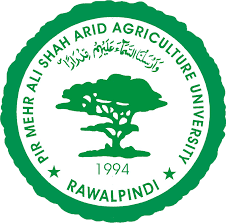
**LAB MANUAL**

**Programming Fundamentals**

**CSC-101**



University Institute of Information Technology PMAS-Arid Agriculture University, Rawalpindi

Lab Manual

Programming Fundamentals

CSC 101

[4(3-3)]

Mr. Muhammad Ahsan Arshad

University Institute of Information Technology

PMAS-Arid Agriculture University, Rawalpindi

## CSC-101 Programming Fundamentals

## (List of Labs)

***Lab #1***

Programming and Debugging using Dev C++ and Microsoft Visual C++

***Lab #2***

Variables and Arithmetic Operators

***Lab #3***

Decision Control Statements

***Lab #4***

for , while and Do-While loop, Continue, Switch, Break

***Lab #5***

Arrays

***Lab #6***

Arrays, practicing with loops

***Lab #7***

Practicing Multi-dimensional Arrays

***Lab #8***

Structures in C++

***Lab #9***

Functions (Call by value and reference)

***Lab #10***

Implementation of cstring and its functions

***Lab #11***

Implementation of Pointers

***Lab #12***

Dynamic Memory Allocation

***Lab #13***

File handling in C++

# Lab No. 1

# Programming, Debugging, Dev C++, Microsoft Visual C++

#### Introduction:

The aim of this lab is to install and create a new project in Dev C++ and Microsoft Visual C++ and debug the very first program in this environment.

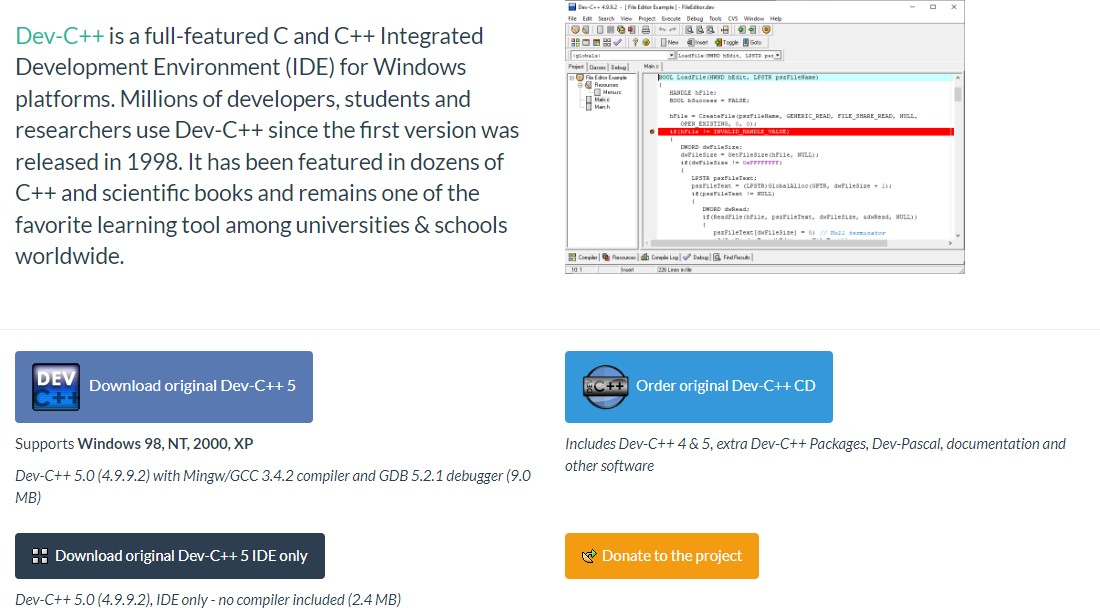
**Objectives of this Lab:**

### To learn about C++ compilers

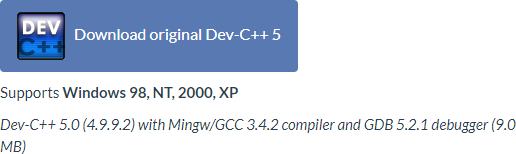
* **Making a project**
* **Adding source files to the Project created.**
* **Writing First program in Dev C++ and Visual C++**
* **Debugging**

## How to Install Dev C++:

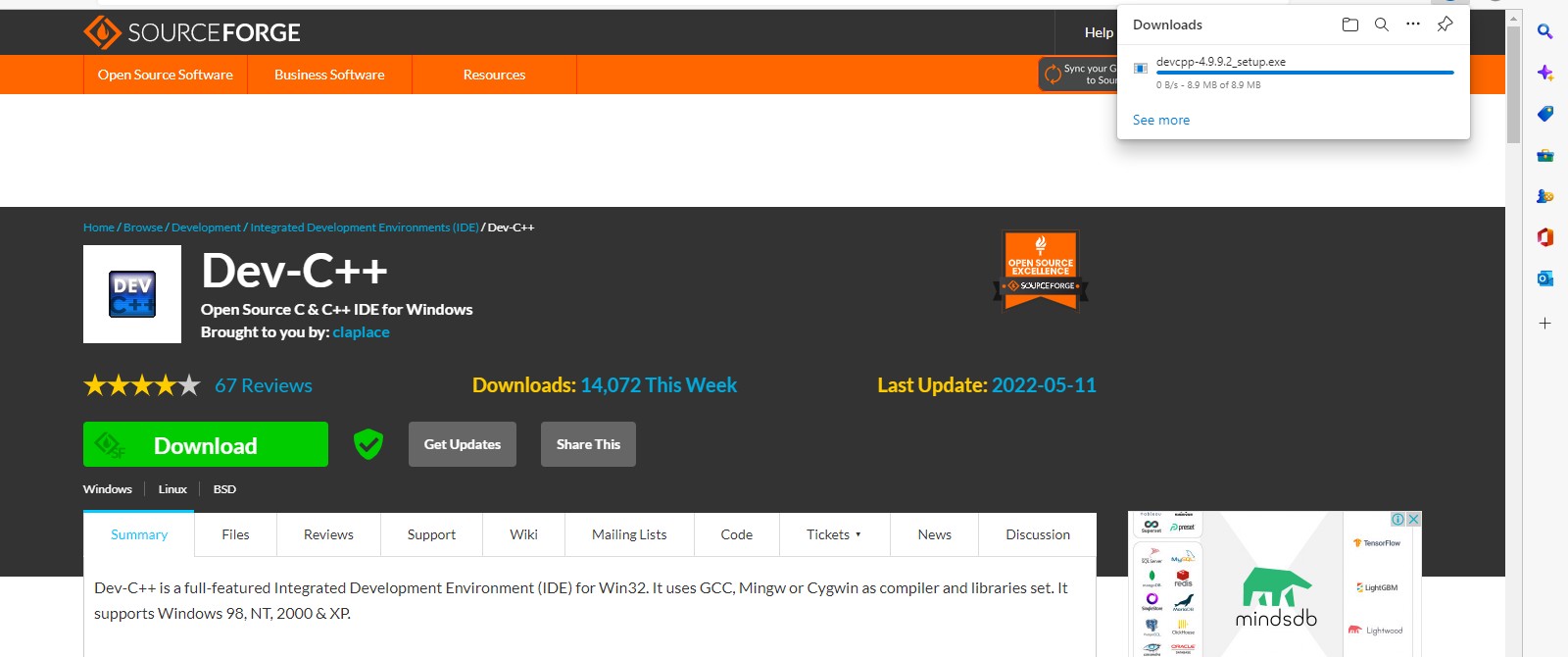
**Step 1):** First, you must download the Dev C++ on your Windows machine. Visit Download Dev C++: <http://www.bloodshed.net/>



**Step 2):** Under package Dev-C++ 5.0 (4.9.9.2) with Mingw/GCC 3.4.2 compiler and GDB 5.2.1 debugger (9.0 MB)

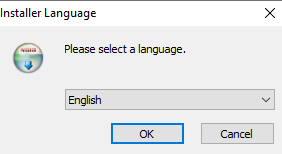


**Step 3):** This package will download C++ .exe file for Windows that can be used to install on Windows 7/8/10

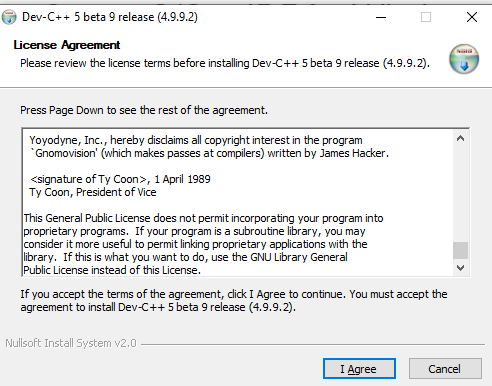


**Step 4):** You will direct to the Source Forge website, and your C++ download will start automatically.

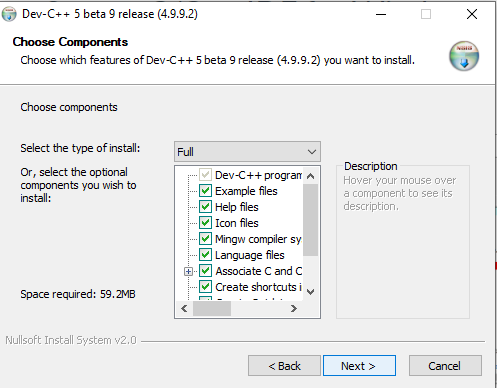
* Click on the save button to save. By default, it is saved in the “Downloads” folder.
* When download completes, go to the saved .exe file and click on it to Run.
* The installer will ask you a language to select. Select “English” and click on “OK”.



**Step 5):** Then screen for license agreement will appear. Click on “I Agree” to proceed further.

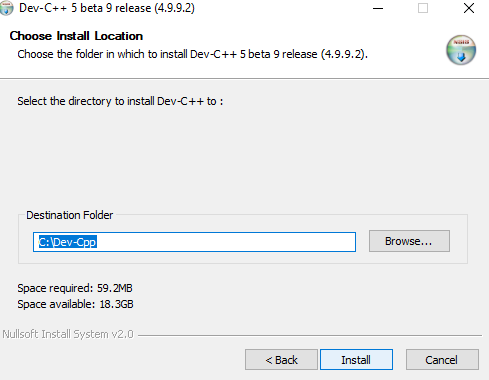


**Step 6):** In this step, you can see different components of Dev C++ that will be installed with this package. Just click on the “next” button.

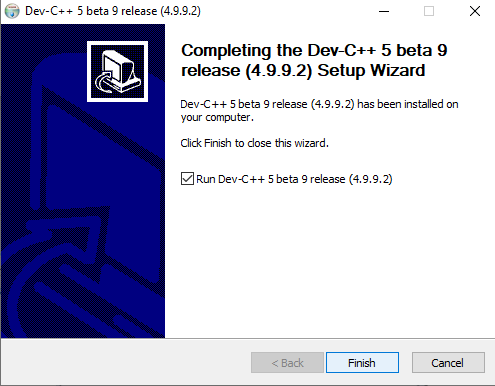


**Step 7):** In this step,

1. By default, the destination folder is in C drive. You are free to change this destination folder but make sure you have enough memory.
2. Click on the “Install” button



**Step 8):** Now, Dev C++ is installed successfully on your Windows. Select” Run Dev C++” to run it and click on the” Finish” button.

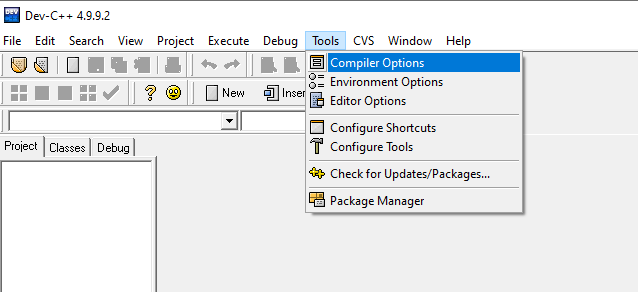


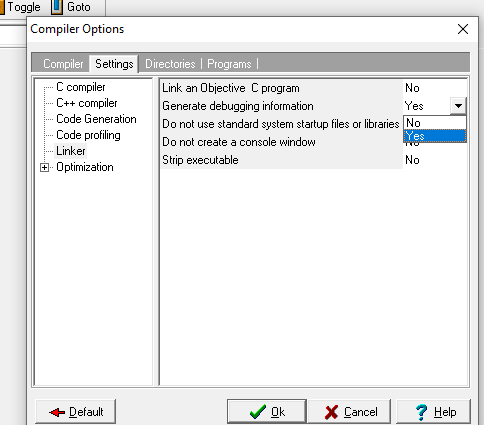
## How to create a small program in Dev:

**Step 1:** Configure Dev-C++.

We need to modify one of the default settings to allow you to use the debugger with your programs.

* Go to the "Tools" menu and select "Compiler Options".



In the "Settings" tab, click on "Linker" in the left panel, and change "Generate debugging information" to "Yes"

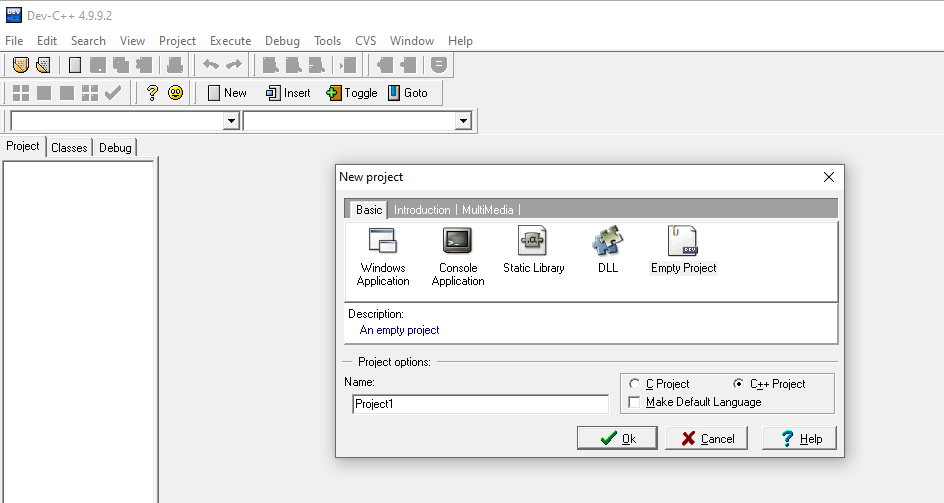
Click "OK".

## Step 2: Create a new project.

* + Go to the "File" menu and select "New", and "Project".
  + Choose "Empty Project" and make sure "C++ project" is selected.

o Here you will also give your project a name. You can give your project any valid filename, but keep in mind that the name of your project will also be the name of your final executable.

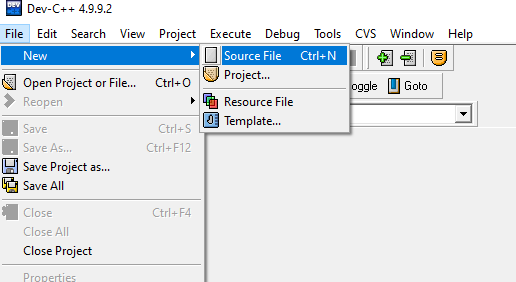
* + Once you have entered a name for your project, click "OK".
  + Dev-C++ will now ask you where to save your project.



### Step 3: Create/add source file(s).

You can add empty source files one of two ways:

* + Go to the "File" menu and select "New Source File" (or just press CTRL+N) OR
  + Go to the "Project" menu and select "New File".



* + - Note that Dev-C++ will not ask for a filename for any new source file until you

attempt to:

* 1. Compile
  2. Save the project
  3. Save the source file
  4. Exit Dev-C++

### First Program:

Now try to run the following code:

#include <iostream>

using namespace std;

int main()

{

cout<<"Welcome to batch 2022 :) \n";

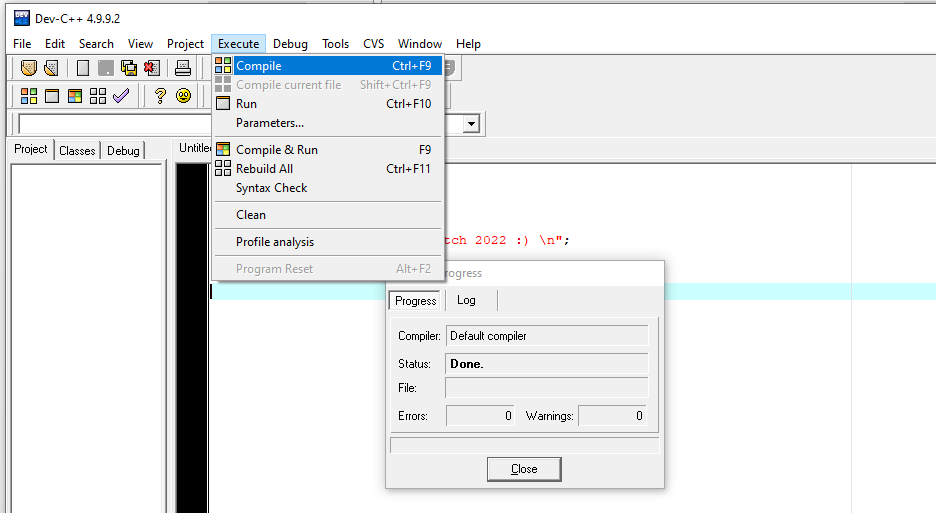
return 0;

}

### Step 4: Compile.

Once you have entered all of your source code, you are ready to compile.

* + - Go to the "Execute" menu and select "Compile" (or just press CTRL+F9)



Once your project successfully compiles, the "Compile Progress" dialog box will have a status of "Done". At this point, you may click "Close

### Step 5: Execute.

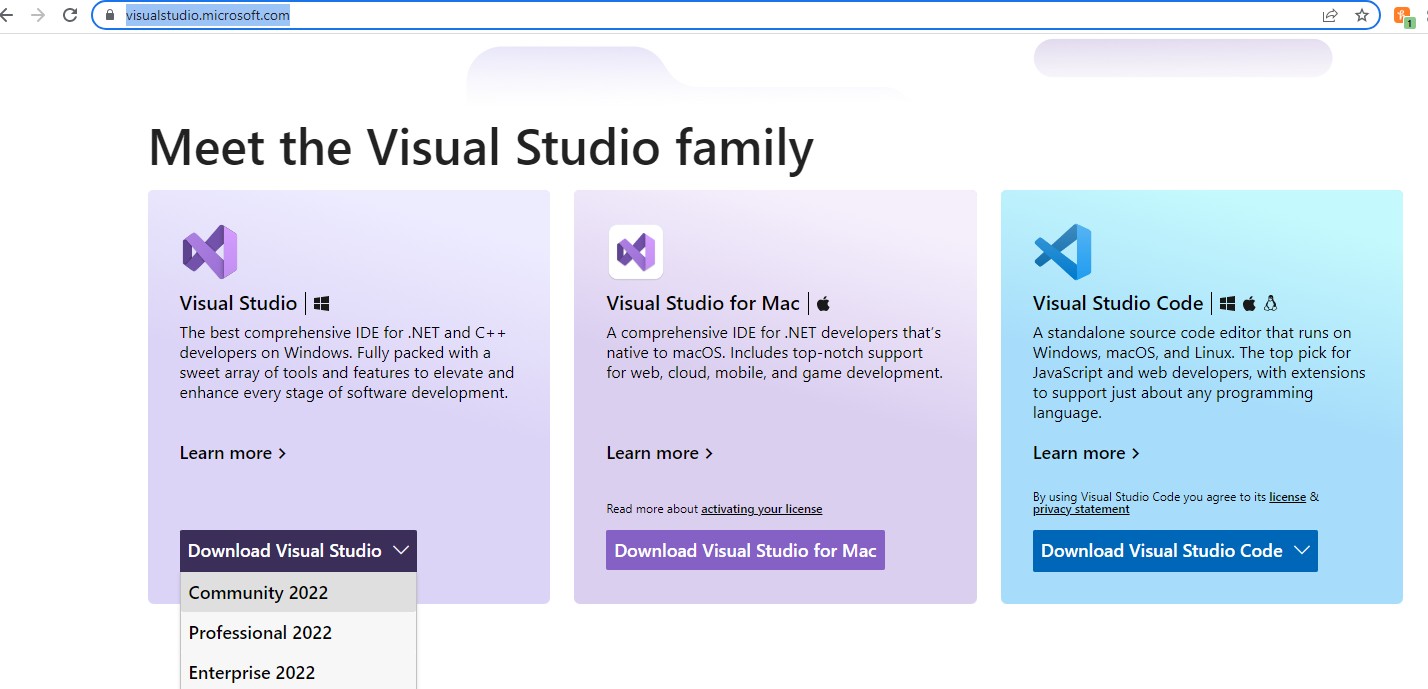
You can now run your program.

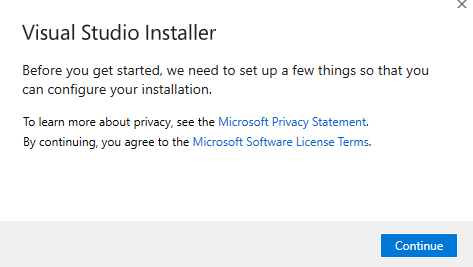
* + - Go to the "Execute" menu, and choose "Run".

**How to install Visual Studio:**

## Step 1: Visual Studio Download

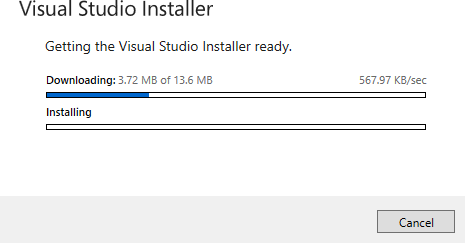
First, you need to download the Visual Studio Ultimate version from Microsoft Download Page <https://visualstudio.microsoft.com/>



**Step 2:** Double Click on Application File Once the software is downloaded, you can extract and double-click on the vs\_ultimate.exe file. Please be sure to start the installation with admin access to avoid any unnecessary permission issues.

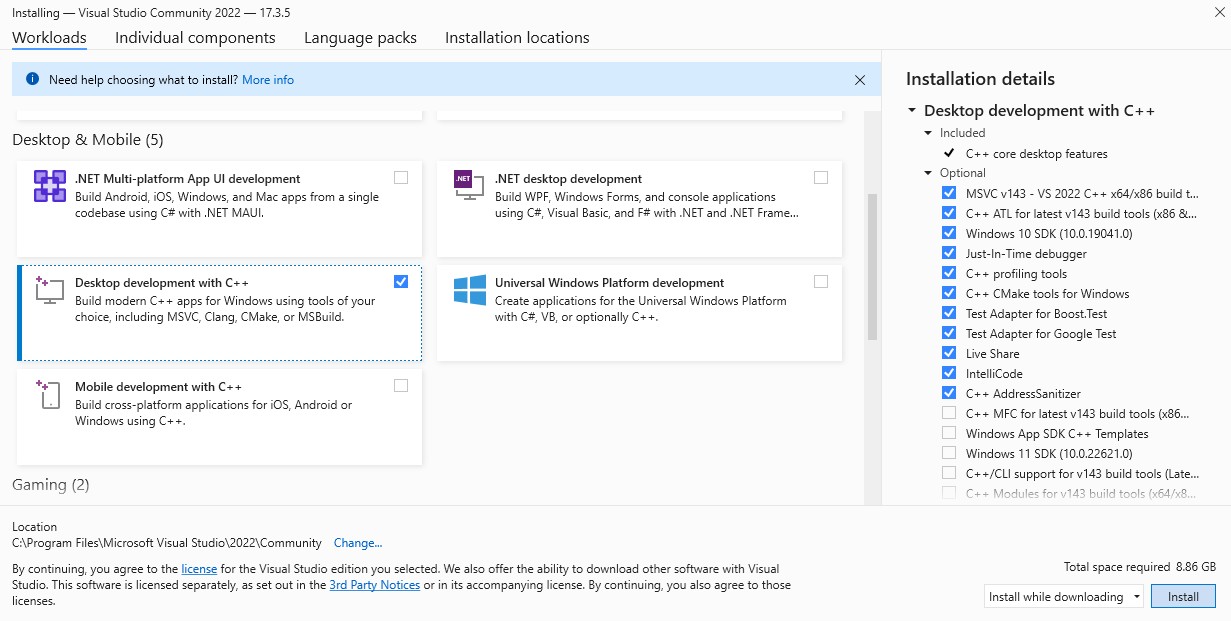
### Step 3: Installation Started

Now you can see that it calculates the amount of space required and check the amount space available and will start the installation process.

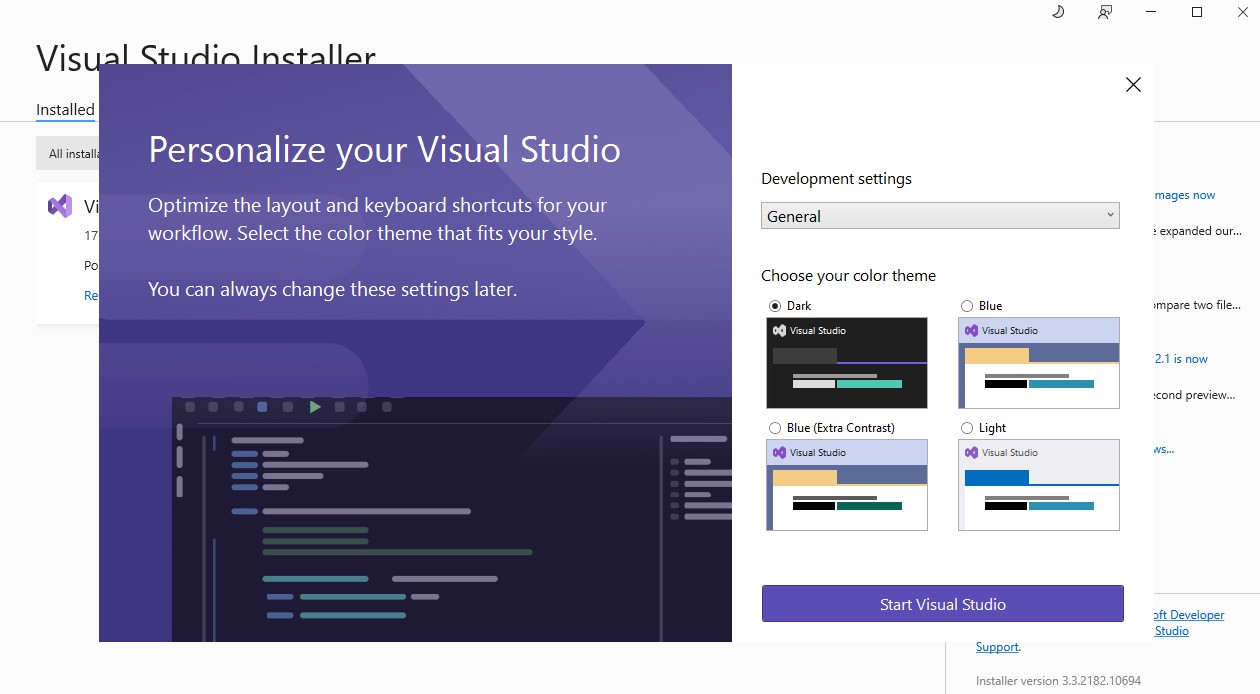


### Step 4: Successful Installation

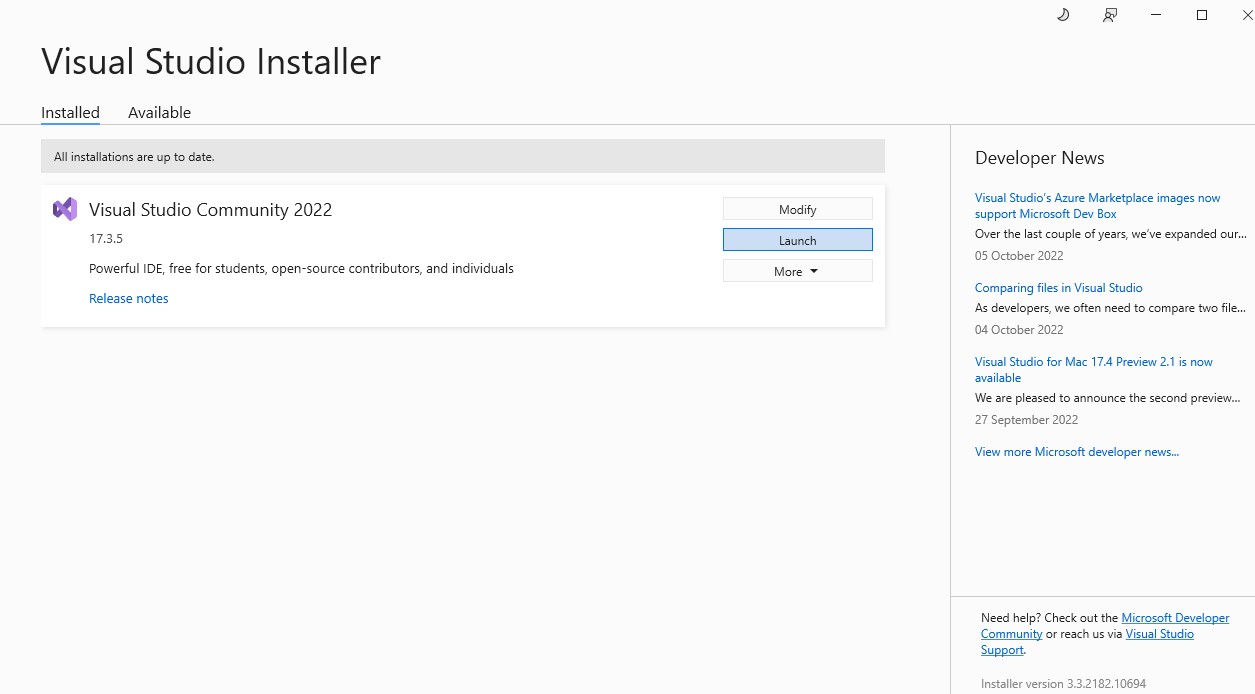
The entire Installation process takes some time usually depends on the number of features you have selected. Once it completes without any error then you will see the below setup successful Screen.



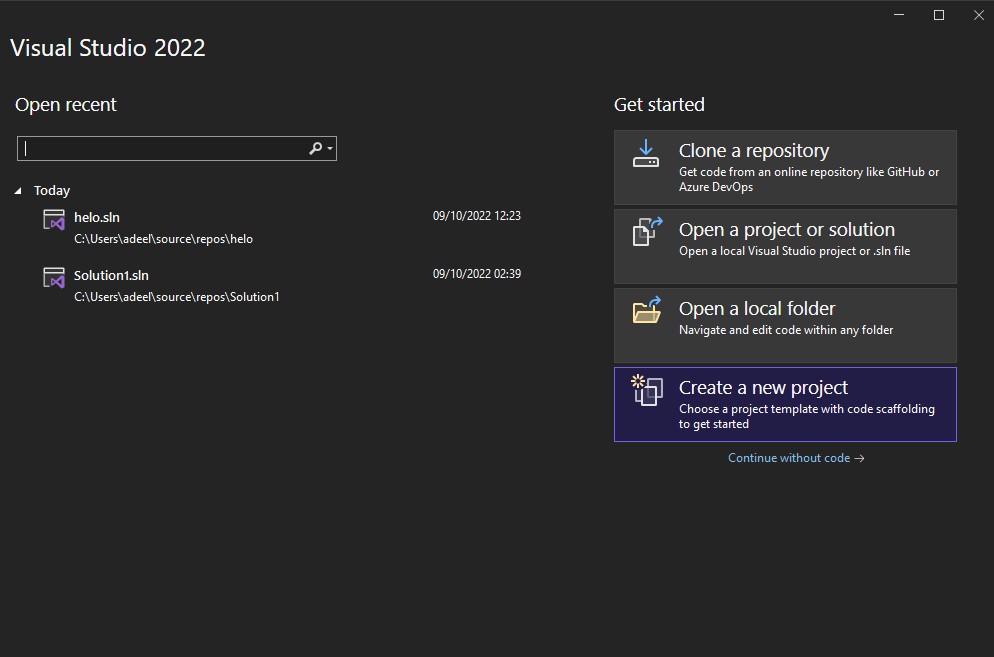
Now select Desktop development C++ and click on Install.



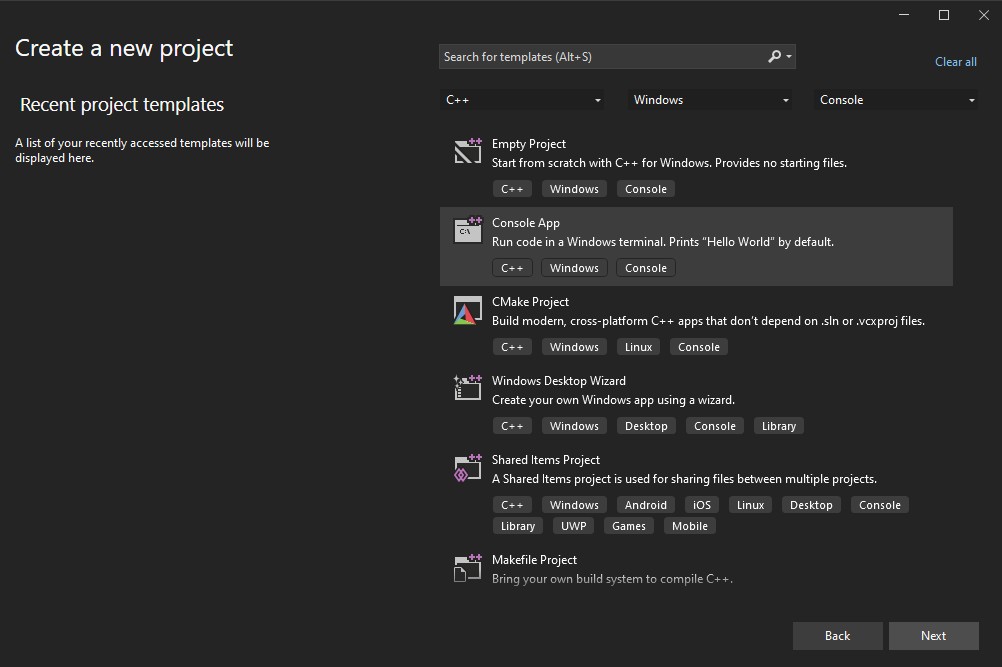
Personalize your Visual Studio.



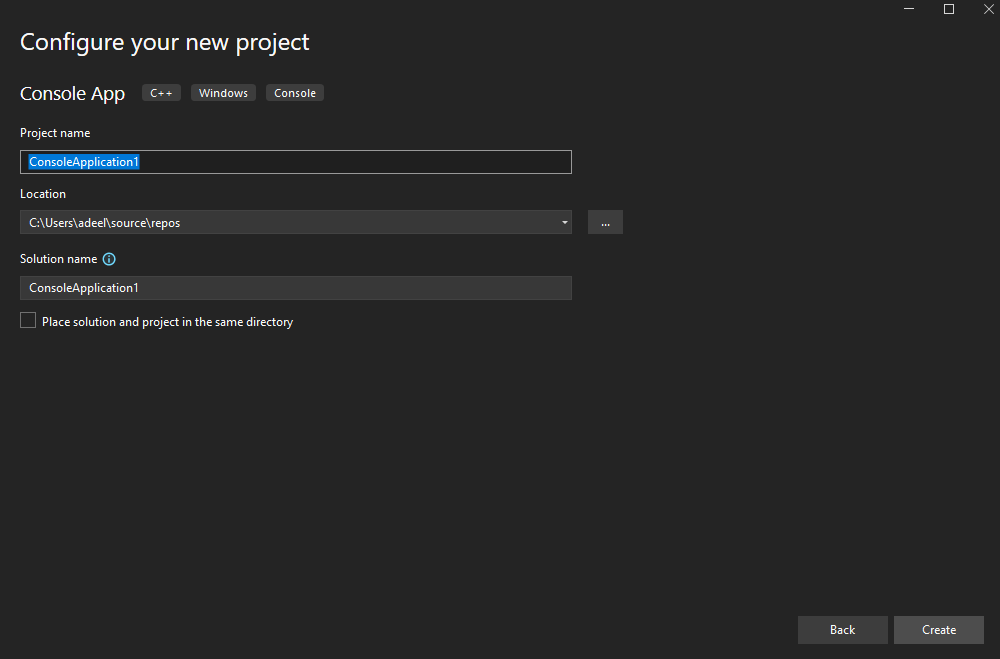
After personalizing your Visual Studio theme click on Launch.



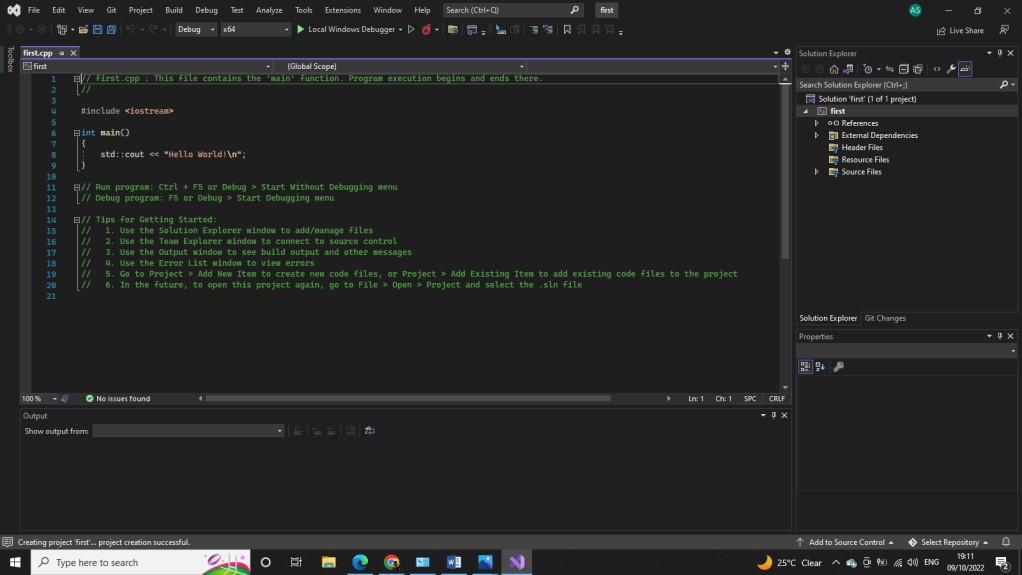
Click on Create a new project.



In create a new project select C++, Windows, and Console in the top search bar Select Console App and click Next.



In Configure your new project enter project name -> Select Location and click on Create.



### Step 5: Write the following program in the window.

### #include <iostream>

using namespace std;

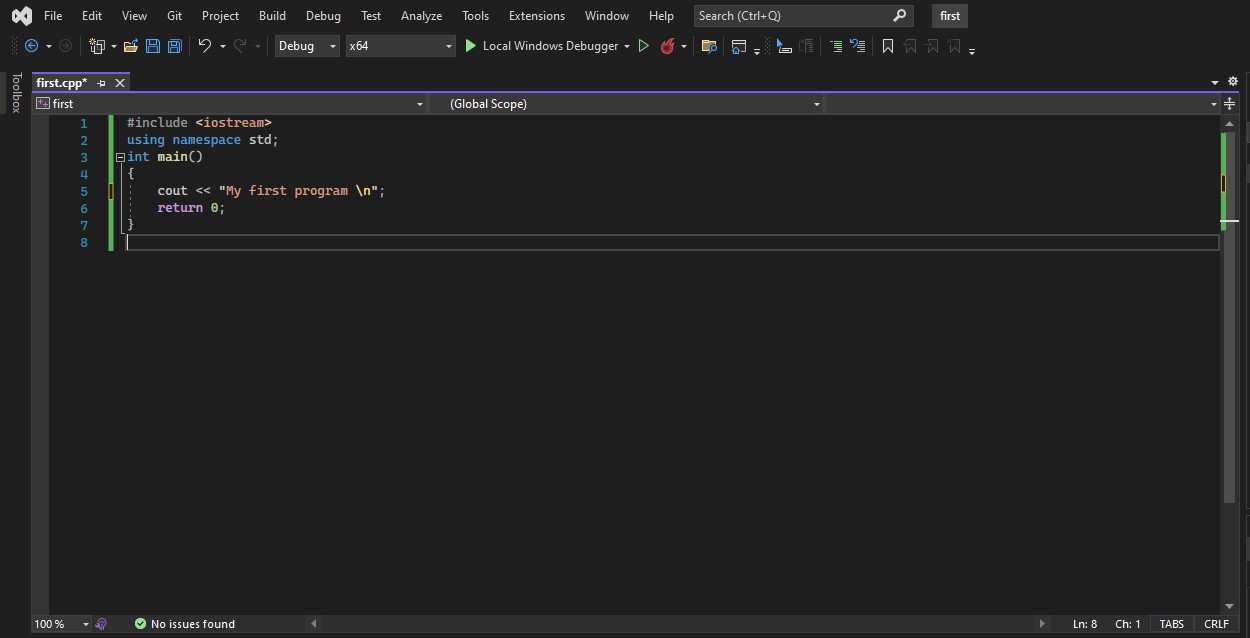
int main()

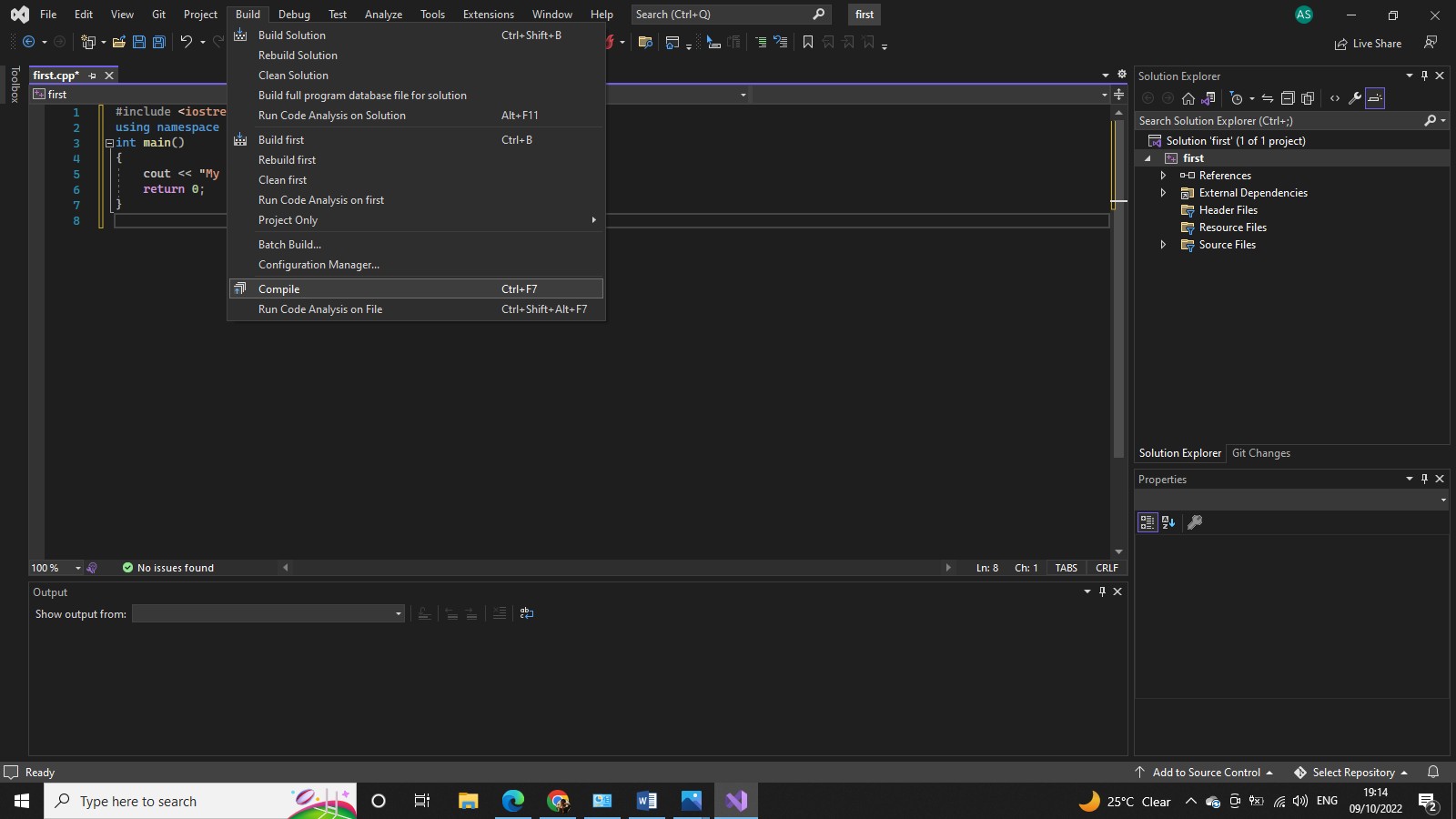
{

cout<<"My first program \n";

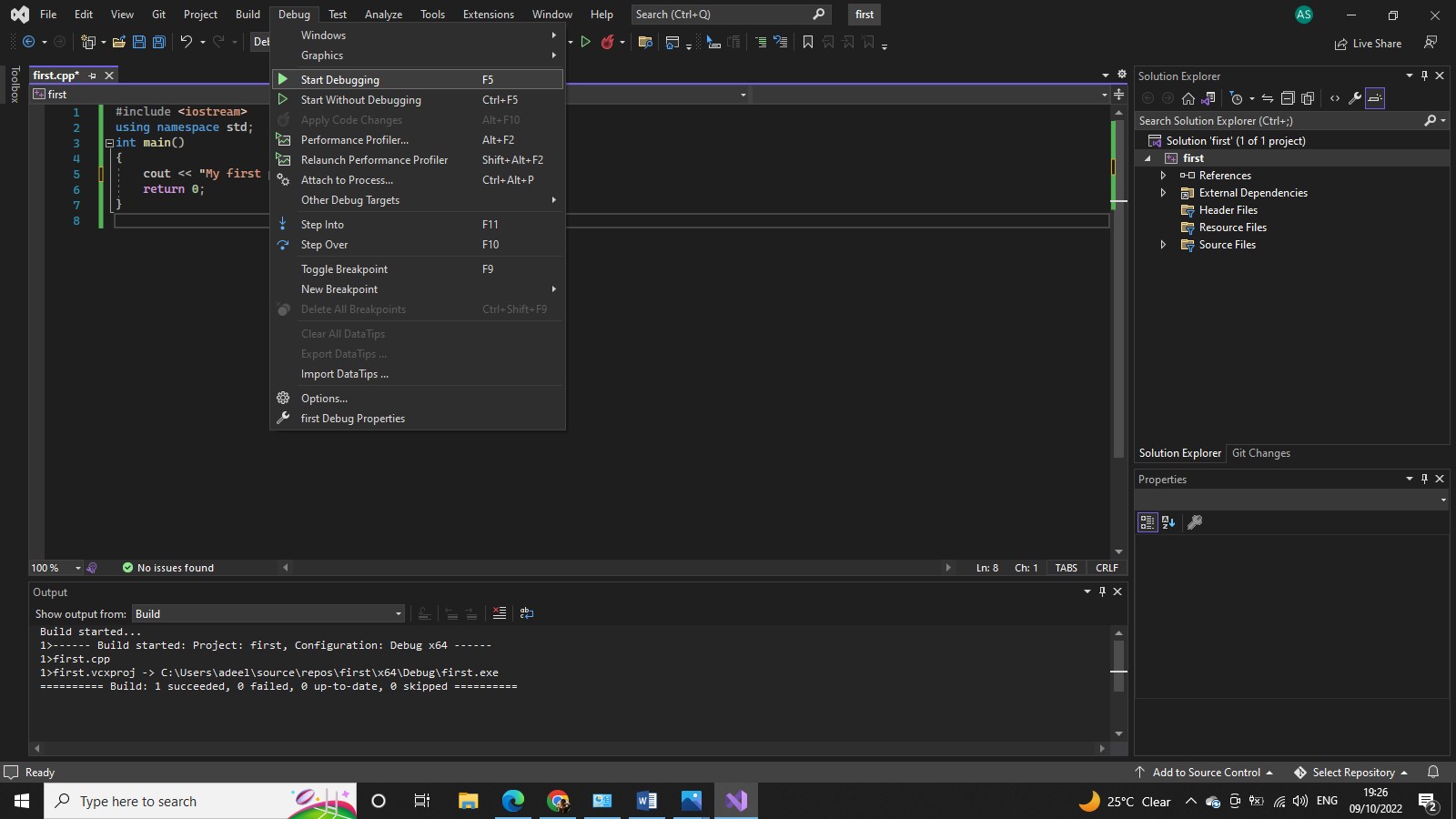
return 0;

}



Now go to Build Menu and click **“Compile”** or press ctrl + F7

After successful compiling it’s time to execute your program. Click on debug menu bar and select star debugging or press F5



#### A console window wit

#### Debugging:

Debugging is the name given to the process of removing bugs (errors) from computer programs.

#### What a Debugger Can Do?

* Step through the program one statement at a time, either "over" or "into" functions.
* Run the program up to a certain point (either to the cursor or to a "breakpoint") and then stop.
* Show the values (or contents) of variables at each point during the execution of the program.

In this lab we will use step over only to execute your code line by line.

#include<iostream>

using namespace std;

int main()

{

Cout<<"First statement executed using step\n";

Cout<<"Second statement executed using step\n";

Cout<<"Third statement executed using step\n";

return 0;

}

First **compile** the above program and then **Execute** it, all the three statements will be printed on the screen. Now again compile the file and execute one statement at a time by clicking on **Debug→Step Over** (or F10). You will see that on the output screen (black screen) the statements will appear one by one.

# Lab No. 2

Variables and Arithmetic Operators

# **Introduction:**

The aim of this lab is to learn how to create variables of primitive data types in C++ compilers (e.g. Visual Studio, Dev C++) and how to use comments and operators to perform mathematical operations.

## Objectives of this lab:

### How to use comments in a program

* **How create variables of different data types**
* **How to use arithmetic operators in a program**

Write following line of code and see the effect of comments.

#include <iostream>

using namespace std;

int main()

{

//declaring integer and character vairables

int a; char ch;

/\*Initializing the

variables

\*/

a=10; ch=’b’;

cout<<“The value of ch is: \n”<<ch;

cout<<“The value of a is: \n”<<a;

return 0;

}

In this example you saw various ways of declaring variables of various data types and how to write comments in C++ programs. Also, notice the use of “\n” simply move the cursor of output to the next line.

Write the following program in the editor and see the effect of the program and compare your result with the following output.

#include <iostream>

using namespace std;

int main()

{

int a = 72;

char b = 'A';

cout<<"\na equals "<<a;

cout<<"\na equals "<<(char)a;

cout<<"\nb equals "<<(int)b;

cout<<"\nb equals "<<b;

return 0;

}

**Output:**

a equals 72 a equals H b equals 65 b equals A

The reason why this works is because a character constant is just an integer from 0 to 255. When we convert integer 72 into char by using (char)a it converts into character H. Similarly for the character A to integer conversion is 65 because the ASCII value of A is 65.

#### Introduction of “cin” statement in your program:

#include <iostream>

using namespace std;

int main()

{

int a,b;

cout<<“Enter value of a: “;

cin>>a;

cout<<“Enter value of b: “;

cin>>b;

cout<<“The value of a is: \n”<<a;

cout“The value of b is: \n”<<b;

cout<<“Enter new value for both separated by a space: \n”<<;

cin>>a;

cout<<” “;

cin>>b;

cout<<”New values are:\n”;

cout<<a<<” “<<b;

return 0;

}

**Introduction of Arithmetic operators in your program:**

After writing this program you will see that how one can add two numbers and following the same way we can use subtraction, multiplication and division operators.

#include <iostream>

using namespace std;

int main()

{

int a,b;

int sum;

cout<<“Enter value of a: \n“;

cin>>a;

cout<<“Enter value of b: \n“;

cin>>b;

sum=a+b;

cout<<“sum is: \t”;

cout<<sum<<”\n”;

return 0;

}

# LAB No. 3

Decision Control Statements

## Objectives of this lab:

### Comparison/Relational Operators

* **Logical Operators**
* **If Statement**
* **If - else statement**
* **Else - if Statement**

Starting from a very simple form of if statement

Example: Write a program in which it takes a number from keyboard as an input and if the number is greater than 100 it prints “The number is greater than hundred”.

**Code:**

#include <iostream>

using namespace std;

int main()

{

int number ;

cout<<“Enter an integer\n”;

cin>>number;

if ( number >100 )

cout<<“The number is greater than 100\n”;

return 0;

}

Try below example and see the effect of if-else statement

#include <iostream>

using namespace std;

int main()

{

int a,b ;

cout<<“Enter first number\n”;

cin>>a;

cout<<“Enter second number\n”;

cin>>b;

if ( a >=b )

// this condition can also be written as if(a>b || a==b)

cout<<a<<”\t”<<b<<”\t”<<a-b; //we can use ‘\t’ for single character

else

cout<<b<<”t”<<a<<”\t”<<b-a;

return 0;

}

Now trying example of Nested if statement

#include <iostream>

using namespace std;

int main()

{

int number ;

cout<<“Enter an integer\n”;

cin>>number;

if ( number <100 )

{

cout<<“Yes the number is less than 100”<<endl;

if ( number <50)

{

cout<<“ and number is also less than 50”<<endl;

}

else

{

cout<<“ but the number is not less than 50”<<endl;

}

}

else

cout<<“No the number is not less than 100”<<endl;

}

Try this example for if-else-if statement

Write a program which takes marks as input and shows the out put as follows:

**Marks Output**

Greater than or equal to 80 Passed: Grade A

Greater than or equal to 65 Passed: Grade B

Greater than or equal to 50 Passed: Grade C

Less than 50 Failed

#include <iostream>

using namespace std;

int main()

{

int marks;

cout<<“Enter an marks\n”;

cin<<marks<<endl;

if (marks >= 80)

cout<<"Passed: Grade A\n";

else if (marks >= 65)

cout<<"Passed: Grade B\n";

else if (marks >= 50)

cout<<"Passed: Grade C\n"<<endl;

else

cout<<"Failed"<<endl;

return 0;

}

Try to modify the above code by adding D grade , criteria is shown below:

**Marks Output**

Greater than or equal to 80 Passed: Grade A

Greater than or equal to 65 Passed: Grade B

Greater than or equal to 50 Passed: Grade C

**Greater than or equal to 40 Passed: Grade D**

Less than 50 Failed

#### Take Home:

1. Write a program, which takes *age* as input from user and prints appropriate message depending upon the following conditions:
   * If age less than 6 then prints, “What a nice child!”
   * If age is between 6 and 9 then prints, “That’s a good age!”
   * If age is between 9 and less than 20 then prints, “Ah! In the prime of life”
   * If age between 20 and < than 30 then prints, “Watch out, younger ones are gaining on you.”
   * More than 30 then it prints, “Well, have fun, and don’t look back.”
2. Write a program which takes 3 numbers as input e.g. a = 30, b = 54 and c = 6 and print output as follows:

**Sample output:**

Min number entered is 6 Max number entered is 54

(These are the values of variables which are taken from keyboard).

1. Write a program that take a number N as input and display on the screen whether N is odd or even. (Hint: if N is divided by 2 and its remainder is 0 then it is even, use % as remainder operator)

**Sample output:**

Enter a number: 8

8 is even

OR

Enter a number: 9

9 is Odd

# LAB No. 4

For, while and Do-While loop, Continue, Switch, Break

## Objectives of this lab:

### To understand the basics of for loop

* + **Concept of while loop**
  + **To understand the usage of Do-While loop**
  + **Concept of switch statement, continue and break statement**

Starting from a very simple and basic example:

**Example 1:** The following code will **print hello** ten times

#include <iostream>

using namespace std;

int main()

{

int count;

for (count =1; count <=10; count++)

cout<<"Hello\n";

return 0;

}

Try the example below and shows the output.

**Example 2:** Calculate simple table of 2 using a for loop

#include <iostream>

using namespace std;

int main()

{

int count;

for (count =1; count <=10; count++)

cout<<"2 x "<<count<<" = "<<(2\*count)<<endl;

return 0;

}

Following examples shows some variations of the for loop

int main()

{

for(int i=0, j=10 ; i<10 ; i++, j-- )

{

cout<<"\n”<<i<<j;

}

return 0;

}

Expression 1 could be written before the body of for loop.

#### Example 3:

int i = 0;

for( ; i < 10; i++)

cout<<“\n”<< i;

Expression 3 could be written in the body of the for loop.

#### Example 4:

int i = 0;

for( ; i < 10; ){

cout<<“\n”<<i;

i++;

}

We can also omit expression 2 so this will be an infinite loop.

#### Example 6:

int i = 0;

for( ; ; ){

cout<<“\n”<< i;

i++;

}

This code will run indefinite times starting from 0. We can use logical operators in expression 2.

#### Example 7:

#include <iostream>

using namespace std;

int main(){

int i=0,j=10;

for(;i<10 && j >4;){

cout<<"\n"<<i<<j;

i++;

j--;

}

return 0;

}

**The *while* loop:**

Try example below and shows the output:

**Example 1:** Print Numbers from 1 to 10 using a while loop

#include <iostream>

using namespace std;

int main()

{

int counter = 1;

while (counter <= 10) // condition

{

cout<<“Counter now reads \n”<<counter;

counter++; // Same as counter=counter + 1 (increment)

}

return 0;

}

Our 2nd example is based on a while-loop that keeps on running until a certain condition is reached (a certain value is entered by the user).

**Example 2:** Printing the numbers you entered using a while loop

#include <iostream>

using namespace std;

int main()

{

int flag; //flag is just an integer variable

cout<<“Enter any number: ( -1 to quit) \n”;

cin>>flag;

cout<<“Entering the while loop now...\n”;

while(flag != -1) {

cout<<“Enter any number: ( -1 to quit)\n ”;

cin>>flag;

cout<<“You entered \n”<<flag;

}

cout<<“Out of loop now \n”;

return 0;

}

**The *do while* loop:**

Try example below and shows the output:

#include <iostream>

using namespace std;

int main( )

{

int x = 0; // Create a local variable 'x'

do

{

x=x+1; // Increment the variable 'x' by 1

} while (x < 3);

return 0;

}

Try some more programs like the one above and show the result.

#### Switch Statement:

Just look at the following example and examine the output:

#include <iostream>

using namespace std;

int main() {

int a;

cout<<"Pick a number from 1 to 4:\n";

cin>>a;

switch (a) {

case 1:

cout<<"You chose number 1\n";

case 2:

cout<<"You chose number 2\n";

case 3:

cout<<"You chose number 3\n";

case 4:

cout<<"You chose number 4\n”;

default:

cout<<"That's not 1,2,3 or 4!\n";

}

return 0;

}

#### The Break Statement:

Try the following example and think on the output

#include <iostream>

using namespace std;

int main() {

int a;

cout<<"Pick a number from 1 to 4:\n";

cin>>a;

switch (a) {

case 1:

cout<<"You chose number 1\n";

break;

case 2:

cout<<"You chose number 2\n";

break;

case 3:

cout<<"You chose number 3\n";

break;

case 4:

cout<<"You chose number 4\n";

break;

default:

cout<<"That's not 1,2,3 or 4!\n";

}

return 0;

}

#### Continue Statement:

This is different than break. Continue statement means skip the remaining statements/line of code of that iteration of loop body and move on to next iteration. Try the example below and compare it with the one above.

int index=0;

for(index=1;index<=10; index++)

{

if(index==4||index==5) continue;

cout<<index;

}

cout<<"\nLoop terminated” ;

**Take Home**

1. Write a program to generate a list of first 100 odd numbers using while, do while and for loops.
2. Write a function which takes as input, a number, total multiplicands and user option to get e (for even) or o (for odd) multiplicands and print table of that number.

Example:

**Input:** Number=3, Total Multiples 15, Multiples types: e

#### Output:

3 \* 2 = 6

3 \* 4 = 12

3 \* 6 = 18

3 \* 8 = 24

……………………….

……………………….

3 \* 14 = 42

1. Write a program to find the sum of digits of the number entered by the user also print it in reverse order. For example, user enters 1234, the sum should be 10 and the program should print 4321. (HINT: use modulus operator)
2. Write a program to ask the user his/her CGPA and print his/her grade accordingly. If grade is C or better give good remarks otherwise leave an advice.
3. Write a program which has the following output screen. (use loops to control output of the program)



1. Write a program to provide following functionality of a calculator using switch case statement.
   * Addition of two integers
   * Subtraction of two integers
   * Multiplication of two integers
   * Division of two integers
   * Addition of two Floating Point Numbers
   * Subtraction of two Floating Point Numbers
   * Multiplication of two Floating Point Numbers
   * Division of two Floating Point Numbers
   * Sine
   * Cosine
   * Tangent
   * Square root
   * Square
   * Cube

User should be able to select his desired operation from the Menu given to him. The program should only terminate when user selects exit operation from the MENU.

For sine, cosine, Tangent and Square root you can use functions available in math.h library.

1. Write a program that takes as input any number of seconds (as int) and then converts it in hours, minutes and seconds. For example, if you enter 7802 the program should print:

2 hrs 10 mins 2 secs

(Hint: Use integer division and modulus operators)

# Lab 5 Arrays

## Objectives of this lab:

### Declaration of Array

* **Initialization of Array**
* **Accessing elements of Array**
* **Printing arrays**

Let's start by looking at the following code where a single variable is used to store a person's age. Code

#include <iostream>

using namespace std;

int main()

{

int age;

age=23;

cout<<endl<< age;

return 0;

}

## Declaration of Array

Here's is the code snippet to create an array and one way to initialize an array:

#include <iostream>

using namespace std;

int main()

{

int age[4]; //declaration of Array

age[0]=23;//initialization of Array elements

age[1]=34;

age[2]=65;

age[3]=74;

return 0;

}

## Initialization of Array

It is like a variable, an array can be initialized. To initialize an array, we provide initializing values which are enclosed within curly braces in the declaration and placed following an equals sign after the array name. Here is an example of initializing an integer array.

int age[4]={23,34,65,74};

## Printing arrays

#include <iostream>

using namespace std;

int main()

{

int age[4];

age[0]=23;

age[1]=34;

age[2]=65;

age[3]=74;

cout<< age<<endl;

return 0;

}

How about printing out each of the values separately? Try this:

#include <iostream>

using namespace std;

int main()

{

int age[4];

age[0]=23;

age[1]=34;

age[2]=65;

age[3]=74;

cout<<age[0]<<endl;

cout<<age[1]<<endl;

cout<<age[2]<<endl;

cout<<age[3]<<endl;

return 0;

}

Lines (10) through line (13) produce the output we are expecting.

Thus there is no single statement in the language that says "print an entire array to the screen". Each element in the array must be printed to the screen individually.

# Lab 6

# Arrays, Practicing with loops

## Objectives of this lab:

* **Copying arrays**
* **Scanning array elements using cin**
* **Dealing with characters using arrays**

## Copying arrays

Suppose that after filling our 4 element array with values, we need to copy that array to another array of 4 int ? Try this:

#include <iostream>

using namespace std;

int main()

{

int age[4];

int same\_age[4];

int i=0;

age[0]=23;

age[1]=34;

age[2]=65;

age[3]=74;

for (;i<4;i++)

same\_age[i]=age[i];

for (i=0;i<4;i++)

cout<<same\_age[i]<<endl;

return 0;

}

## Scanning array elements using cin

int a[5];

cin>>a[0]; // this will scan the value for the very first location of the array.

cout<<a[0]<<endl;

You can also scan the entire elements of the array using a loop. Practice yourself ?

## Dealing with characters using arrays

You can also store characters and other type data (float etc.) in the arrays. Just declare it as we’ve done in the case with int. There is no difference in dealing with characters except you’ve to enclose the value in a single quote. Practice yourself ?

char ar[3];

ar[0]=’a’;

ar[1]=’b’

…..

# Lab 7

# Practicing Multi-dimensional Arrays

## Objectives of this lab:

* **Defining Multi-dimensional arrays**
* **Initialization of Multi-dimensional arrays**
* **Practicing Multi-dimensional arrays**

## Defining multi-dimensional arrays

In C++, we can create an array of an array, known as a multidimensional array. For example:

2-Dimensional Array:

int matrix[10][20];

3-Dimensional Array:

char deptNames[3][10][20];

**2) Multidimensional Array Initialization:**

Like a normal array, we can initialize a multidimensional array in more than one way.

**a). Initialization of two-dimensional array**

int test[2][3] = {2, 4, 5, 9, 0, 19};

The above method is not preferred. A better way to initialize this array with the same array elements is given below:

int test[2][3] = { {2, 4, 5}, {9, 0, 19}};

This array has 2 rows and 3 columns, which is why we have two rows of elements with 3 elements each.

**b). Initialization of three-dimensional array**

int test[2][3][4] = {3, 4, 2, 3, 0, -3, 9, 11, 23, 12, 23, 2, 13, 4, 56, 3, 5, 9, 3, 5, 5, 1, 4, 9};

This is not a good way of initializing a three-dimensional array. A better way to initialize this array is:

int test[2][3][4] = {

{ {3, 4, 2, 3}, {0, -3, 9, 11}, {23, 12, 23, 2} },

{ {13, 4, 56, 3}, {5, 9, 3, 5}, {5, 1, 4, 9} }

};

Notice the dimensions of this three-dimensional array.

**3) Practicing Multidimensional Array:**

**Example 1:**

#include<iostream>

using namespace std;

int main( )

{

int arr[4][2] = {

{ 10, 11 },

{ 20, 21 },

{ 30, 31 },

{ 40, 41 }

} ;

int i,j;

cout<<"Printing a 2D Array:\n";

for(i=0;i<4;i++)

{

for(j=0;j<2;j++)

{

cout<<"\t"<<arr[i][j];

}

cout<<endl;

}

return 0;

}

**Example 2:**

#include<iostream>

using namespace std;

int main( )

{

int s[2][2];

int i, j;

cout<<"\n2D Array Input:\n";

for(i=0;i<2;i++)

{

for(j=0;j<2;j++)

{

cout<<"\ns["<<i<<"]["<<j<<"]= ";

cin>>s[i][j];

}

}

cout<<"\nThe 2-D Array is:\n";

for(i=0;i<2;i++)

{

for(j=0;j<2;j++)

{

cout<<"\t"<<s[i][j];

}

cout<<endl;

}

return 0;

}

**Example 3:**

#include<iostream>

using namespace std;

int main()

{

int m1[5][5], m2[5][5], m3[5][5];

int i, j, r, c;

cout<<"Enter the no.of rows of the matrices to be added(max 5):";

cin>>r;

cout<<"Enter the no.of columns of the matrices to be added(max 5):";

cin>>c;

cout<<"\n1st Matrix Input:\n";

for(i=0;i<r;i++)

{

for(j=0;j<c;j++)

{

cout<<"\nmatrix1["<<i<<"]["<<j<<"]= ";

cin>>m1[i][j];

}

}

cout<<"\n2nd Matrix Input:\n";

for(i=0;i<r;i++)

{

for(j=0;j<c;j++)

{

cout<<"\nmatrix2["<<i<<"]["<<j<<"]= ";

cin>>m2[i][j];

}

}

cout<<"\nAdding Matrices...\n";

for(i=0;i<r;i++)

{

for(j=0;j<c;j++)

{

m3[i][j]=m1[i][j]+m2[i][j];

}

}

cout<<"\nThe resultant Matrix is:\n";

for(i=0;i<r;i++)

{

for(j=0;j<c;j++)

{

cout<<"\t"<<m3[i][j];

}

cout<<endl;

}

return 0;

}

**Example Do yourself:**

Multiply two matrices A and B of order 2x2 using your matrices multiplication concepts?

# LAB NO. 8

Structures in C++

## Objectives of this lab:

### To understand the concept of Structure

#### Let start from a simple example:

The following program will declare a structure of two member functions and two data member.

#include<iostream>

using namespace std;

struct test

{

int x, y;

void set(int a, int b)

{

x=a;

y=b;

}

void get()

{

cout<<”The value of X is: ”<<x<<endl;

cout<<”The value of Y is: ”<<y<<endl;

}

};

int main()

{

Test obj;

obj.set(10,20);

obj.get();

return 0;

}

#### A more precise example:

#include<iostream>

using namespace std;

struct Games

{

char Name[80];

char Rating;

bool Played;

long NumberOfKills;

}

Unreal, Blizzard[3];

typedef Games\* Gamesptr;

void main()

{ Unreal.Rating=1;

strcpy(Unreal.Name,"Unreal");

Unreal.Played=True;

Unreal.NumberOfKills=100;

Games Quake;

Quake.Rating=2;

strcpy(Quake.Name,"Quake");

Quake.Played=True;

Quake.NumberOfKills=100;

GamesList[0]->Rating=1;

strcpy(GamesList[0]->Name, "Warcraft2");

Gamesist[0]->Played = False;

GamesList[0]->NumberOfKills = 0;

}

#### Multiple structures in a Program

One can declare multiple structures in a program and can be call from main

#include<iostream>

using namespace std;

struct student

{

int id;

char name[15];

double gpa;

void GPA()

{

float marks;

cin>> marks;

if(marks>=60 &&<=100)

cout<<”Grand Point Average of the student is between 2 and 4 ”<<endl;

else cout<<”Grand Point Average of the student is between 0 and 1.99 ”<<endl;}

# LAB No. 9

Functions (Call by value and by reference), Random function

## Objectives of this lab:

### To understand the Concept of functions

* **Passing value to a function by value**
* **Passing value to a function by reference**

Let’s do an example which calls a function which prints ten asterisks (\*) in line. (\*\*\*\*\*\*\*\*\*\*)

#include<iostream>

using namespace std;

void asteriks(); // prototype declaration

void main(){

asteriks(); // Function calling

}

void asteriks(){ // Function definition

int i=0;

for(;i<10;i++)

cout<<"\*";

}

Lets go one step ahead, function asterisks (int a) with a single argument.

#include<iostream>

using namespace std;

void asteriks(int n); // prototype declaration

void main(){

asteriks(7); // Function calling

}

void asteriks(int num){ // Function definition

int i=0;

for(;i<num;i++)

cout<<"\*";

}

Now an example of a function with two parameters

Here is an example of a function which calculates the sum of two numbers passed as arguments to the function and returns the sum to the calling function.

int add(int no1,int no2 ){

int sum = no1+no2;

return sum;

}

Lets do another example of making a program using functions which will tell us whether the input number is even or odd.

#include<iostream>

using namespace std;

int is\_even(int n); // (Prototype declaration)

void main()

{

int number, test;

cout<<"Enter a number to test even or odd";

cin>>number;

test = is\_even(number); // (Function calling)

if(test==0)

cout<<"\nThe number is odd";

else

cout<<"\nThe number is even";

}

int is\_even(int n){ //Function definition

int remainder;

remainder= n%2;

if(remainder==1)

return 0;

else

return 1;

}

One more way to write the same example is

#include<iostream>

using namespace std;

int is\_even(int n){ //Function definition

int remainder;

remainder= n%2;

if(remainder==1)

return 0;

else

return 1;

}

void main()

{

int number, test;

cout<<"Enter a number to test even or odd\n";

cin>>number;

test = is\_even(number); //(Function calling)

if(test==0)

cout<<"\nThe number is odd";

else

cout<<"\nThe number is even";

}

Another way to write the same program

#include<iostream>

using namespace std;

int is\_even(int n); // (Prototype declaration)

void main()

{

int number;

cout<<"Enter a number to test even or odd\n";

cin>>number;

if(is\_even(number)) //Function call within if condition

cout<<"\nThe number is even";

else

cout<<"\nThe number is odd";

}

int is\_even(int n){ //Function definition

int remainder;

remainder= n%2;

if(remainder==1)

return 0;

else

return 1;

}

#### Compile and run this program.

#include<iostream>

using namespace std;

float avg(float no1, float no2, float no3); //(Prototype declaration)

void main()

{

float n1,n2,n3,result;

cout<<"Enter three number to find avg");

cin>>n1>>n2>>n3;

result=avg(n1,n2,n3); // (Function calling)

cout<<"\nThe average is \n”<<result;

}

float avg(float a, float b, float c){

cout<<"\nEntering the function\n";

float average= (a+b+c)/3;

return average;

}

### Call by Value:

#### Example 1:

void func( int );

int main( ) {

int i = 8;

func( i );

cout<<“ the value of i is ”<< i;

return 0;

}

void func( int i) {

i = i + 10;

}

#### Example 2:

void interchange(int,int);

int main()

{

int x=50, y=70;

interchange(x,y);

cout<<“ x= ”<<x<<” y= ”<<y;

return 0;

}

void interchange(int x1, int y1)

{

int z1;

z1=x1;

x1=y1;

y1=z1;

cout<<“x1 is = “<<x1<<”y1 is “<<y1;

}

#### Call by Reference: Example 3:

#include<iostream>

using namespace std;

void interchange(int&, int&);

int main()

{

int x=50, y=70;

cout<<“ x= ”<<x<<” y= ”<<y;

interchange(x,y);

cout<<“ x= ”<<x<<” y= ”<<y;

return 0;

}

void interchange(int & x1, int & y1)

{

int z1;

z1=x1;

x1=y1;

y1=z1;

}

Here the function is called by reference. In other words address is passed by using symbol “&”

The main difference between them can be seen by analyzing the output of program1 and program2.

**Exercise Questions:**

**Program1:**

Without changing the main function, complete the program for the following output:



#include <iostream>

using namespace std;

int main()

{

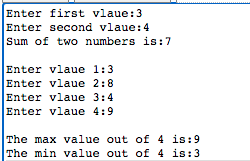
greet();

return 0;

}

**Program2:**

Without changing the main function, complete the program for the following output:



#include <iostream>

using namespace std;

int main() {

addTwoNumber();

cout<<endl<<endl;

findMaxMin();

return 0;

}

**Solutions:**

**Program1:**

Without changing the main function, complete the program for the following output:



#include <iostream>

using namespace std;

int main()

{

greet();

return 0;

}

**Solution:**

#include <iostream>

using namespace std;

// declaring a function

void greet();

int main() {

// calling the function

greet();

return 0;

}

// defining a function

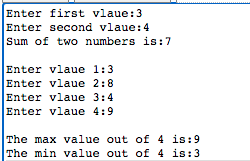
void greet() {

cout << "Hello there!";

}

**Program2:**

Without changing the main function, complete the program for the following output:



#include <iostream>

using namespace std;

int main() {

addTwoNumber();

cout<<endl<<endl;

findMaxMin();

return 0;

}

**Solution:**

#include <iostream>

using namespace std;

// function declaration

void addTwoNumber();

void findMaxMin();

int main() {

addTwoNumber();

cout<<endl<<endl;

findMaxMin();

return 0;

}

// function definitions

void addTwoNumber()

{ int first,second;

cout<<"Enter first vlaue:";

cin>>first;

cout<<"Enter second vlaue:";

cin>>second;

cout<<"Sum of two numbers is:"<<(first+second);

}

void findMaxMin()

{

int A[4];

for(int i=0;i<4;i++)

{ cout<<"Enter vlaue "<<i+1<<":";

cin>>A[i];

}

int max=A[0];

int min=A[0];

for(int i=0;i<4;i++)

{ if(A[i]>max)

max=A[i];

if(A[i]<min)

min=A[i];

}

cout<<"\nThe max value out of 4 is:"<<max;

cout<<"\nThe min value out of 4 is:"<<min;

}

# LAB NO. 10

Implementation of cstring and its functions

## Objectives of this lab:

### How cstrings defined and implemented

* + **Built in functions of cstring**
  + **How cstring functions are useful for our programs**

**Example1: The strlen() function in C++ returns the length of the given C-string. It is defined in the cstring header file**

#include <iostream>

#include <cstring>//In some compilers this header file is not required e.g. cpp.sh

using namespace std;

int main() {

// initialize C-string

char dept[] = "University Institute of IT";

// print the length of the dept string

cout << strlen(dept);

return 0;

}

**Example2:The strcpy() function in C++ copies a character string from source to destination.**

#include <iostream>

#include <cstring>

using namespace std;

int main() {

char dept[] = "University Institute of IT";

char dept2[30];

strcpy(dept2,dept);//first argument is destination

cout << dept2;

return 0;

}

**Example3:The strcat() function in C++ appends a copy of a string to the end of another string.**

#include <iostream>

#include <cstring>

using namespace std;

int main() {

char dept[] = "University Institute of IT";

char dept2[]= " University Institute of MS";

strcat(dept,dept2);

cout << dept;

return 0;

}

**Example4: The strcmp() function in C++ compares two null-terminating strings (C-strings).**

#include <iostream>

#include <cstring>

using namespace std;

int main() {

char dept[] = "University Institute of IT";

char dept2[]= " University Institute of MS";

int result=strcmp(dept,dept2);

if(result==0)

cout<<"Both strings are equal";

else

cout<<"Both strings are not equal";

//A positive value will return if the first differing character in first string is greater than the corresponding character in second string, otherwise return negative value. It does not depend on the length of string.

return 0;

}

**Example5: The strncmp() function in C++ compares a specified number of characters of two null terminating strings.**

#include <iostream>

#include <cstring>

using namespace std;

int main() {

char dept[] = "University Institute of IT";

char dept2[]= "University Institute of MS";

int result=strncmp(dept,dept2,20);

//cout<<result;

if(result==0)

cout<<"Both strings are equal";

else

cout<<"Both strings are not equal";

return 0;

}

**Example6: The strerror() takes an argument: errnum which is an integer value that represents the error code. This function converts the error code to a suitable string that describes the error.**

#include <cstring>

#include<fstream>

#include <iostream>

using namespace std;

int main()

{

/\* example.txt does not exist \*/

ifstream fin("example.txt");

if (!fin)

cout << "Error opening file : " << strerror(errno) << endl;

return 0;

}

**Example7: The strtok() function in C++ returns the next token in a C-string (null terminated byte string)**

#include <cstring>

#include <iostream>

using namespace std;

int main() {

char dept[] = "University Institute of Information Technology";

char\* word=strtok(dept, " ");;

int count=0;

while(word!=NULL)

{

count++;

cout << "token "<<count<<" = " << word << endl;

word = strtok(NULL, " ");

}

return 0;

}

**Example8: The strstr() function in C++ finds the first occurrence of a substring in a string.**

#include <cstring>

#include <iostream>

using namespace std;

int main() {

char dept[] = "University Institute of Information Technology";

char target[]="Info";

char \*p = strstr(dept, target);

if (p)

cout << "'" << target << "' is present in \"" << dept << "\" at position " << p-dept;

else

cout << target << " is not present in \"" << dept << "\"";

return 0;

}

**Example9: The memchr() function in C++ searches in a string for the first occurrence of a character in a specified number of characters.**

#include <iostream>

using namespace std;

int main()

{

char ptr[] = "University Institute of Information Technology";

char ch = 'z';

int count = 15;

if (memchr(ptr,ch, count))

cout << ch << " is present in first " << count << " characters of \"" << ptr << "\"";

else

cout << ch << " is not present in first " << count << " characters of \"" << ptr << "\"";

return 0;

}

**Example10: The memcmp() function in C++ compares a specified number of characters of two strings:**

#include <iostream>

using namespace std;

int main()

{

char str1[] = "Hello World!";

char str2[] = "Hello Earth!";

int result;

result = memcmp(str1, str2, 5);

if (result==0)

cout <<"First 5 characters are same";

else

cout <<"First 5 characters are not same";

return 0;

}

# LAB NO. 11

Pointers

## Objectives of this lab:

### Pointer initialization

* + **Simple pointer manipulation**
  + **Pass parameter by reference**
  + **Dynamic Data and Pointers**
  + **Pointers and arrays**

#### Pointer initialization:

When declaring pointers we may want to explicitly specify which variable we want them to point to:

Write the following code and see the output.

#include <iostream>

using namespace std;

int main () { int number;

int \*point = &number;

cout<<\*point<<endl;

return 0;

}

#### Simple Pointer Manipulation:

Pointers are a type of variable that allow you to specify the address of a variable.

Let start an example, following code will exchange two variable try it and check whether two values exchange or not.

#include <iostream>

using namespace std;

void exchange (int x, int y);

int main () {

int a = 5;

int b = 9;

cout << "This program attempts to exchange two values." << endl;

cout << "Values before the exchange:" << endl;

cout << "a= " << a << " b= " << b << endl;

exchange(a, b); // code that calls the function

cout << "Values after the exchange:" << endl;

cout << "a= " << a << " b= " << b << endl;

return 0;

}

// function for passing by value

void exchange (int x, int y) {

int temp;

temp = x;

x = y;

y = temp;

} // end exchange

**Output:** In output you see that two values are not exchange with each other, if that is what you really wanted to do, you should have used pointers to pass the parameters by address (pointers) or references to pass the parameters by reference.

### Now try the following code, or you can modify the program above.

#include <iostream>

using namespace std;

void exchange (int \* x, int \* y);

int main () {

int a = 5;

int b = 9;

cout << "This program exchanges 2 values." << endl;

cout << "Values before the exchange:" << endl;

cout << "a= " << a << " b= " << b << endl;

exchange(&a, &b); // code that calls the function

cout << "Values after the exchange:" << endl;

cout << "a= " << a << " b= " << b << endl;

}

// function for passing by address (or pointers)

void exchange (int \* x, int \* y) {

int temp;

temp = \*x;

\*x = \*y;

\*y = temp;

return; // Optional for void functions

} // end exchange

One can use following code for call by reference.

#include <iostream>

using namespace std;

void exchange (int& x, int& y);

int main () {

int a = 5;

int b = 9;

cout << "This program exchanges 2 values." << endl;

cout << "Values before the exchange:" << endl;

cout << "a= " << a << " b= " << b << endl;

exchange(a, b); // code that calls the function

cout << "Values after the exchange:" << endl;

cout << "a= " << a << " b= " << b << endl;

}

// function for passing by reference

void exchange (int& x, int& y) {

int temp;

temp = x;

x = y;

y = temp;

return; // Optional for void functions

} // end exchange

Now, when the function is executed, the values of a and b will be changed in the main program.

#### Pointers and arrays:

The concept of array is very much bound to the one of pointer. In fact, the identifier of an array is equivalent to the address of its first element, as a pointer is equivalent to the address of the first element that it points to, so in fact they are the same concept.

Let try an example and watch the output.

**Example 1:**

#include <iostream>

using namespace std;

int main ()

{

int numbers[5];

int \* p;

p = numbers; \*p = 10;

p++; \*p = 20;

p = &numbers[2]; \*p = 30;

p = numbers + 3; \*p = 40;

p = numbers; \*(p+4) = 50;

for (int n=0; n<5; n++)

cout << numbers[n] << ", ";

return 0;

}

**Example 2:** we can use a pointer variable to change the value of another pointer. See how it happens

#include<iostream> using namespace std;

int main()

{

int a = 50; // initialize integer variable a

cout<<"The value of 'a': "<<a<<endl; // show the output of a

// change the value of 'a' using pointer 'b'

\*b = 100;

// transfer the address of 'a' to pointer 'b'

b = &a;

// declare an integer pointer b

int \* b;

cout<<"The value of 'a' using \*b: "<<a<<endl;// show the output of a

// change the value of 'a' using pointer to pointer 'c'

\*\*c = 200;

// transfer the address of 'b' to pointer to pointer 'c'

c = &b;

// declare an integer pointer to pointer 'c'

int \*\*c;

cout<<"The value of 'a' using \*\*c: "<<a<<endl;// show the output of a

return 0;

}

# LAB NO. 12

Dynamic Memory Allocation

## Objectives of this lab:

* + **To understand the concept of Dynamic memory allocation**
  + **New and Delete operator**

#### Dynamic Memory Allocation:

Let us try following program to demonstrate the use of dynamic data and pointers.

**Example 1:**

#include <iostream>

using namespace std;

int main() {

int x, \*p1, \*p2;

x = 1;

p1 = new int;

\*p1 = 5;

p2 = new int;

\*p2 = 3;

cout << "p1 is " << \*p1 << "\np2 is " << \*p2 << "\nx is " << x << endl; // Line 15

x = \*p2;

p1 = &x;

cout << "p1 is " << \*p1 << "\np2 is " << \*p2 << "\nx is " << x << endl; // Line 18

return 0;

}

**Example 2:** This example covers some basic usage of new and delete operator.

#include <iostream> #include <cstring>

using namespace std; int main ()

{

double \*d; // d is a variable whose purpose

// is to contain the address of a

// zone where a double is located

d = new double; // new allocates a zone of memory

// large enough to contain a double

// and returns its address.

// That address is stored in d.

\*d = 45.3;

// The number 45.3 is stored

// inside the memory zone

// whose address is given by d.

cout << "Type a number: ";

cin >> \*d;

\*d = \*d + 5;

cout << "Result: " << \*d << endl;

delete d; // delete deallocates the

// zone of memory whose address

// is given by pointer d.

// Now we can no more use that zone.

d = new double[15]; // allocates a zone for an array

// of 15 doubles. Note each 15

// double will be constructed.

// This is pointless here but it

// is vital when using a data type

// that needs its constructor be

// used for each instance.

d[0] = 4456;

d[1] = d[0] + 567;

cout << "Content of d[1]: " << d[1] << endl;

delete [] d;

// delete [] will deallocate the

// memory zone. Note each 15

// double will be destructed.

// This is pointless here but it

// is vital when using a data type

// that needs its destructor be

// used for each instance (the ~

// method). Using delete without

// the [] would deallocate the

// memory zone without destructing

// each of the 15 instances. That

// would cause memory leakage.

int n = 30;

d = new double[n]; // new can be used to allocate an

// array of random size.

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

{

d[i] = i;

}

delete [] d;

char \*s;

s = new char[100];

strcpy (s, "Hello!");

cout << s << endl;

delete [] s;

return 0;

}

**Example 3:** Following example is used to dynamically allocate memory to an array of user define size.

#include<iostream> using namespace std;

int main()

{

int size;

cout<<”Enter size of array”<<endl; cin>>size;

int \*p=new int[size];

for (n=0; n<i; n++)

{

cout << "Enter number: ";

cin >> p[n];

}

cout << "You have entered: ";

for (n=0; n<i; n++)

cout << p[n] << ", ";

delete[] p;

return 0;

}

**Lab Exercise Problems:**

1. Declare a dynamic data array to contain the scores for 10 students. Then, step through the array assigning the counter as the value of the score for that array element. Finally, write another for loop that displays each score. Remember that you have to use the pointer to access the array.
2. A pointer variable can be use to read every element of an array and then print all the element of the array until pointer reaches to the end of the array, write a program that stores your name in an array and then initialize a pointer for reading and printing every character of the array.
3. According to the Gregorian calendar it was Monday on the date 1/1/01.if any year is input through the keyboard write a program to find out what is the on 1st January of this year.
4. A Pointer can be used to point it to another pointer, use it in a program and change the value of original variable using second pointer through first pointer.
5. Design a simple program that illustrate the basic usage of New and Delete operator.
6. Design a program that dynamically allocates memory to an array of user defined size.
7. Design a program that allocates an array of random size.

# LAB NO. 13

File Handling in C++

#### Introduction:

The aim of this lab is to understand the concept of filing (permanent data files).

**Objectives of this Lab:**

### To understand the concept of File Handling in C++

* **File Input with Characters**
* **File read and write**

Let us start with a very basic example:

Write following code in the editor and see what happens. Note that you will have to create a test text file.

#include <fstream>

#include <iostream>

using namespace std;

int main( )

{

ofstream Savefile("test.txt");

return 0;

}

**Make a following change to above code**

#include <fstream>

#include <iostream>

using namespace std;

int main( ){

ofstream Savefile("test.txt");

Savefile<< "UIIT Department, Arid University";

return 0;

}

**Now check out your file, and not down what happens to it.**

Writing multiple lines to your file, write down following code in your editor

#include<fstream.h>

int main( )

{

ofstream outfile("test.txt");

outfile<< "This is first line of Program\n";

outfile<< "This is second line of Program\n";

return 0;

#### File Input with Characters

Write down following code and write characters to your file.

#### Example 1: Write a C++ program which reads a string, character by character from a permanent file test.txt present in the working directory and displays on the console screen.

#include <fstream>

#include <iostream>

using namespace std;

int main( )

{

char ch;

ifstream Readfile("test.txt");

while(Readfile)

{

Readfile.get(ch);

cout<< ch;

}

cout<< endl;

return 0;

}

#### Example 2: Write a C++ program which writes a string constant into a permanent file test.txt present in the working directory.

#include <fstream>

#include <iostream>

#include <string>

using namespace std;

void main( ){

String str = "If you start judging the people then you will have no time to love them!";

ofstream Savefile("test.txt"); //for specified path we need “D:\\folderName”

Savefile<<str;

cout << "File written\n";

Savefile.close();

}

#### Example 3: Write a C++ program which reads data from a file record.txt through a read function. One record consists of 3 values, i.e. rollnumber, name and cgpa. If cgpa is >=3.5, then write that record in another file search.txt through a separate write function.

#include<iostream>

#include<fstream>

using namespace std;

struct student

{

int r;

char name[20];

float cgpa;

};

void readRecord();

void writeRecord(student);

int main()

{

readRecord();

return 0;

}

void readRecord()

{

student s;

ifstream fin("record.txt");

while(fin>>s.r>>s.name>>s.cgpa)

if(s.cgpa >=3.5)

writeRecord(s);

fin.close();

}

void writeRecord(student x)

{

ofstream fout("search.txt");

fout<<x.r<<x.name<<x.cgpa<<endl;

fout.close();

}

# **Practice Exercise:**

# Write a C++ program which read password from file pwd.txt and also get password from user. Compare both and if matched, display “Password Matched Successfully”, otherwise display “Invalid Password”