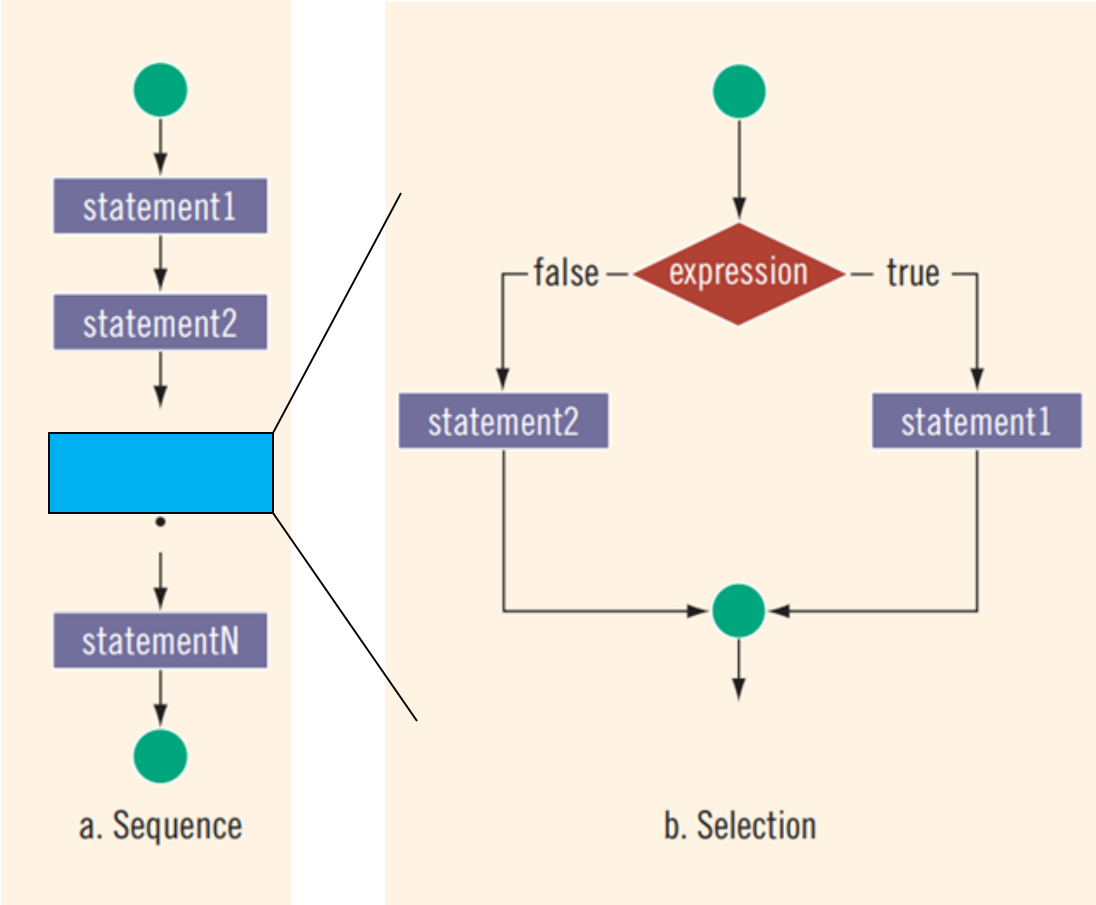
Programming Principles (MT162)

Lecture 5

Dr. Ahmed Fathalla

Control Structures I (Selection)

Control Structures



Control Structures

- A computer can proceed:
  - In sequence
  - Selectively (branch) - making a choice
  - Repetitively (iteratively) - looping
- Some statements are executed **Only If** certain **conditions** are met.
- A **condition** is met if it evaluates to `true`.

Relational Operators

- The following Table lists the C+ + relational operators.

- Relational operators:
  - Allow comparisons
  - Require two operands (binary)
  - Evaluate to `true` or `false`

Operator	Description
(==)	equal to
(!=)	not equal to
(<)	less than
(<=)	less than or equal to
(>)	greater than
(>=)	greater than or equal to

Logical Operators (two or more logical expressions)

Logical (Boolean) operators enable you to combine logical expressions.

Operator	Description
(!)	not
(&&)	and
(II)	or

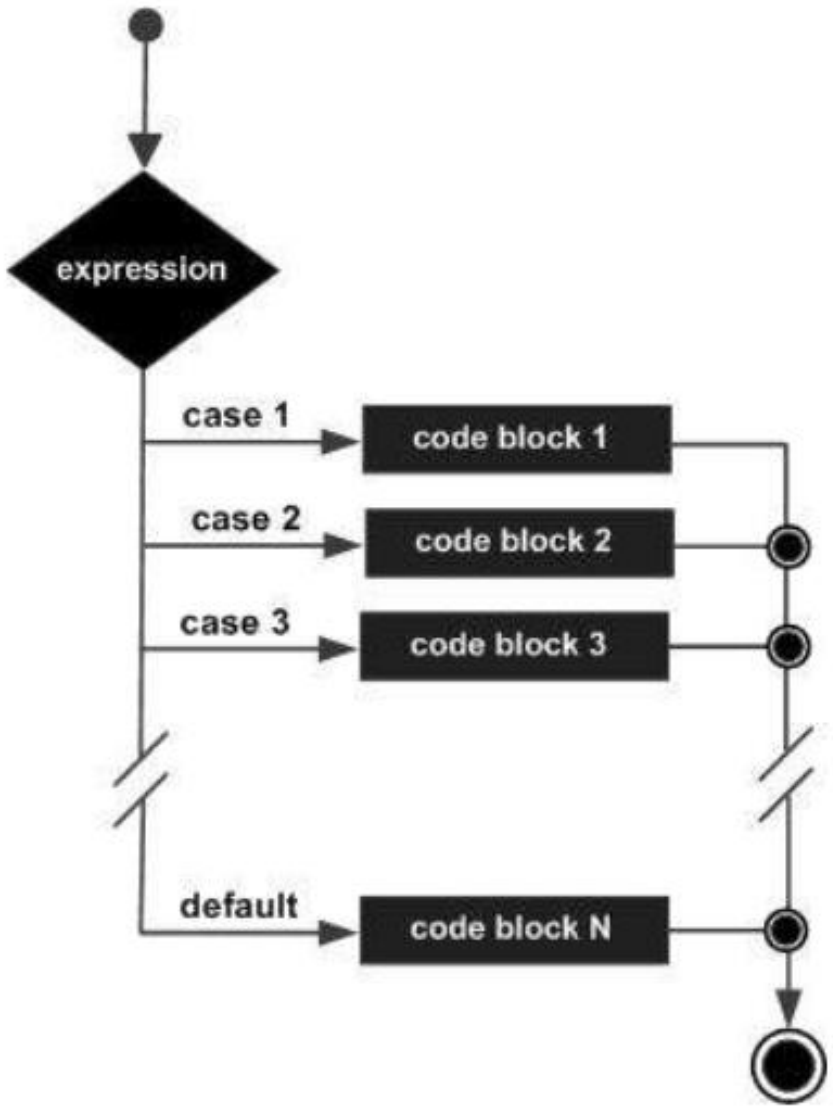
INPUT	OUTPUT
0	1
1	0

INPUT 1	INPUT 2	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	1

INPUT 1	INPUT 2	OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

Switch statement

```
switch (expression)
{
case value1:
    statements1
    break;
case value2:
    statements2
    break;
    .
    .
    .
case valuen:
    statementsn
    break;
default:
    statements
}
```



Exercise_1: write a Program to display month name according to the month number.

```
int main()
{
        int month;
        cout<<" Enter a number from 1-6.";
        cin>>month;
        switch (month)
            {
              case 1: cout<< "The month is January";   break;
              case 2:cout<< "The month is February";  break;
              case 3:cout<<"The month is March";       break;
              case 4:cout<<"The month is April";         break;
              case 5:cout<<"The month is May";         break;
              case 6:cout<<"The month is June";         break;
             }
        return 0;
}
```

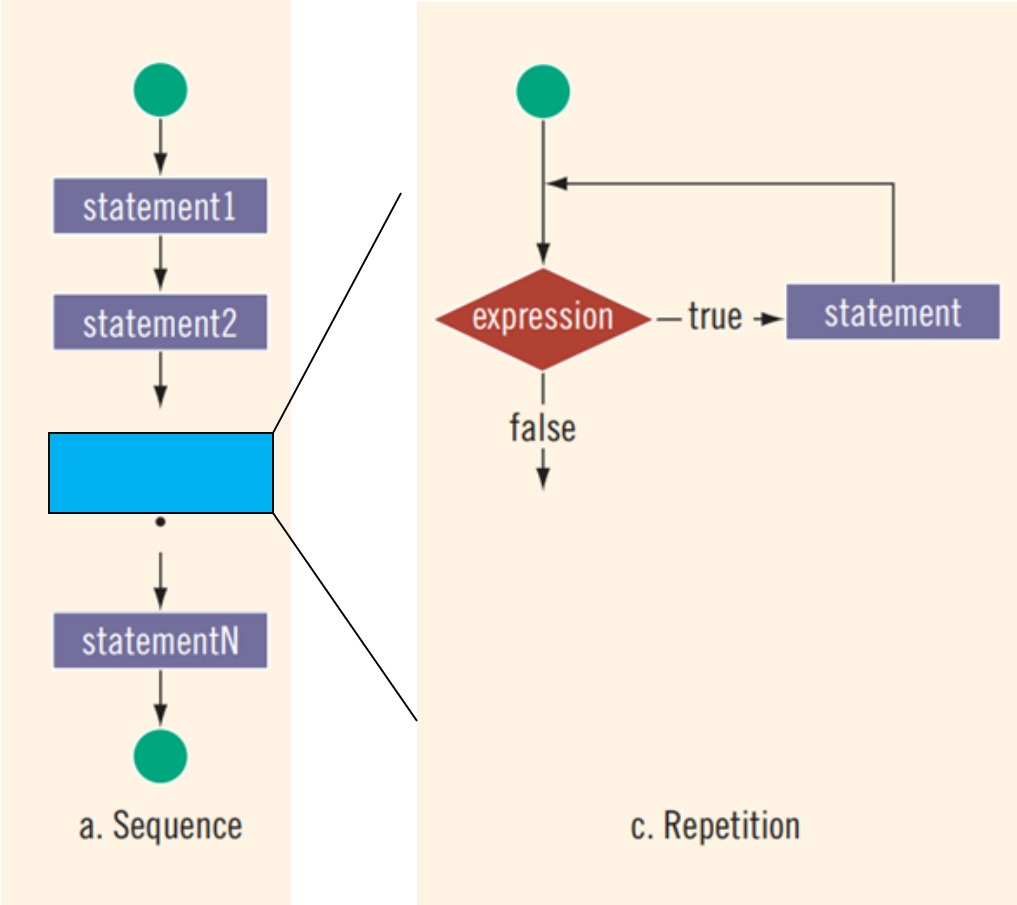
Exercise_2:

write a program to ask the user for an Operator and Two Operands, then Perform the Operation. (using if-else and Switch statements)

```
int main()
{
   float a,b; char op;
   cout<<"Enter two real numbers followed by one these characters:+, -, *, or /: ";
   cin>>a>>b>>op;
   switch(op)
   {
      case '+':cout<<a<<" + "<<b<<" = "<<a+b;break;
      case '-':cout<<a<<" - "<<b<<" = "<<a-b;break;
      case '*':cout<<a<<" * "<<b<<" = "<<a*b;break;
      case '/':
               if (b==0)
                  {
                   cout<<"Can not divide by 0";   break;
                  }
               cout<<a<<" / "<<b<<" = "<<a/b;
   }
  return 0;
}
```

Control Structures II (Repetition)

Control Structures

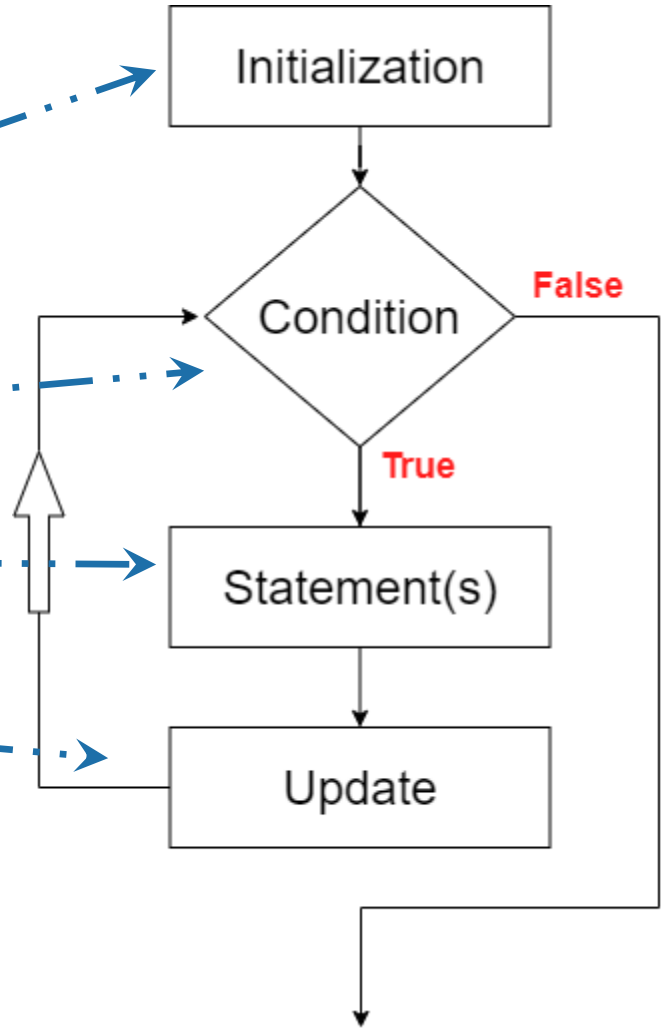


Why Is Repetition Needed?

- Repetition allows you to efficiently use variables.
- Can input, add, and average multiple numbers using a limited number of variables
- For example, to add five numbers:
  - Declare a variable for each number, input the numbers and add the variables together
  - Create a loop that reads a number into a variable and adds it to a variable that contains the sum of the numbers

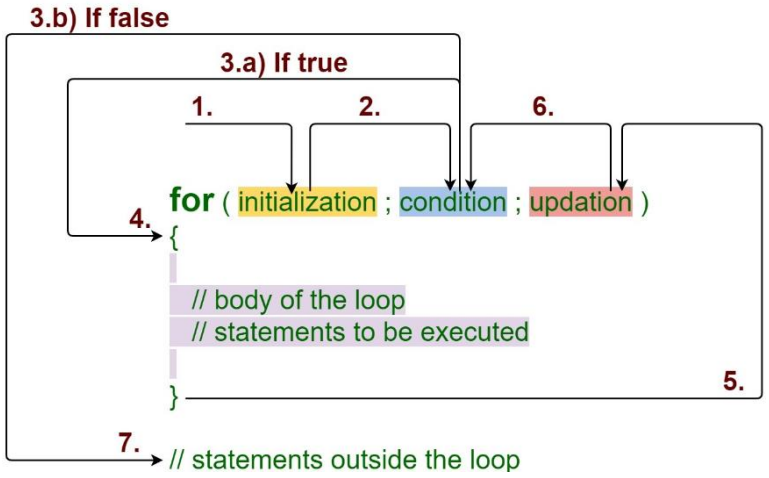
Loops

- Main components
  - Initialization.
  - Condition.
  - Statement(s) (What to do)
  - Update.

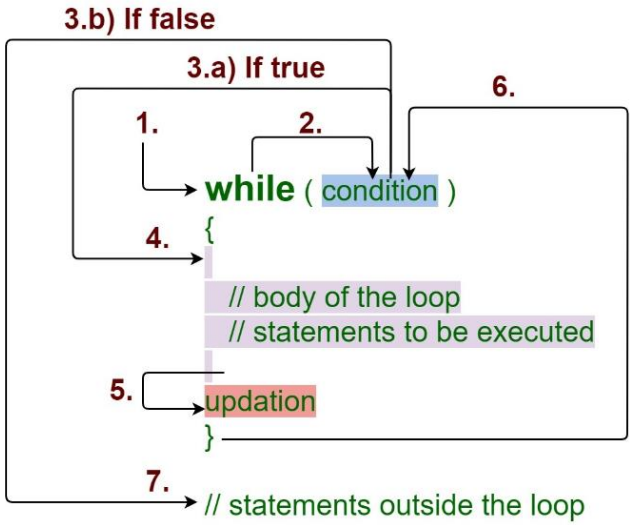


Loop types

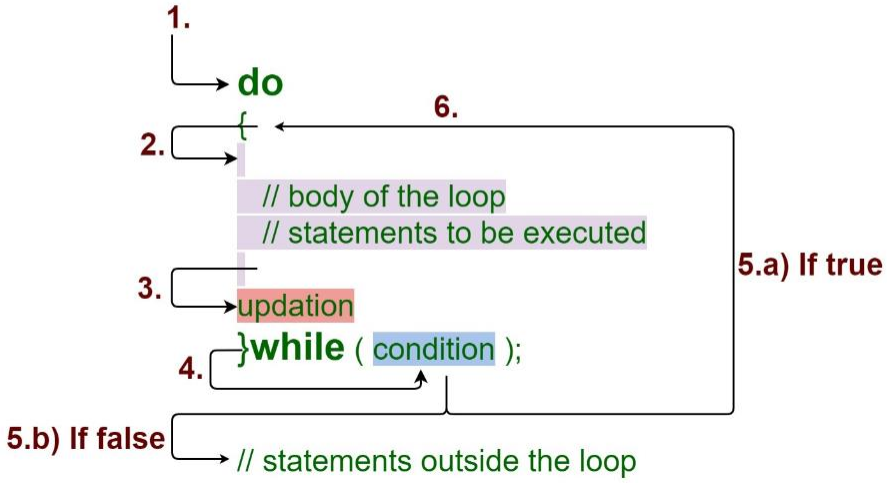
For Loop



While Loop



Do - While Loop

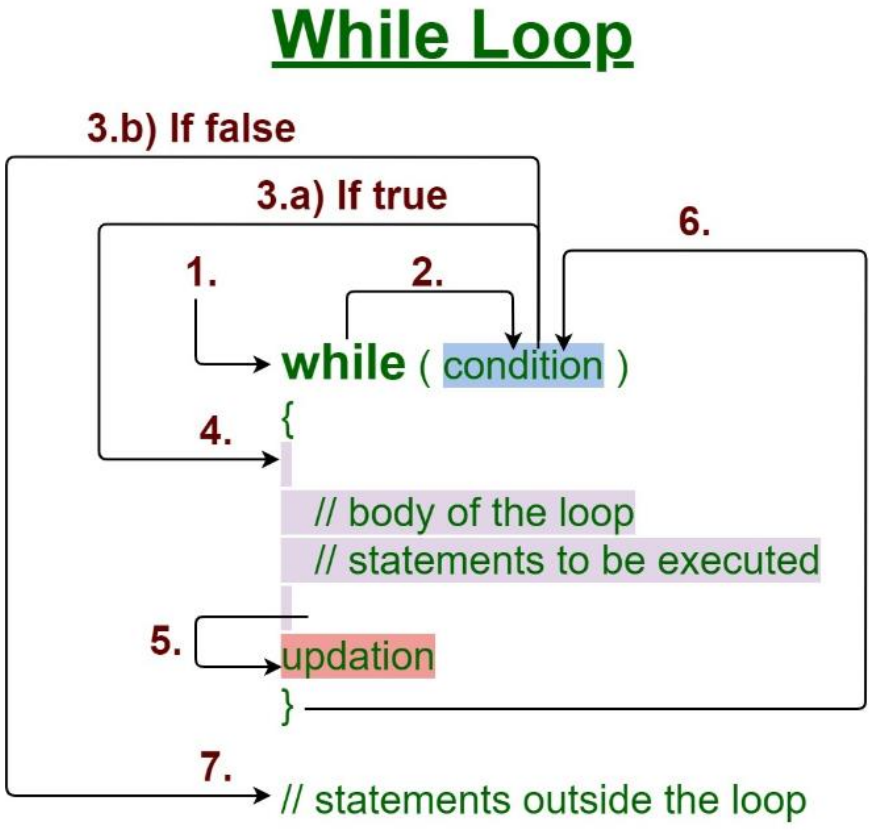


`while` Looping (Repetition) Structure

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

```
while (expression)
    statement
```

  `while` is a reserved word
- Statement can be simple or compound
- Expression acts as a decision maker and is usually a logical expression
- Statement is called the **body of the loop**
- The parentheses are part of the syntax

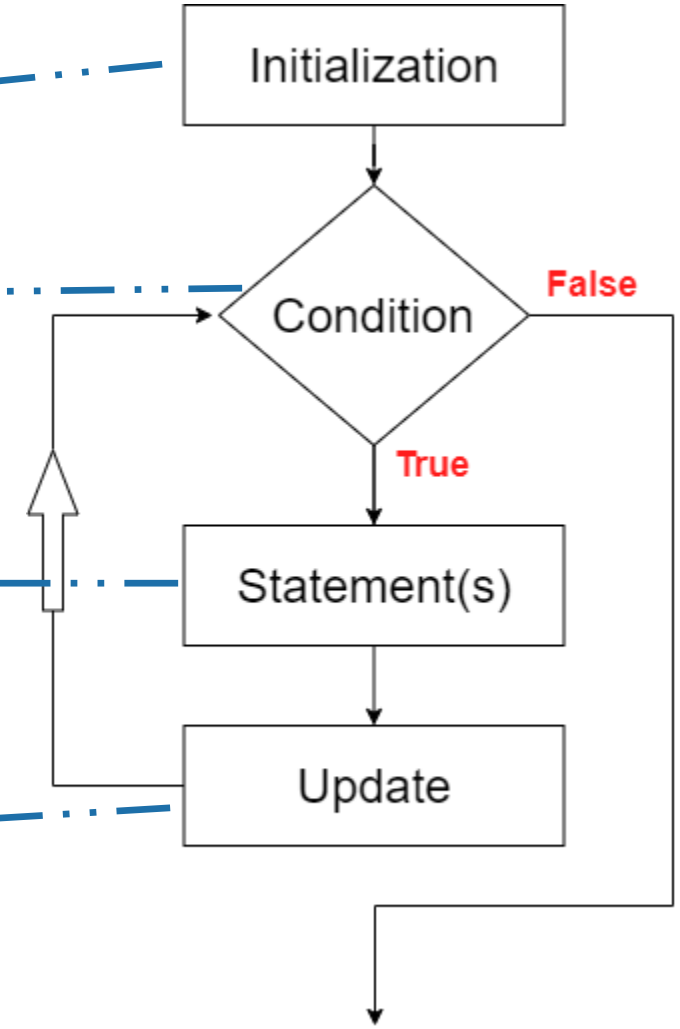


While loop

```
Initialization;
while (condition)
{
        statement_1;
        statement_2;
        statement_n;

        update;
}
```

body of the loop



Exercise_1: Print numbers between [1-100].

```
int main()
{
    int i = 1;
    while (i<=10)
    {
        cout<<i<<endl;
        i += 1;
    }
}
```

Exercise_2: Print "**Hello World**" 100 times.

```
int main()
{
    int i = 1;
    while (i<=10)
    {
        cout<<"Hello world!"<<endl;
        i += 1;
    }
}
```