Programming Fundamental

Week 11 –

Repetition controller structure in the C programming language

Dr. Maria Irmina Prasetiyowati, S.Kom,. M.T.
Alethea Suryadibrata, S.Kom., M.Eng.
Putri Sanggabuana Setiawan, S.Kom, M.T.I.
Januar Wahjudi, S.Kom., M.Sc.
Drs Slamet Aji Pamungkas, M.Eng
Kursehi Falgenti S.Kom., M.Kom.

Weekly Learning Outcomes for Subjects (Sub-CPMK):

Sub-CPMK 0614: Students are able to create simple programs with elements of selection control, repetition control, functions or procedures, and implement arrays and pointers in the C programming language (C6).

Review

1. IF…
2. Nested IF
3. Switch Case

Outline

1. For
2. While…
3. Do…While…
4. Break
5. Continue

Repetition Essentials

- A loop is a group of instructions the computer executes repeatedly while some loop repetition **condition** remains **true**

- 2 kinds of repetition
  - Counter-controlled repetition
  - Sentinel-controlled repetition

Counter-Controlled Repetition

- Counter-controlled repetition is sometimes called definite repetition because we know in advance exactly how many times the loop will be executed
- A loop control variable is used to count the number of repetitions
  - **Initialization :** Loop control variable is set to an initial value before the while statement is reached
  - **Testing :** Loop control variable is tested before the start of each loop repetition
  - **Updating :** Loop control variable is updated (incremented / decremented) during each iteration

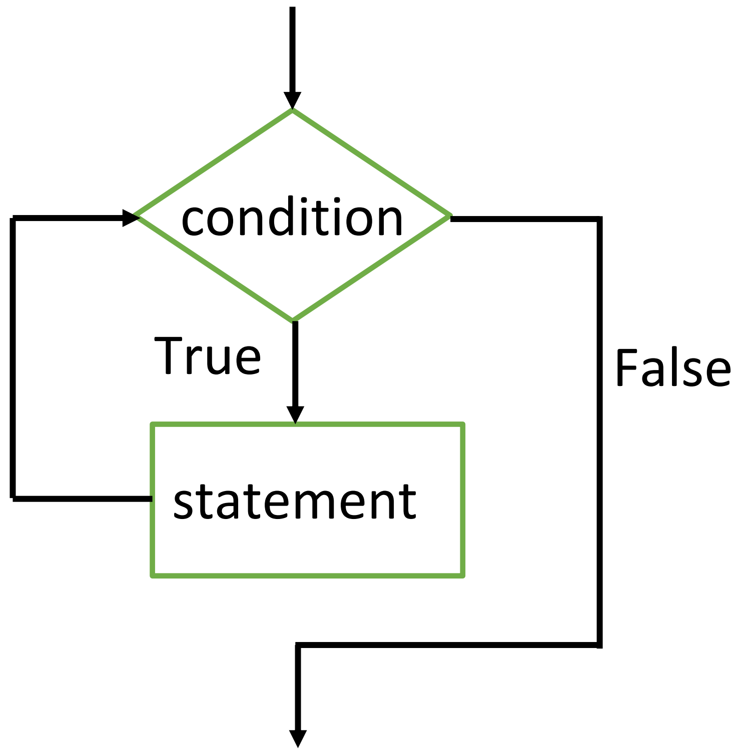
Sentinel-Controlled Repetition

- Sentinel-controlled repetition is sometimes called indefinite repetition because it’s not known in advance how many times the loop will be executed
- Sentinel values are used to control repetition when
  - The precise number of repetitions is not known in advance
  - The loop includes statements that obtain data each time the loop is performed
- The sentinel value indicates “end of data”
- The sentinel is entered after all regular data items have been supplied to the program
- Sentinels must be distinct from regular data items

While Statement

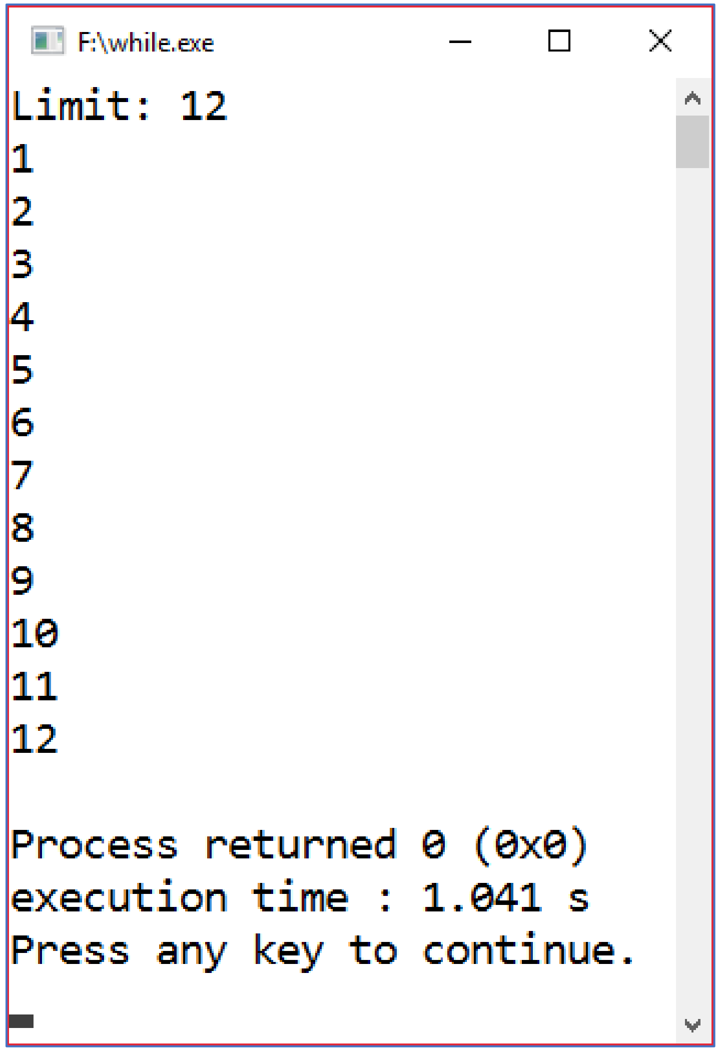
```
while(loop_repetition_condition)
{
      statement;
}
```

- The loop is repeated when the condition is true (when its value is not 0)
- The loop is exited when the condition is false

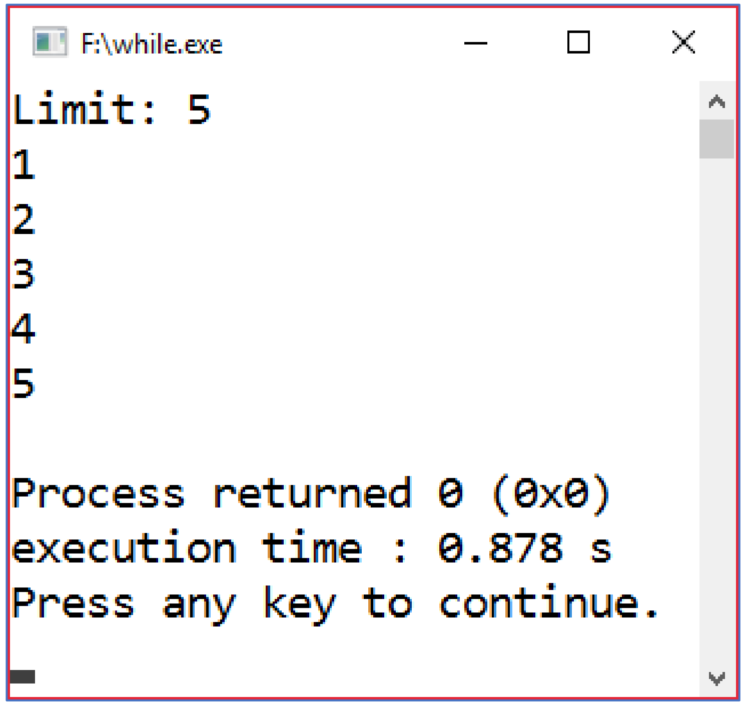


While Statement

```c
#include <stdio.h>

int main()
{
    int iCounter, iLimit;

    printf("Limit: ");
    scanf("%d",&iLimit);

    iCounter = 1;
    while(iCounter <= iLimit)
    {
        printf("%d\n",iCounter);
        iCounter++;
    }

    return 0;
}
```



```
Limit: 12
1
2
3
4
5
6
7
8
9
10
11
12

Process returned 0 (0x0)
execution time : 1.041 s
Press any key to continue.
```

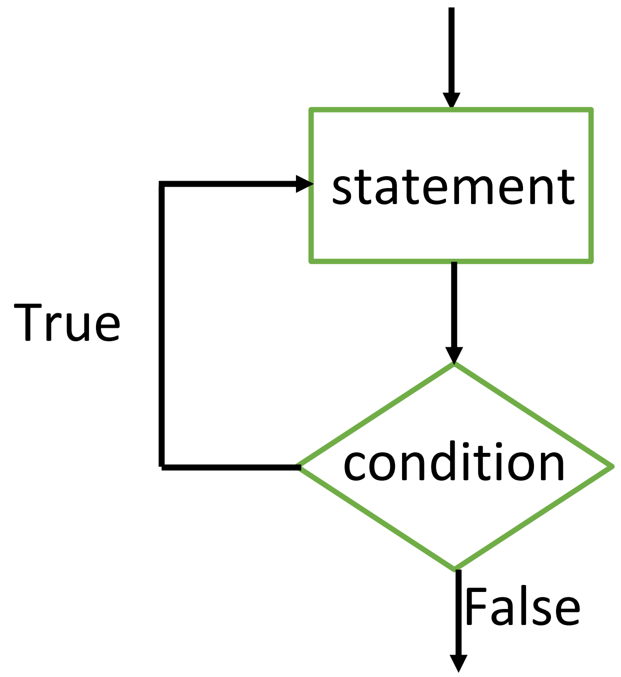


```
Limit: 5
1
2
3
4
5

Process returned 0 (0x0)
execution time : 0.878 s
Press any key to continue.
```

Do...While Statement

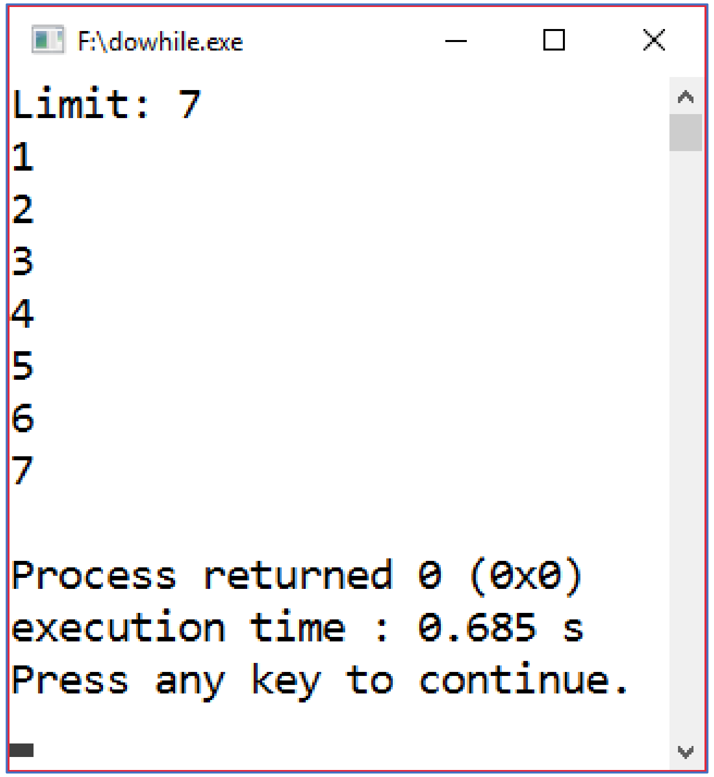
```
do
{
      statement;
}while(loop_repetition_condition);
```

- The do...while statement is similar to the while statement
- In the **while statement**, the loop repetition condition is tested at the beginning of the loop **before** the body of the loop is performed
- The **do...while statement** tests the loop repetition condition **after** the loop body is performed
- Therefore, the loop body will be executed **at least once**



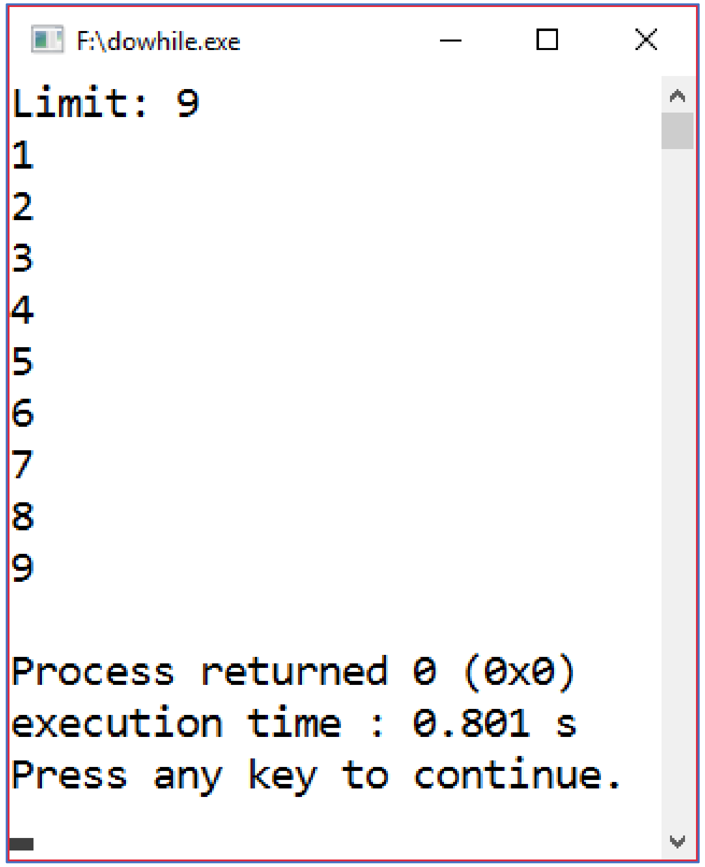
Do...While Statement

```c
#include <stdio.h>

int main()
{
    int iCounter, iLimit;

    printf("Limit: ");
    scanf("%d",&iLimit);

    iCounter = 1;
    do
    {
        printf("%d\n",iCounter);
        iCounter++;
    }while(iCounter <= iLimit);

    return 0;
}
```



F:\dowhile.exe

```
Limit: 7
1
2
3
4
5
6
7

Process returned 0 (0x0)
execution time : 0.685 s
Press any key to continue.
```

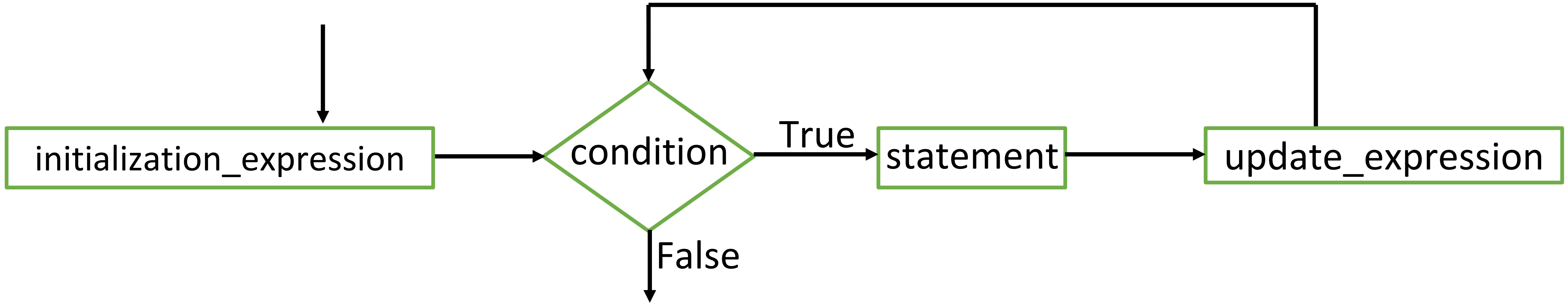


F:\dowhile.exe

```
Limit: 9
1
2
3
4
5
6
7
8
9

Process returned 0 (0x0)
execution time : 0.801 s
Press any key to continue.
```

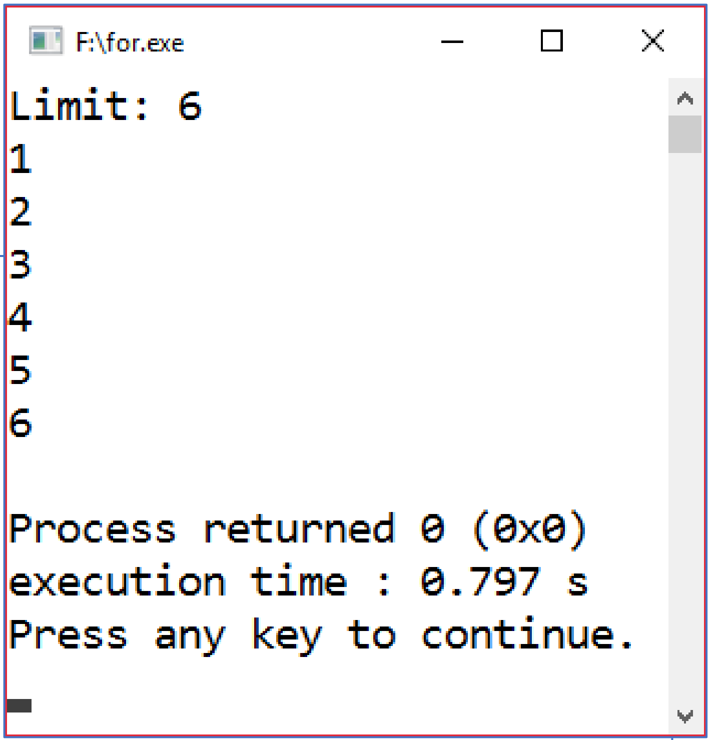
For Statement

```
for(initialization_expression;loop_repetition_condition;update_expression)
{
      statement;
}
```



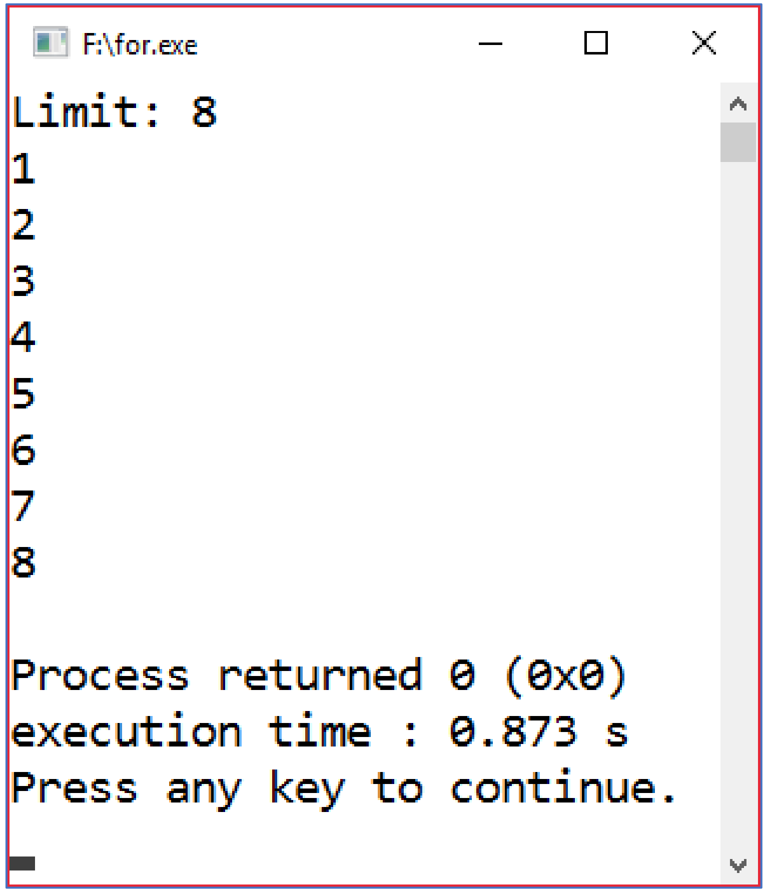
For Statement

```c
#include <stdio.h>

int main()
{
    int iCounter, iLimit;

    printf("Limit: ");
    scanf("%d",&iLimit);

    for(iCounter = 1;iCounter <= iLimit;iCounter++)
    {
        printf("%d\n",iCounter);
    }

    return 0;
}
```



F:\for.exe

```
Limit: 6
1
2
3
4
5
6

Process returned 0 (0x0)
execution time : 0.797 s
Press any key to continue.
```



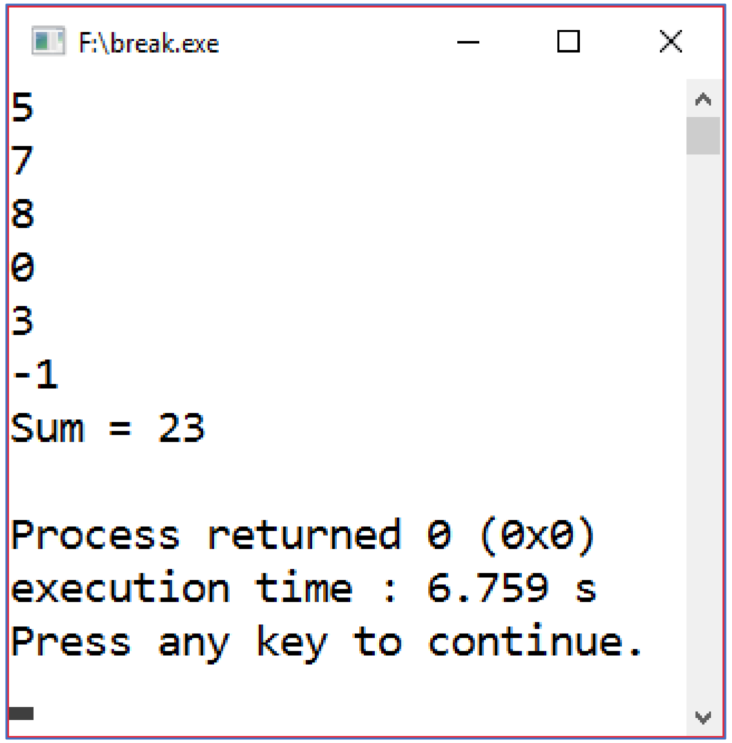
F:\for.exe

```
Limit: 8
1
2
3
4
5
6
7
8

Process returned 0 (0x0)
execution time : 0.873 s
Press any key to continue.
```

Break Statement

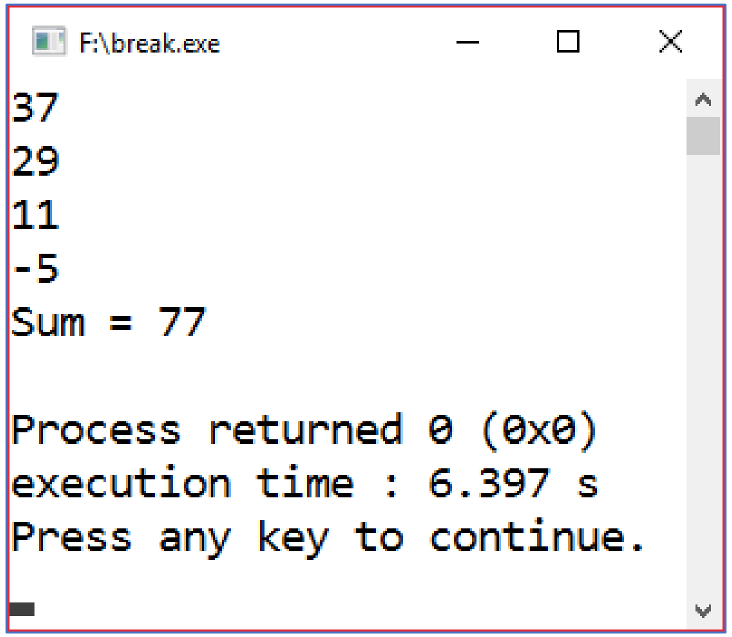
- The break statement, when executed in a **while**, **do…while**, **for**, or **switch** statement, causes an **immediate exit** from the statement

```c
#include <stdio.h>

int main()
{
    int iSum = 0, iNumber;

    while(1)
    {
        scanf("%d",&iNumber);
        if(iNumber < 0) break;
        iSum += iNumber;
    }
    printf("Sum = %d\n",iSum);

    return 0;
}
```



F:\break.exe

```
5
7
8
0
3
-1
Sum = 23

Process returned 0 (0x0)
execution time : 6.759 s
Press any key to continue.
```

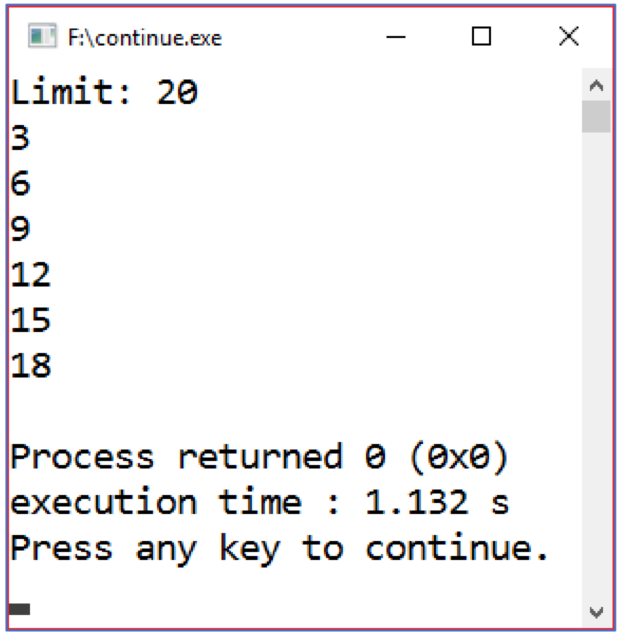


F:\break.exe

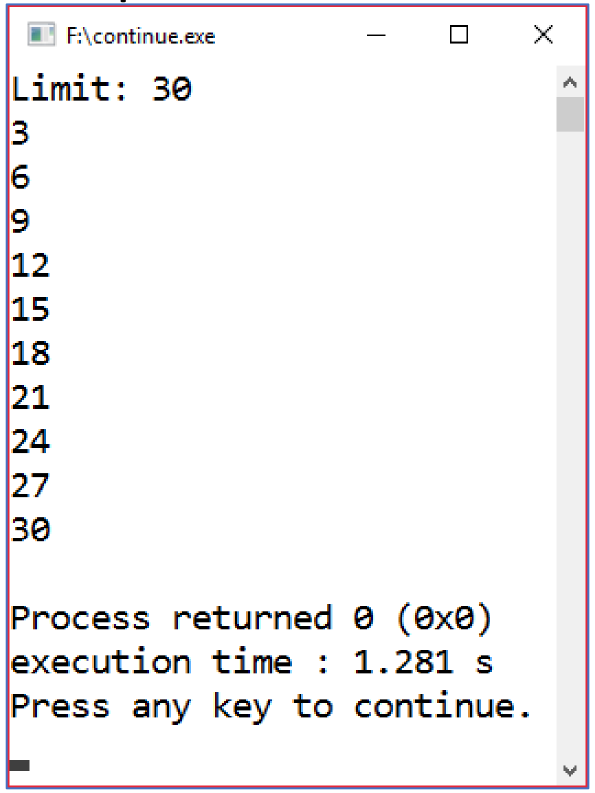
```
37
29
11
-5
Sum = 77

Process returned 0 (0x0)
execution time : 6.397 s
Press any key to continue.
```

Continue Statement

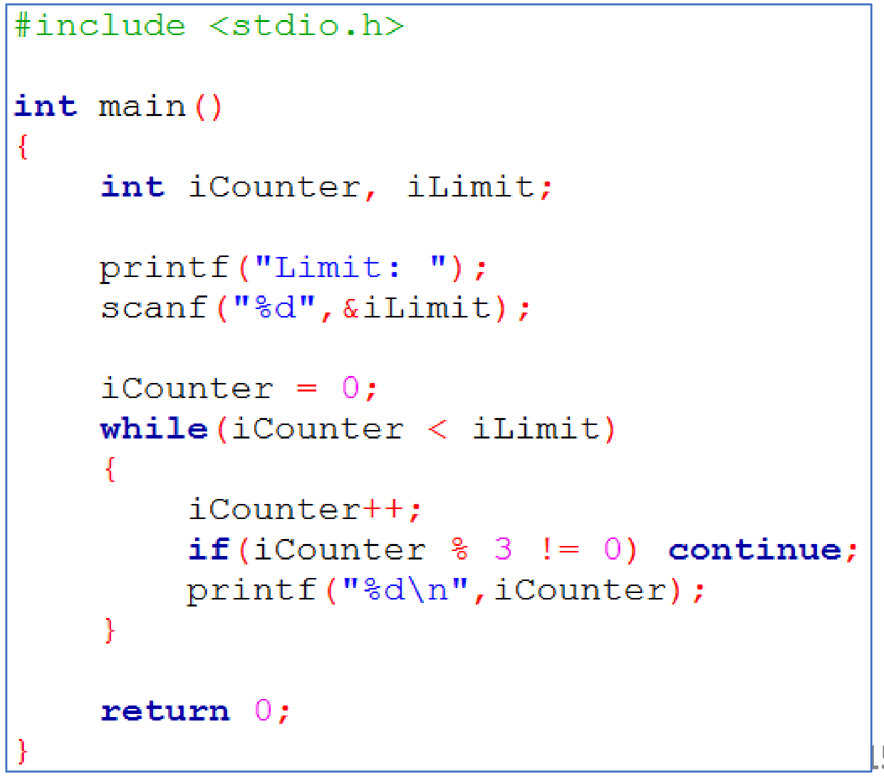
- The **continue** statement, when executed in a **while**, **do...while**, or **for** statement, **skips** the remaining statements in the body of that control statement and **performs the next iteration** of the loop
- In **while** and **do...while** statements, the loop repetition condition is evaluated immediately after the continue statement is executed
- In the **for** statement, the update expression is executed, then the loop repetition condition is evaluated



```
F:\continue.exe
Limit: 20
3
6
9
12
15
18

Process returned 0 (0x0)
execution time : 1.132 s
Press any key to continue.
```



```
F:\continue.exe
Limit: 30
3
6
9
12
15
18
21
24
27
30

Process returned 0 (0x0)
execution time : 1.281 s
Press any key to continue.
```

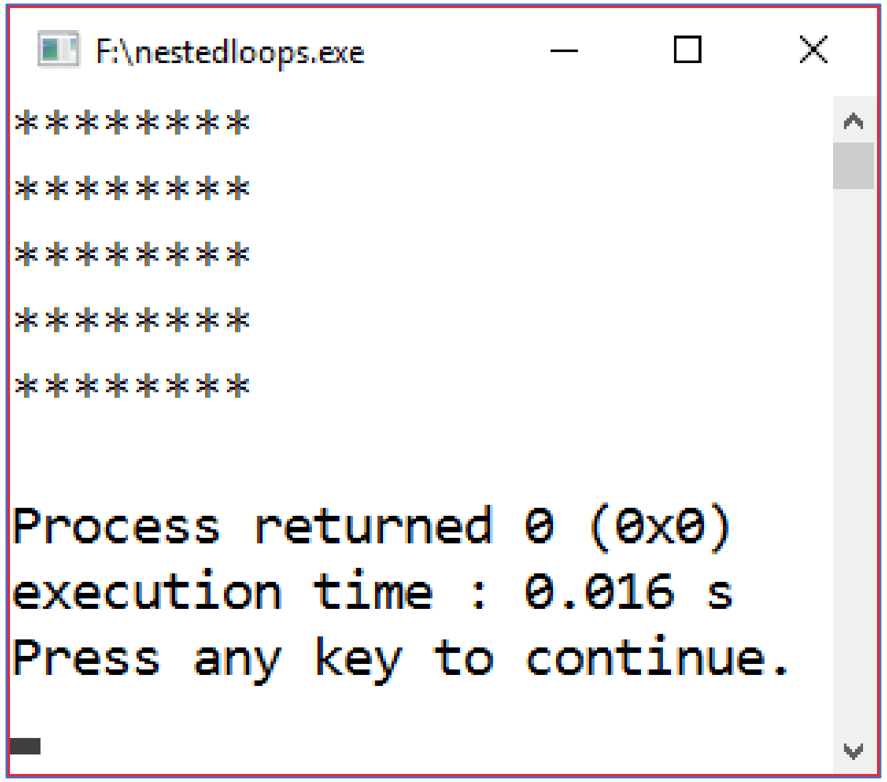


```
#include <stdio.h>

int main()
{
    int iCounter, iLimit;

    printf("Limit: ");
    scanf("%d",&iLimit);

    iCounter = 0;
    while(iCounter < iLimit)
    {
        iCounter++;
        if(iCounter % 3 != 0) continue;
        printf("%d\n",iCounter);
    }

    return 0;
}
```

Nested Loops

- Nested loops consist of an outer loop with one or more inner loops
- Each time the outer loop is repeated, the inner loops are reentered, their loop control expressions are reevaluated, and all required iterations are performed

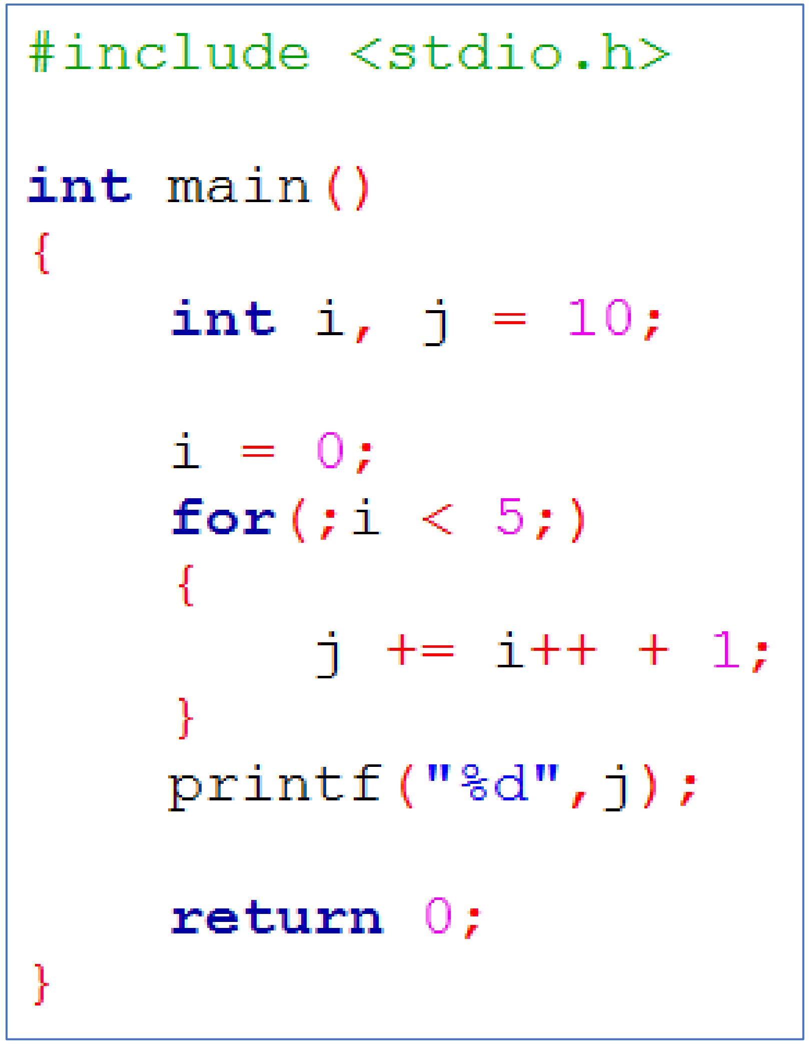
```c
#include <stdio.h>

int main()
{
    int iRow, iColumn;

    for(iRow = 0;iRow < 5;iRow++)
    {
        for(iColumn = 0;iColumn < 8;iColumn++)
        {
            printf("*");
        }
        printf("\n");
    }

    return 0;
}
```


F:\nestedloops.exe

```
********
********
********
********
********


Process returned 0 (0x0)
execution time : 0.016 s
Press any key to continue.
```

Practice 1

- What does the following code print?



```
#include <stdio.h>

int main()
{
    int i, j = 10;

    i = 0;
    for(;i < 5;)
    {
        j += i++ + 1;
    }
    printf("%d",j);

    return 0;
}
```

Practice 2

- What does the following code print?

```
#include <stdio.h>

int main()
{
    int i, j = 0;

    i = 10;
    while(1)
    {
        i--;
        if(i == 0) break;
        if(i % 2 == 0) continue;
        j += i;
    }
    printf("%d",j);

    return 0;
}
```

Practice 3

- What does the following code print?

```
#include <stdio.h>

int main()
{
    int i, j;

    for(i = 0;i < 3;i++)
    {
        for(j = 0;j < 5;j++)
        {
            printf("%3d",5*i+j+1);
        }
        printf("\n");
    }

    return 0;
}
```

Practice 4

- Write **for** statements that print the following sequence of values.

```
20 14 8 2 -4 -10
```

Practice 5

- Write **while** statements that print the following sequence of values.

```
19 27 34 40 45
```

Practice 6

- The factorial function is used frequently in probability problems. The factorial of a positive integer ***n*** (written ***n*!** and pronounced "*n* factorial") is equal to the product of the positive integers from 1 to ***n***. Write a program that evaluates the factorials of the integers from ***p*** to ***q*** (***p*** and ***q*** are inputted by user). The screen dialogue should appear as follows:

```
1 5
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
```

```
3 8
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
```

Practice 7

- Write a program that prints the following pattern. Use for loops to generate the pattern. All asterisks (*) should be printed by a single printf statement of the form **printf("*");**. The screen dialogue should appear as follows:

```
3
***
* *
***
```

```
4
****
*  *
*  *
****
```

```
5
*****
*   *
*   *
*   *
*****
```

```
7
*******
*     *
*     *
*     *
*     *
*     *
*******
```

NEXT WEEK’S OUTLINE

1. Modular Programming
2. Function Prototype
3. Function invocation (void)
4. Variable scope
5. Recursion

REFERENCES

- Hanly, Jeri R. and Koffman, Elliot B., 2013, Problem Solving and Program Design in C, Seventh Edition, Pearson Education, Inc.
- Deitel, Paul and Deitel, Harvey, 2016, C How to Program, Eighth Edition, Pearson Education, Inc.

Visi

Menjadi Program Studi Strata Satu Informatika **unggulan** yang menghasilkan lulusan **berwawasan internasional** yang **kompeten** di bidang Ilmu Komputer (*Computer Science*), **berjiwa wirausaha** dan **berbudi pekerti luhur**.

INFORMATIKA UMN UNIVERSITAS MULTIMEDIA NUSANTARA

Misi

1. Menyelenggarakan pembelajaran dengan teknologi dan kurikulum terbaik serta didukung tenaga pengajar profesional.
2. Melaksanakan kegiatan penelitian di bidang Informatika untuk memajukan ilmu dan teknologi Informatika.
3. Melaksanakan kegiatan pengabdian kepada masyarakat berbasis ilmu dan teknologi Informatika dalam rangka mengamalkan ilmu dan teknologi Informatika.