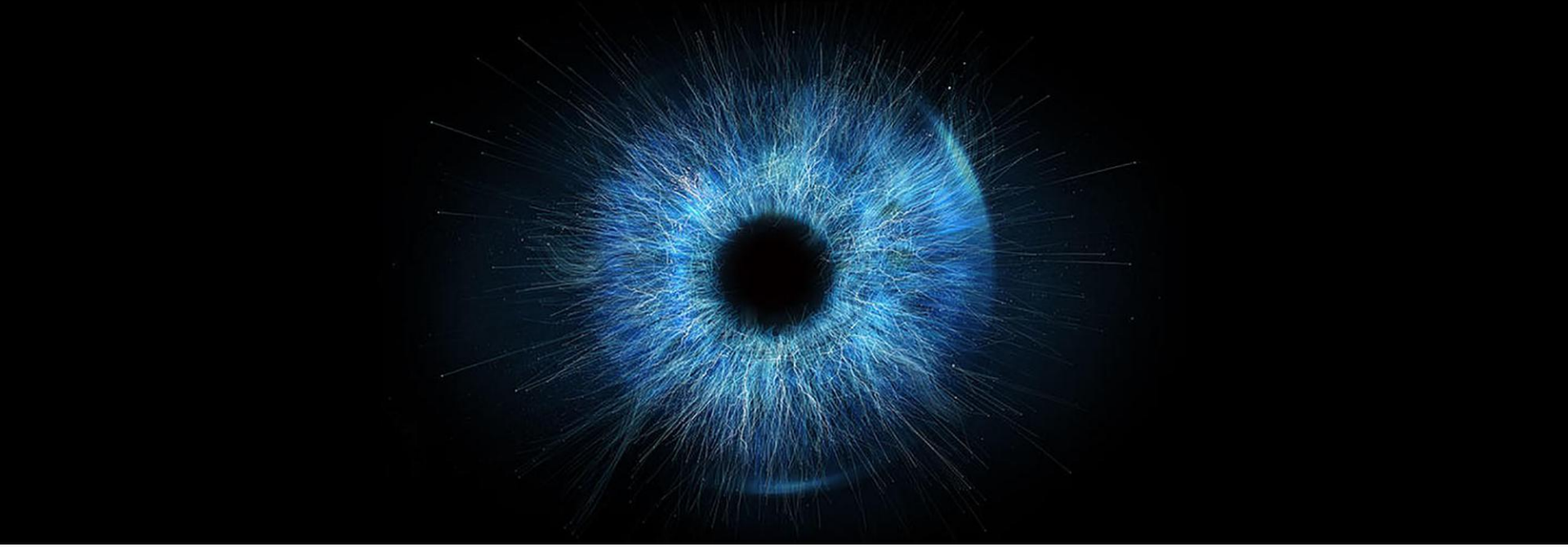


Fundamentals of machine vision algorithms

Exercise 1 – basics of c++

Doc.dr.sc. Filip Šuligoj

fsuligoj@fsb.hr

University of Zagreb

FSB Faculty of mechanical engineering and naval arhitecture

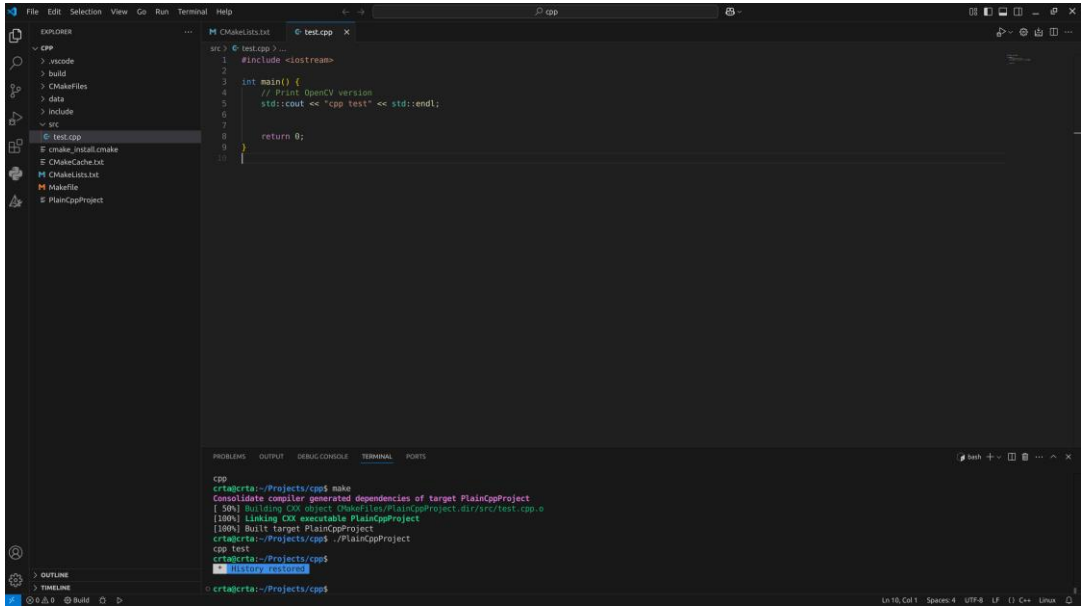
Exercises

C++ introduction

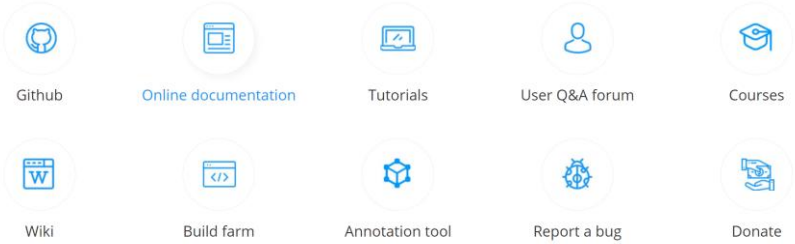
https://www.tutorialspoint.com/cplusplus/index.htm

Linux and VS code IDE + OpenCV

Linux Ubuntu + VS code



OpenCV https://opencv.org/



Practical tasks

Assignment Tasks

1.Project Setup:

- *Build the project using VS Code (refer to your CMakeLists.txt) and run main.cpp.*

2.Basic Integer Operations:

- *Declare global integer variables.*
- *Perform addition and output the result using cout.*

3.Function Implementation:

- *Create a function (outside of main()) that multiplies two global integers.*
- *Call this function from main() and display the product.*

4.Loop Constructs:

- ***For Loop:*** *Print numbers from 1 to 101.*
- ***Nested For Loops:*** *Use a loop within a loop to multiply the current indices and output the result.*
- ***While Loop:*** *Print numbers from 1 to 101, using the increment operator (++) in the loop condition.*

5.Conditional Statement:

- *Implement an if…else statement that evaluates a variable's value and prints different messages accordingly.*

```cpp
#include <iostream>
#include <cmath>      // For math functions: sqrt(), abs(), pow(), etc.
#include <cstring>    // For C-style string functions like strcpy()
#include <string>     // For C++ string operations
#include <vector>     // For using the vector container

using namespace std;

// Global variables and function declaration
int a = 10;
int b = 0;

// Function to multiply two integers (demonstrates function definition)
int multiplicationTest(int x, int y) {
    return x * y;
}

int main() {
    // 1. Integer arithmetic: add two ints and output the result.
    cout << "Initial value of a: " << a << endl;
    b = 5;
    int sum = a + b;
    cout << "Sum of a and b: " << sum << endl;

    // 2. Function call: multiply two integers using multiplicationTest.
    int product = multiplicationTest(a, b);
    cout << "Product of a and b: " << product << endl;

    // 3. Demonstrate cmath functions.
    double sqrtValue = sqrt(a);
    cout << "Square root of a: " << sqrtValue << endl;
    double absoluteValue = abs(-a);
    cout << "Absolute value of -a: " << absoluteValue << endl;
    double powerValue = pow(a, 2);
    cout << "a squared: " << powerValue << endl;

    // 4. For loop: print numbers from 1 to 101.
    cout << "\nFor loop output (1 to 101):" << endl;
    for (int i = 1; i <= 101; i++) {
        cout << i << " ";
    }
    cout << endl;

    // 5. Nested loops: multiply indices and print the result.
    cout << "\nNested loops multiplication (1-5 for simplicity):" << endl;
    for (int i = 1; i <= 5; i++) {
        for (int j = 1; j <= 5; j++) {
            cout << i << " * " << j << " = " << multiplicationTest(i, j) << "\t";
        }
        cout << endl;
    }

    return 0;
}
```

Practical tasks

Assignment Tasks

6.Math Functions:

- *Include <cmath> and demonstrate the use of abs(), floor(), sqrt(), and pow() with both integer and double types.*

7.String Handling:

- ***C-style Strings:***
  - *Include <cstring>, declare two char arrays, and copy one into the other using strcpy().*
- ***C++ Strings:***
  - *Include <string>, declare two std::string variables, concatenate them using the + operator, and output the result.*

8.Vector Operations:

- *Initialize a std::vector<int> with a few integers.*
- *Append an additional element to the vector using push_back().*
- *Print the vector's contents to confirm the new member has been added.*

9.Documentation:

- *Add inline comments throughout your code to explain each basic notion and operation.*

```cpp
    // 6. While loop: print numbers from 1 to 101.
    cout << "\nWhile loop output (1 to 101):" << endl;
    int counter = 1;
    while (counter <= 101) {
        cout << counter << " ";
        counter++;  // Increment the counter (same as counter++)
    }
    cout << endl;

    // 7. If-else condition: check the value of 'a' and print a message.
    cout << "\nIf-else condition:" << endl;
    if (a > 10) {
        cout << "a is greater than 10" << endl;
    } else {
        cout << "a is not greater than 10" << endl;
    }

    // 8. C-style string operation: copy one char array into another.
    char charName1[10] = "Alice";
    char charName2[10];
    strcpy(charName2, charName1);  // Copies contents of charName1 to charName2
    cout << "\nC-style string copy: " << charName2 << endl;

    // 9. C++ string concatenation: combine two strings.
    string str1 = "Hello, ";
    string str2 = "World!";
    string combined = str1 + str2;
    cout << "C++ string concatenation: " << combined << endl;

    // 10. Vector operation: initialize a vector and add a member at the end.
    vector<int> intVector = {1, 2, 3, 4, 5};  // Initialize vector with 5 integers.
    intVector.push_back(6);                   // Add element '6' at the end.
    cout << "\nVector contents after adding an element:" << endl;
    for (size_t i = 0; i < intVector.size(); i++) {
        cout << intVector[i] << " ";
    }
    cout << endl;

    return 0;
}
```

Independent challenge

Task Description:

1.Array Initialization:

- *Use a std::vector<int> populated with a predefined set of integers (20 random values).*

3.Binary Search Implementation:

- *Write a separate function that performs binary search on the array/vector. This function should:*
  - *Accept the array and a target integer as parameters.*
  - *Return the index of the target if found.*
  - *Return an indication (e.g., -1 or a custom message) if the target value is not present.*

4.User Interaction Loop:

- *In the main() function, implement a loop that allows the user to repeatedly search for a number in the sorted array.*
- *Provide a clear prompt and handle user inputs robustly, including invalid entries.*

5.Error Handling:

- *Ensure proper error checking for edge cases (e.g., empty array, invalid number format).*
- *Use conditionals to manage unexpected inputs or states.*

6.Program Structure and Documentation:

- *Organize your code into functions and maintain modularity.*
- *Include appropriate header files such as <iostream> and <vector>.*
- *Comment your code to explain logic and flow.*