C++ Programming

ARRAYS AND FUNCTIONS

Arrays

Stores fixed-size sequential collection of elements of the same data type

Declaring an array:

- <type> <array name>[array size];

Example:

- Int numbers[10];
- Int marks[] = {10, 23, 89 ,1, 23};

Declaring two dimensional array

- <type> <array name>[array size];

Example:

- Int array[5][8];

Question

1. Write a program to input data to an array of 5 integers. Print the contents of the array in the reverse order you entered.
2. Write a C++ program to create an array of size 10 and read numbers from the keyboard. Display the numbers greater than mean value.
3. Write a C++ program to find the transpose of a 5 x 5 matrix. Ask the user to enter the elements from the keyboard.

Functions

A function is a group of statements that together perform a task.

Every C++ program has at least one function, which is **main().**

The general form of a C++ function definition is as follows,

```
return_type function_name( parameter list ) {
   body of the function
}
```

Function prototype - declaration of the function and must appear before the function is invoked.

- **return_type function_name(parameter_list);**

Functions - example

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

int max(int num1, int num2);

int main () {
  int a = 100;
  int b = 200;
  int ret;
  ret = max(a, b);
  cout << "Max value is : " << ret << endl;
  return 0;
}

int max(int num1, int num2) {
  // local variable declaration
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

Function Prototype

Function Heading

Return Statement

Parameter passing

When an argument is passed by value, a copy of the argument’s value is made and passed to the called function.

With pass by reference, the caller gives the called function the ability to access the caller’s data directly.

Parameter passing - example

```
#include<iostream>
using namespace std;
int squareByValue(int number); // pass by value
void squareByReference( int & number );  // pass by reference
int main()
{
        int x = 2;
        int z = 4;
        // square by value
        cout << x << "- Before squareByValue" << endl;
        cout << "Value returned " << squareByValue(x) << endl;
        cout << x << "- After squareByValue" << endl;
        // square by reference
        cout << z << "- Before squareByReference" << endl;
        squareByReference(z);
        cout << z << "- After squareByValue" << endl;

        return 0;
} // end main
```

```
// Pass by value
int squareByValue( int number )
{
        return number*= number;
}
// Pass by reference
void squareByReference( int &number )
{
        number*= number;
}
```

Default Arguments

A Function will return default values when no argument is passed to the function.

```
#include<iostream>
using namespace std;
//prototype that specifies default arguments
int boxVolume(int length = 1, int width = 1, int height = 1);

int main(){
        cout << “Box volume :” << boxVolume() << endl;   //no arguments given
        cout << “Box volume :” << boxVolume(10) << endl; //specify length only
        return 0;
}
int boxVolume(int length, int width, int height)
{
        return length * width * height;
}
```

```
Output:
1
10
```

Question

1. Write a C++ program to find the cube of the numbers from 1 to 10. Write a function called cube to return the cube of a number when the number is given as a parameter.

Scope Rules

The portion of the program where an identifier can be used is known as its scope

```
#include<iostream>
using namespace std;
void useLocal();
void useGlobal();
int x = 1; //global variable
int main() {
        cout << “Global is “ << x << endl;
        int x = 12;
         cout << “x in main is “ << x << endl;
        {   // start new block
                int x = 7;
                 cout << “now x is “ << x << endl;
        }
        cout << “x in main is “ << x << endl;
        useLocal();
        useGlobal();
        useGlobal();
}
```

```
void useLocal()
{
        int x = 25;
        cout << “local is “ << x << endl;
}
void useGlobal()
{
        cout << “global is “ << x << endl;
        x = 50;
}
```

Function Overloading

Function overloading is a feature in C++ where two or more functions can have the same name but different parameters.

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

void print(int i) {
  cout << " Here is int " << i << endl;
}

void print(double  f) {
  cout << " Here is float " << f << endl;
}

void print(char const *c) {
  cout << " Here is char* " << c << endl;
}

int main() {
  print(10);
  print(10.10);
  print("ten");
  return 0;
}
```

Output:

```
Here is int 10
Here is float 10.1
Here is char* ten
```

Function Templates

Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type.

A template is a blueprint or formula for creating a generic class or a function.

Function Templates - example

```
#include <iostream>
#include <string>

using namespace std;

template <typename T>
inline T const& Max (T const& a, T const& b) {
   return a < b ? b:a;
}
```

Output:
Max(i, j): 39
Max(f1, f2): 20.7
Max(s1, s2): World

```
int main () {
   int i = 39;
   int j = 20;
   cout << "Max(i, j): " << Max(i, j) << endl;

   double f1 = 13.5;
   double f2 = 20.7;
   cout << "Max(f1, f2): " << Max(f1, f2) << endl;

   string s1 = "Hello";
   string s2 = "World";
   cout << "Max(s1, s2): " << Max(s1, s2) << endl;

   return 0;
}
```

Using Integers

Using Floats

Using Strings

Math Library Functions

The <math.h> header provides a collection of functions that enable you to perform common mathematical calculations.

```
#include <stdio.h>
#include <math.h>
using namespace std;

int main () {
   cout << sqrt( 90.0 );
   cout << cos( 90.0 );
   cout << pow(2,7);
   return(0);
}
```

Question

1. Write a program to calculate the hypotenuse of a right angled triangle given the other two lengths