



## Week 2 - Lecture 3

# Control Statements - Iteration

Edited by: Dr. Wooi Ping Cheah  
Autumn 2022

# Overview

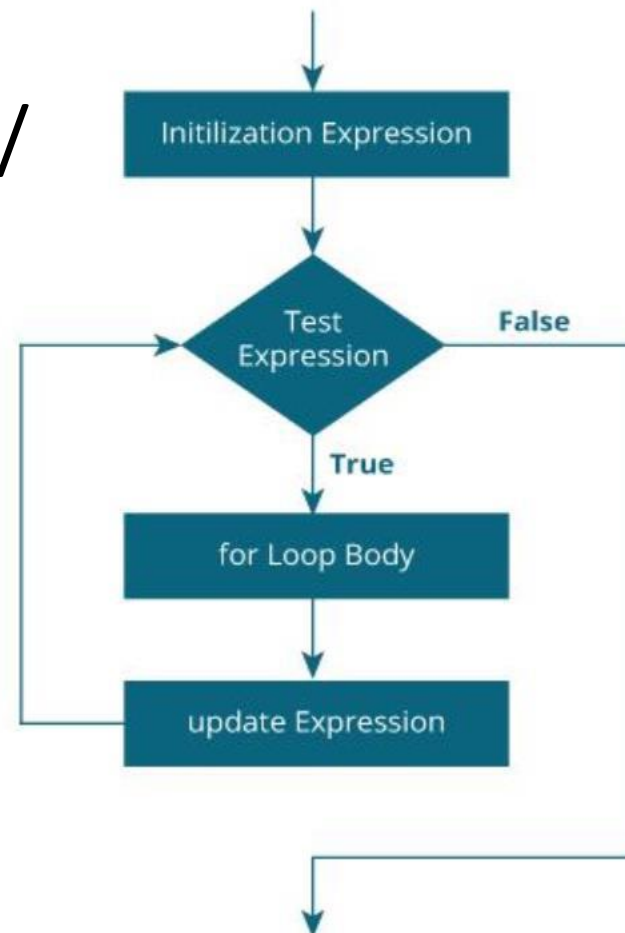
- **Control Structure in C**
- Return value of scanf
- break and continue

# Control Structure in C

- Sequence structure
- Selection structure
  - if
  - if ... else ...
  - switch
- Iteration structure
  - while
  - do ... while ...
  - for

# for Loop in C

- `for(initialisation; condition; update)`  
  {  
    /\* statements \*/  
  }



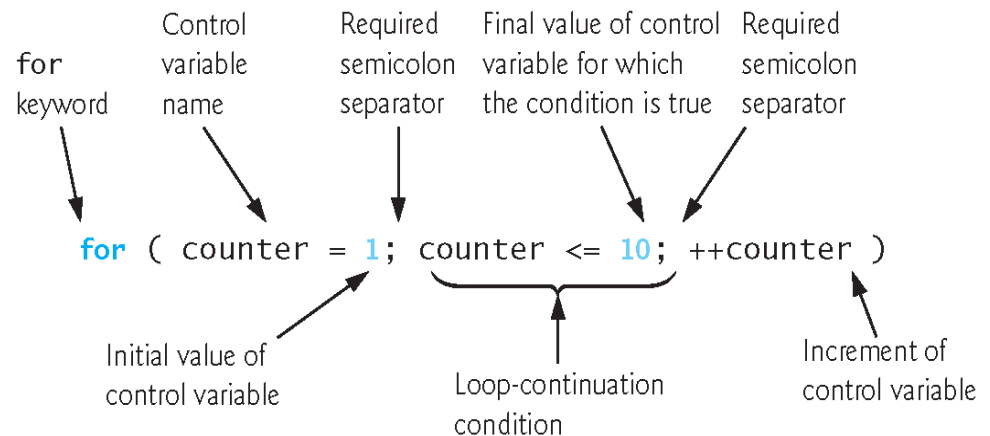
# for Loop in C (2)

- The general format of the `for` statement is

```
for ( expression1; expression2; expression3 ) {  
    statement  
}
```

where

- *expression1* initializes the loop-control variable
- *expression2* is the loop-continuation condition
- *expression3* increments the control variable.



# Example: for Loop

```
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int input = 0;
7      int sum = 0;
8
9      printf("Please enter an input: ");
10     scanf("%d", &input);
11
12     sum = sum + input;
13
14     printf("Please enter an input: ");
15     scanf("%d", &input);
16
17     sum = sum + input;
18
19     printf("The sum of two numbers is %d\n", sum);
20
21     return 0;
22 }
```

```
25  #include <stdio.h>
26
27  int main(void)
28  {
29      int counter = 0;
30      int input = 0;
31      int sum = 0;
32
33      for(counter = 0; counter < 2; counter++)
34      {
35          printf("Please enter an input: ");
36          scanf("%d", &input);
37
38          sum = sum + input;
39      }
40
41      printf("The sum of two numbers is %d\n", sum);
42
43      return 0;
44 }
```

# Example: for Loop (2)

The following examples show methods of varying the control variable in a **for** statement.

Vary the control variable from 1 to 100 in increments of 1.

```
for ( i = 1; i <= 100; ++ i )
```

Vary the control variable from 100 to 1 in increments of -1 (decrements of 1).

```
for ( i = 100; i >= 1; --i )
```

Vary the control variable from 7 to 77 in steps of 7.

```
for ( i = 7; i <= 77; i += 7 )
```

Vary the control variable from 20 to 2 in steps of -2.

```
for ( i = 20; i >= 2; i -= 2 )
```

Vary the control variable over the following sequence of values: 2, 5, 8, 11, 14, 17.

```
for ( j = 2; j <= 17; j += 3 )
```

Vary the control variable over the following sequence of values: 44, 33, 22, 11, 0.

```
for ( j = 44; j >= 0; j -= 11 )
```

# Take Home exercise.

Consider the following problem statement:

A person invests \$1000.00 in a savings account yielding 5% interest. Assuming that all interest is left on deposit in the account, calculate and print the amount of money in the account at the end of each year for 10 years. Use the following formula for determining these amounts:

$$a = p(1 + r)^n$$

where

p is the original amount invested (i.e., the principal)

r is the annual interest rate

n is the number of years

a is the amount on deposit at the end of the n<sup>th</sup> year.

This problem involves a loop that performs the indicated calculation for each of the 10 years the money remains on deposit.



# Take Home exercise.

```
1 // Fig. 4.6: fig04_06.c
2 // Calculating compound interest.
3 #include <stdio.h>
4 #include <math.h>
5
6 // function main begins program execution
7 int main( void )
8 {
9     double amount; // amount on deposit
10    double principal = 1000.0; // starting principal
11    double rate = .05; // annual interest rate
12    unsigned int year; // year counter
13
14    // output table column heads
15    printf( "%4s%21s\n", "Year", "Amount on deposit" );
16
17    // calculate amount on deposit for each of ten years
18    for ( year = 1; year <= 10; ++year ) {
19
20        // calculate new amount for specified year
21        amount = principal * pow( 1.0 + rate, year );
22
23        // output one table row
24        printf( "%4u%21.2f\n", year, amount );
25    } // end for
26 } // end function main
```

# Take Home exercise.

| Year | Amount on deposit |
|------|-------------------|
| 1    | 1050.00           |
| 2    | 1102.50           |
| 3    | 1157.63           |
| 4    | 1215.51           |
| 5    | 1276.28           |
| 6    | 1340.10           |
| 7    | 1407.10           |
| 8    | 1477.46           |
| 9    | 1551.33           |
| 10   | 1628.89           |

# Omitting Expressions

- `int a = 0;`  
  `for(; a < 5; a++) { ... }`
- `for(a = 0; a < 5;){`  
  `a++;`  
  `}`
- `for(a = 0; ; a++) { ... }`
- `for( ; ; ) { ... }`
- `for(; a < 5; ) { ... }`

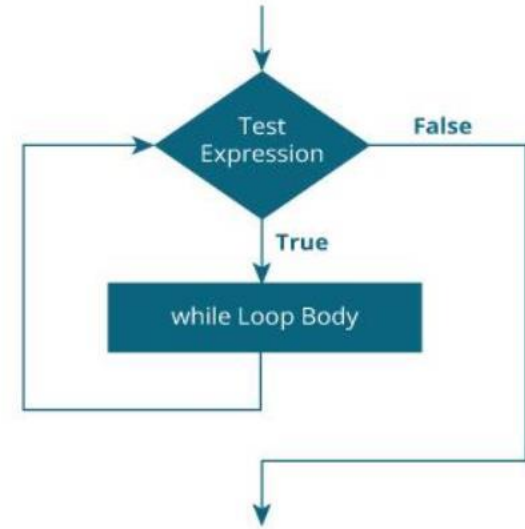
Infinite loops

While loop

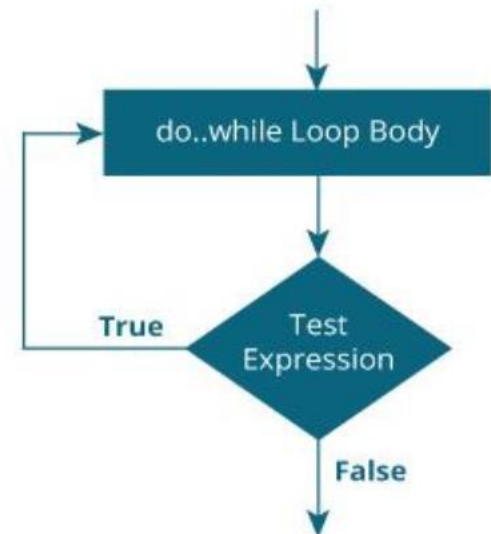
# while and do-while Loop in C

- `while(test)`  
  {  
    /\* statements \*/  
  }

- `do`  
  {  
    /\* statements \*/  
  }  
  while(test);



Source: <https://www.programiz.com/c-programming/c-do-while-loops>



# Example: while and do-while Loop

```
70 #include <stdio.h>
71
72 int main(void)
73 {
74     int input = 0;
75     int sum = 0;
76
77     do
78     {
79         printf("Please enter an input: ");
80         scanf("%d", &input);
81
82         if(input > 0)
83         {
84             sum = sum + input;
85         }
86     }while(input > 0);
87
88     printf("The sum of two numbers is %d\n", sum);
89
90     return 0;
91 }
92
```

```
95 #include <stdio.h>
96
97 int main(void)
98 {
99     int counter = 0;
100     int input = 0;
101     int sum = 0;
102     int limit = 0;
103
104     printf("Please enter the number of input: ");
105     scanf("%d", &limit);
106
107     while(counter < limit)
108     {
109         printf("Please enter an input: ");
110         scanf("%d", &input);
111
112         sum = sum + input;
113         counter = counter + 1;
114     }
115
116     printf("The sum of two numbers is %d\n", sum);
117
118     return 0;
119 }
```

# Example: Menu

```
263 #include <stdio.h>
264
265 int main(void)
266 {
267     int keepGoing = 1;
268
269     do
270     {
271         // Print the menu and get the user's choice.
272         printf("1) Add\n");
273         printf("2) Multiply\n");
274         printf("3) Mod\n");
275         printf("4) Quit\n");
276         printf("Enter option: ");
277
278         int option = 0;
279         scanf("%d", &option);
280
281         if(option == 1)
282         {
283             // Add 2 numbers.
284             int n1 = 0;
285             printf("Enter first number: ");
286             scanf("%d", &n1);
287
288             int n2 = 0;
289             printf("Enter second number: ");
290             scanf("%d", &n2);
291
292             int ans = n1 + n2;
293             printf("The answer is %d\n\n", ans);
294         }
295         else if(option == 2)
296         {
297             // Multiply 2 numbers.
298             int n1 = 0;
299             printf("Enter first number: ");
300             scanf("%d", &n1);
```

```
331
332         int n2 = 0;
333         printf("Enter second number: ");
334         scanf("%d", &n2);
335
336         int ans = n1 * n2;
337         printf("The answer is %d\n\n", ans);
338     }
339     else if(option == 3)
340     {
341         // Modulus
342         int n1 = 0;
343         printf("Enter first number: ");
344         scanf("%d", &n1);
345
346         int n2 = 0;
347         printf("Enter second number: ");
348         scanf("%d", &n2);
349
350         int ans = n1 % n2;
351         printf("The answer is %d\n\n", ans);
352     }
353     else if(option == 4)
354     {
355         // Quit
356         keepGoing = 0;
357     }
358 }while(keepGoing);
359
360 return 0;
361 }
```

# Overview

- Control Structure in C
- **Return value of scanf**
- break and continue

# Understanding Input Buffer

```
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int input = 0;
7      int sum = 0;
8
9      printf("Please enter an input: ");
10     scanf("%d", &input);
11
12     sum = sum + input;
13
14     printf("Please enter an input: ");
15     scanf("%d", &input);
16
17     sum = sum + input;
18
19     printf("The sum of two numbers is %d\n", sum);
20
21     return 0;
22 }
```

```
C:\Users\z2017233\Desktop>iteration
Please enter an input: 1
Please enter an input: 2
The sum of two numbers is 3

C:\Users\z2017233\Desktop>iteration
Please enter an input: 1 2
Please enter an input: The sum of two numbers is 3

C:\Users\z2017233\Desktop>iteration
Please enter an input: 1 2 3
Please enter an input: The sum of two numbers is 3

C:\Users\z2017233\Desktop>iteration
Please enter an input: 1 2 3 4
Please enter an input: The sum of two numbers is 3
```

- scanf will read from buffer without waiting for any user input if the buffer is **NOT** empty.



# Return value of scanf()

- One way to validate user's input.

```
181 #include <stdio.h>
182
183 int main(void)
184 {
185     int day = 0;
186     int month = 0;
187     int year = 0;
188     int count = 0;
189
190     printf("Please enter your date of birth (dd/mm/yyyy): ");
191     count = scanf("%d/%d/%d", &day, &month, &year);
192
193     printf("There are %d correctly formatted values\n", count);
194     printf("The values are %d, %d and %d\n", day, month, year);
195
196
197     return 0;
198 }
```

```
C:\Users\z2017233\Desktop>iteration
Please enter your date of birth (dd/mm/yyyy): 01/02/1234
There are 3 correctly formatted values
The values are 1, 2 and 1234

C:\Users\z2017233\Desktop>iteration
Please enter your date of birth (dd/mm/yyyy): 01 02 1234
There are 1 correctly formatted values
The values are 1, 0 and 0
```

# Overview

- Control Structure in C
- Return value of scanf
- **break and continue**

# break vs continue Statement

- The **break** statement, when executed in a loop (e.g., while), causes an immediate exit from that statement. Program execution continues with the next statement
- The **continue** statement, when executed in loop, skips the remaining statements in the body of that control statement and performs the next iteration of the loop.

# break vs continue Statement (2)

- **int i;**  
**for(i = 1; i < 10; i++){**  
    **if(i == 5)**  
        **break;**  
    **printf("%d ", i);**  
**}**  
**printf("End = %d\n", i);**

Completely  
terminates a loop

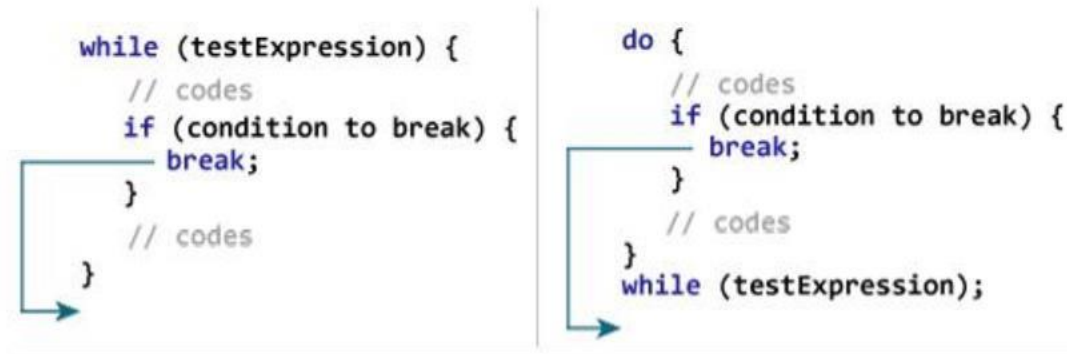
Displays 1 2 3 4 End = 5

- **int i;**  
**for(i = 1; i < 10; i++){**  
    **if(i < 5)**  
        **continue;**  
    **printf("%d ", i);**  
**}**

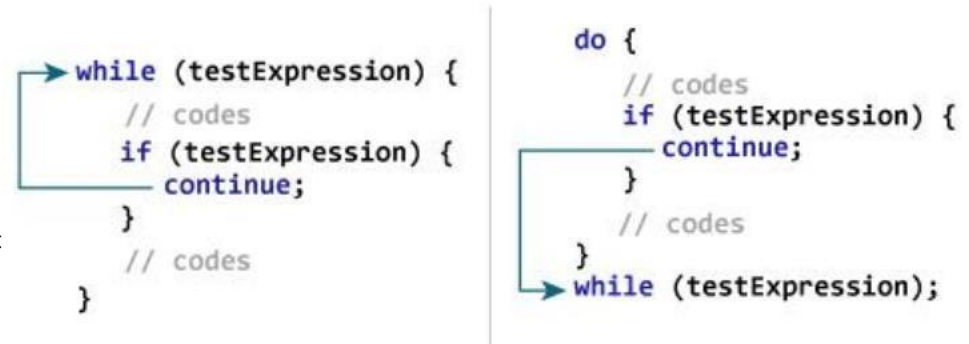
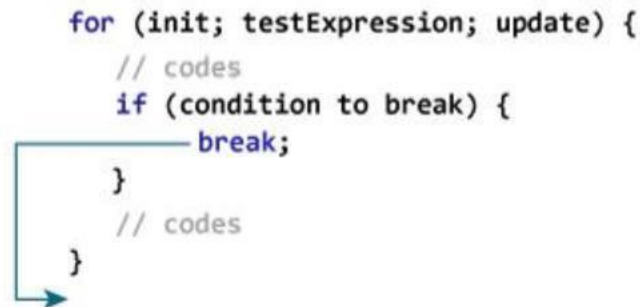
Terminates the **current**  
**iteration** of a loop

Displays 5 6 7 8 9

# break vs continue Statement (3)



Note the directions of the arrows!!

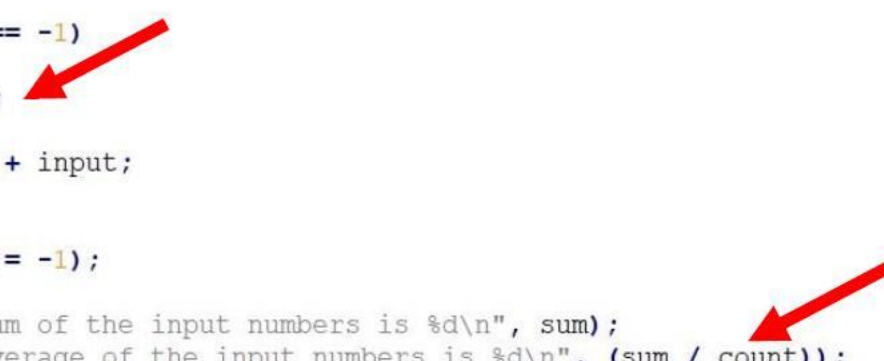


Source: <https://www.programiz.com/c-programming/c-break-continue-statement>

# Using break

- In this example, watch out for integer division.

```
122  #include <stdio.h>
123
124  int main(void)
125  {
126      int input = 0;
127      int sum = 0;
128      int count = 0;
129
130      do
131      {
132          printf("Please enter an input (-1 to quit): ");
133          scanf("%d", &input);
134
135          if(input == -1)
136          {
137              break;
138          }
139          sum = sum + input;
140          count++;
141
142      }while(input != -1);
143
144      printf("The sum of the input numbers is %d\n", sum);
145      printf("The average of the input numbers is %d\n", (sum / count));
146
147      return 0;
148  }
```



# Summary

- Control Structure in C
- Return value of scanf
- break and continue