



Fundamentals of programming I

Lab 3

Switch Statements

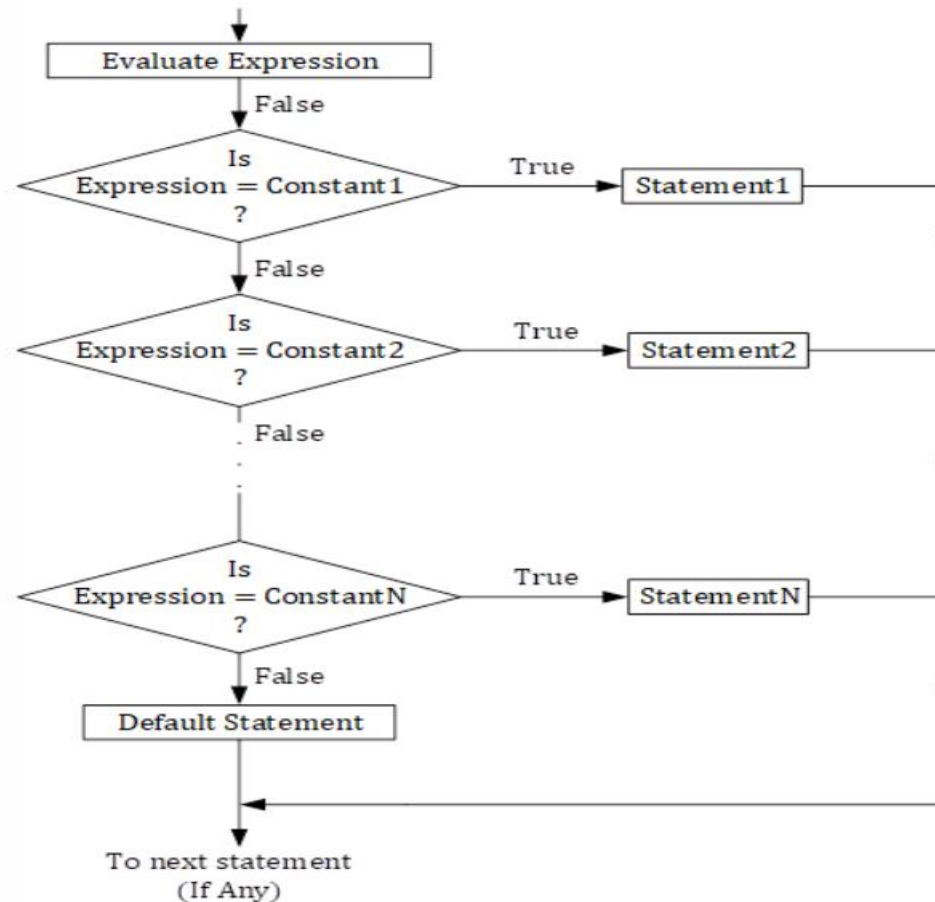
- Use the switch statement to select one of many code blocks to be executed.


Syntax

```
switch(expression) {  
  case x:  
    // code block  
    break;  
  case y:  
    // code block  
    break;  
  default:  
    // code block  
}
```

This is how it works:

- The switch expression is evaluated once
- The value of the expression is compared with the values of each case
- If there is a match, the associated block of code is executed





Rules that must be remembered while using switch statement:

- Expression and constant must be of type either integer or character.
- All the cases must be distinct.
- The block of statement under default will be executed when none of the cases match the value of expression.
- The break statement causes exit from switch statement. If it is not present, after executing the case which match the value of the expression, the following other cases are executed.

Example

```
int day = 4;
switch (day) {
    case 1:
        cout << "Monday";
        break;
    case 2:
        cout << "Tuesday";
        break;
    case 3:
        cout << "Wednesday";
        break;
    case 4:
        cout << "Thursday";
        break;
    case 5:
        cout << "Friday";
        break;
    case 6:
        cout << "Saturday";
        break;
    case 7:
        cout << "Sunday";
        break;
}
// Outputs "Thursday" (day 4)
```



Example


```
int day = 4;
switch (day) {
    case 6:
        cout << "Today is Saturday";
        break;
    case 7:
        cout << "Today is Sunday";
        break;
    default:
        cout << "Looking forward to the Weekend";
}
// Outputs "Looking forward to the Weekend"
```

Example

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int x=5;
5
6      switch (x/2)
7      {
8          case 1:
9              cout << "Choice 1"; break;
10         case 2:
11             cout << "Choice 2"; break;
12         case 3:
13             cout << "Choice 3"; break;
14         default:
15             cout << "Not 1, 2 or 3"; break;
16     }
17 }
18
```

Output

Choice 2



Control Statements : Repetition statements

- There are three methods by way of which we can repeat a part of a program.
- They are:
 - (a) Using a while statement.
 - (b) Using a do-while statement.
 - (C) Using a for statement.

While Loop

- The while loop loops through a block of code as long as a specified condition is true:

Syntax

```
initialization expression;  
while (test_expression)  
{  
    // statements  
  
    update_expression;  
}
```

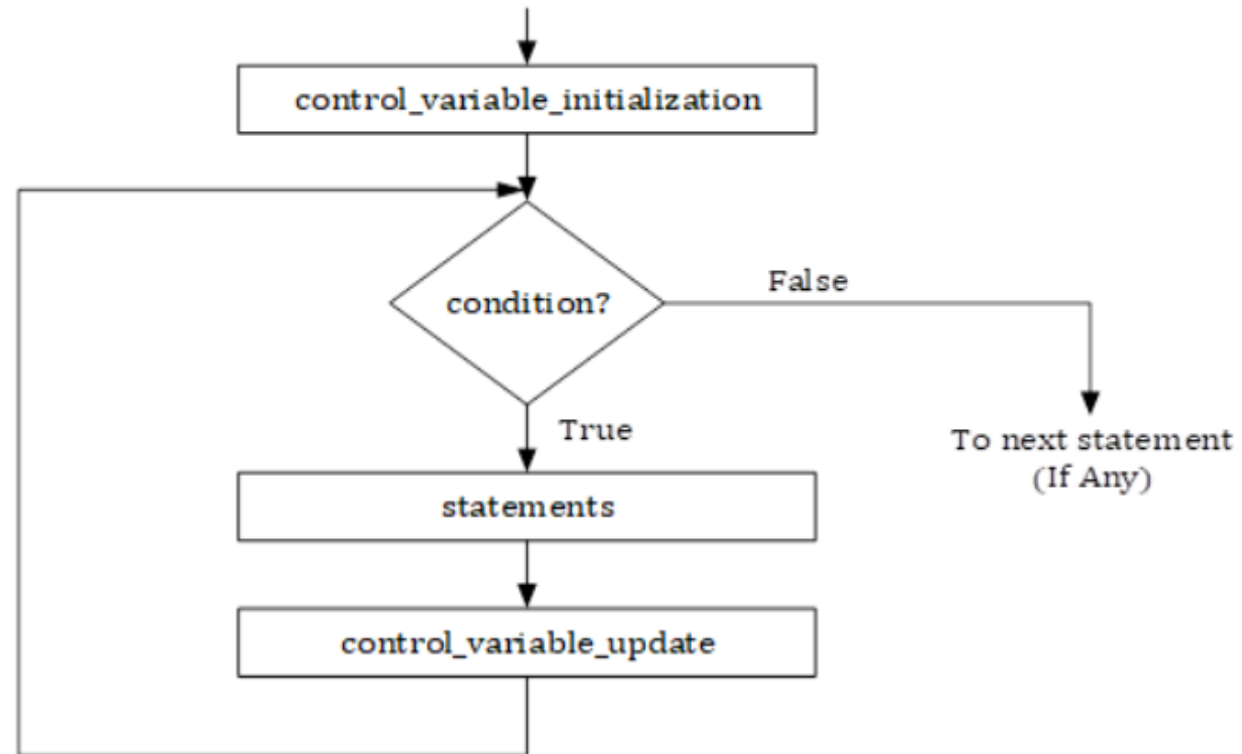
- loop counter is any numeric variable (int , float ,....).
- The initial value of the loop counter is up to you, but you must increment it to avoid endless loops.
- The condition being tested may be relational, logical or even numeric.

while (i <= 10)

while (i >= 10 && j <= 15)

While (3) , While (3 + 5)

while Loop Flowchart



Example

```
int i = 0;
while (i < 5) {
    cout << i << "\n";
    i++;
}
```

Output

0
1
2
3
4



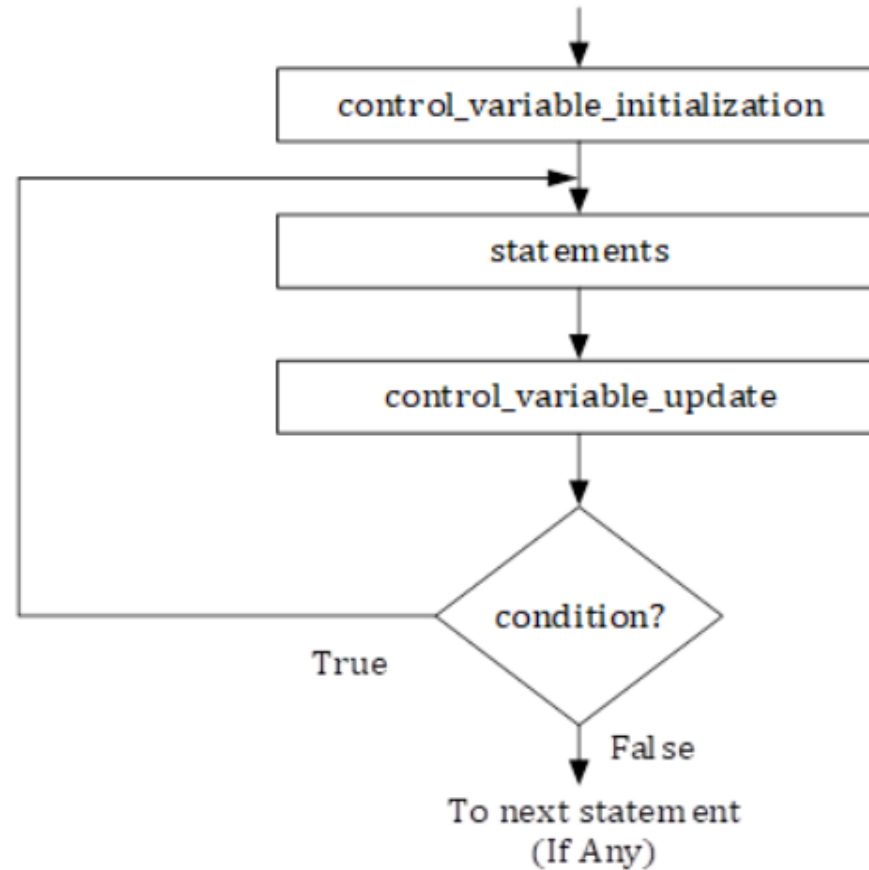
The do-While Loop

- The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

```
initialization expression;  
do{  
    // statements  
  
    update_expression;  
} while (test_expression);
```

do-while Loop Flowchart





Example

```
int i = 0;  
do {  
    cout << i << "\n";  
    i++;  
}  
while (i < 5);
```

Output

0
1
2
3
4



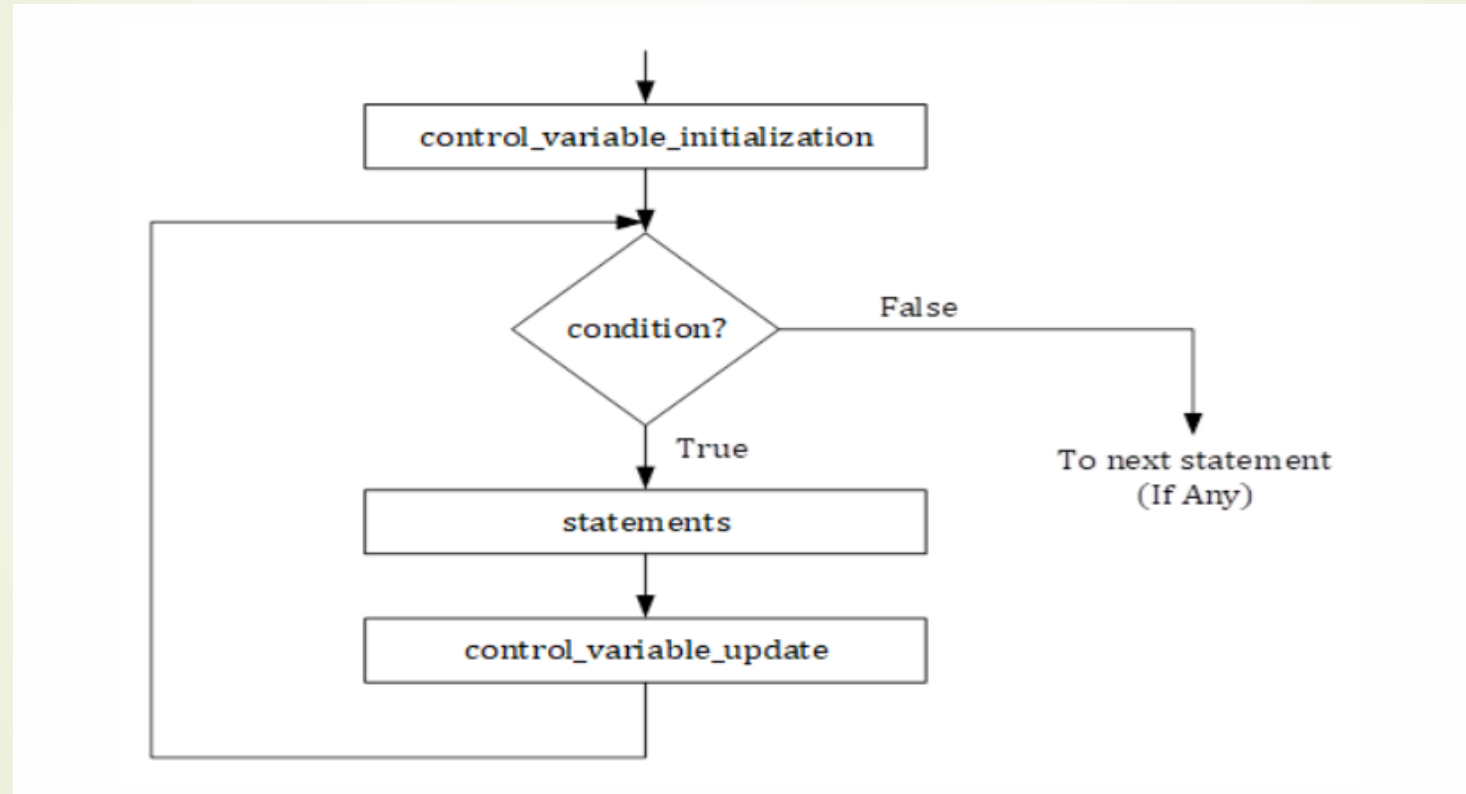
For Loop

- When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop:

Syntax

```
for (initialization expr; test expr; update expr)
{
    // body of the loop
    // statements we want to execute
}
```

For Loop Flowchart





Example

```
for (int i = 0; i < 5; i++) {  
    cout << i << "\n";  
}
```

Output

0
1
2
3
4

Example

- This example will only print even values between 0 and 10

```
for (int i = 0; i <= 10; i = i + 2) {  
    cout << i << "\n";  
}
```

Output

0
2
4
6
8
10

Example

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      int n=0, ceven=0, codd=0;
7      for (int i = 1; i <= 7; ++i) {
8          cout<<"Enter the number"<<endl;
9          cin>>n;
10         if(n%2==0){
11             ceven++;
12         }
13         else if(n%2!=0){
14             codd++;
15         }
16     }
17
18     cout<<"the odd number ="<<codd<<endl;
19     cout<<"the even number="<<ceven;
20     return 0;
21 }
22
23
```

Output

```
Enter the number
34
Enter the number
56
Enter the number
12
Enter the number
65
Enter the number
23
Enter the number
87
Enter the number
21
the odd number =4
the even number=3
```



The foreach Loop

- which is used exclusively to loop through elements in an array (or other data sets).

Syntax

```
for (type variableName : arrayName) {  
    // code block to be executed  
}
```

Example

```
int myNumbers[5] = {10, 20, 30, 40, 50};  
for (int i : myNumbers) {  
    cout << i << "\n";  
}
```

Output

10
20
30
40
50

Nested Loops

- A loop within another loop is called a nested loop.

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int weeks = 3, days_in_week = 7;
6
7      for (int i = 1; i <= weeks; ++i) {
8          cout << "Week: " << i << endl;
9
10         for (int j = 1; j <= days_in_week; ++j) {
11             cout << "    Day:" << j << endl;
12         }
13     }
14
15     return 0;
16 }
17
18
```

Output

```
Week: 1
    Day:1
    Day:2
    Day:3
    Day:4
    Day:5
    Day:6
    Day:7
Week: 2
    Day:1
    Day:2
    Day:3
    Day:4
    Day:5
    Day:6
    Day:7
Week: 3
    Day:1
    Day:2
    Day:3
    Day:4
    Day:5
    Day:6
    Day:7
```

Example

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      for (int i = 1; i <= 5; ++i) {
7
8          for (int j = 1; j <= 6; ++j) {
9              cout << "*";
10             }
11             cout<<endl;
12         }
13
14         return 0;
15     }
16
17
```

Output

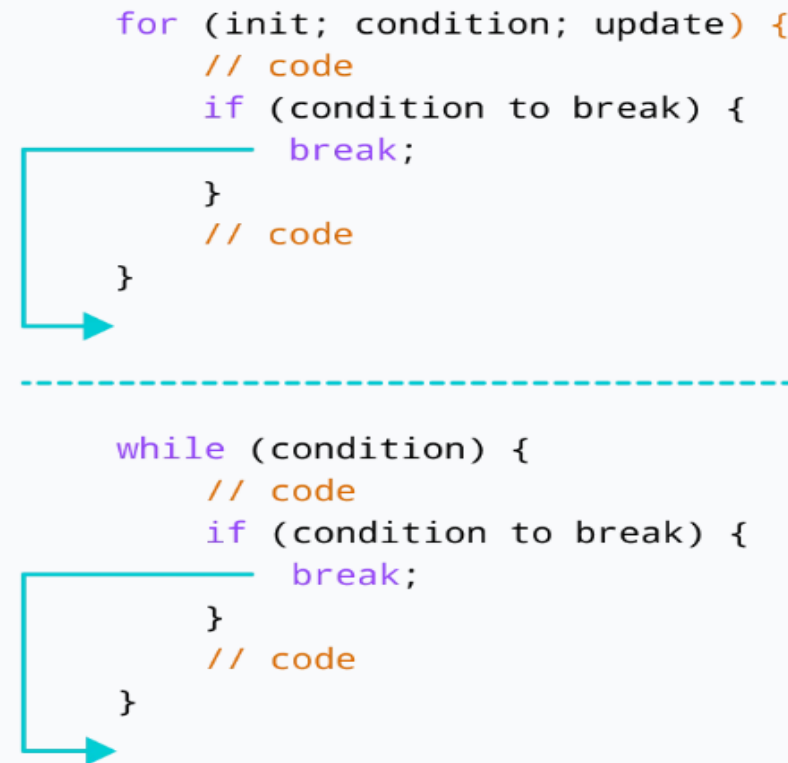
```
*****
*****
*****
*****
*****
```

break

- It was used to "jump out" of a switch statement.
- The break statement can also be used to jump out of a loop.

```
for (init; condition; update) {  
    // code  
    if (condition to break) {  
        break;  
    }  
    // code  
}
```

```
while (condition) {  
    // code  
    if (condition to break) {  
        break;  
    }  
    // code  
}
```



break with for loop

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5  for (int i = 0; i < 10; i++) {
6      if (i == 4) {
7          break;
8      }
9      cout << i << "\n";
10 }
11
12     return 0;
13 }
14
15
```

Output

0
1
2
3

break with while loop

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int i = 0;
6      while (i < 10) {
7
8          if (i == 4)
9              break;
10
11         cout << i << "\n";
12         i++;
13     }
14
15     return 0;
16 }
17
```

Output

0
1
2
3

break with Nested loop

```
#include <iostream>
using namespace std;

int main() {
    int number;
    int sum = 0;

    // nested for loops

    // first loop
    for (int i = 1; i <= 3; i++) {
        // second loop
        for (int j = 1; j <= 3; j++) {
            if (i == 2) {
                break;
            }
            cout << "i = " << i << ", j = " << j << endl;
        }
    }

    return 0;
}
```

Output

```
i = 1, j = 1
i = 1, j = 2
i = 1, j = 3
i = 3, j = 1
i = 3, j = 2
i = 3, j = 3
```

Continue

- The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

```
for (init; condition; update) {  
    // code  
    if (condition to continue) {  
        continue;  
    }  
    // code  
}
```

```
while (condition) {  
    // code  
    if (condition to continue) {  
        continue;  
    }  
    // code  
}
```

continue with for loop

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      for (int i = 1; i <= 5; i++) {
7          // condition to continue
8          if (i == 3) {
9              continue;
10         }
11
12         cout << i << endl;
13     }
14
15     return 0;
16 }
17
18
19
```

Output

1
2
4
5

continue with while loop

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int i = 0;
6      while (i < 10) {
7          if (i == 4) {
8              i++;
9              continue;
10         }
11         cout << i << "\n";
12         i++;
13     }
14
15     return 0;
16 }
17
18
19
```

Output

0
1
2
3
5
6
7
8
9

continue with Nested loop

```
#include <iostream>
using namespace std;

int main() {
    int number;
    int sum = 0;

    // nested for loops

    // first loop
    for (int i = 1; i <= 3; i++) {
        // second loop
        for (int j = 1; j <= 3; j++) {
            if (j == 2) {
                continue;
            }
            cout << "i = " << i << ", j = " << j << endl;
        }
    }

    return 0;
}
```

Output

```
i = 1, j = 1
i = 1, j = 3
i = 2, j = 1
i = 2, j = 3
i = 3, j = 1
i = 3, j = 3
```



Thank You !