SINAI UNIVERSITY

CSW 232
Computer Programming (1)

SPRING 2024

Lecture 05 - Repetition II

Instructor: Dr. Tarek Abdul Hamid

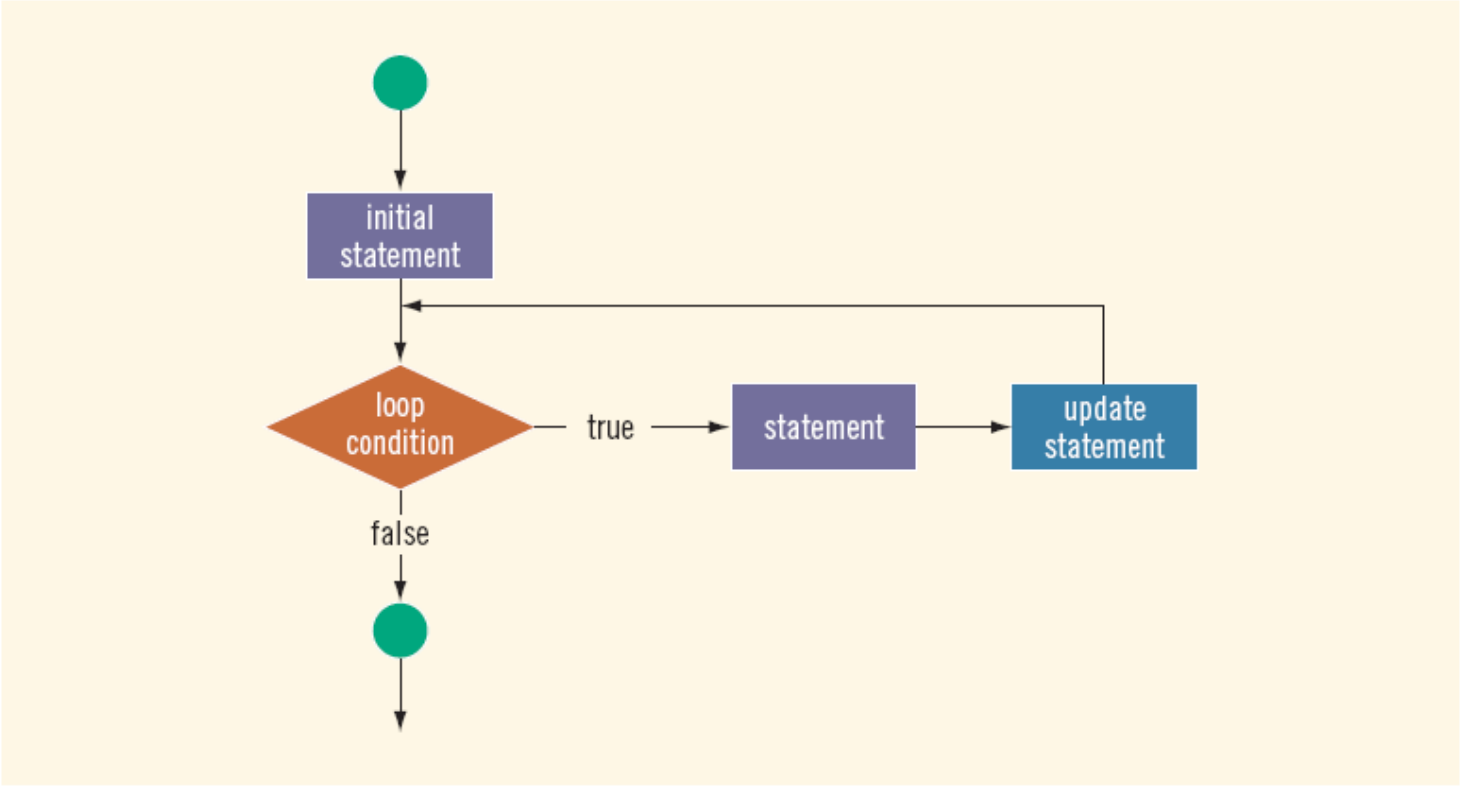
for Looping (Repetition) Structure

- The general form of the `for` statement is:

```
for (initial statement; loop condition; update statement)
    statement
```

- The `initial statement`, `loop condition`, and `update statement` are called `for` loop control statements
  - `initial statement` usually initializes a variable (called the `for` **loop control**, or `for` **indexed**, **variable**)
- In C++, `for` is a reserved word

for Looping (Repetition) Structure



for Looping (Repetition) Structure Example

Write a program to print the first 10 non negative integers

for Looping (Repetition) Structure Example

```
for (i = 0; i < 10; i++)
    cout << i << " ";
cout << endl;
```

for Looping (Repetition) Structure Example

Write a program to outputs Hello! and a star (on separate lines) five times

for Looping (Repetition) Structure Example

1. The following `for` loop outputs `Hello!` and a star (on separate lines) five times:

```
for (i = 1; i <= 5; i++)
{
    cout << "Hello!" << endl;
    cout << "*" << endl;
}
```

2. Consider the following `for` loop:

```
for (i = 1; i <= 5; i++)
    cout << "Hello!" << endl;
    cout << "*" << endl;
```

for Looping (Repetition) Structure

- C++ allows you to use fractional values for loop control variables of the `double` type
  - Results may differ
- The following is a semantic error:

The following `for` loop executes five empty statements:

```
for (i = 0; i < 5; i++);      //Line 1
    cout << "*" << endl;      //Line 2
```

- The following is a legal `for` loop:

```
for (;;)
    cout << "Hello" << endl;
```

for Looping (Repetition) Structure

SU SINAI UNIVERSITY

You can count backward using a **for** loop if the **for** loop control expressions are set correctly.

For example, consider the following **for** loop:

```
for (i = 10; i >= 1; i--)
    cout << " " << i;
cout << endl;
```

The output is:

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

You can increment (or decrement) the loop control variable by any fixed number. In the following **for** loop, the variable is initialized to 1; at the end of the **for** loop, i is incremented by 2. This **for** loop outputs the first 10 positive odd integers.

```
for (i = 1; i <= 20; i = i + 2)
    cout << " " << i;
cout << endl;
```

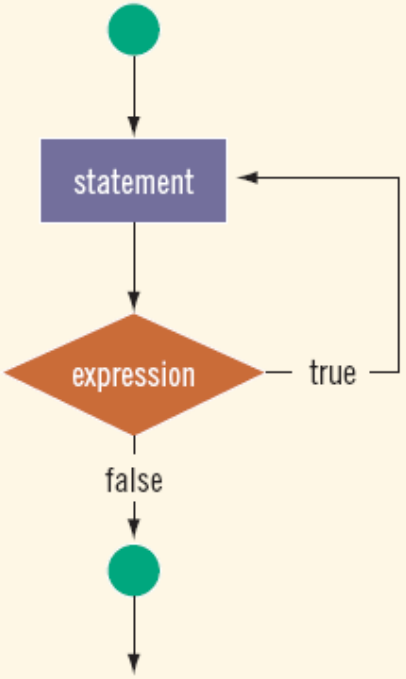
do...while Looping (Repetition) Structure

- General form of a do...while:

```
do
    statement
while (expression);
```

- The statement executes first, and then the expression is evaluated
- To avoid an infinite loop, body must contain a statement that makes the expression false
- The statement can be simple or compound
- Loop always iterates at least once

do...while Looping (Repetition) Structure



do...while Looping (Repetition) Structure

```
i = 0;

do
{
    cout << i << " ";
    i = i + 5;
}
while (i <= 20);
```

The output of this code is:

```
0 5 10 15 20
```

do...while Looping (Repetition) Structure

Consider the following two loops:

a.
```
i = 11;
while (i <= 10)
{
    cout << i << " ";
    i = i + 5;
}
cout << endl;
```

b.
```
i = 11;
do
{
    cout << i << " ";
    i = i + 5;
}
while (i <= 10);

cout << endl;
```

In (a), the `while` loop produces nothing. In (b), the `do...while` loop outputs the number 11 and also changes the value of `i` to 16.

Example

Write a program to check the Divisibility Test by 3 and 9

Example

```
sum = 0;

do
{
    sum = sum + num % 10; //extract the last digit
                          //and add it to sum
    num = num / 10;       //remove the last digit
}
while (num > 0);

cout << "The sum of the digits = " << sum << endl;

if (sum % 3 == 0)
    cout << temp << " is divisible by 3" << endl;
else
    cout << temp << " is not divisible by 3" << endl;

if (sum % 9 == 0)
    cout << temp << " is divisible by 9" << endl;
else
    cout << temp << " is not divisible by 9" << endl;
```

Choosing the Right Looping Structure

- All three loops have their place in C++
  - If you know or can determine in advance the number of repetitions needed, the `for` loop is the correct choice
  - If you do not know and cannot determine in advance the number of repetitions needed, and it could be zero, use a `while` loop
  - If you do not know and cannot determine in advance the number of repetitions needed, and it is at least one, use a `do...while` loop

`break` and `continue` Statements

- `break` and `continue` alter the flow of control
- `break` statement is used for two purposes:
  - To exit early from a loop
    - Can eliminate the use of certain (flag) variables
  - To skip the remainder of the `switch` structure
- After the `break` statement executes, the program continues with the first statement after the structure

`break` and `continue` Statements

- `continue` is used in `while`, `for`, and `do...while` structures
- When executed in a loop
  - It skips remaining statements and proceeds with the next iteration of the loop

Example

Write a program To create the following pattern:

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

Example

```
for (i = 1; i <= 5 ; i++)
{
      for (j = 1; j <= i; j++)
            cout << "*";
      cout << endl;
}
```

Nested Control Structures

- What is the result if we replace the first `for` statement with the following?

```
for (i = 5; i >= 1; i--)
```

- Answer:

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


Thanks!