-selection `if` statement.

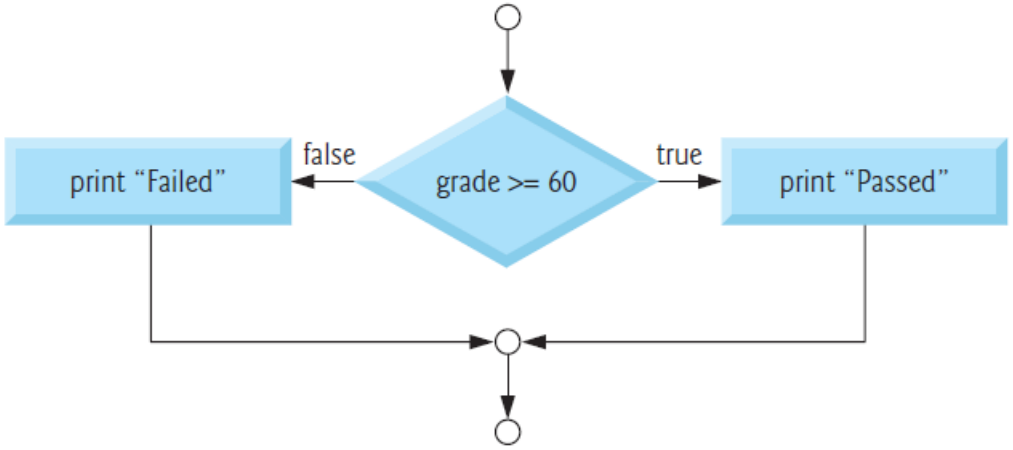


Fig. 3.3 | Flowcharting the double-selection `if...else` statement.

ساختارهای کنترلی: عبارت تکرار while

عبارت تکرار(حلقه)، این امکان را می‌دهد که یک دستور یا مجموعه ای از دستورات تا زمانی که شرط درست باشد، تکرار شود.

```
while(شرط)
    دستور;
```

```
while(شرط){
    دستور۱;
    دستور۲;
    ...
    nدستور;
}
```

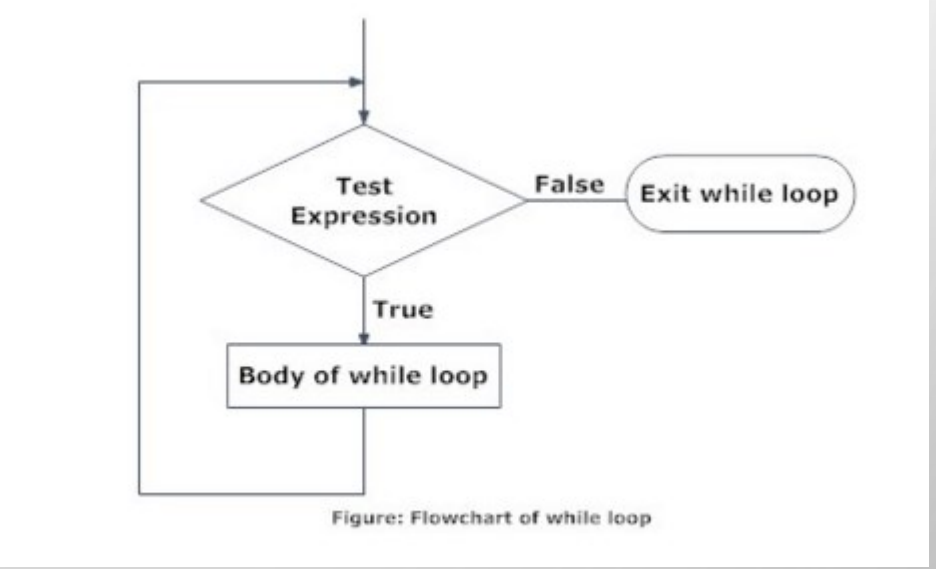


Figure: Flowchart of while loop

ساختارهای کنترلی: عبارت تکرار while

- یک مثال از while
  - بخشی از برنامه (تکه برنامه)
  - برنامه‌ای را تصور کنید که می خواهد اولین توان 3 را بدست آورد که از 100 بیشتر است
- فرض کنید مقدار اولیه متغیر product برابر 3 است.
- وقتی دستور while تمام شود، product، جواب مورد نظر را خواهد داشت.

```
product = 3;
while ( product <= 100 ) {
   product = 3 * product;
} /* end while */
```

ساختارهای کنترلی: عبارت تکرار while

- اول، مقدار متغیرproduct، برابر 3 است
- متغیر product در 3 ضرب می‌شود و به ترتیب مقادیر 9، 27 و 81 را می‌گیرد
- وقتی که مقدار product، برابر 243 می شود شرط اینکه 100=>product نقض می شود
- بنابراین از حلقه خارج شده و عدد 243 را چاپ می کند
- اجرای برنامه از دستور بعد از while ادامه می یابد

**اگر در بدنه while دستوری که در نهایت موجب نادرست شدن شرط while شود، قرار ندهیم، خطای منطقی به نام حلقه بی‌نهایت بوجود می‌آید که در آن دستور while هرگز خاتمه نمی‌یابد.

ساختارهای کنترلی: عبارت تکرار while

- فلوچارت زیر، تکرار را نشان می دهد
- خط جریان، از مستطیل دوباره به لوزی تصمیم‌گیری برمی‌گردد تا هر دفعه شرط حلقه چک شود و در صورت نادرست بودن، خارج شود.
- در این نقطه، از دستور while خارج شده و کنترل به دستور بعد از آن منتقل می‌شود.

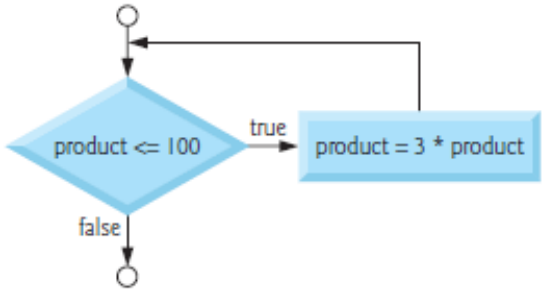


Fig. 3.4 | Flowcharting the `while` repetition statement.

تکرار با کنترل توسط شمارنده

از کلاسی که شامل 10 دانشجو است، آزمونی گرفته شده است. نمرات این آزمون اعداد صحیح در بازه 0 تا 100 هستند. برنامه‌ای بنویسید که میانگین نمرات این کلاس را محاسبه کرده و در خروجی نمایش دهد.

- این تکنیک از یک متغیر به نام **شمارنده** استفاده می‌کند تا تعداد دفعاتی که یک مجموعه دستور باید اجرا شود را مشخص کند.
- در این مثال، وقتی شمارنده از 10 بیشتر شد از دستور تکرار خارج می شویم.
- به تکرار با کنترل توسط شمارنده، «تکرار معین» هم می‌گویند چون تعداد تکرارها قبل از ورود به حلقه مشخص است.

تکرار با کنترل توسط شمارنده

میانگین نمرات کلاس

```
1  /* Fig. 3.6: fig03_06.c
2     Class average program with counter-controlled repetition */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main( void )
7  {
8     int counter; /* number of grade to be entered next */
9     int grade; /* grade value */
10    int total; /* sum of grades input by user */
11    int average; /* average of grades */
12
13    /* initialization phase */
14    total = 0; /* initialize total */
15    counter = 1; /* initialize loop counter */
16
17    /* processing phase */
18    while ( counter <= 10 ) { /* loop 10 times */
19       printf( "Enter grade: " ); /* prompt for input */
20       scanf( "%d", &grade ); /* read grade from user */
21       total = total + grade; /* add grade to total */
22       counter = counter + 1; /* increment counter */
23    } /* end while */
24
25    /* termination phase */
26    average = total / 10; /* integer division */
27
28    printf( "Class average is %d\n", average ); /* display result */
29    return 0; /* indicate program ended successfully */
30 } /* end function main */
```

نگهداری جمع نمرات

شمارنده تکرار حلقه

ادامه ...

خروجی برنامه

```
Enter grade: 98
Enter grade: 76
Enter grade: 71
Enter grade: 87
Enter grade: 83
Enter grade: 90
Enter grade: 57
Enter grade: 79
Enter grade: 82
Enter grade: 94
Class average is 81
```

نکته:

متغیری که مقداردهی اولیه نشده، حاوی «مقدار ناخواسته» از آخرین مقداردهی در مکان ذخیره شده متغیر در حافظه است.

اگر متغیر ابتدای برنامه مقداردهی نشود نتیجه برنامه احتمالا غلط است بنابراین تمام شمارنده‌ها و مجموع‌ها باید مقداردهی اولیه شوند.

ساختارهای کنترلی: تکرار با کنترل توسط نگهبان

برنامه‌ای بنویسید که نمرات تعدادی از دانشجویان را از ورودی دریافت کرده و میانگین آنها را محاسبه کرده و در خروجی چاپ کند. تعداد دانشجویان در هر بار اجرای برنامه متفاوت می باشد.

- در مثال قبل، تعداد نمرات از قبل مشخص و برابر با 10 بود.
- در این مثال، مشخص نیست که در حین اجرای برنامه، نمرات چه تعداد دانشجو توسط کاربر وارد خواهد شد.
- چطور برنامه مشخص کند که چه موقع باید خواندن نمرات را متوقف نماید؟ چطور بفهمیم که چه زمانی میانگین را محاسبه و در خروجی چاپ کنیم؟

یک راه برای حل مسئله، استفاده از مقداری خاص به نام مقدار نگهبان (مقدار سیگنال) است.

Flag

- مقدار نگهبان را مقدار ساختگی یا مقدار پرچم نیز می نامند.
- مقدار نگهبان «پایان داده ورودی» را مشخص می‌کند.
- کاربر، نمره دانشجویان را تا زمانی که نمره‌ها مقدار مجاز دارند، وارد می‌کند.
- سپس کاربر مقدار نگهبان را وارد می کند تا نشان دهد که نمره آخر وارد شده است.
- تکرار با کنترل توسط نگهبان اغلب «تکرار نامعین» نامیده می‌شود چون تعداد تکرار قبل از حلقه مشخص نیست.
- مقدار نگهبان باید بگونه‌ای انتخاب شود که با «مقدار ورودی مجاز»، اشتباه نشود.
- چون مقدار یک نمره نمی تواند منفی باشد، 1- مقدار نگهبانی است که برای این مسئله انتخاب می کنیم.
- بنابراین، یک اجرا از برنامه می تواند یک رشته ورودی باشد مانند 74،75،96،95، 89 و 1-
- برنامه، میانگین نمره‌های کلاس یعنی 74،75،96،95و89 را محاسبه و چاپ می‌کند.
- عدد 1- که مقدار نگهبان است، نباید در محاسبه میانگین کلاس وارد شود.

در نظر بگیرید

- اگرچه همه نمرات عدد صحیح هستند، ولی میانگین احتمال دارد به صورت اعشاری باشد.
- نوع داده‌ای int را نمی توان برای میانگین اعشاری استفاده کرد.
- در برنامه نوع داده‌ای float برای اعداد اعشاری معرفی شده است.
- همچنین عملگری به نام عملگر cast (تبدیل نوع) برای مدیریت محاسبه میانگین معرفی شده است.

کد

میانگین نمرات کلاس با تعداد نامشخصی از دانشجو

```
1  /* Fig. 3.8: fig03_08.c
2     Class average program with sentinel-controlled repetition */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main( void )
7  {
8     int counter; /* number of grades entered */
9     int grade; /* grade value */
10    int total; /* sum of grades */
11
12    float average; /* number with decimal point for average */
13
14    /* initialization phase */
15    total = 0; /* initialize total */
16    counter = 0; /* initialize loop counter */
17
18    /* processing phase */
19    /* get first grade from user */
20    printf( "Enter grade, -1 to end: " ); /* prompt for input */
21    scanf( "%d", &grade ); /* read grade from user */
22
```

ادامه کد

```
23      /* loop while sentinel value not yet read from user */
24      while ( grade != -1 ) {
25         total = total + grade; /* add grade to total */
26         counter = counter + 1; /* increment counter */
27
28         /* get next grade from user */
29         printf( "Enter grade, -1 to end: " ); /* prompt for input */
30         scanf("%d", &grade); /* read next grade */
31      } /* end while */
32
33      /* termination phase */
34      /* if user entered at least one grade */
35      if ( counter != 0 ) {
36
37         /* calculate average of all grades entered */
38         average = ( float ) total / counter; /* avoid truncation */
39
40         /* display average with two digits of precision */
41         printf( "Class average is %.2f\n", average );
42      } /* end if */
43      else { /* if no grades were entered, output message */
44         printf( "No grades were entered\n" );
45      } /* end else */
46
47      return 0; /* indicate program ended successfully */
48   } /* end function main */
```




```
Enter grade, -1 to end: 75
Enter grade, -1 to end: 94
Enter grade, -1 to end: 97
Enter grade, -1 to end: 88
Enter grade, -1 to end: 70
Enter grade, -1 to end: 64
Enter grade, -1 to end: 83
Enter grade, -1 to end: 89
Enter grade, -1 to end: -1
Class average is 82.50
```

```
Enter grade, -1 to end: -1
No grades were entered
```

خروجی

```
Enter grade, -1 to end: 75
Enter grade, -1 to end: 94
Enter grade, -1 to end: 97
Enter grade, -1 to end: 88
Enter grade, -1 to end: 70
Enter grade, -1 to end: 64
Enter grade, -1 to end: 83
Enter grade, -1 to end: 89
Enter grade, -1 to end: -1
Class average is 82.50
```

```
Enter grade, -1 to end: -1
No grades were entered
```

دستور do while

- do while مشابه while است
- در while، «شرط حلقه» در ابتدای حلقه چک می شود
- در do while «شرط حلقه» بعد از اجرای دستور (بدنه حلقه) چک می شود، بنابراین دستورات بدنه حلقه حداقل یک بار اجرا می شوند
- در do while حتی اگر یک دستور بود بهتر است از آکولاد باز و بسته استفاده شود تا با WHILE اشتباه نشود

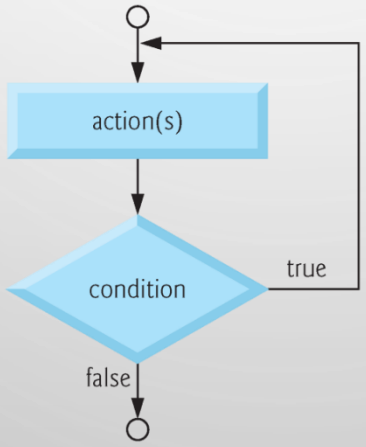


Fig. 4.10 | Flowcharting the do...while repetition statement.

مثال

مثال: برنامه ای بنویسید که اعداد 1 تا 10 را چاپ کند (do...while)

```
1  /* Fig. 4.9: fig04_09.c
2     Using the do/while repetition statement */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main( void )
7  {
8     int counter = 1; /* initialize counter */
9
10    do {
11       printf( "%d  ", counter ); /* display counter */
12    } while ( ++counter <= 10 ); /* end do...while */
13
14    return 0; /* indicate program ended successfully */
15 } /* end function main */
```

```
1  2  3  4  5  6  7  8  9  10
```

Fig. 4.9 | do...while statement example.

فلوچارت حلقه‌ها

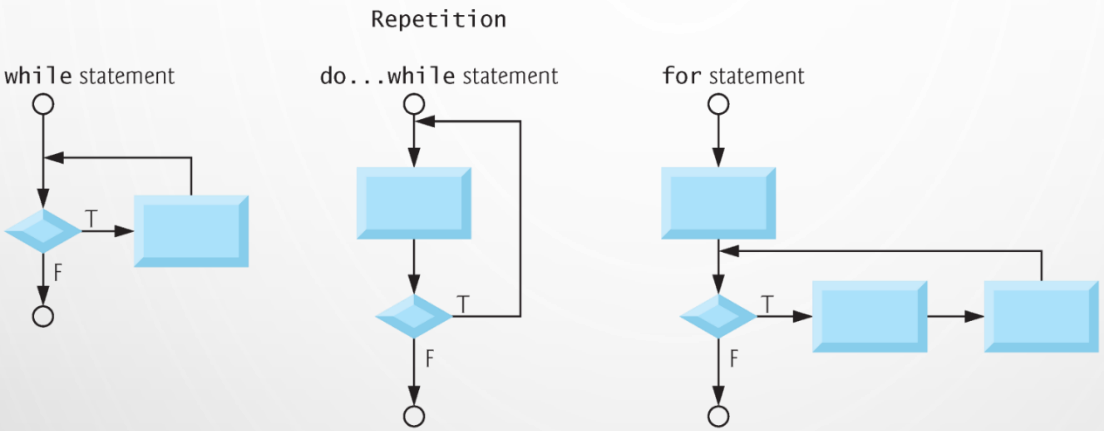


Fig. 4.17 | C's single-entry/single-exit sequence, selection and repetition statements. (Part 2 of 2.)

مثال do while

برنامه ای بنویسید که عددی را از ورودی خوانده، وارون آن را در خروجی چاپ کند.

```c
#include<stdio.h>
int main (void)
{
 int num,digit=0;
 printf("Enter a Number: ");
 scanf("%d",&num);
 printf("Inverse=");
 do{
    digit=num%10;
    printf("%d", digit);
    num/=10;
 } while(num!=0);
 return 0;
}
```

```
 Enter a Number: 45678

Inverse=87654
--------------------------------
Process exited after 7.099 seconds with retur
Press any key to continue . . .
```

دستور break

خروج زود هنگام از حلقه‌ها و صرفنظرکردن از بقیه دستورات

```
1  /* Fig. 4.11: fig04_11.c
2     Using the break statement in a for statement */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main( void )
7  {
8     int x; /* counter */
9
10    /* loop 10 times */
11    for ( x = 1; x <= 10; x++ ) {
12
13       /* if x is 5, terminate loop */
14       if ( x == 5 ) {
15          break; /* break loop only if x is 5 */
16       } /* end if */
17
18       printf( "%d ", x ); /* display value of x */
19    } /* end for */
20
21    printf( "\nBroke out of loop at x == %d\n", x );
22    return 0; /* indicate program ended successfully */
23 } /* end function main */
```

Fig. 4.11 | Using the `break` statement in a `for` statement. (Part 1 of 2.)

ادامه ...

در این برنامه بدنه دستور FOR، 4 مرتبه اجرا می شود

```
1 2 3 4
Broke out of loop at x == 5
```

Fig. 4.11 | Using the `break` statement in a `for` statement. (Part 2 of 2.)

مثال

برنامه ای بنویسید که تعدادی عدد را از ورودی خوانده و تعداد اعداد زوج و فرد را مشخص کرده و به خروجی ببرد. مقدار نگهبان، عدد صفر است

```
#include<stdio.h>
int main (void)
 {
 int num,count=0,n=0;
 while(1)
 {
     printf("Enter a number:");
     scanf("%d",&num);
     if(num==0)
      break;
     n++;
     if(num%2==0)
       count++;
 }
 printf("\n evens=%d, odds=%d",count,n-count);
 return 0;
}
```

```
Enter a number:16
Enter a number:18
Enter a number:29
Enter a number:34
Enter a number:64
Enter a number:27
Enter a number:1
Enter a number:0

 evens=4, odds=3
--------------------------------
Process exited after 21.05 seconds with return value 0
Press any key to continue . . .
```

دستور continue

- صرفنظر از بقیه دستورات حلقه و رفتن به ابتدای حلقه
  - while
  - do while

  } ابتدای حلقه: شرط حلقه

  - for

  } ابتدای حلقه: تغییر مقدار (بخش سوم for) و سپس شرط حلقه

- تنها تفاوت معادل‌سازی حلقه for با استفاده while:
  - اگر continue در بدنه حلقه قبل از دستور تغییر مقدار استفاده شود، در while تغییر مقدار متغیر انجام نمی شود

مثال

```
1  /* Fig. 4.12: fig04_12.c
2     Using the continue statement in a for statement */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main( void )
7  {
8     int x; /* counter */
9
10    /* loop 10 times */
11    for ( x = 1; x <= 10; x++ ) {
12
13       /* if x is 5, continue with next iteration of loop */
14       if ( x == 5 ) {
15          continue; /* skip remaining code in loop body */
16       } /* end if */
17
18       printf( "%d ", x ); /* display value of x */
19    } /* end for */
20
21    printf( "\nUsed continue to skip printing the value 5\n" );
22    return 0; /* indicate program ended successfully */
23 } /* end function main */
```

Fig. 4.12 | Using the `continue` statement in a `for` statement. (Part 1 of 2.)