Programming Language 1 (MT261)

Lecture 1 & 2

Dr. Ahmed Fathalla

Identifiers

- The C++ identifier is a name used to identify a **variable**, **function**, **class**, or any other user-defined item.
- Consist of letters, digits, and the underscore character (_)
- Must begin with a letter or underscore
- C++ is **case sensitive**
  - `NUMBER` is not the same as `number`

Identifiers

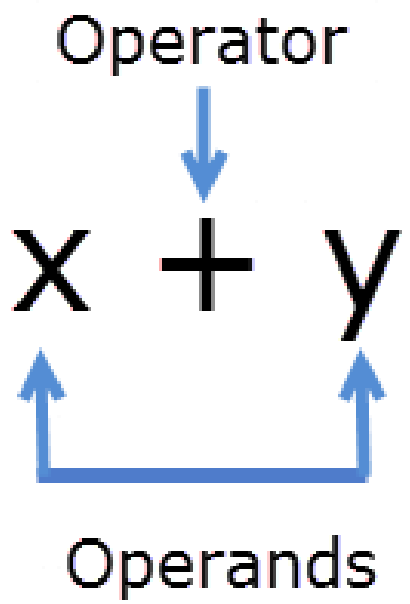
TABLE 2-1 Examples of Illegal Identifiers

Illegal Identifier	Description
`employee Salary`	There can be no space between `employee` and `Salary`.
`Hello!`	The exclamation mark cannot be used in an identifier.
`one+two`	The symbol + cannot be used in an identifier.
`2nd`	An identifier cannot begin with a digit.

Data Types

- char
- bool
- integer (int)
- float
- double

Arithmetic Expression



Arithmetic Operators

- C++ arithmetic operators:
  - + addition
  - - subtraction
  - * multiplication
  - / division
  - % modulus operator
  - ++, -- Increment and Decrement Operators

Arithmetic Operators

- Operators can be unary or binary.

- Examples of **Unary** operators:
  - unary minus(-)
  - increment(++)
  - decrement(- -)
  - NOT(!)

Operator Precedence

- Expression:

  Example: `2 + 3 * 5`

- All operations inside of () are evaluated first
- *, /, and % are at the same level of precedence and are evaluated next
- + and – have the same level of precedence and are evaluated last.

Operator Precedence

```
3 * 7 - 6 + 2 * 5 / 4 + 6
```

- When operators are on the same level
  - Performed from left to right (associativity)

```
(((3 * 7) - 6) + ((2 * 5) / 4 )) + 6
                = 23
```

Expressions

- If all operands are integers
  - Expression is called an **integral expression**
    - Yields an integral result
    - Example: `2 + 3 * 5`
- If all operands are floating-point
  - Expression is called a **floating-point expression**
    - Yields a floating-point result
    - Example: `12.8 * 17.5 - 34.50`

Mixed Expressions

- Mixed expression:
  - Has operands of different data types
  - Contains integers and floating-point
- Examples of mixed expressions:

```
2 + 3.5
6  /  4 + 3.9
5.4  *  2 – 13.6 + 18  /  2
```

Mixed Expressions (continued)

- Evaluation rules:
  - If operator has same types of operands
    - Evaluated according to the type of the operands
  - If operator has both types of operands
    - Integer is changed to floating-point
    - Operator is evaluated
    - Result is floating-point
  - Entire expression is evaluated according to precedence rules

Exercise:

Write a program that takes hours and minutes as input and outputs the total number of minutes (e.g., 1 hour 30 minutes = 90 minutes)

Write a program to compute the area and perimeter of a rectangle 3 inches wide by 5 inches long.

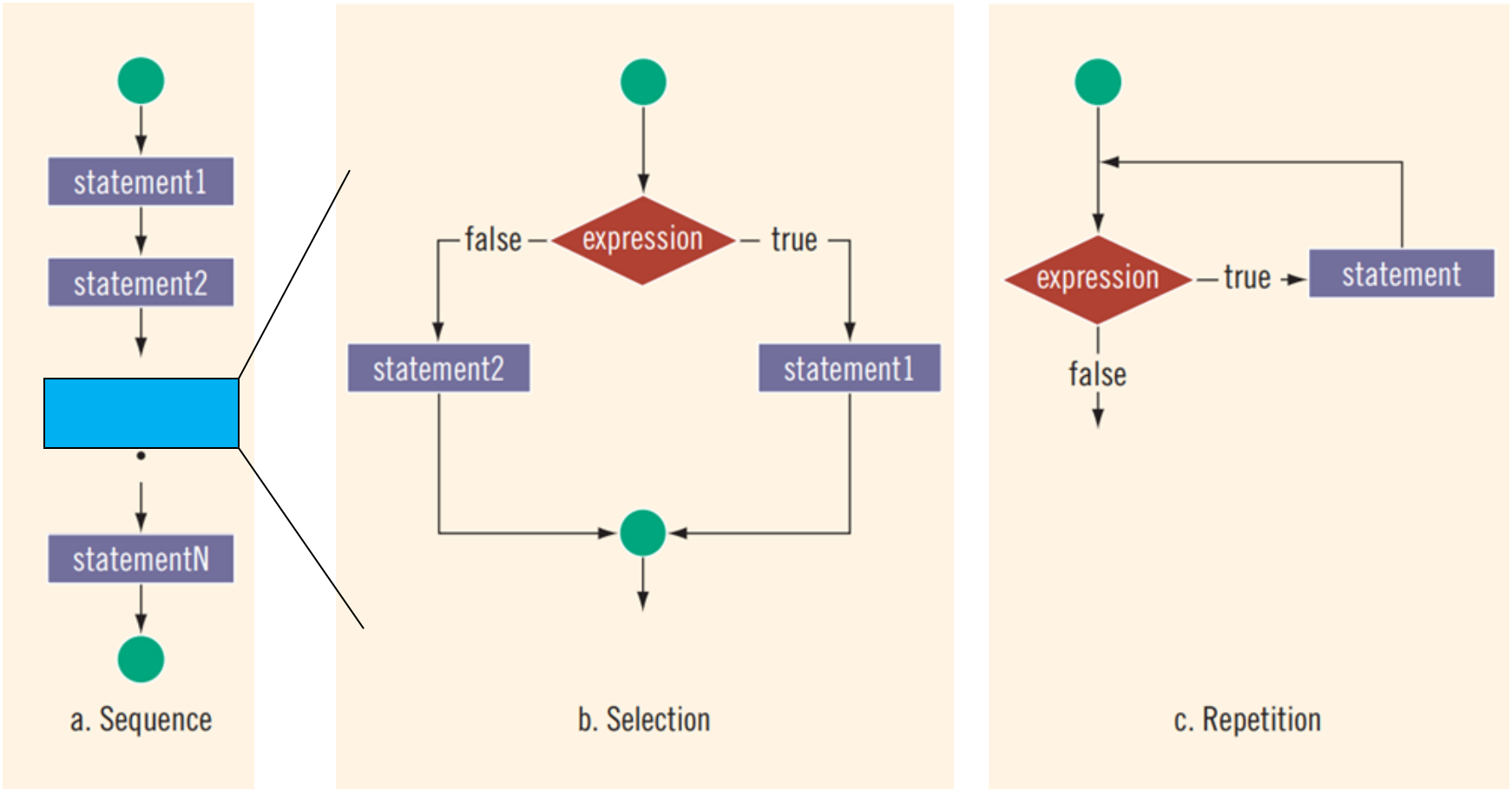
Write a program that converts Celsius to Fahrenheit. The formula is

$$F = {}^{9}/_{5}C + 32.$$

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.`

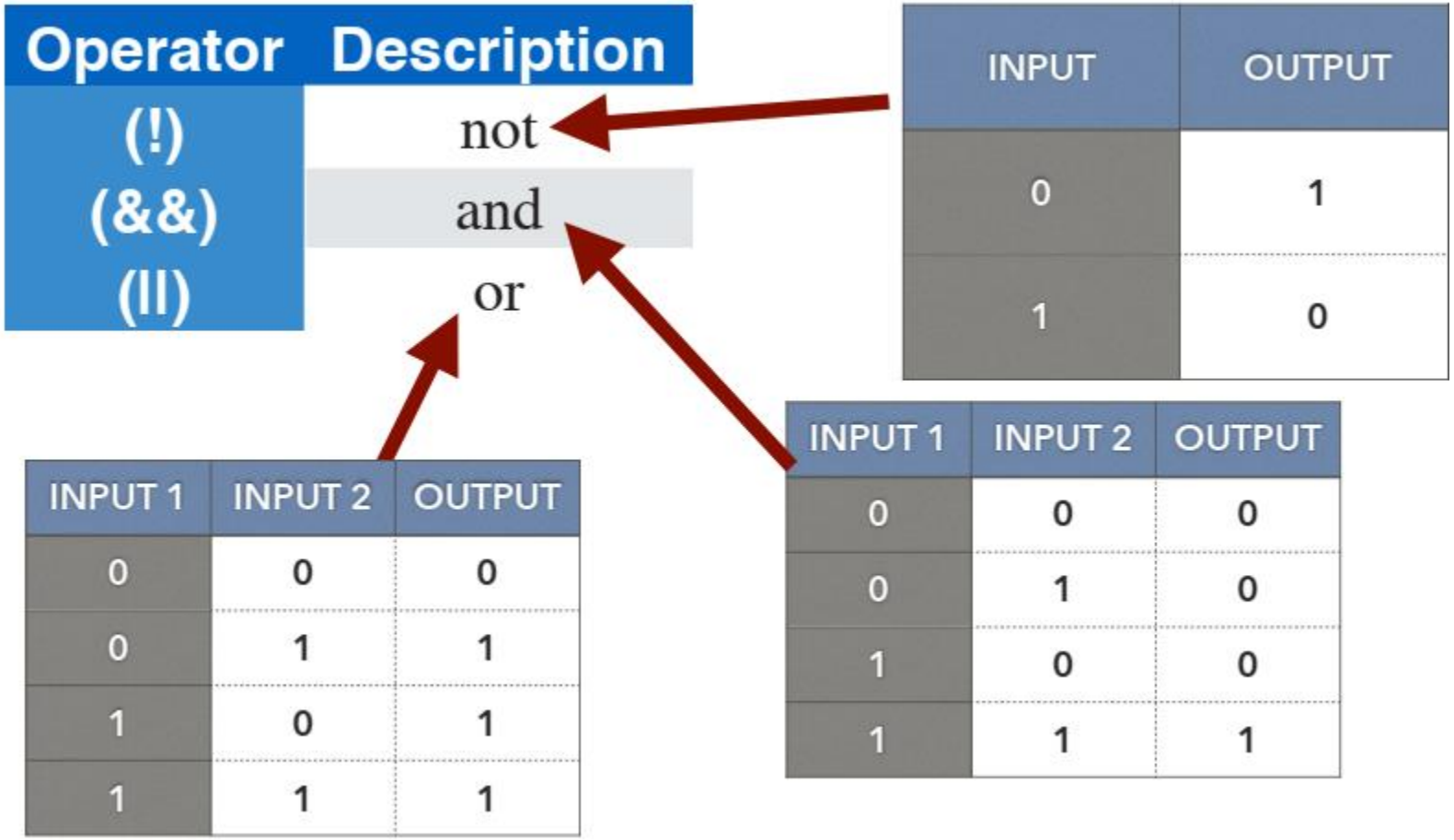
Programs Processing



Relational Operators

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

Logical (Boolean) Operators



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

INPUT	OUTPUT
0	1
1	0

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

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

Complex logical expressions

Operator	Priority
!, +, - (Unary)	first
*, /, %	second
+, -	third
<, <=, >=, >	fourth
!=, ==	fifth
&&	sixth
||	seventh
assignment operator (=)	last

IF … else statement

One way selection

```
if (expression)
    statement
```

Two way selection

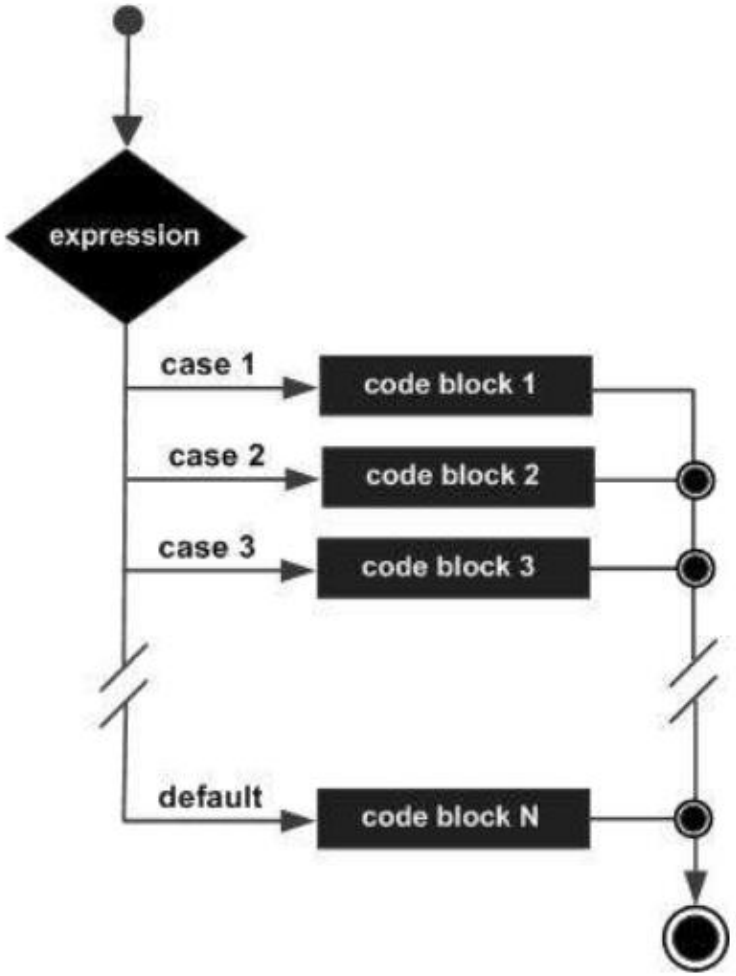
```
if (expression)
    statement1
else
    statement2
```

Suggested IDE/software for C++

- Windows/Desktop
  - Visual studio.
  - CodeBlocks
- Online compiler:
  - https://www.onlinegdb.com/online_c++_compiler
  - https://onecompiler.com/cpp
  - https://riju.codes/cpp
- Mobile application:
  - Cxxdroid - C++ compiler IDE
  - CppDroid - C/C++ IDE

Switch statement

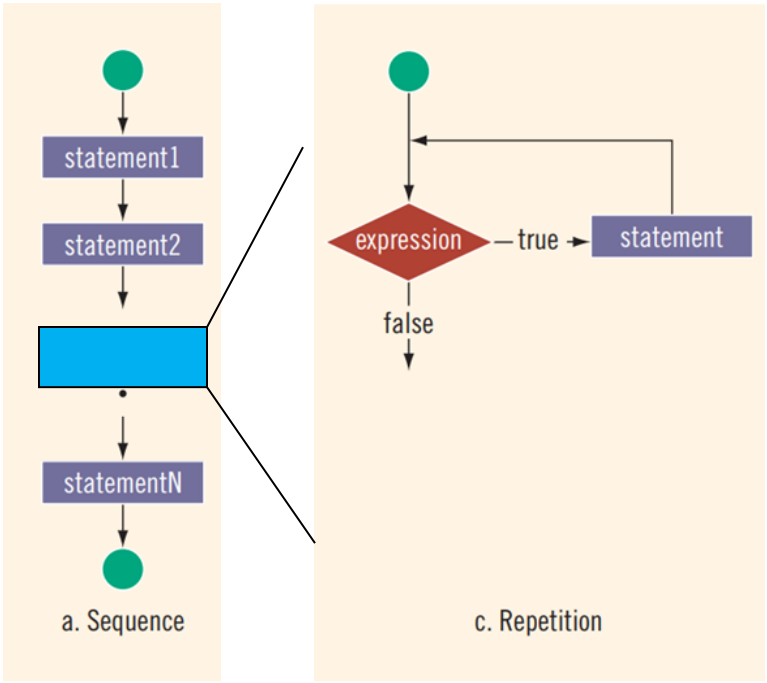
```
switch (expression)
{
case value1:
    statements1
    break;
case value2:
    statements2
    break;
    .
    .
    .
case valuen:
    statementsn
    break;
default:
    statements
}
```



Exercise: 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;
}
```

Control Structures

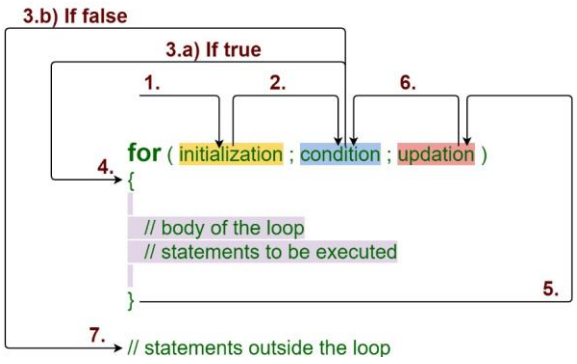


Why Is Repetition Needed?

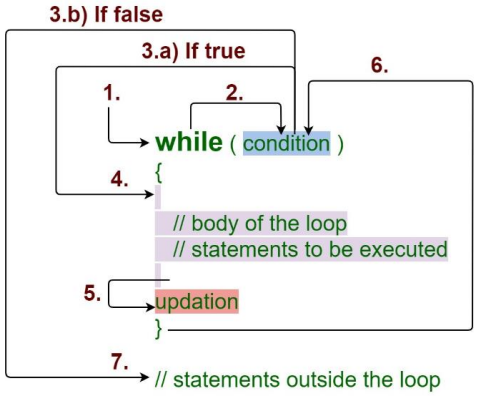
- Repetition allows you to **efficiently use variables**.
- Can input, add, and average multiple numbers using a limited number of variables
- For example, to add five numbers:
  - Declare a variable for each number, input the numbers and add the variables together
  - Create a loop that reads a number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

Loops

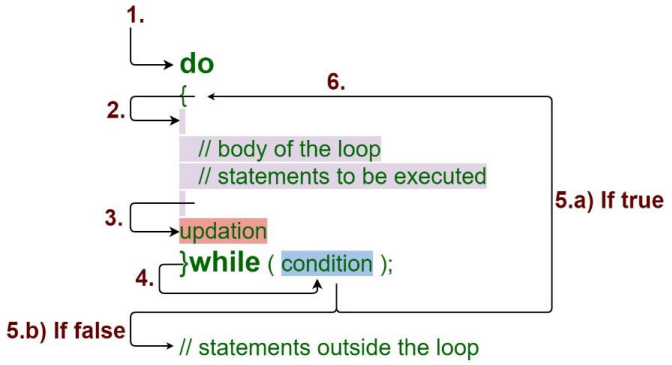
For Loop



While Loop

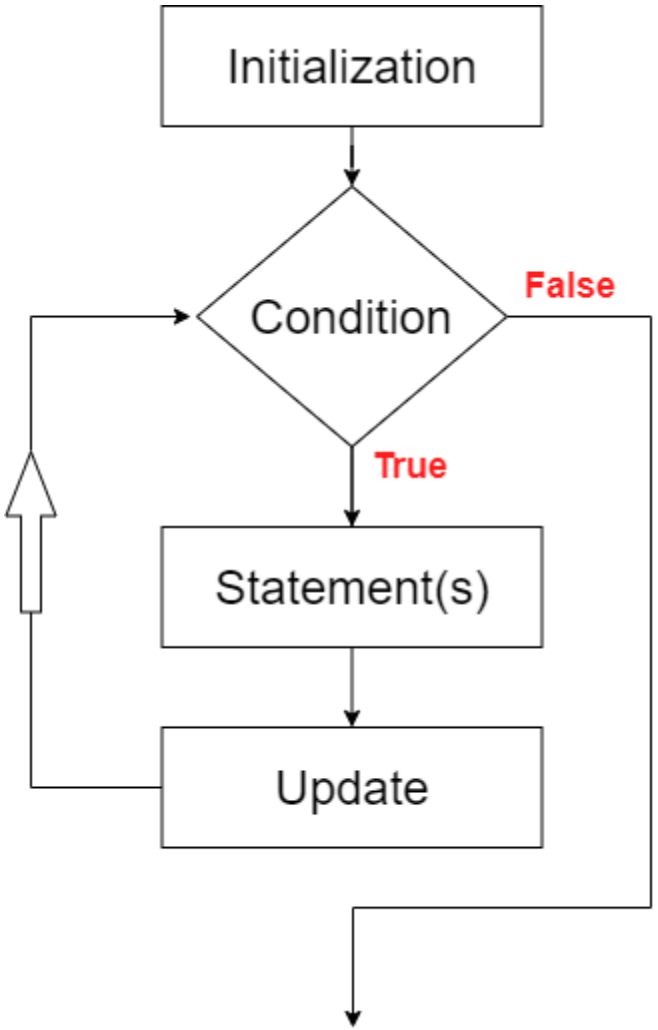


Do - While Loop



Loops

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



Entry-controlled and Exit-controlled loops

- The loop in which test condition is checked in the beginning of the loop are known as **entry controlled loop**.
  - For Example: **while** & **for** loops.

- When statements inside the loop body is executed first and then the condition is checked that loop is known to be as **exit controlled loop**.
  - For Example: **do-while** loop.

Choosing the correct looping

- Number of repetitions is known => **for loop**.
- Number of repetitions unknown + could be zero => **while loop**.
- Number of repetitions unknown + at least 1 => **do...while loop**.

Exercise

- Print all numbers between [1, 100]

```
int main()
{
    for(int z=0; z<=100; z=z+1)
    {
        cout<<z;
    }
    return 0;
}
```

//Solution_1

```
#include <iostream>
using namespace std;
int main()
{
    for(int z=0; z<=100;    )
    {
        cout<<z++;
    }
    return 0;
}
```

//Solution_2

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

int main()
{
    int z=1;
    for(;;)
    {
        if (z>100)
            break;
        cout<<z;
        z++;
    }
    return 0;
}
```

break and continue Statements

- The **break** statement, when executed it provides an immediate exit from the loop structure.
  - The break statement is typically used to exit early from a loop.
  - After the break statement executes, the program continues to execute with the first statement after the structure.
- The **continue** statement is used in while, for, and do.. while structures. When the continue statement is executed in a loop, it skips the remaining statements in the loop and proceeds with the next iteration of the loop.

Exercise

- Print all numbers between [1, 100]
  Solution_2

```
int main()
{
    for(int z=0; z<=100; z=z+1)
    {
        cout<<z;
    }
    return 0;
}
```

```
int main()
{
    int z=1;
    for(;;)
    {
        if (z>100)
            break;
        cout<<z;
        z++;
    }
    return 0;
}
```

Exercises:

1. Can you always convert a while loop into a for loop? Convert the following while loop into a for loop.

```
int i = 1, sum = 0;
while (sum < 10000)
{
    sum = sum + i;
    i++;
}
```

Infinite loop

- **Infinite loop**: continues to execute endlessly
  - Avoided by including statements in loop body that assure exit condition is eventually `false`
- Example of infinite loops:

```
for( ; ; ) {
    cout<<"This loop will run forever.\n";
}

while(true)
{
     cout<<"This loop will run forever.\n";
}
```

Nested Control Structures

1. Write a c++ program to find the multiplication table of a given number
2. Modify the program to find the multiplication table of for all numbers between 1:10

Part_1

```
int n;
cin>>n;
for(int i=1; i<=n; i++)
{
   cout<<n*i<<"\t";
}
cout<<"\n";
```

Part_2

```
int n;
for(n=1;n<=10;n++)
   {
      for(int i=1; i<=n; i++)
      {
         cout<<n*i<<"\t";
      }
      cout<<"\n";
   }
```

Exercise:

Write a program to calculate HCF (Highest Common Factor) of two given number.

Easy solution: https://onlinegdb.com/JIcIAWV78
Better solution: https://onlinegdb.com/I7VvI8Wk9

Write a program to check given number is prime or not.

Write a program that calculates the sum of the digits of an integer. For example, the sum of the digits of the number 2155 is 2 + 1 + 5 + 5 or 13.The program should accept any arbitrary integer typed in by the user.

Write a program to enter the numbers till the user inputs 0. Then, display the maximum and minimum number entered.