

OOP Lab 5: Operator overloading

Date: 16-10-2020

Total marks: 15+15=30

Deadline: same day before 11:59 pm (sharp)

Submission instructions

1. Submit your file in a .zip or .rar format. Name your file with your Roll number (e.g. BSEF19M009.zip or BSEF19M009.rar)
2. Make a separate folder for each lab task (e.g. task1, task2...). All the headers and .cpp files must be in respective folder with a screenshot of the output of a program of the respective task.
3. Every Cpp file must contain your name and roll number(In comments) at the top of each program
4. Don't enclose your code in comments otherwise it will not be evaluated.

Instructions: (MUST READ)

- Understanding the problem statement is a part of that question. Do whatever you are asked and what you understand. NO QUERIES ARE ALLOWED as this lab is overly simple
- No compensation or makeup lab.
- Don't discuss with peers. Changing variable names/ changing for to while loop will not help you in hiding cheating attempt!
- Cheating cases will simply be dropped out and their future lab submissions will NOT be graded.
- You are not allowed to consult Internet. Plagiarism cases will be strictly dealt.
- You must submit the lab solution BEFORE 11:59 pm. Even a few minutes late submissions will not be considered. Make sure to do proper management of time/Internet connectivity/power failure or whatever issue is possible!

General instructions for all tasks: (marks will be deducted if the instructions are violated)

Note: All the programs should be implemented using class. You can take input in main() function and then call appropriate methods/ member functions of a designed class to set and get values. **YOU MUST CREATE A SEPARATE CPP AND HEADER FILE(S) FOR CLASS DECLARATION AND DEFINITION.**

- The attributes of class should be declared as **private** and member functions as **public**.
- All the member functions should be declared inside the class and defined outside the class.
- You should not initialize the attributes while declaring them in class. The values should be assigned using member functions only. E.g. you cannot declare like:

```
Class Person
{
    Private:
    int age=25;
```

}

- The values should be initialized using a constructor. There must be a constructor in your defined class.
- All inputs should be taken in *main()* and all the final results should also be reported/ displayed in the main function.
- All the logic should be implemented in class' member functions. Main() should only input and output relevant values by calling relevant functions of the class.

Task 1:

A polynomial is an expression consisting of variables and coefficients e.g. $x^2 - 4x + 7$. Implement a class of Polynomial with two data members: 1) dynamic array of coefficients and 2) degree. Coefficients are the constant values multiplied with variable in a polynomial. E.g. in $x^2 - 4x + 7$, coefficient of 'x' is 4.

If coef is an array of coefficients then:

coef[0] would hold all coefficients of x^0

coef[1] would hold all x^1

coef[n] = x^n ...

Degree is largest exponent of that variable. E.g. . in $x^2 - 4x + 7$, degree is 2.

Example:

For polynomial $P = 3x^4 + 4x^2 + 16$, if someone tries to access $P[4]$ then it should get 3.

Similarly if someone tries to access $P[3]$ it should get 0 as there is no term with degree 3

You have to do following tasks:

- In default constructor, initialize coefficient array with 0s.
- Implement setter method that takes three parameters, 1) coefficient value (int c, e.g.), 2) index of array at which this value (c) should be stored at, and 3) degree of whole polynomial.
- Implement following functions:
 - Additions of polynomials (**operator + and +=**)
 - Subtraction of polynomials (**operator - and -=**)
 - Greater than **operator >** (compared on the basis of degree - if the degrees are equal then compare on the basis of constants)
 - Smaller than **operator <**
 - Equal to **operator ==** (both degree and constants)
 - **Operator <= , operator >= , operator !=**

Implement the default, parameterized, and copy constructors, destructors and accessor methods as needed.

Main function should do the following:

1. Declare two polynomials and set their values ONLY by taking input from the user.

2. Declare third polynomial and use copy constructor to set it to polynomial 1's contents. Make sure you perform deep copy otherwise ZERO will be granted for this part!
3. Declare fourth polynomial, one by one apply the following functions, and display the resultant fourth object side by side.
 - a. +, -=, >, ==, != operator (you need to design all (as mentioned above) but only call these mentioned here in main to check)
4. Make sure to have a destructor that deallocates the allocated memory.

Task 2:







Part 1: Configuring and running a sample code (This part is not graded but obviously you need to perform this step to attempt the task given at the end. Therefore, no need to submit this configuration task details. You need to submit only the class that needs to be designed; instructions given at the end)

Configuring OpenCV with Visual C++

Step 1: Download and extract the pre-built library

Download the latest binary from opencv's Github repository (opencv-4.1.2-vc14_vc15.exe).

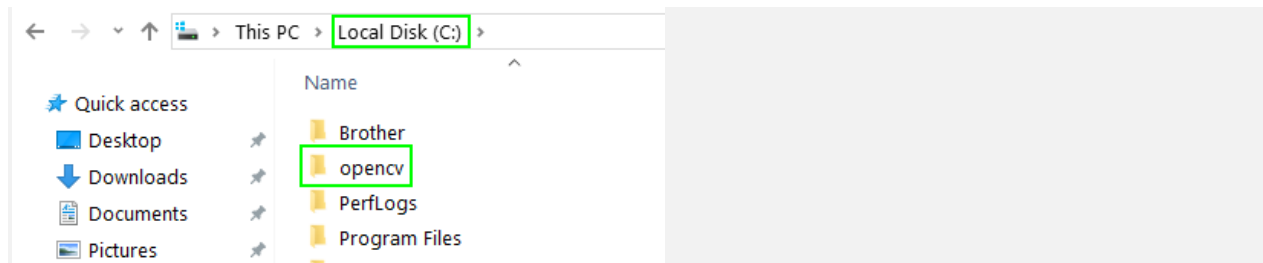
<https://github.com/opencv/opencv/releases/tag/4.1.2>.

 opencv-4.1.2-android-sdk.zip	218 MB
 opencv-4.1.2-docs.zip	80.1 MB
 opencv-4.1.2-ios-framework.zip	147 MB
 opencv-4.1.2-vc14_vc15.exe	203 MB
 Source code (zip)	
 Source code (tar.gz)	

Run the downloaded .exe file to extract the archive.

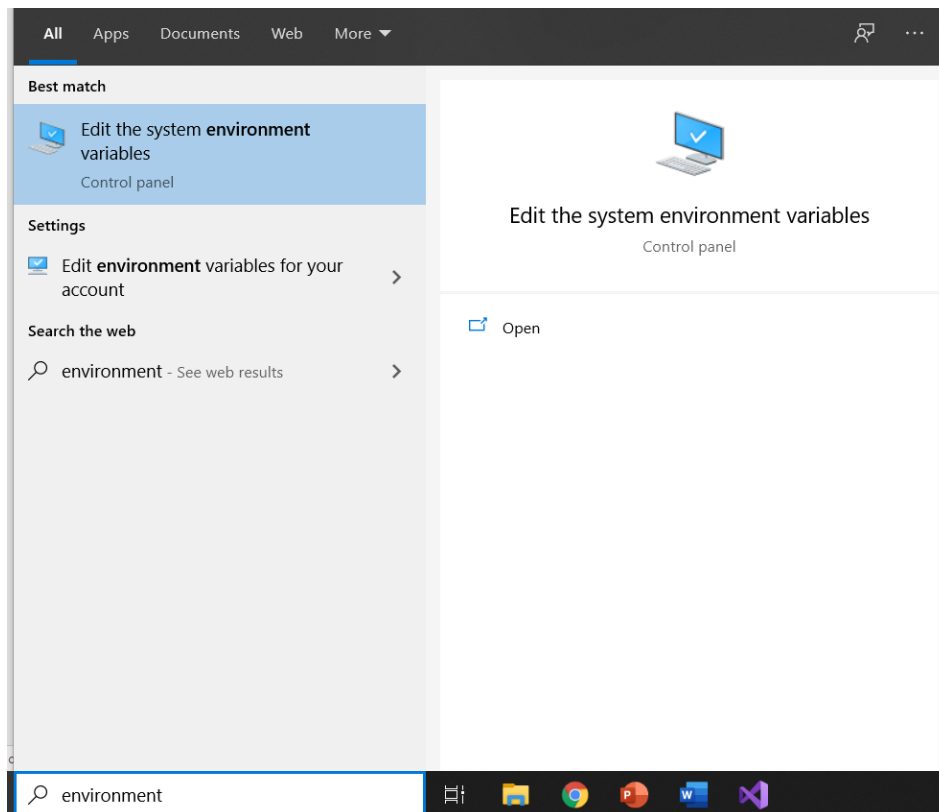
For the sake of uniformity, this tutorial will assume that you've extracted the contents to c:\

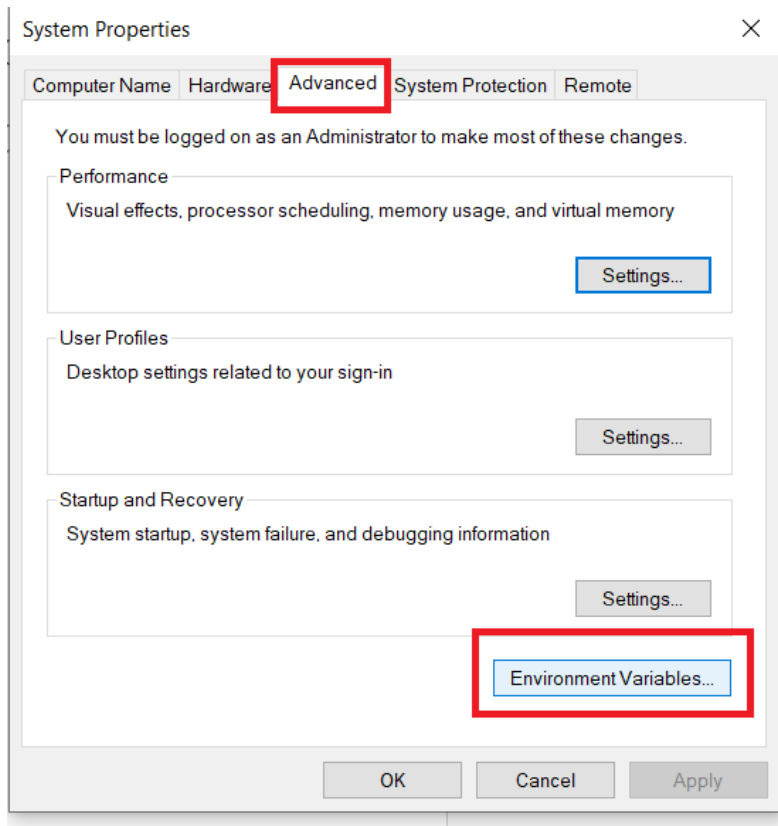
Extract to C:\



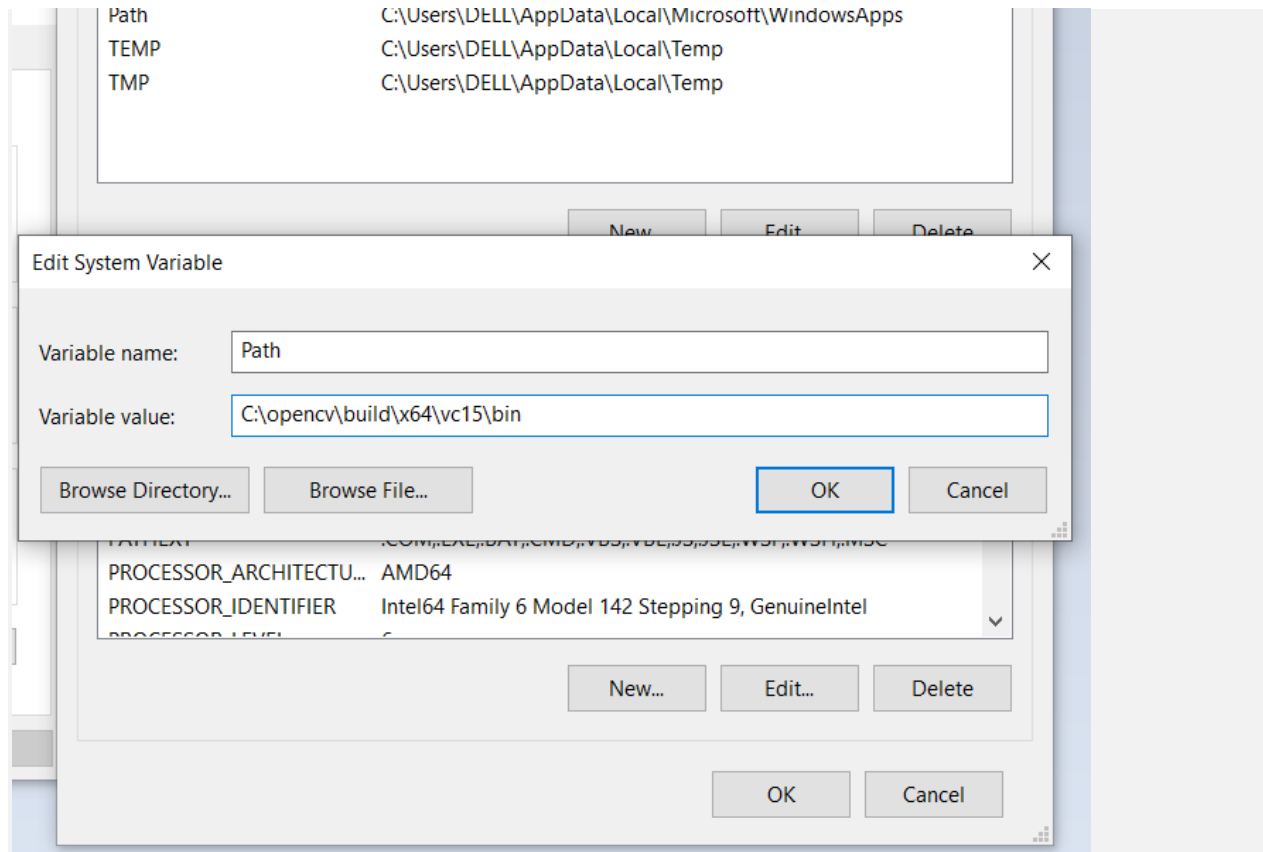
Step 2: Add to path

Open Environment variables dialog box





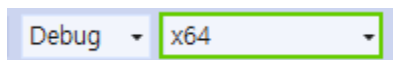
Add opencv's bin directory to path.



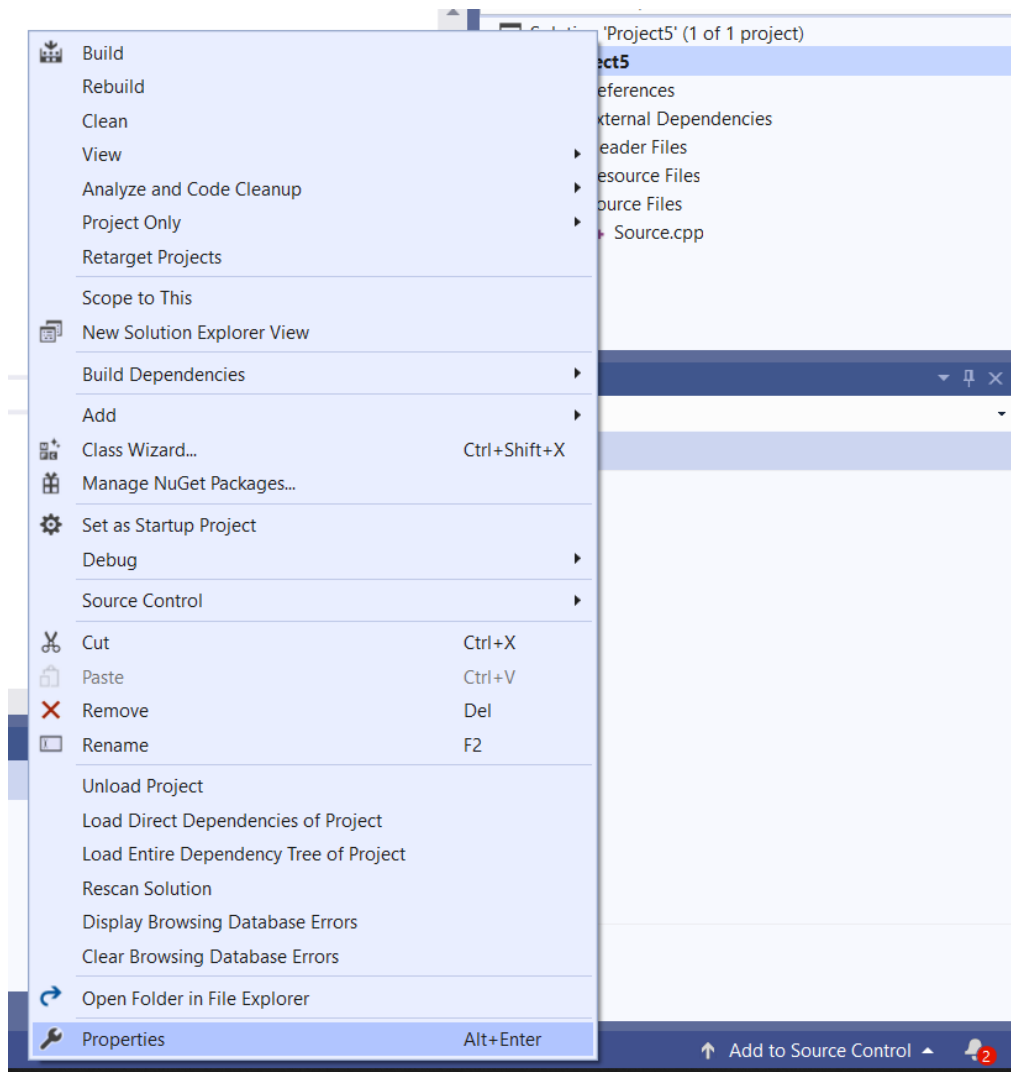
Step 3: Create a project in Visual Studio 2019

Step 4: Adding additional libraries and includes

1. Set platform target to x64

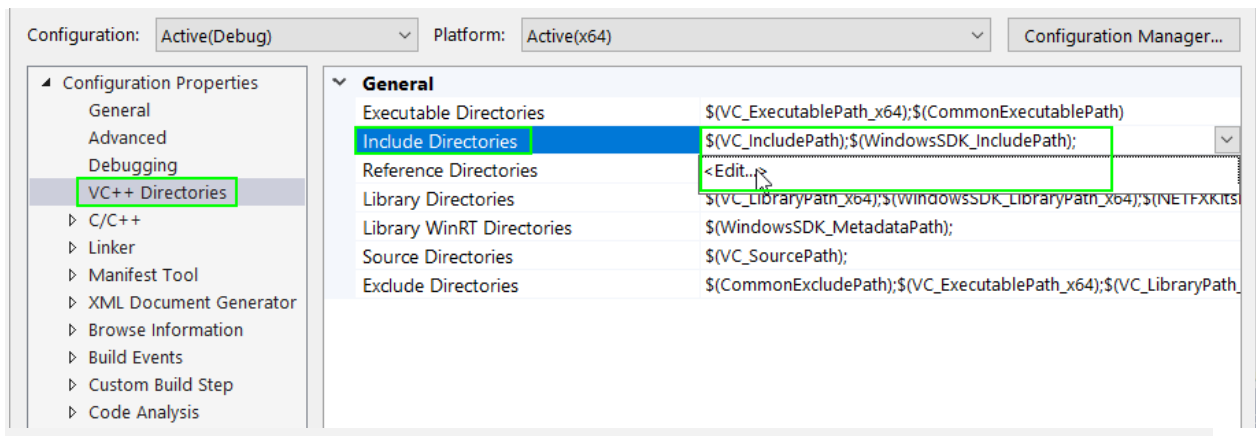


Now, go to Project → *YourProjectName* properties in menu.

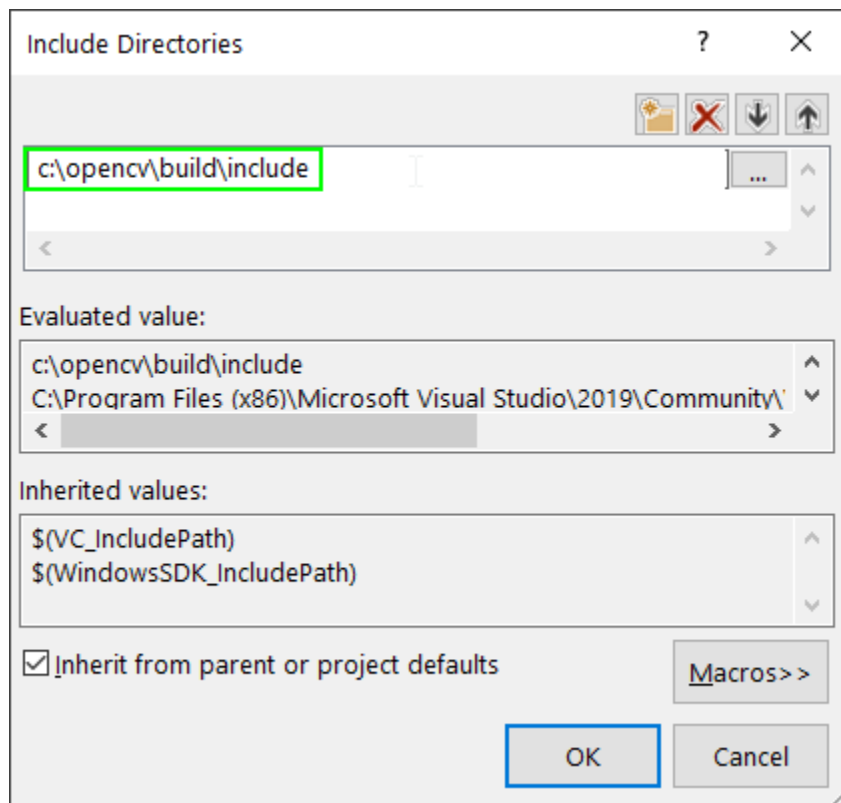


2. Add to Include Directories

3. go to VC++ Directories on the left and click on **Include Directories** row. Once you see the down arrow on the rightmost part of the row, click on the arrow, and select **<Edit...>**.

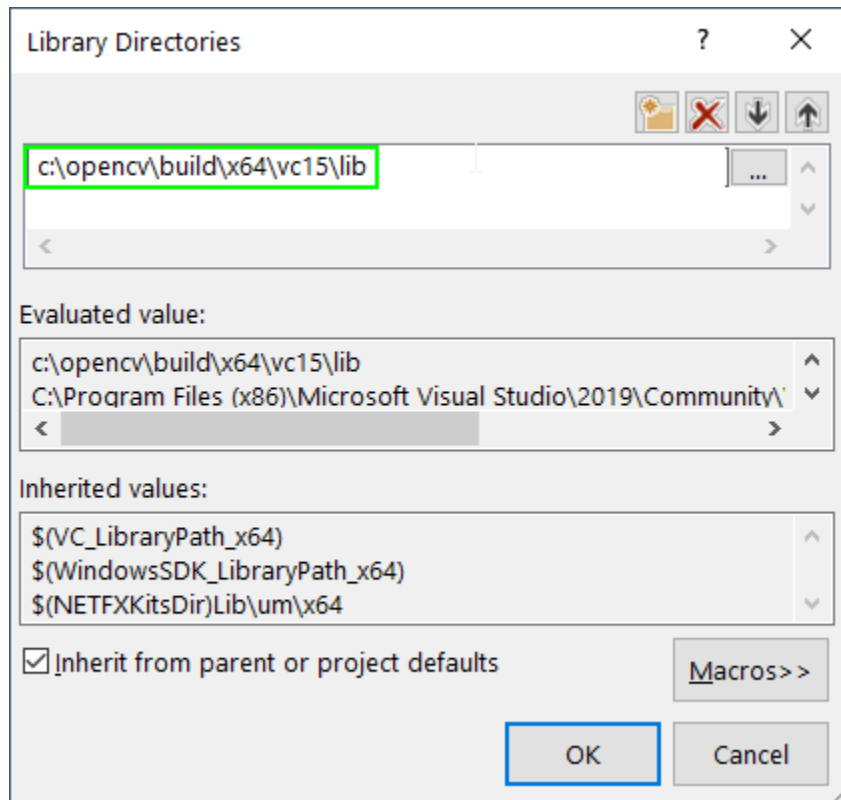
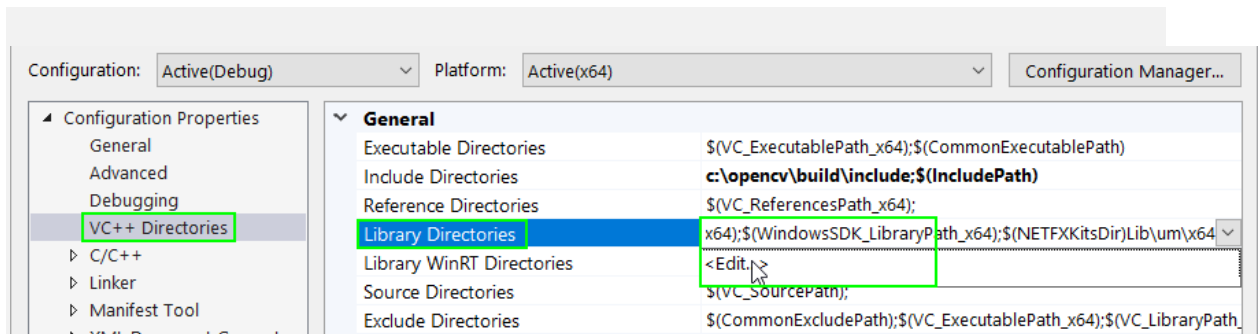


In Include Directories window, add c:\opencv\build\include



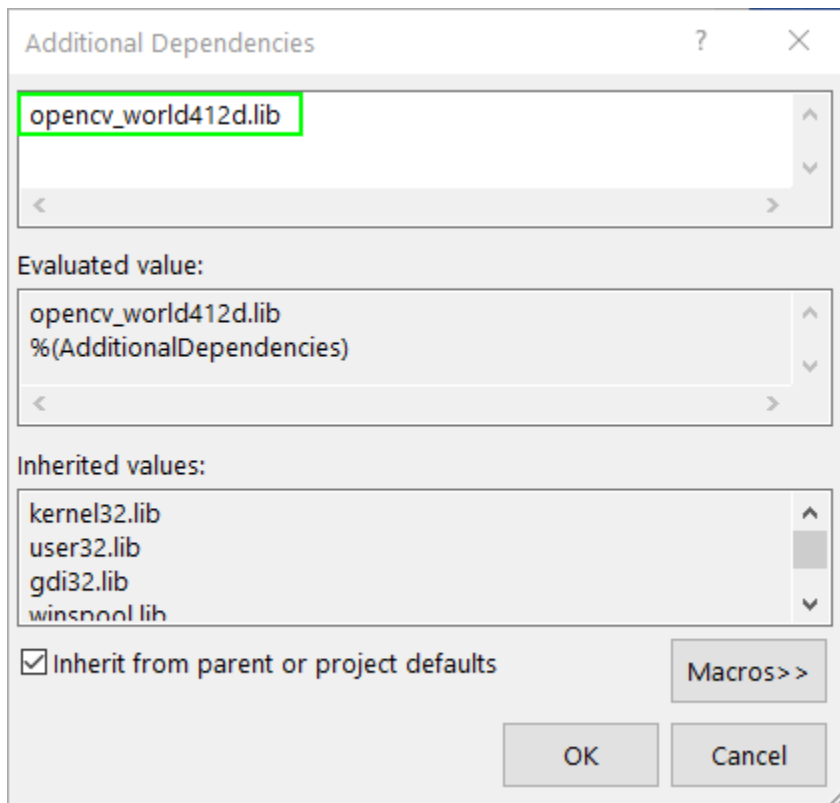
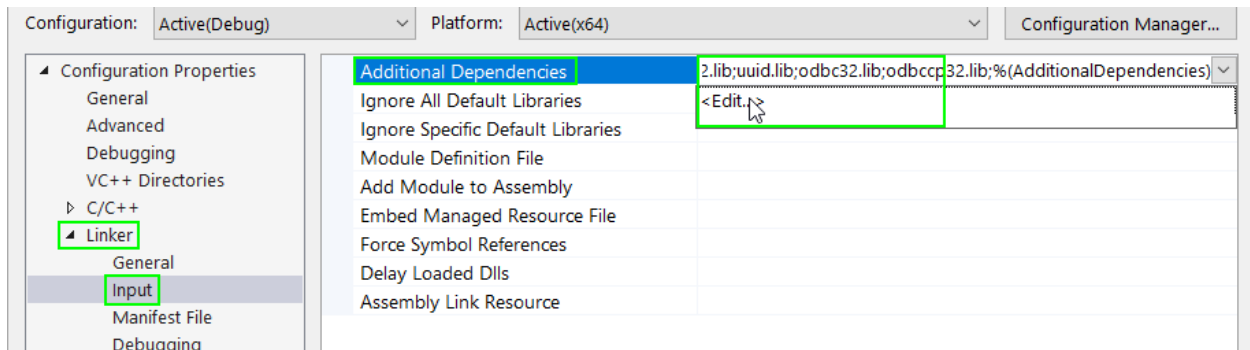
4. Add to Library Directories

Click OK. In the same tab, look for **Library Directories**. Again, click on the down arrow and select **<Edit...>**.



5. Add Additional Dependencies

Linker — Input — Additional Dependencies



Step 4: Run the following code that displays an image

Make sure to change the path to image inside imread function.

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <iostream>
```

```
using namespace cv;
using namespace std;
```

```

int main()
{
    // Read the image file
    Mat image = imread("D:/Fiverr_project/1.jpg");

    // Check for failure
    if (image.empty())
    {
        cout << "Could not open or find the image" << endl;
        cin.get(); //wait for any key press
        return -1;
    }

    String windowName = "OpenCV Test"; //Name of the window

    namedWindow(windowName); // Create a window

    imshow(windowName, image); // Show our image inside the created window.

    waitKey(0); // Wait for any keystroke in the window

    destroyWindow(windowName); //destroy the created window

    return 0;
}

```

TASK: Designing class and drawing shape

Design a class **Circle** with the following data members:

Centerx- x coordinate of central point

Century- y coordinate of central point

Diameter- Diameter of a circle

RedColorVal- value for red component used while coloring the circle

GreenColorVal- value for green component used while coloring the circle

BlueColorVal- value for blue component used while coloring the circle

Note: Range of color must be between 0 to 255. No need for validation checks, you can take this assumption while hardcoding values.

- Design a copy constructor that copies one object to another
- Design an overloaded + operator function that adds two circles and returns the resultant circle.
- Design a function **MoveUp** that moves a central point of circle up by 1 point.
- Design a function **MoveDown** that moves a central point of circle down by 1 point.
- Design a function **MoveLeft** that moves a central point of circle left by 1 point.
- Design a function **MoveRight** that moves a central point of circle right by 1 point.

Note: These four functions do not take any circle object in argument.

Main function should do the following:

1. Create four circles with hardcoded values. Each circle should have a unique color. Ensure this while hardcoding values.
2. Create a fifth circle, color it white, align it to point (0,0), and updates it with the sum of any two circles designed in step1. Use overloaded operator + for taking sum. Taking sum means just adding their diameters.
3. Move the circles as:
 - a. Move circle 1 up
 - b. Move circle 2 down
 - c. Move circle 3 left
 - d. Move circle 4 right
4. Find which of the four circles (in step 1) are concentric and label them accordingly.
5. Draw these four circles graphically. You can use the following reference code to achieve that. Don't confuse yourself with the **circle** written there. This circle is not an object of your designed class. It belongs to opencv for drawing a circle to the window. Make sure to differentiate your class' name with this.

You can draw various shapes like Circle, Rectangle, Line, Ellipse, Polyline, Convex, Polyline, Polyline on an image using the respective methods of the org.opencv.imgproc package.

You can draw a circle on an image using the method circle() of the imgproc class. Following is the syntax of this method –

circle(img, center, radius, color, thickness)

This method accepts the following parameters –

mat – A Mat object representing the image on which the circle is to be drawn.

point – A Point object representing the center of the circle.

radius – A variable of the type integer representing the radius of the circle.

scalar – A Scalar object representing the color of the circle. (BGR)

thickness – An integer representing the thickness of the circle; by default, the value of thickness is 1.

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <iostream>
```

```
using namespace cv;
using namespace std;
```

```
int main()
```

```

{
    //circle(img, center, radius, color, thickness)
    /* mat - A Mat object representing the image on which the circle is to be drawn.
point - A Point object representing the center of the circle.
radius - A variable of the type integer representing the radius of the circle.
scalar - A Scalar object representing the color of the circle. (BGR)
thickness - An integer representing the thickness of the circle;
by default, the value of thickness is 1.
*/
    Mat image = Mat::zeros(300, 600, CV_8UC3);
    //draw a circle
    circle(image, Point(350, 150), 100, Scalar(231, 123, 255), -100);
    //show the circle to window
    imshow("OOP Lab Task 5", image);
    waitKey(0);
    return 0;
}

```