Lecture Notes

Site: Moodle LMS
Course: Object Oriented Programming(Div-A)
Book: Lecture Notes

Printed by: Mahesh Jagtap
Date: Wednesday, 8 July 2020, 9:01 AM

Table of contents

1. Introduction to C++

- C++ - general purpose programming language - high performance applications

- It is developed by Bjourne Stroustrup

- It is an object oriented language means it uses classes and object to write the program.

- Major updates has been done in 1998, 2011, 2014, 2017, 2020, so because this there different standards of C++ such as C++98, C++11, C++14, C++17, C++20.

- It is one of the most famous and widely used programming language.

Applications developed in C++

- Adobe
- Maya
- Mozilla Firefox
- Finacle(by Infosys)
- Operating Systems
- Embedded Systems
- Graphical Apps/Games

Job Opportunities for C++

- C++ Developer
- Qt-C++ Developer
- Game Developer
- Device Driver Developer

Required Software:

- GNU C/GCC Compiler
  - Ubuntu: You can install gcc using this command: sudo apt install g++
  - Windows: You can install Cygwin/Mingw
- Integrated Development Environment
  - Code Blocks IDE
  - CLion
  - Eclipse IDE for C++ Developers
- Editors
  - Gedit
  - Vi/Emacs
  - Notepad/Notepad++
  - Sublime Text

2. Syntax of C++ Program

The standard syntax for writing a C++ program is as follows:

header file – included to perform input output operations
class definition – define the class
user function definition – define user defined function declared in the class
main function definition – starts execution of program

Simple Example:

```
#include<iostream>   //header file
using namespace std;   //namespace to avoid name conflicts
int main(){             //execution starts here
      cout<<"Hello Students";    //statements to be executed
      return 0;             //ends the function and return value
}
```

Explanation:

- First line includes the header file iostream to perform input and output oprations. every c++ program requires this header file.
- Second line represents that you are using namespace std to avoid name conflicts among variables and functions.
- Third line represents that you are starting the execution from main function. remember every C++ program must have a main function defined in it. function definition should be start with { bracket and end with } bracket. main function must be written with int as a return type.
- Fourth line represents that the statements which we want to execute after calling this function. remember every executable statement in C++ must be terminated with ; (semicolon).
- Fifth line represents that the function is returning a zero value.

3. Comments in C++

- Comments are written to make your program readable and understandable. Comments are not executed.
- There are two types of comments in C++
  - Single line comments
  - Multi line comments
- Single Line Comments
  - These comments are written with //
  - e.g. //This is a simple program
  - These are helpful when you are writing a short comment.
- Multi Line Comments
  - These comments are written with /* */
  - e.g. /* This is a program

    to print a value*/
  - These are helpful when you are writing long comments which covers multiple lines.
- It is a good practice to write comments in the program in every programming language.
- You can write nested comments but you can only write single line comment within multi line comments.
- e.g. /* This is program to //add two numbers */

4. Data Types & Variables in C++

Data Types in C++

- Data type is a reserved keyword in C++ used to specify the type of a value in program.
- There are various data types in C++ as follows:
- **Basic Data Types/Primitive Data Types**
  - These data types are already predefined/built in the system.
  - keywords are int, float char, double, boolean, void
  - Following table shows the details about this type.

Data Type	Keyword	Size	Range of Values
Integer	int	4 bytes	-2147483648 to 214748647
Float	float	4 bytes	-3.4E+38 to +3.4E+38
Double	double	8 bytes	-1.7E+308 to +1.7E+308
Character	char	1 byte	-128 to +127 or 0 to 255
Boolean	bool	1 byte	0 and 1
Empty	void	1 byte	No value

- **Derived Data Types:**
  - These data types are derived from built in data types.
  - e.g. arrays, pointers, functions, references.
- **User Defined Data Types:**
  - These data types are defined by the user.
  - e.g. Structures, Union, Enumeration, Class

Variables in C++

- Variables are containers for storing data values.
- A variable is a name which is associated with a value that can be changed.
- Variables needs to be declared in the program to allocate the memory space for a value.
- Variable Declaration Syntax: ***data_type variable_name;***
- For example, suppose i want to a variable to hold integer variable. i will declare variable as follows:

```
int number;    //this is just declaration
```

```
int number = 20;   //this is declaration with initialization
```

- In above example when we declare number variable as an integer, computer will allocate a 4 byte memory space for this variable in main memory i.e. RAM.
- Variable values are stored into computer main memory(RAM).
- So the same way you can declare variables for other types of data using above syntax.

```
  ◦ e.g. float result; char ch; double salary; boolean flag;
```

Naming Rules for variables:

- All variable names must begin with a letter of the alphabet or an underscore(_).
- After the first initial letter, variable names can also contain letters and numbers.
- Variable names are case sensitive means uppercase and lowercase variables are considered as different.
- No spaces or special characters are allowed.
- You cannot use a C++ keyword (a reserved word) as a variable name.
- Following are some of the valid variable names.

```
    num123, _num1, number, Number, num_one
```

- Following are some of the invalid variable names:

```
    12number, @number, num#1, number one, num@one
```

- Example Program

```
//program to declare variables
#include<iostream>
using namespace std;
int main(){
         int number = 100;
         float flt_Number = 23.87;
         double number1 = 232.1123;
         char ch = 'P';
         bool flag = false;
```

```
        //Following are output statements to print the values on screen.
```

```
         cout<<"Integer:"<<number<<endl;
         cout<<"Float:"<<flt_Number<<endl;
         cout<<"Double:"<<number1<<endl;
         cout<<"Character:"<<ch<<endl;
         cout<<"Boolean:"<<flag<<endl;
         return 0;
}
```

5. Input & Output in C++

- In every programming language, there are input and output statement to get data and display data.
- We include *iostream* header file in every C++ program. This file is responsible for the process of input and output.
- Input Statement:
  - cin is an object of istream class from the header file iostream.
  - cin used to take input from the keyboard.
  - Syntax: ***cin>>variable_name1;***
  - Here cin is an object which refers to the keyboard, ask for entering the value and store the value in variable.
  - e.g. ***cin>>number;***
  - So above statement will ask for the value from the keyboard and then store the value in number variable.
  - You can cascade multiple variable inputs like this ***cin>>number1>>number2>>number3;***
  - >> is called as extraction operator.
- Output Statement:
  - cout is an object of ostream class from the header file iostream.
  - cout is used to display/print the output on the screen/terminal.
  - Syntax: ***cout<<variable_name; or cout<<"string to be printed";***
  - Here cout is an object which refers to the monitor, writes the value/data on monitor.
  - e.g. ***cout<<"Hello People";*** //here Hello People will be printed on monitor/terminal.
  - You can also cascade multiple output statements like this ***cout<<number1<<number2<<number3;***
  - << is called as insertion operator.
- Example Program

```
//Program to take input and display output
#include<iostream>
using namespace std;
int main(){
  int num1, num2, num3;      //3 variable declared
  cout<<"Enter two numbers:"<<endl;      // show message to understand user
  cin>>num1>>num2;           //take two numbers input
  num3 = num1 + num2;    //add two numbers and store value into num3 variable
  cout<<"Sum="<<num3<<endl;   //print the sum value
  return 0;
}
```

6. Manipulators in C++

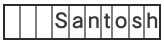
- Manipulators are used to format the output of a C++ program.
- There are various manipulators available in C++ as follows:
  - setw
  - setfill
  - setprecesion
  - endl

- **setw manipulator**
  - This is manipulator is used to set the width of an ouput.
  - By default, it displays the output in right alignment.
  - **Syntax**:

```
setw(width);
```

  - **Example**:

```
cout<<setw(10)<<"Santosh"';
```

  - The above statement will display the string as follows:
  - 

  - we can use other flags with setw to align the output such as left, right.
  - Example: to display above string in left alignment you need to write the same code as follows:

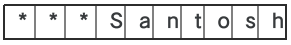
```
cout<<setw(10)<<left<<"Santosh";
```

- **setfill manipulator:**
  - This manipulator fills the characters in the remaining spaces of setw columns.
  - **Syntax**:

```
setfill(character);
```

  - **Example:**

```
cout<<setw(10)<<setfill('*')<<"Santosh";
```

  - The output of the statements will look like as follows:
  - 

- **setprecision manipulator**
  - This is used with floating point numbers.
  - It is used to set the number of digits printed to the right of the decimal point.
  - It comes in two forms: 1. fixed 2. scientific
  - fixed is used when you want to show float number in fixed format.
  - **Example:** *cout<<setprecision(2)<<fixed<<10.231;*
  - scientific is used to display float numbers in scientific format
  - **Example:** *cout<<setprecision(2)<<scientific<<234.21232*

- **endl manipulator**
  - This is used to move the cursor to new line.
  - Example: *cout<<"Santosh"<<endl;*

7. Operators in C++

- Operators are symbols that perform operations on variables and values. For example, + is an operator used for addition, while - is an operator used for subtraction.
- Operators in C++ can be classified into 6 types:
  - Arithmetic Operators
  - Assignment Operators
  - Relational Operators
  - Logical Operators
  - Bitwise Operators

- **Arithmetic Operators:**

Operator	Operation
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`%`	Modulo Operation (Remainder after division)

- Assignment Operators:

Operator	Example	Equivalent to
`=`	`a = b;`	`a = b;`
`+=`	`a += b;`	`a = a + b;`
`-=`	`a -= b;`	`a = a - b;`
`*=`	`a *= b;`	`a = a * b;`
`/=`	`a /= b;`	`a = a / b;`
`%=`	`a %= b;`	`a = a % b;`

- **Relational Operators**

Operator	Meaning	Example
`==`	Is Equal To	`3 == 5` gives us **false**
`!=`	Not Equal To	`3 != 5` gives us **true**
`>`	Greater Than	`3 > 5` gives us **false**
`<`	Less Than	`3 < 5` gives us **true**
`>=`	Greater Than or Equal To	`3 >= 5` give us **false**
`<=`	Less Than or Equal To	`3 <= 5` gives us **true**

- **Logical Operators**

Operator	Example	Meaning
`&&`	expression1 **&&** expression 2	Logical AND. True only if all the operands are true.
`||`	expression1 **||** expression 2	Logical OR. True if at least one of the operands is true.
`!`	**!**expression	Logical NOT. True only if the operand is false.

- **Bitwise Operators**

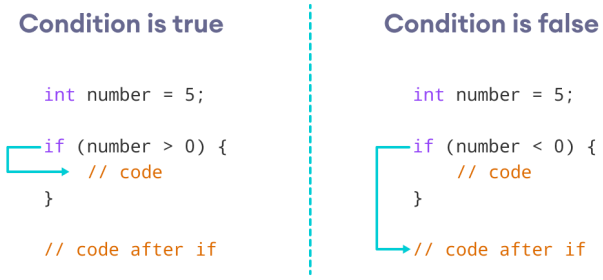
Operator	Description
`&`	Binary AND
`|`	Binary OR
`^`	Binary XOR
`~`	Binary One's Complement
`<<`	Binary Shift Left
`>>`	Binary Shift Right

8. Conditional Statements in C++

- In computer programming, we use the if statement to run a block code only when a certain condition is met.
- For example, assigning grades (A, B, C) based on marks obtained by a student.
  - if the percentage is above 90, assign grade A
  - if the percentage is above 75, assign grade B
  - if the percentage is above 65, assign grade C

- There are three forms of if...else statements in C++.
- - if statement
  - if...else statement
  - if...else if...else statement
- if statement
  - The if statement evaluates the condition inside the parentheses ( ).
    - If the condition evaluates to true, the code inside the body of if is executed.
    - If the condition evaluates to false, the code inside the body of if is skipped.
  - Syntax:

```
if (condition) {

   // statements to be executed

}
```



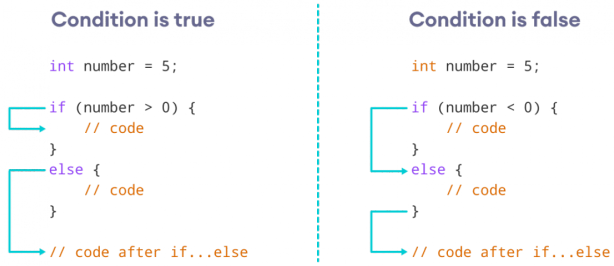
Example:

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

int main() {
    int num1, num2;

    cout << "Enter two numbers: ";
    cin >> num1 >> num2;

    // checks if the number is greater than another number
    if (num1 > num2) {
        cout << num1 <<" is greater than "<< num2 << endl;
    }

    return 0;
}
```

if else statement:

- The if statement can have an optional else clause. Its syntax is:
- Syntax:

```
if (condition) {
    // block of code if condition is true
}
else {
    // block of code if condition is false
}
```



Example:

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

int main() {
    int num1, num2;

    cout << "Enter two numbers: ";
    cin >> num1 >> num2;

    // checks if the number is greater than another number
    if (num1 > num2) {
        cout << num1 <<" is greater than "<< num2 << endl;
    }
```

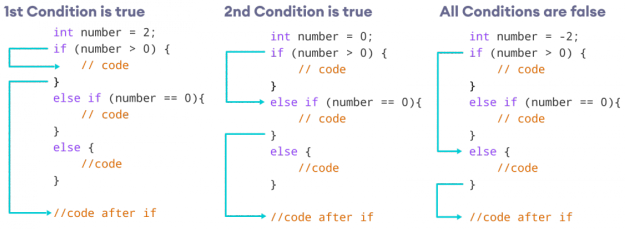
```
    else {
        cout << num1 <<" is less than "<< num2 << endl;
    }
```

```
return 0;
```

```
}
```

- if...else...else if statement
  - The if...else statement is used to execute a block of code among two alternatives. However, if we need to make a choice between more than two alternatives, we use the if...else if...else statement.
  - Syntax:

```
if (condition1) {
    // code block 1
} else if (condition2){
    // code block 2
} else {
    // code block 3
}
```



Example:

```
// Program to check whether an integer is positive, negative or zero
#include <iostream>
using namespace std;

int main() {
     int number;

    cout << "Enter an integer: ";
    cin >> number;
    if (number > 0) {
        cout << "You entered a positive integer: " << number << endl;
    }
    else if (number < 0) {
      cout << "You entered a negative integer: " << number << endl;
     }
    else {
        cout << "You entered 0." << endl;
    }
     cout << "This line is always printed.";
    return 0;
}
```

9. Switch Statement

The switch statement allows us to execute a block of code among many alternatives.

The syntax of the switch statement in C++ is:

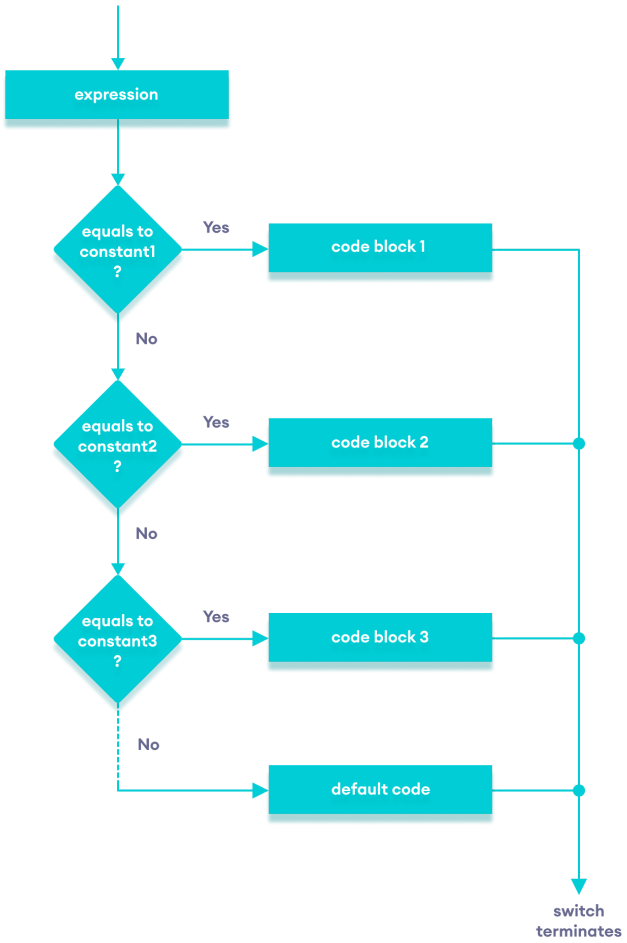
```
switch (expression)  {
    case constant1:
        // code to be executed if
        // expression is equal to constant1;
        break;

    case constant2:
        // code to be executed if
        // expression is equal to constant2;
        break;
        .
        .
        .
    default:
        // code to be executed if
        // expression doesn't match any constant
}
```

How does the switch statement work?

The expression is evaluated once and compared with the values of each case label.

If there is a match, the corresponding code after the matching label is executed. For example, if the value of the variable is equal to constant2, the code after case constant2: is executed until the break statement is encountered.

If there is no match, the code after default: is executed.



Example:

```
// Program to build a simple calculator using switch Statement
#include <iostream>
using namespace std;

int main() {
    char oper;
    float num1, num2;
    cout << "Enter an operator (+, -, *, /): ";
    cin >> oper;
    cout << "Enter two numbers: " << endl;
    cin >> num1 >> num2;

    switch (oper) {
        case '+':
            cout << num1 << " + " << num2 << " = " << num1 + num2;
            break;
        case '-':
            cout << num1 << " - " << num2 << " = " << num1 - num2;
            break;
        case '*':
            cout << num1 << " * " << num2 << " = " << num1 * num2;
            break;
        case '/':
            cout << num1 << " / " << num2 << " = " << num1 / num2;
            break;
        default:
            // operator is doesn't match any case constant (+, -, *, /)
            cout << "Error! The operator is not correct";
            break;
    }

    return 0;
}
```

10. Loops in C++

In computer programming, loops are used to repeat a block of code.

For example, let's say we want to show a message 100 times. Then instead of writing the print statement 100 times, we can use a loop.

That was just a simple example; we can achieve much more efficiency and sophistication in our programs by making effective use of loops.

There are 3 types of loops in C++.

- for loop
- while loop
- do...while loop

for loop

```
for (initialisation; condition; increment/decrement) {
    // body of-loop
}
```

Here,

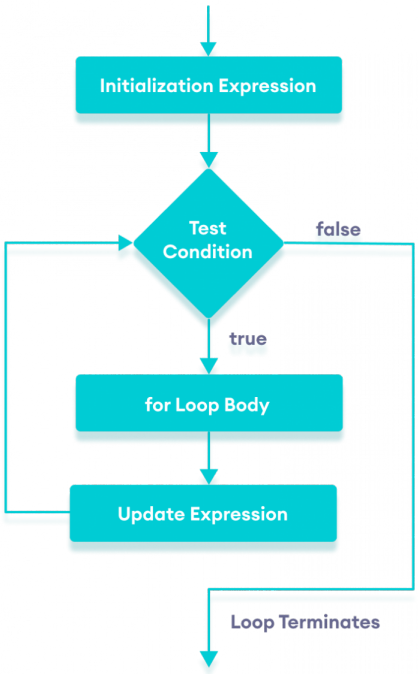
initialisation – initialises variables and is executed only once

condition – if true, the body of for loop is executed

if false, the for loop is terminated

increment/decrement – updates the value of initialised variables and again checks the condition

Flowchart of for Loop in C++



Example:

```
#include <iostream>
using namespace std;
int main() {
        for (int i = 1; i <= 5; ++i) {
        cout << i << " ";
    }
    return 0;
}
```

Ranged Based for Loop

In C++11, a new range-based for loop was introduced to work with collections such as arrays and vectors. Its syntax is:

```
for (variable : collection) {
    // body of loop
}
```

Here, for every value in the collection, the for loop is executed and the value is assigned to the variable.

Example:

```
#include <iostream>
using namespace std;
int main() {

    int num_array[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

    for (int n : num_array) {
        cout << n << " ";
    }

    return 0;
}
```

---

while Loop

Syntax:

```
while (condition) {
    // body of the loop
}
```

Here,

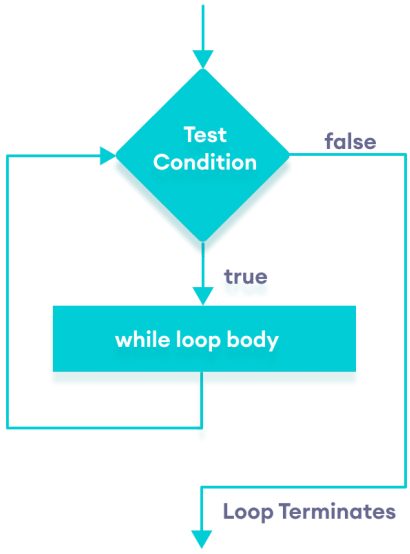
A while loop evaluates the condition

If the condition evaluates to true, the code inside the while loop is executed.

The condition is evaluated again.

This process continues until the condition is false.

When the condition evaluates to false, the loop terminates.



Example:

```
// C++ Program to print numbers from 1 to 5

#include <iostream>
using namespace std;
int main() {
    int i = 1;

    // while loop from 1 to 5
    while (i <= 5) {
        cout << i << " ";
        ++i;
    }

    return 0;
}
```

do...while Loop

The do...while loop is a variant of the while loop with one important difference: the body of do...while loop is executed once before the condition is checked.

Syntax:

```
do {
   // body of loop;
}
while (condition);
```

Here,

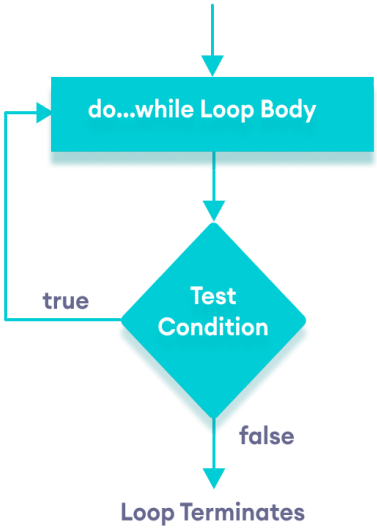
The body of the loop is executed at first. Then the condition is evaluated.

If the condition evaluates to true, the body of the loop inside the do statement is executed again.

The condition is evaluated once again.

If the condition evaluates to true, the body of the loop inside the do statement is executed again.

This process continues until the condition evaluates to false. Then the loop stops.



Example:

```cpp
// C++ Program to print numbers from 1 to 5

#include <iostream>
using namespace std;
int main() {
    int i = 1;

    // do...while loop from 1 to 5
    do {
        cout << i << " ";
        ++i;
    }
    while (i <= 5);

    return 0;
}
```

11. Functions in C++

- Function is a block of code which performs a specific task.
- Function is used to make your program modular.
- Function increases the re usability of the code.
- Function Declaration:

```
     data_type func_name(parameters);
```

- Function Definition:

```
     data_type func_name(paramters){
      //statements
      }
```

- Function Call:

```
      func_name(paramters);
```

Types:

1. Predefined(Library) – already defined in the library. we need to just call in in our program.
2. User Defined – defined by us as per our requirement.

Example:

```
#include<iostream>
using namespace std;
void printStars();   //function declaration
int main(){
    cout<<"Santosh Nagargoje"<<endl;
    printStars();   //function call
    cout<<"Modern College of Engineering"<<endl;
    printStars();
}
void printStars(){    //function definition
     cout<<"**************************************"<<endl;
}
```

Output:

```
Santosh Nagargoje
**********************************
Modern College of Engineering
**********************************
```

- based upon the return value and parameters, functions are again classified as follows:
  - function without parameter without return value
  - function without parameter with return value
  - function with parameter without return value
  - function with parameter with return value

- function without parameter without return value: