C++ Programming

Sadik Saifi

Oct 24, 2022

Contents

1 Basics Of Programming Languages

Programming is the process of creating a set of instructions which tells a computer how to perform a task.

1.1 Types Of Programming Languages

- Low Level Languages.
- High Level Languages.

1.1.1 Low Level Programming Languages

Assembly Level Programming Language Low-level programming that is intended to communicate directly with hardware.

1.1.2 High Level Programming Languages

Easy to understand and less complex than assembly level language/machine code.

Types Of High Level Programming Languages.

- Procedural
- Functional
- Object-Oriented

Procedural:

- It is written in set of procedures which executes in a structure/serial order.
- Procedures are also known as subroutine/function.
- e.g. C, Basic, Pascal etc.

Functional:

- We functions to write the code.
- Functions are chunks/block of code which can use over and over again.
- They can also take parameters.
- e.g. Python, JavaScript etc.

Object-Oriented:

- We use object and classes.
- Classed are user defined prototype/blueprints which can be used to create objects.
- Object are real life entity.
- e.g. C++, Java, C# etc.

Class: We can define some properties, attributes, methods etc.

Object: Instance of a class which allows to use variables and methods from class.

2 CPP Programming Language.

- It is Object-Oriented Programming Language.
- But it also has support for procedural programming.
- Initially it was intended to developed as a superset of C but later on it became a new programming language.
- Developed by **Bjarne Stroustrup** in **1979**.

2.1 C++ Sample Program.

```
#include<iostream>    /*iostream is directive & it is processed by
                        preprocessor.
                        Preprocessor is program that compiler runs.
                        #include telling our program to include the header
                        files like iostream.
                        iostream containes input output functions.
                      */
using namespace std;  /*It is telling our program that there is namespace of
                        name std which we have to use in our program.
                        e.g cout belongs to std namespace
                              - std:cout
                      */
int main() {          /*'int main' is the main function which the entry point
                        of a program.*/
  int val;            //declaring a variable.
  cout<<"Hello";      //printing the output.
  cin>>val;           //taking input.
  cout<<val;          //printing the output.
  return 0;           //return 0 indicates, program has executed successfully.
  cout<<"Hi";         //this won't be executed.
}
```

2.2 C++ Variables

- Variables are just containers to store our value where our code is executed.
- As C++ is statically typed language, so you can't store different type of value in different type of container.
  e.g.

```
int a = 5;          // correct
int b = "f";        // incorrect
int e = "6";        // incorrect
char c = "a"        // correct
char d = "4"        // correct
```

2.3 C++ Data Types

Primary	Derived	User Defined
Integer	Function	Class
Character	Array	Structure
Boolean	Pointer	Union
Floating Point	Reference	Enum
Double Floating Point	-	-
Void	-	-
Wide Character		

2.4 C++ Operators

- Arithmetic Operators
  - e.g. +, -, *, /, %, ++, – etc.
- Relational Operators
  - e.g. ==, !, >, <, >=, <= etc.
- Logical Operators
  - e.g. &&, ||, ! etc.
- Assignment Operators
  - e.g. =, +=, -=, /=, %= etc.
- Bitwise Operators
  - e.g. ~, «, », |, &, ^ etc.
  - a « b = a x 2^b
  - a » b = a / 2^b
- Misc Operators
  - e.g sizeOf, ?exp1:exp2, comma Operator, dot & arrow Operators, casting Operator, & Address Operator, * Pointer Operator etc.
- Uninary Operators
  - e.g. +, -, ++, –, ! etc.

3 Conditionals Statements

- If else Statements.
- Nested If else Statements
- Ternary Statements

3.1 If else Statements

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

3.2 Nested If else Statements

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

3.3 Ternary Statements

```
int number = -4;
string result;
// Using ternary operator
result = (number > 0) ? "Positive Number!" : "Negative Number!";
```

3.4 Switch Case Statements

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

4 C++ Loops/Iterative Statements

Loops are used when you want to do repetitive task in the program.

4.1 Types Of Loops

- While Loop
- For Loop
- Do-while Loop

4.2 While Loop

```
while(codition){
    //code
  }
```

4.3 For Loop

```
for (init-statement; condition; final-expression) {
    //code
  }
```

4.3.1 Multiple Variables in For Loop

```
for (int i=0, j=4; i<4, j>0; i++, j--) {
    // code
  }
```

4.4 For Loop vs While Loop

```
// For Loop
for (init-statement; condition; final-expression) {
    //code
  }

// while Loop
init-statement
while(condition) {
    // code
    final-expression
  }
```

4.5 Do-while Loop

```
do {
    code
} while(condition);
```

5 Function

5.1 How to declare a function.

```
returnType functionName(parameter1, parameter2){
    // Statements
  }
```

5.2 Function Prototype

Like if you want define your function-A after the function-B but you want to call function-B in inside of function-A.

e.g.

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

int add(int, int);    // prototype

int main() {
    cout<<add(2,3)<<endl;
  }

int add(int a, int b){
    return a+b;
  }
```

5.3 Scope of Variables

5.3.1 Accessing global variable.

We access the global variable by using **scope resolution** operator (::).

e.g.

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

int a = 6;
int main() {
    int a = 5;
    cout<<a<<endl;      //5
    cout<<::a<<endl;    //6
}
```

5.4 Parameters

5.4.1 Formal Parameters & Actual Parameters.

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

void add(int param1, int param2){ // param1 & param2 are formal parameters.
    cout<<param1+param2<<endl;
}

int main() {
    int a = 5;
    int b = 7;    // Actual parameters.
    add();
}
```

5.5 Pass by Value and Pass by Reference.

5.5.1 Pass by Value.

Copy of the actual variable get copied into the formal variable.

e.g.

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

void add(int param1, int param2){ // param1 & param2 are formal parameters.
    cout<<param1+param2<<endl;
}
int main() {
    int a = 5;
    int b = 7;    // Actual parameters.
    add(a,b);
}
```

5.5.2 Pass by Reference.

Variables it self gets used in the function.

e.g.

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

void print(int &param) {
  cout << param << endl;
  param = 2;
}
int main() {
  int a = 5;

  print(a);
  cout<<a<<endl;
}
```

5.5.3 Default values of parameters in a function.

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

void add(int a, int b=0, int c=2){  // b & c default values
    return a+b+c;
}

int main() {
    add(3);     // 5
    add(3,2)    // 7
    add(3,2,3)  // 8
}
```

6 Array

- An array is data structure which stores a collection of items.
- It store homogeneous items(same data type).
- It has contiguous memory.
- **Representation of Array**
  - |1|2|3|4|5|
  - Length of this array is 5.
- We access the elements by their indexes(starts from 0).

6.1 Syntax Of An Array

```
// datatype arrayName[arraySize];
int array[5];
```

6.2 Array Literal

```
int array[] = { 1, 2, 3, 4, 5 };
```

6.3 Types Of Array

- Single dimensional or One-dimensional array.
- Multidimensional array.

6.4 Sample Array Program

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

int main() {
  int array[] = {1, 2, 3, 4, 5};

  cout << sizeOf(array) << endl;                  // size of the array.
                                                  // 20 bits.
  cout << sizeOf(array[0]) << endl;               // size of an elements
                                                  // 4 bits
  cout << sizeOf(array)/sizeOf(array[0]) << endl; // length of the array.
                                                  // 5
}
```