National University of Computer & Emerging Sciences, Karachi Computer Science Department

Course Code: CL-1004	Course : Object Oriented Programming Lab

Spring 2023, Lab Manual - 01

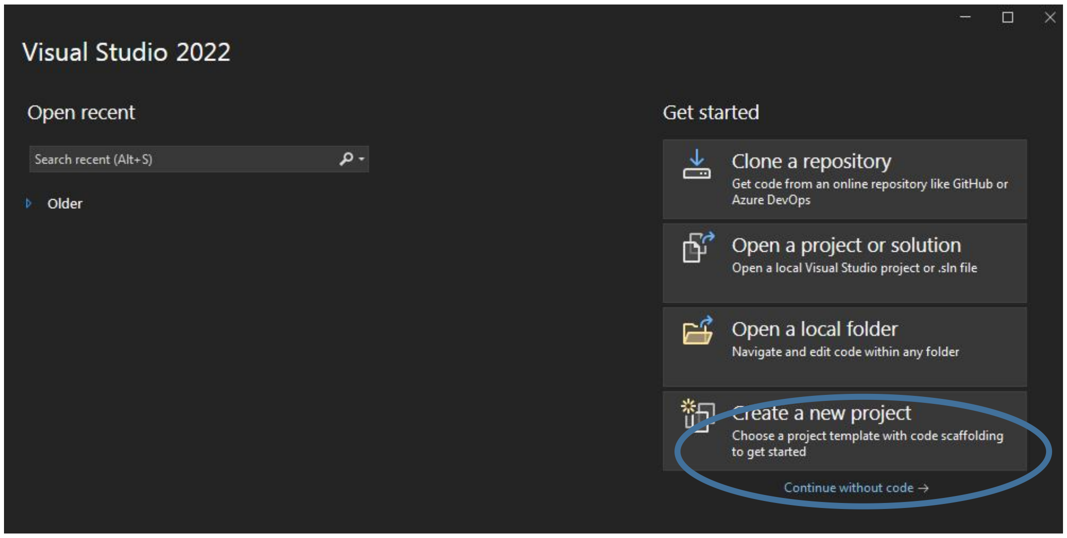
Contents

INTRODUCTION TO IDE (VISUAL STUDIO 2022)

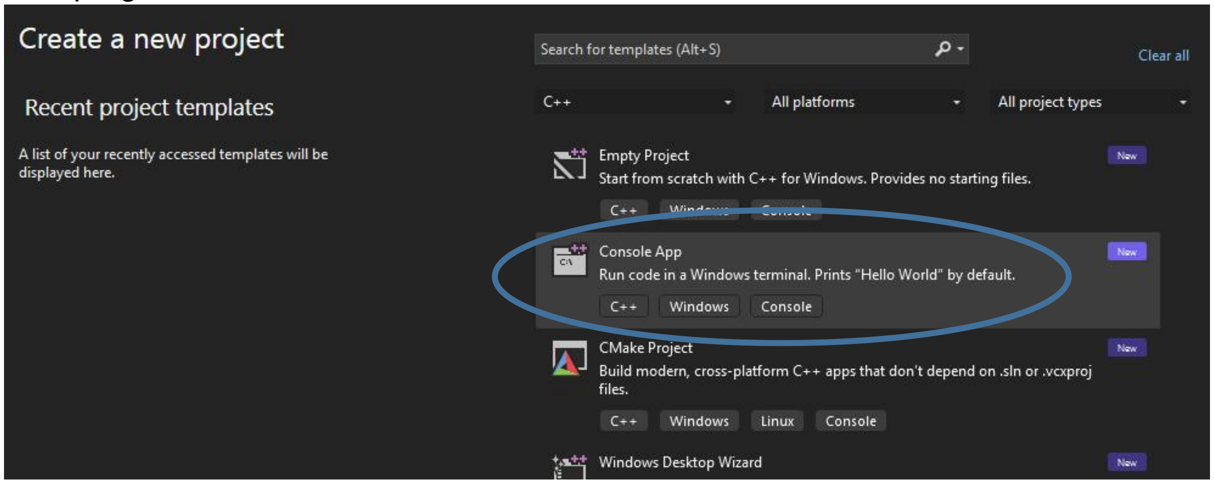
Visual Studio is an Integrated Development Environment(IDE) developed by Microsoft to develop GUI (Graphical User Interface), console, Web applications, web apps, mobile apps, cloud, and web services, etc.

GETTING STARTED WITH VISUAL STUDIO 2022

1. First, you have to download and install the Visual Studio. For that, you can refer to https://visualstudio.microsoft.com/vs/ . Download the **Community 2022**.
2. After you installed the IDE, open the IDE and the following window will appear on your screen. Select Create new project.


Visual Studio 2022
Open recent
Search recent (Alt+S)
Older
Get started
Clone a repository
Get code from an online repository like GitHub or Azure DevOps
Open a project or solution
Open a local Visual Studio project or .sln file
Open a local folder
Navigate and edit code within any folder
Create a new project
Choose a project template with code scaffolding to get started
Continue without code →

3. After creating a new project, create a C++ console application to write and run C++ programs.



Create a new project

Recent project templates

A list of your recently accessed templates will be displayed here.

Search for templates (Alt+S)

Clear all

C++ | All platforms | All project types

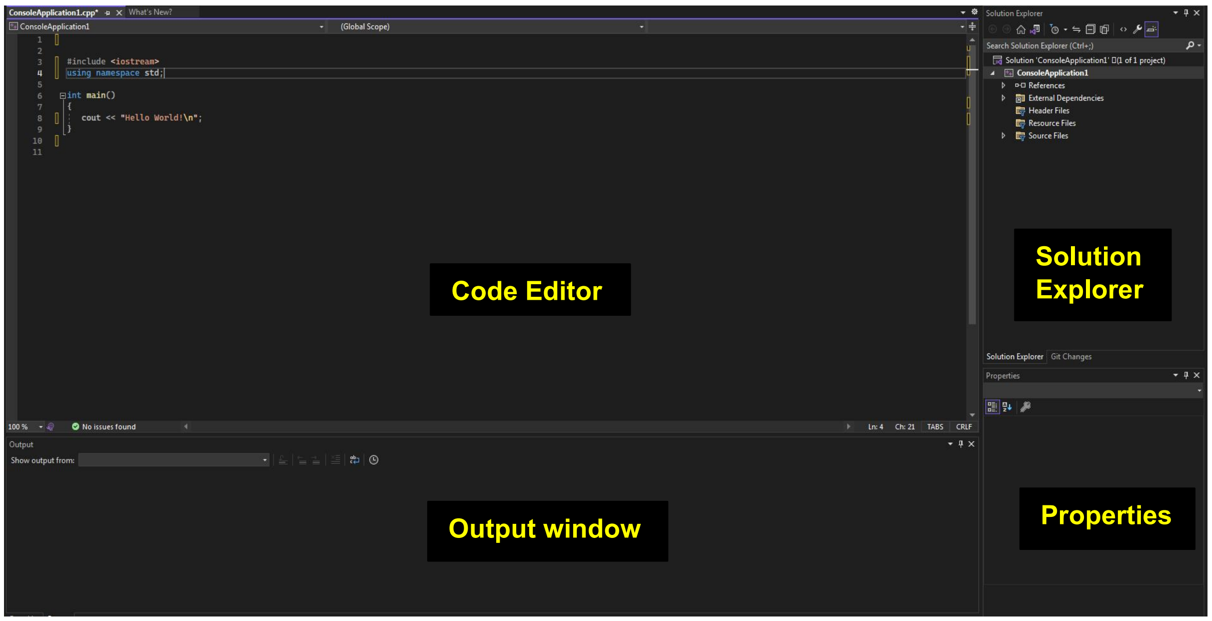
Empty Project — New
Start from scratch with C++ for Windows. Provides no starting files.
C++ | Windows | Console

Console App — New
Run code in a Windows terminal. Prints "Hello World" by default.
C++ | Windows | Console

CMake Project — New
Build modern, cross-platform C++ apps that don't depend on .sln or .vcxproj files.
C++ | Windows | Linux | Console

Windows Desktop Wizard — New

4. After creating a C++ file, you will get the following window.



```

#include <iostream>
using namespace std;

int main()
{
    cout << "Hello World!\n";
}

```

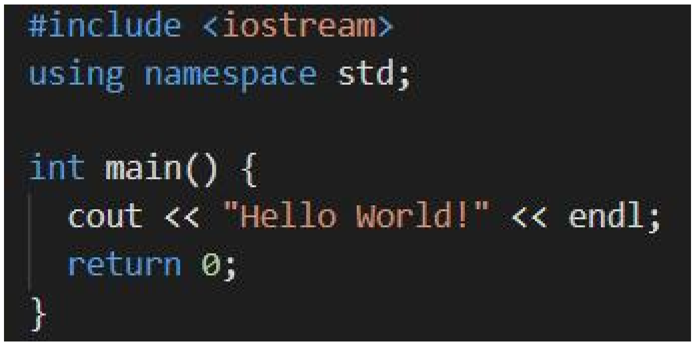
- **<u>Code Editor</u>**: Where the user will write code.
- **<u>Output Window</u>**: Here the Visual Studio shows the outputs, compiler warnings, error messages and debugging information.
- **<u>Solution Explorer</u>**: It shows the files on which the user is currently working.
- **<u>Properties</u>**: It will give additional information and context about the selected parts of the current project.

SKELETON OF C++ PROGRAM

A C++ program is structured in a specific and particular manner. In C++, a program is divided into the following three sections:

1. Standard Libraries Section
2. Main Function Section
3. Function Body Section

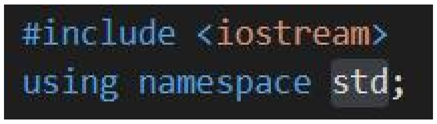
For example, let's look at the implementation of the Hello World program:



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

int main() {
  cout << "Hello World!" << endl;
  return 0;
}
```

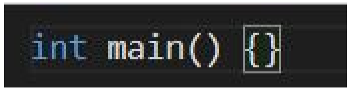
STANDARD LIBRARIES SECTION



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

- #include is a specific preprocessor command that effectively copies and pastes the entire text of the file, specified between the angle brackets, into the source code.
- The file <iostream>, which is a standard file that should come with the C++ compiler, is short for input-output streams. This command contains code for displaying and getting an input from the user.
- namespace is a prefix that is applied to all the names in a certain set. iostream file defines two names used in this program - cout and endl.

MAIN FUNCTION SECTION



```cpp
int main() {}
```

- The starting point of all C++ programs is the main function.
- This function is called by the operating system when your program is executed by the computer.

- { signifies the start of a block of code,   and } signifies the end.

INPUT/OUTPUT IN C++

- C++ is very similar to the C Language.
- For the input/output stream we use <iostream> library (in C it was <stdio>).
- For taking input and out we cout and cin (in C it was printf and scanf).
    - cout uses insertion (<<) operator.
    - cin uses extraction (>>) operator.

SAMPLE C++ CODE:

```
#include <iostream>

using namespace std;

int main()

{

int var = 0;

cout << "Enter an Integer value: "; cin >> var;

cout << "Value of var is : " << var; return 0;

}
```

Sample Run: In this sample run, the user input is shaded. Enter an Integer value: 12

Value of var is : 12

ARRAYS

An Array is a collection of fixed number of elements of same data type.

1-D ARRAY

- 1-D Array is a form of array in which elements are arranged in a form of List.
- To declare a 1D array you need to specify the data type, name and array size.

dataType arrayName [arraySize] ;

- Following is the declaration of a 1D array.

int numArray[5];

where;

Data Type: Integers (int)

Array Name: numArray

Array Size: 5

- To access array elements, you use the array name along with the index in subscript operator **“[ ]”**.

  numArray[0], numArray[1], numArray[2], numArray[3], numArray[4]

  where;

  Index of the array starts with zero ‘0’.

  Index of the last element is always ‘size - 1’ (in this case it is 4).

Example Code for 1-D Array

Program to read five numbers, find their sum, and print the numbers in reverse order.

```
#include <iostream>
using namespace std;
int main()
{
int item[5]; //Declare an array item of five components int sum = 0;
int counter;
cout << "Enter five numbers: ";
for (counter = 0; counter < 5; counter++)
{
cin >> item[counter];
sum = sum + item[counter];
}
cout << endl;
cout << "The sum of the numbers is: " << sum << endl; cout << "The numbers in
reverse order are: ";
//Print the numbers in reverse order.
for (counter = 4; counter >= 0; counter--)
```

```
        cout << item[counter] << " ";
cout << endl;
return 0;
}
```

Sample Run: In this sample run, the user input is shaded. Enter five numbers: 12 76 34 52 89

The sum of the numbers is: 263

The numbers in reverse order are: 89 52 34 76 12

2-D ARRAY

1. 2-D Array is a collection of fixed collection of elements arranged in rows and columns.
2. To declare a 2D array you need to specify the data type, name and no. of rows and columns.

   `dataType arrayName [ rowSize ][ columnSize ] ;`
3. Following is the declaration of a 2D array.

   `int numArray[5][5];`

   where;
   - Data Type: Integers
   - Array Name: numArray
   - Rows: 5
   - Columns: 5
4. To access array element you use the array name along with the row Index and column Index in subscript operator "[ ][ ]".

   numArray[0][0], numArray[1][1], numArray[2][2], numArray[3][3], numArray[4][4].

   where;
   - Index for the rows and columns of the array starts with zero '0'.
   - Index of the last element in rows and columns is always **'sizeofRow - 1'** and **'sizeofColumn -1'** respectively (in this case it is 4).

Example Code for 2-D Array:

Program to read a 2D array of size 3x3 find the sum for each row, print the sum line by line.

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

```
{
int item[3][3];      //Declare an array of size 3x3 int sum = 0;
int row, col;
cout << "Enter array elements: " << endl;
for (row = 0; row < 3; row++)
{
for (col = 0; col < 3; col++)
{
cin >> item[row][col]; sum = sum + item[row][col];
}
cout << "The sum of row " << i << " : " << sum <<
endl;
}
cout << endl; return 0;
}
```

Sample Run: In this sample run, the user input is shaded. Enter array elements:

```
12 76 34
The sum of row 0 : 122 52 89 48
The sum of row 1 : 189 22 63 99
The sum of row 2 : 184
```

POINTERS

A Pointer is a variable whose content is a memory address.

SINGLE POINTERS

1. To declare a single pointer variable you need to specify the data type, an asterisk symbol (*) and the name of the pointer variable.

   dataType *ptrName;

2. Following is the declaration of a Pointer variable.

   int *ptr;

where;

- DataType: Integer
- Name: ptr

3. Pointer variable holds the memory address of the variable which is of same data type (integer in this case).
4. To assign the memory address of any variable to the pointer variable we use Address of Operator (&).

int intVar = 5;

ptr = &intVar;

In this statement ptr now holds the memory address of an integer variable ‘intVar’.

5. To access the value at the memory address (currently stored) in the variable we use Dereferencing Operator (*).
6. Do not confuse this with the symbol used for the declaration of a pointer.

int intVar2 = *ptr;

In this statement another integer variable ‘intVar2’ is now initialized with the value at the memory address which is stored in ptr (that is the value of intVar).

Example Code for Single Pointers

The following program illustrates how pointer variables work:

```
#include <iostream>

using namespace std;

int main()

{

int *p;

int x = 37;

cout << "Line 1: x = " << x << endl; //Line 1

p = &x; //Line 2

//Line 3

cout << "Line 3: *p = " << *p << ", x = " << x << endl;

*p = 58; //Line 4

//Line 5
cout << "Line 5: *p = " << *p << ", x = " << x << endl; cout << "Line 6: Address of p = "
<< &p << endl; //Line 6 cout << "Line 7: Value of p = " << p << endl; //Line 7

cout << "Line 8: Value of the memory location " << "pointed to by *p = " << *p << endl;
//Line 8

cout << "Line 9: Address of x = " << &x << endl; //Line 9 cout << "Line 10: Value of x = "
<< x << endl; //Line 10 return 0;

}
```

Sample Run:

```
Line 1: x = 37

Line 3: *p = 37, x = 37

Line 5: *p = 58, x = 58

Line 6: Address of p = 006BFDF4 Line 7: Value of p = 006BFDF0

Line 8: Value of the memory location pointed to by *p = 58 Line 9: Address of x =
006BFDF0

Line 10: Value of x = 58
```

DYNAMIC VARIABLES

1. Variables created during the program execution are called dynamic variables.
2. To create a dynamic variable, we use new operator.

   new dataType [size];   // to allocate an array of variables.

   where

   - The new operator allocates the memory of a designated type.
   - It returns a pointer to the allocated memory.

3. Following is the declaration of a dynamic variable.

   int *p = new int;

   char *cArray = new char[5];

   where;

   - Line 01: creates a single variable of integer type.
   - Line 02: Creates an array of 5 characters.

4. To delete the dynamically allocated memory we use delete operator.

   delete ptrVar;   //to deallocate single dynamic variable

delete [] ptrArray; //to deallocate dynamically created array

5. delete operator is used to free the memory which is dynamically allocated using new operator.

Example Code for Dynamic Variables

```
#include<iostream>

using namespace std;

int main()

{

int* intPtr;

char* charArray; int arraySize;

intPtr = new int; // allocating memory to single variable cout << "Enter an Integer Value:
";

cin >> *intPtr;

cout << "Enter the size of the Character Array : "; cin >> arraySize;

charArray = new char[arraySize]; // allocating memory to array

for (int i = 0; i < arraySize; i++)

        cin >> charArray[i];

for (int i = 0; i < arraySize; i++)

        cout << charArray[i];

return 0;

}
```

Sample Run: In this sample run, the user input is shaded.

```
Enter an Integer Value: 2

Enter the size of the Character Array : 2 a b

ab
```

Lab tasks:

1. Display Sum and Average of Array Elements Using for Loop
2. Write C++ program to add two numbers using pointers
3. Program to reverse an array using the pointer
4. Write a program that swaps the value of two variables using pointers.
5. Write a program in C++ to print a square pattern with $ character.
6. Write a C++ program to find the highest number in a given array.