Data Structures and Algorithms — Lab 1

Topic

Introduction to GitHub and the Implementation of Sorting and Searching Algorithms using Templates.

Objective

- Learn to create, compile, and execute C++ programs using Visual Studio.
- Understand the fundamentals of version control and collaboration using GitHub.
- Understand and implement the Selection Sort algorithm.
- Explore the concepts of Linear Search and Binary Search algorithms.
- Develop code using templates to make the sorting and searching algorithms reusable for different data types.

Outcomes

1. Create and execute C++ programs using Visual Studio (2013/2019/2022).
2. Understand the basics of version control by creating and managing repositories on GitHub.
3. Commit, track, and revert code changes on GitHub.
4. Implement Selection Sort using templates to organize arrays of various data types in ascending order.
5. Use Linear Search with templates to find elements in unsorted arrays of different data types.
6. Apply Binary Search using templates on sorted arrays to improve search efficiency for multiple data types.

Content

The following sections will be covered during this lab session.

1. **Visual Studio** will focus on launching and creating new C++ projects, writing, compiling, and executing C++ code, and troubleshooting common errors in C++ programs.
2. **GitHub** will cover creating accounts and private repositories, uploading code files, managing versions, making changes and committing updates through the web interface, viewing commit history, reverting to previous versions, and uploading Scratch projects for version control.
3. **Selection Sort** will cover implementing the algorithm by repeatedly finding the smallest element in the unsorted part of the array and swapping it with the current element. Students will write, compile, and execute the code, ensuring the program sorts the array correctly.
4. **Linear Search** will involve writing code to search for a specific value by iterating through each element in the array sequentially. Students will compile and execute the code, testing it with different inputs to ensure correctness.
5. **Binary Search** will guide students through implementing the algorithm on sorted arrays, using an iterative method to divide the search space by half at each step. Students will write, compile, and test the code with edge cases such as single-element arrays or missing values.

Tasks

Students are required to complete the following tasks in lab timings.

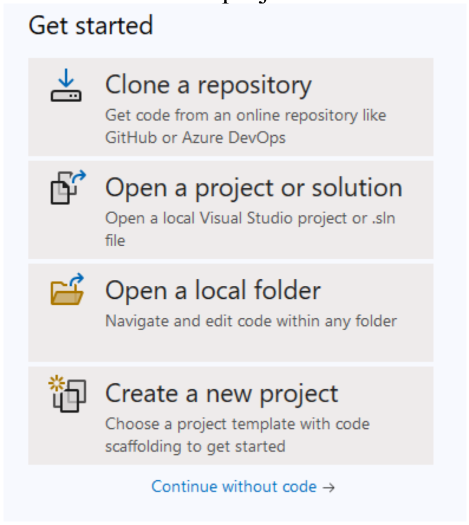
Task 0 – how to create a new project on Visual Studio 2022, 2019 or 2013

Step 1: Launch Visual Studio 2022

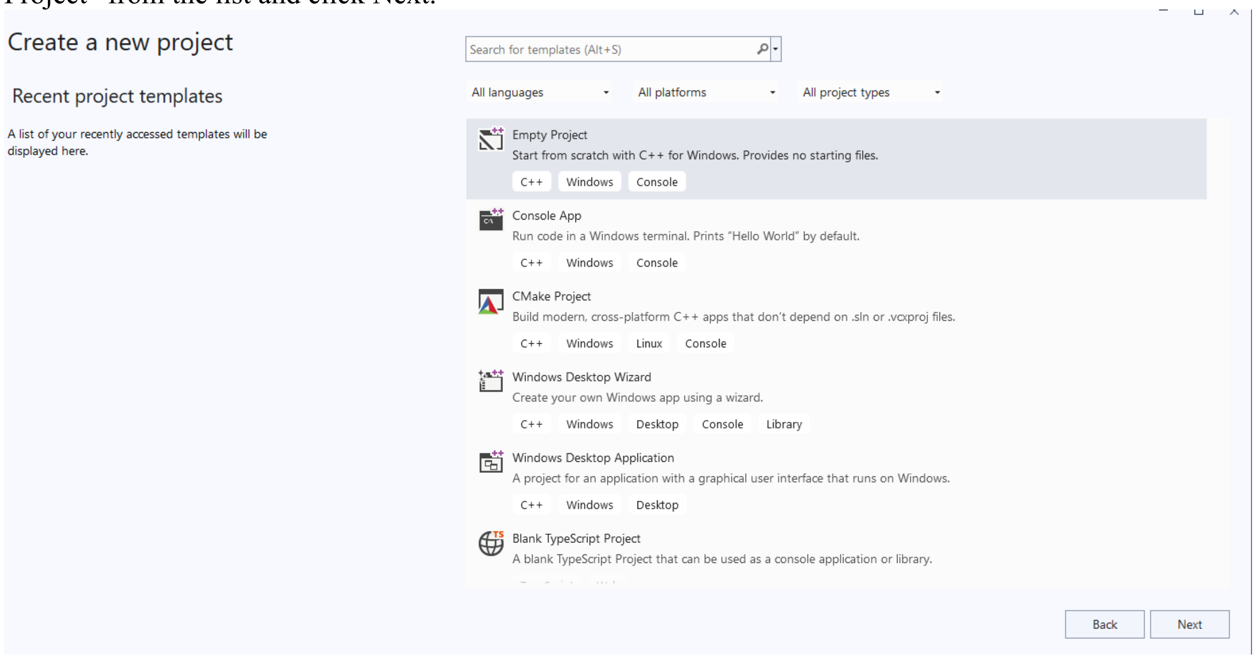
Open Microsoft Visual Studio 2022 from the Start menu or desktop shortcut. If Visual Studio 2022 is not available, similar steps may be followed for Visual Studio 2013 and 2019 (there is a slight change in steps; ask the teacher to help you follow the steps)

Step 2: Create a New Project

Once Visual Studio is open, click on "Create a new project" from the start window.


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 →

In the "Create a new project" dialog, use the search bar to search for "Empty Project". Select "Empty Project" from the list and click Next.



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

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

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

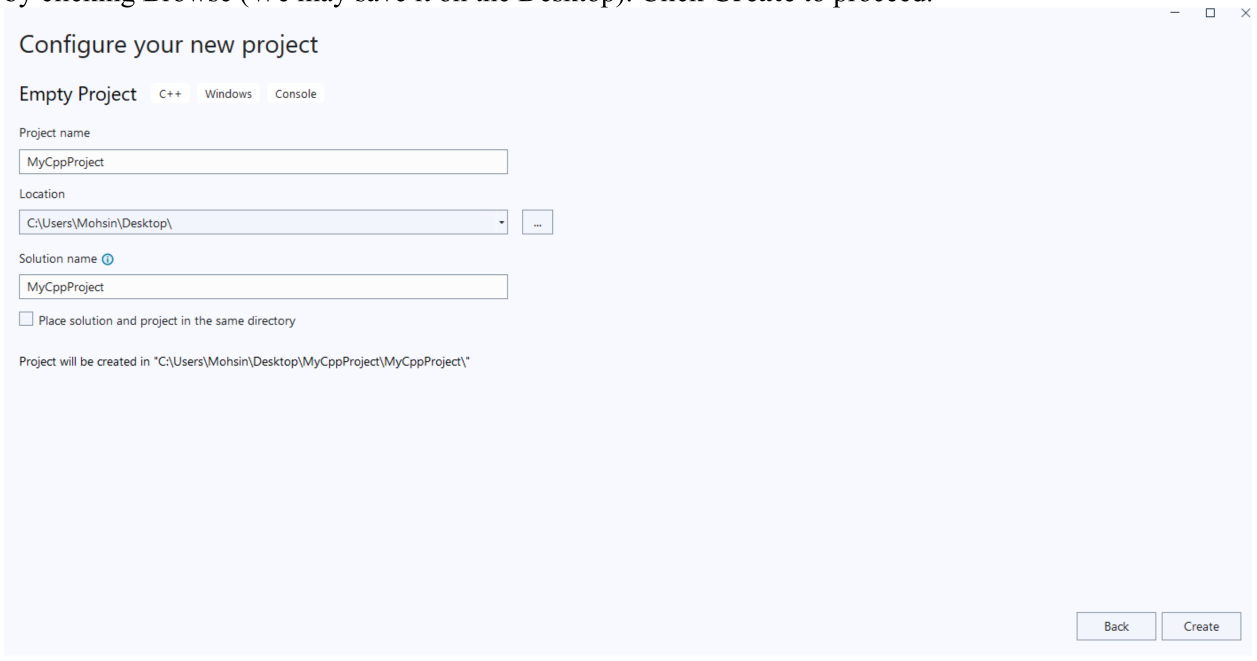
CMake Project
Build modern, cross-platform C++ apps that don't depend on .sln or .vcxproj files.
C++ Windows Linux Console

Windows Desktop Wizard
Create your own Windows app using a wizard.
C++ Windows Desktop Console Library

Windows Desktop Application
A project for an application with a graphical user interface that runs on Windows.
C++ Windows Desktop

Blank TypeScript Project
A blank TypeScript Project that can be used as a console application or library.

Step 3: Configure Your Project
Enter a name for your project (e.g., "MyCppProject"). Choose a folder where you want to save the project by clicking Browse (We may save it on the Desktop). Click **Create** to proceed.



Empty Project C++ Windows Console

Project name: MyCppProject

Location: C:\Users\Mohsin\Desktop\

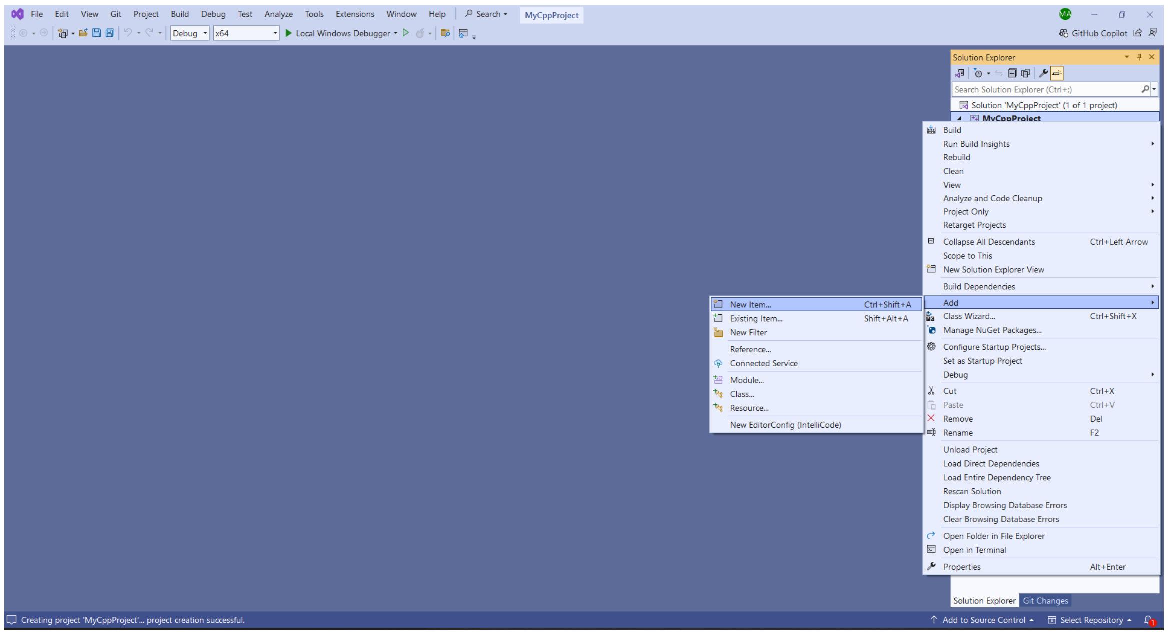
Solution name: MyCppProject

☐ Place solution and project in the same directory

Project will be created in "C:\Users\Mohsin\Desktop\MyCppProject\MyCppProject\"

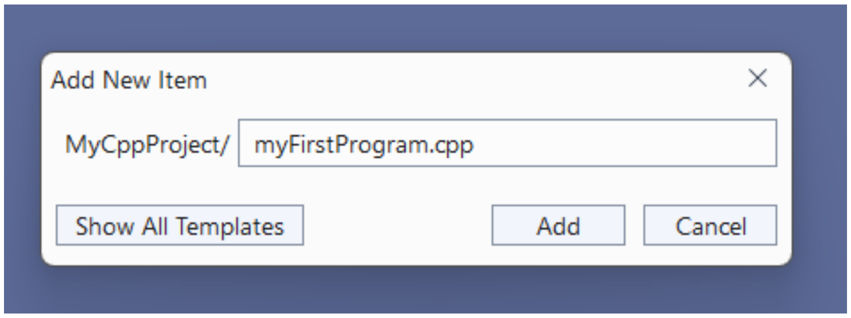
Step 4: Add a .cpp File to the Project

In Solution Explorer (which is usually on the right-hand side), right-click on the Source Files folder. From the context menu, select Add > New Item.



Creating project 'MyCppProject'... project creation successful.

In the “Add New Item” dialog box, select C++ File (.cpp). Enter a name of your choice for your C++ source file (e.g., “**myFirstProgram.cpp**”). Click Add.



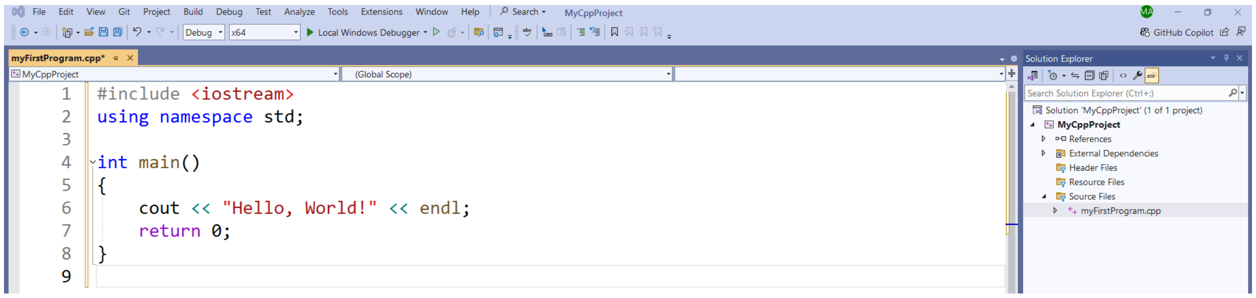
MyCppProject/ myFirstProgram.cpp

Step 5: Write Code in the .cpp File

Visual Studio will now open the **myFirstProgram.cpp** file in the editor. You can now write your C++ code. For example, you can add the following simple program:

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

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



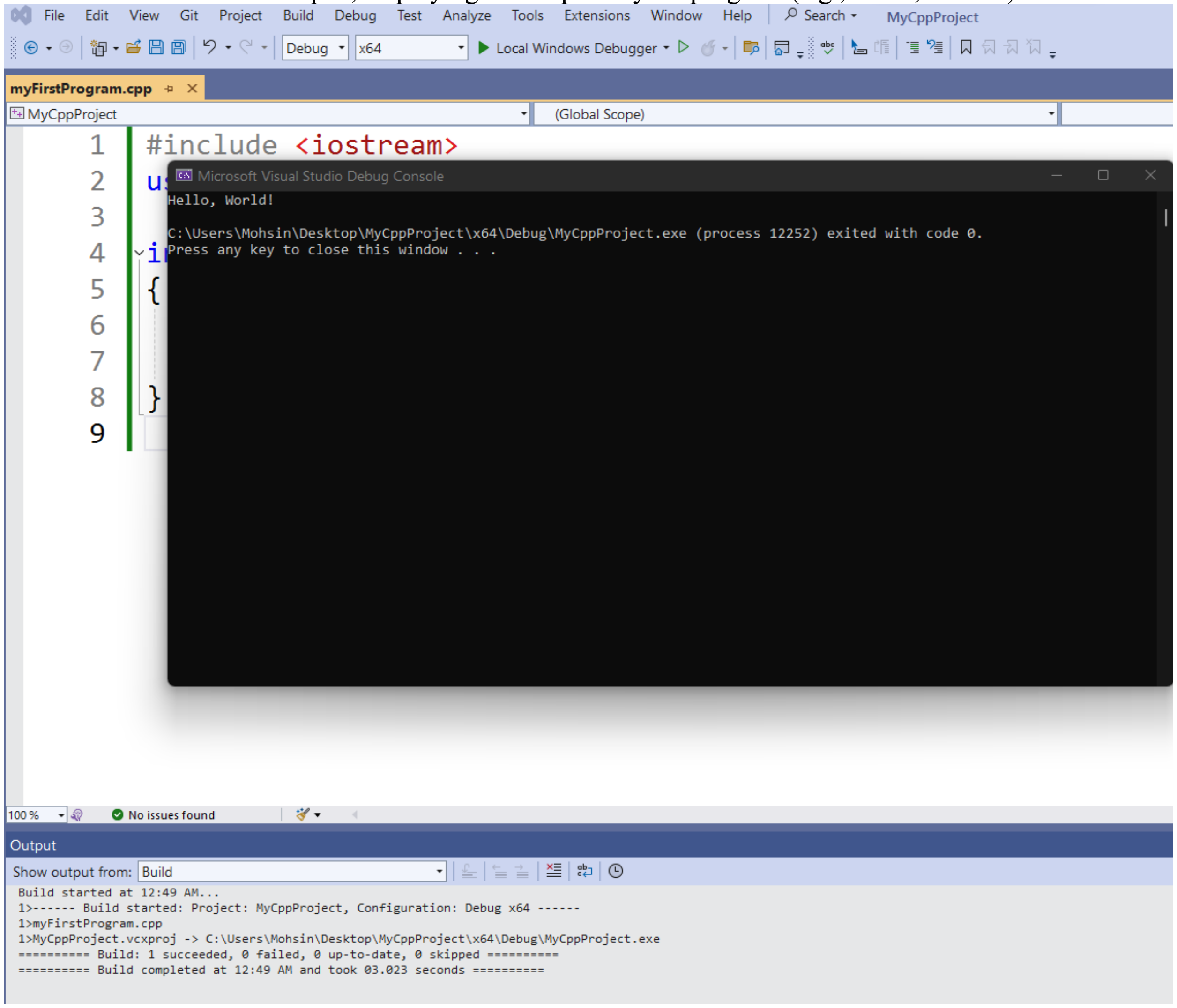
```cpp
#include <iostream>
using namespace std;

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

```

Step 6: Build and Run the Project
After writing the code, press Ctrl+F5.

Step 7: View Output
A console window should open, displaying the output of your program (e.g., Hello, World!).



```
Hello, World!

C:\Users\Mohsin\Desktop\MyCppProject\x64\Debug\MyCppProject.exe (process 12252) exited with code 0.
Press any key to close this window . . .
```

```
Build started at 12:49 AM...
1>------ Build started: Project: MyCppProject, Configuration: Debug x64 ------
1>myFirstProgram.cpp
1>MyCppProject.vcxproj -> C:\Users\Mohsin\Desktop\MyCppProject\x64\Debug\MyCppProject.exe
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
========== Build completed at 12:49 AM and took 03.023 seconds ==========
```

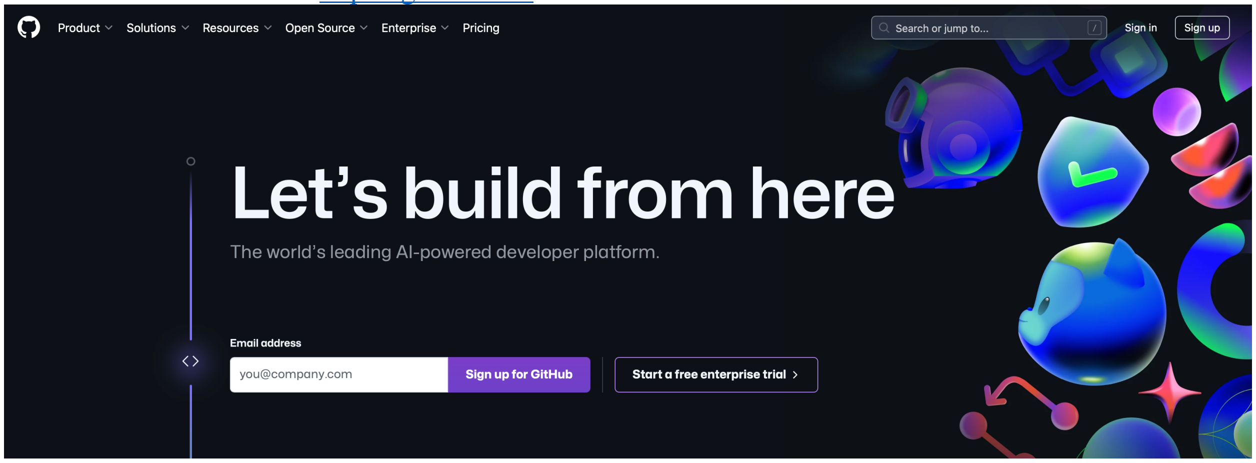
Task 1

GitHub is a web-based platform that allows you to store, manage, and collaborate on software projects. It uses **Git**, a version control system, to track changes to your code over time. GitHub enables you to easily share your code with others, collaborate on projects, and maintain different versions of your work. Whether you are working solo or with a team, GitHub helps ensure that you have a full history of your project, can revert to earlier versions if needed, and efficiently collaborate by merging changes from different contributors.

We use GitHub because it provides a centralized place to store your code, track progress, and collaborate with others. It also integrates with many tools and services, making it easier to work on coding projects from anywhere. In this guide, we will cover how to create an account on GitHub, create your first repository, upload a simple C++ file, make changes to the file and commit the new version, view different versions, and delete a repository when needed.

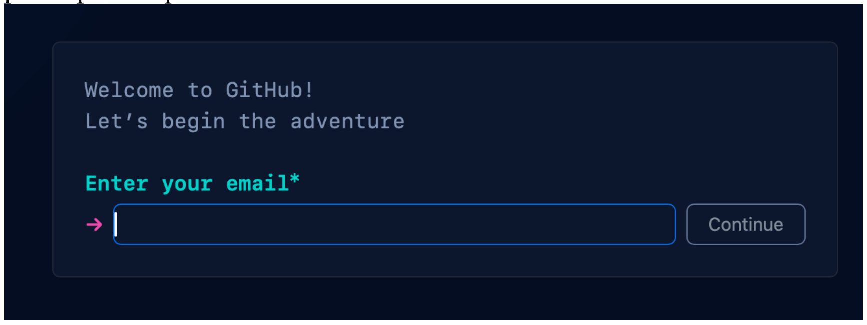
1. Creating an Account on GitHub

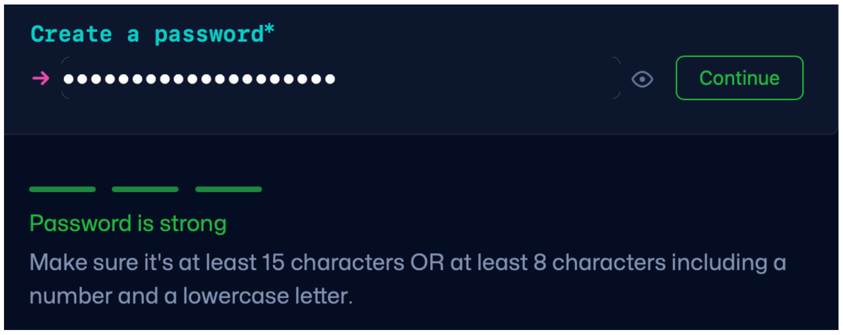
Go to GitHub's website at https://github.com.



Click on the "Sign up" button in the upper-right corner of the homepage.

Enter your details: choose a unique username, enter your email address, and create a strong password and pick up a unique username



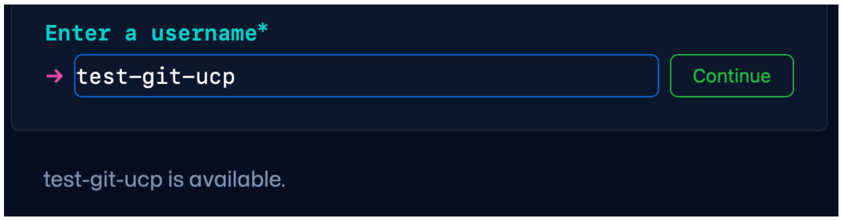


Create a password*

→ ••••••••••••••••••••

Password is strong

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.

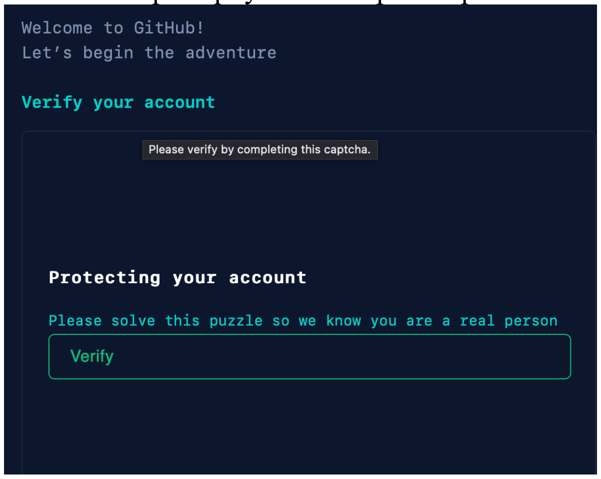


Enter a username*

→ test-git-ucp

test-git-ucp is available.

GitHub will prompt you to complete a puzzle to verify you're not a bot.



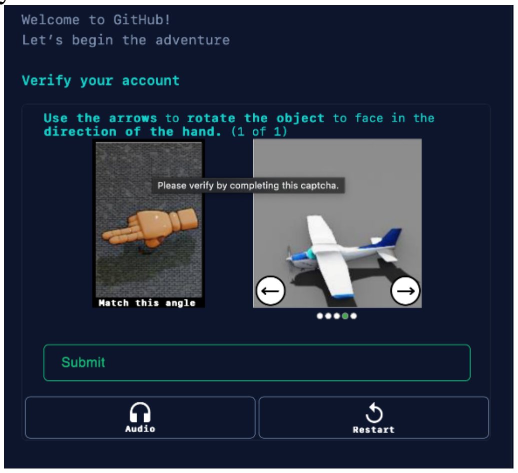
Welcome to GitHub!
Let's begin the adventure

Verify your account

Please verify by completing this captcha.

Protecting your account

Please solve this puzzle so we know you are a real person



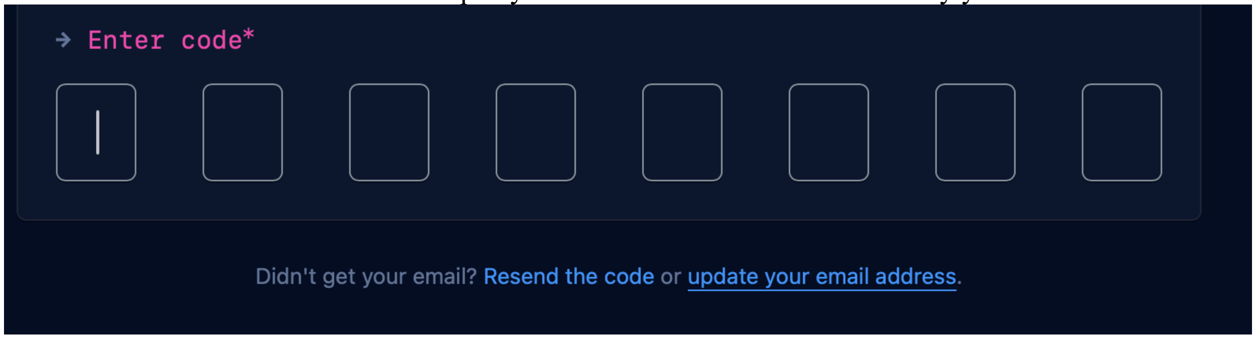
Welcome to GitHub!
Let's begin the adventure

Verify your account

Use the arrows to rotate the object to face in the direction of the hand. (1 of 1)

Please verify by completing this captcha.

You'll receive a confirmation email. Open your email and click the link to verify your account.



→ Enter code*

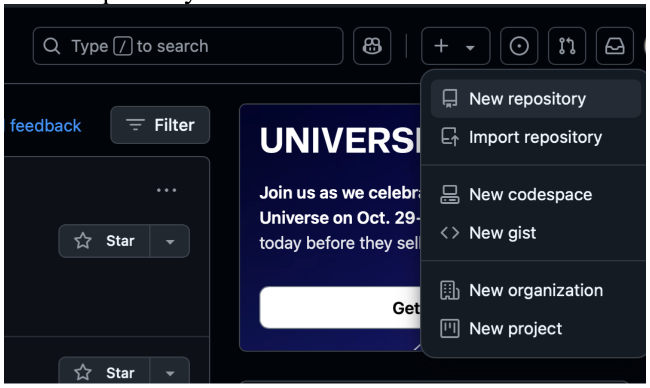
Didn't get your email? Resend the code or update your email address.

If asked, pick a plan: for most beginners, the free plan is sufficient. You'll then be taken to your GitHub dashboard, where you can customize your settings, but you can skip this step if preferred.

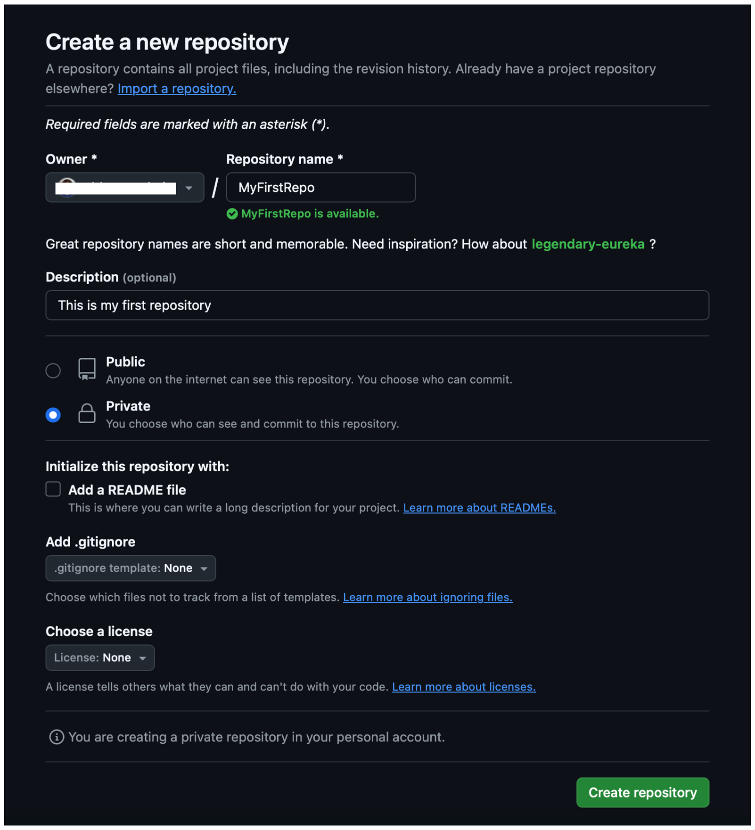
2. Creating Your First Repository and Uploading myFirstProgram.cpp

Step 1: Create a Private Repository

Go to the dashboard. After logging in, click the plus icon in the upper-right corner of the page and select "New repository."



Enter a name for your repository (for example, "MyFirstRepo"). Optionally, add a short description of the repository. Select the "Private" option to ensure the repository is private and only accessible to you (and invited collaborators). Optionally, you can check the option to "Add a README file," but leave other options unchecked for now. Click "Create repository" at the bottom of the page.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner * [redacted] / Repository name * MyFirstRepo

✓ MyFirstRepo is available.

Great repository names are short and memorable. Need inspiration? How about legendary-eureka ?

Description (optional)

This is my first repository

○ Public
Anyone on the internet can see this repository. You choose who can commit.

● Private
You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

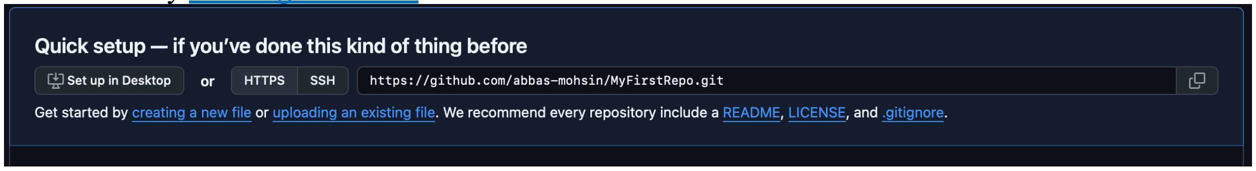
License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

ⓘ You are creating a private repository in your personal account.

Create repository

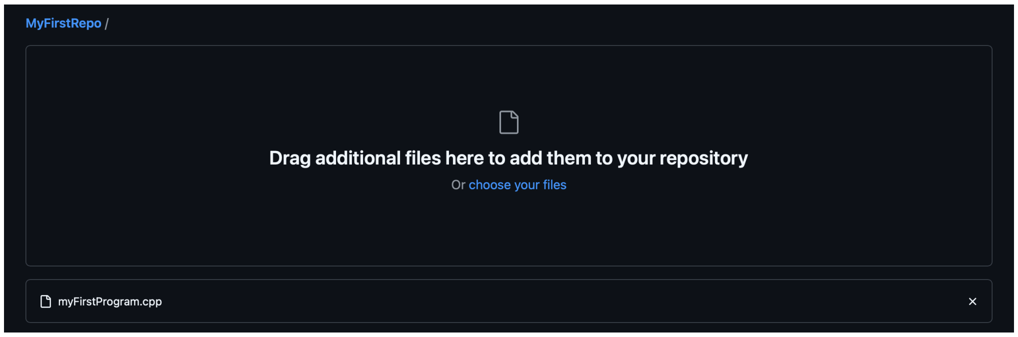
Step 2: Upload myFirstProgram.cpp
Once you're on your repository's main page, click "uploading an existing file" – you will find it right after: "Get started by creating a new file or "


Quick setup — if you've done this kind of thing before

`https://github.com/abbas-mohsin/MyFirstRepo.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

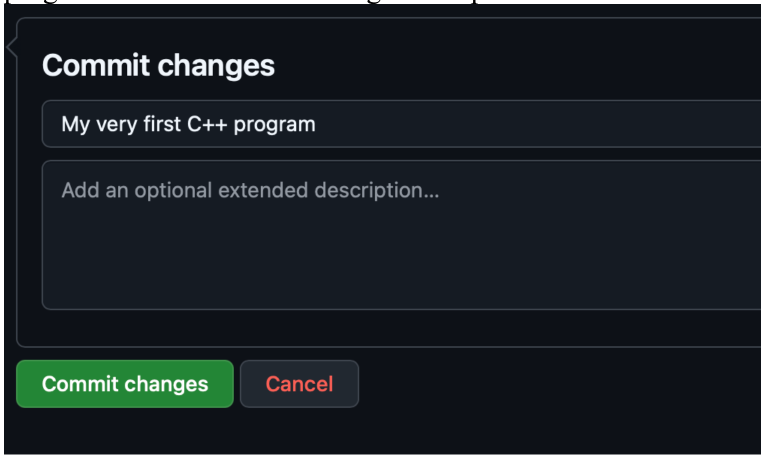
Drag your **myFirstProgram.cpp** file from your computer into the designated area or click "choose your files" to browse and select it.



Drag additional files here to add them to your repository

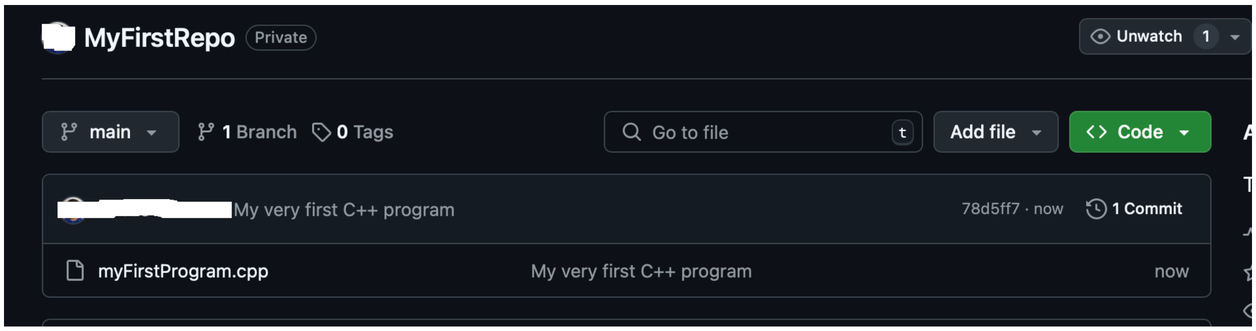
Or choose your files

In the "Commit changes" section, write a message describing the upload, for example, "My very first C++ program" Click "Commit changes" to upload the file.


Commit changes

My very first C++ program

Add an optional extended description...



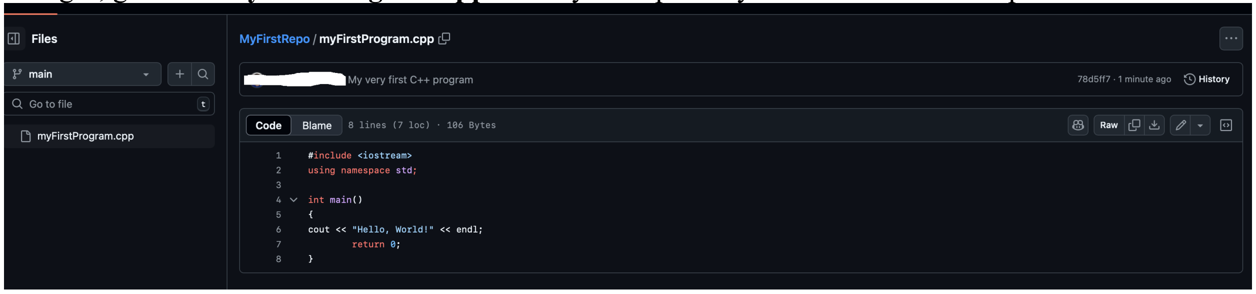
[redacted] My very first C++ program 78d5ff7 · now 1 Commit

myFirstProgram.cpp My very first C++ program now

Your **myFirstProgram.cpp** file is now uploaded to your private GitHub repository.

3. Making Changes and Committing Them

Once you've uploaded your code, you can easily make changes to it through GitHub's website. To make changes, go to the **myFirstProgram.cpp** file in your repository. Click on the file to open it.



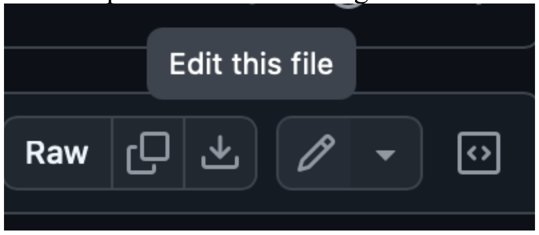
[redacted] My very first C++ program 78d5ff7 · 1 minute ago

8 lines (7 loc) · 106 Bytes

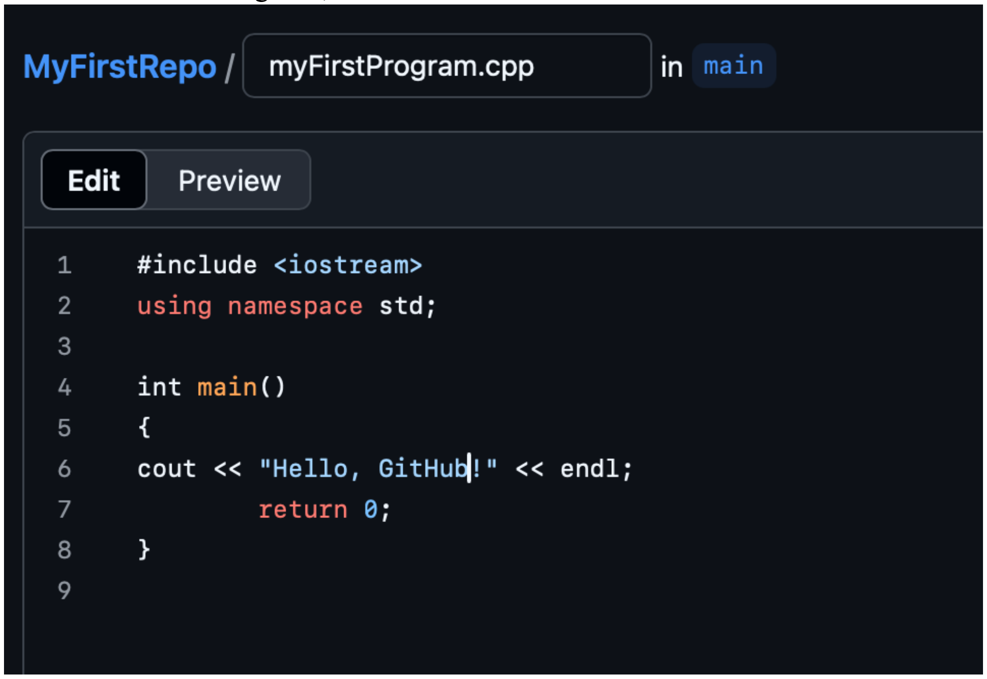
```
#include <iostream>
using namespace std;

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

At the top-right corner of the file, you'll see a pencil icon that says **"Edit this file"**. Click on this pencil icon to open the file for editing.



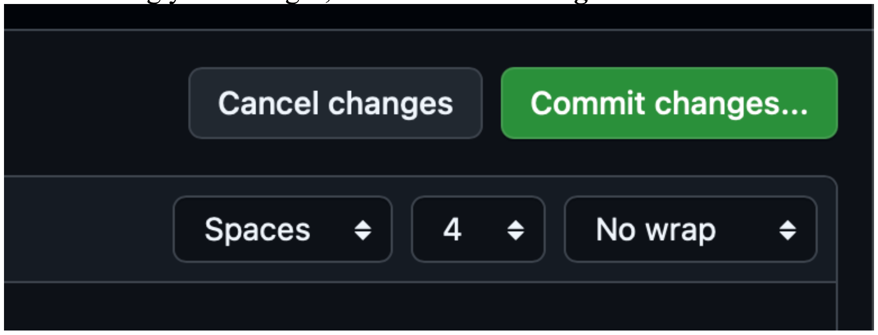
You can now make changes to your code. For example, change the output of your program from `Hello, World!` to something else, like `Hello, GitHub!`



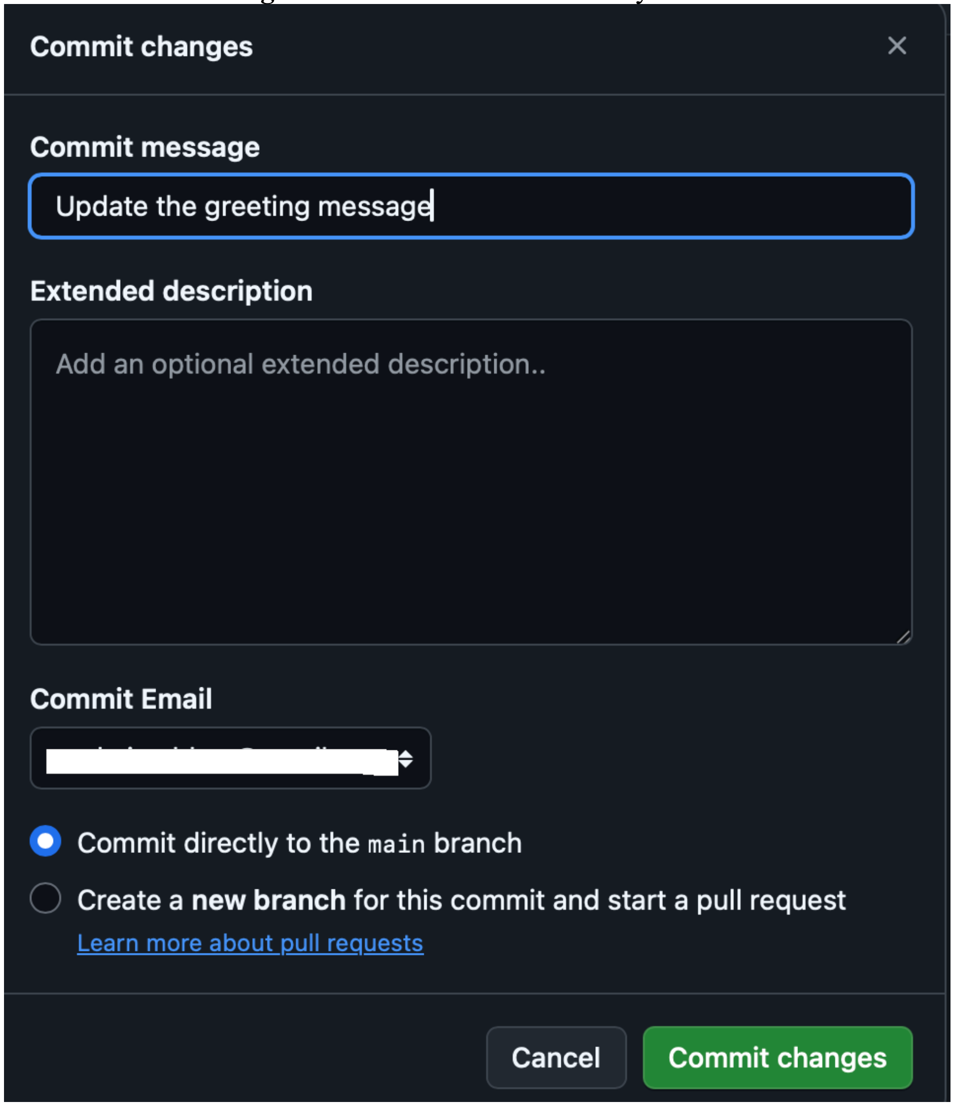
```
#include <iostream>
using namespace std;

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

After making your changes, click **Commit changes**.



In the **Commit message** field, describe your changes (for example, "Update the greeting message").
You can leave the default option **"Commit directly to the main branch"** selected.
Click **"Commit changes"** to save the new version of your file.


Commit changes

Commit message: Update the greeting message

Extended description: Add an optional extended description..

Commit Email: [redacted]

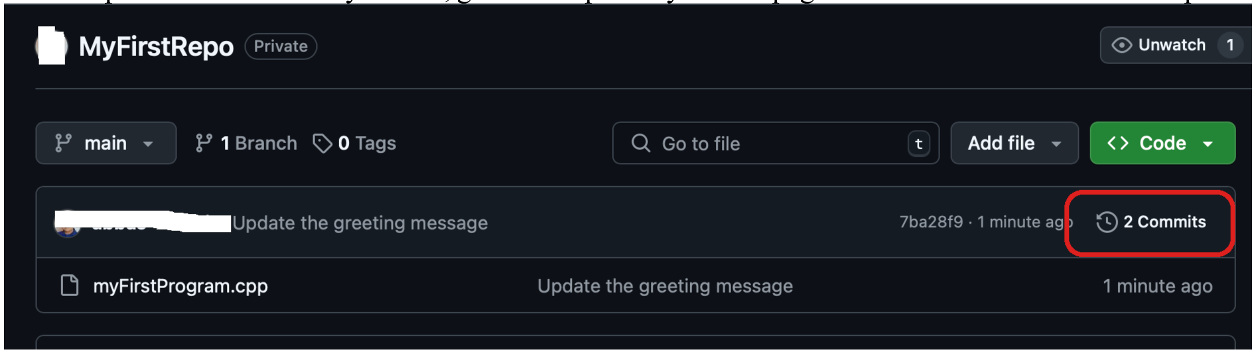
[✓ Commit directly to the `main` branch]

Create a **new branch** for this commit and start a pull request

4. Viewing Different Versions of Your File

Both the original and the updated versions of your file will be saved as part of your repository's history. GitHub tracks these changes as different "versions" of your file, allowing you to revert back to an older version if needed.

To view previous versions of your file, go to the repository's main page and click on the **"Commits"** option.

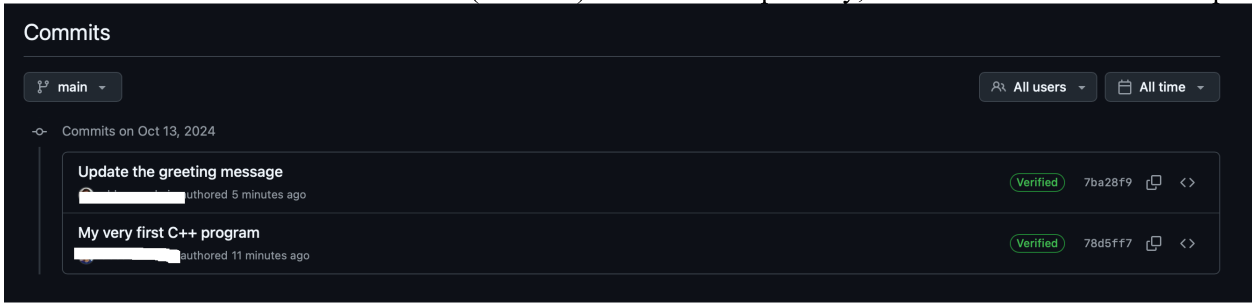


MyFirstRepo Private

[redacted] Update the greeting message 7ba28f9 · 1 minute ago 2 Commits

myFirstProgram.cpp Update the greeting message 1 minute ago

This will show a list of all the commits (versions) made to the repository, with the most recent at the top.



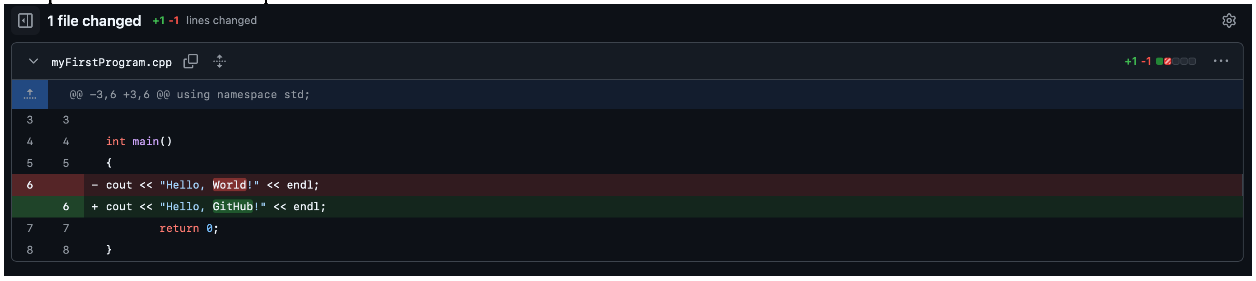
Commits

Commits on Oct 13, 2024

Update the greeting message
[redacted] uthored 5 minutes ago — Verified 7ba28f9

My very first C++ program
[redacted] authored 11 minutes ago — Verified 78d5ff7

Click on any of the commits to see the changes made in that version. You can view the differences between the previous and the updated file.



1 file changed +1 -1 lines changed

myFirstProgram.cpp +1 -1

```
@@ -3,6 +3,6 @@ using namespace std;

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

This way, you can always go back to an earlier version of the file if needed, ensuring you never lose important work.

For tasks 2-4, create a private repository in your account: **Lab-1-DSA.** Upload codes with the names **Task_2**, **Task_3** and **Task_4** on it.

Task 2

Selection sort is a sorting algorithm in which we repeatedly find the next largest (or smallest) element in the array and move it to its final position in the sorted array. Assume that we wish to sort the array in increasing (ascending) order, i.e., the smallest element at the beginning of the array and the largest element at the end. We begin by selecting the smallest element and moving it to the lowest index position. We can do this by swapping the element at the lowest index and the smallest element. We then reduce the effective size of the array by one element and repeat the process on the greater (sub)array. The process stops when the effective size of the array becomes 1 (an array of 1 element is already sorted). The pseudocode of the task is given below:

```
Input: An unsorted array, A of N elements
Output: Sorted array, A

For I = 0:N-1
    SmallSub = I
    For J = I+1:N-1
       If A[J] < A[SmallSub]
          SmallSub = J
       End-If
    End-For
    Swap ( A[I], A[SmallSub] )
End-For
```

Implement the code for selection sort using templates. Test your code using the main function below and upload it to your repository: **Lab-1-DSA.** The name of the code on your repository should be **Task_2.cpp**

```
int main() {
    // Test with an integer array of size 5
    int intArray[5] = {64, 25, 12, 22, 11};
    cout << "Original integer array: ";
    printArray(intArray);
    selectionSort(intArray);
    cout << "Sorted integer array: ";
    printArray(intArray);

    // Test with a string array of size 4
    string stringArray[4] = {"apple", "orange", "banana", "grape"};
    cout << "\nOriginal string array: ";
    printArray(stringArray);
    selectionSort(stringArray);
    cout << "Sorted string array: ";
    printArray(stringArray);

    return 0;
}
```

Task 3

In computer science, linear search or sequential search is a method for finding a particular value in a list that consists of checking every one of its elements, one at a time and in sequence, until the desired one is found. The pseudocode of the task is given below:

```
For I = 0:N-1
   If ( A[I] == Value )
     Return [I];
   End-if
End-For
return -1;
```

Implement the code for linear sort using templates. Test your code using the main function below and upload it to your repository: **Lab-1-DSA.** The name of the code on your repository should be **Task_3.cpp**

```
int main() {
    // Test with an integer array of size 5
    int intArray[5] = {64, 25, 12, 22, 11};
    int intKey = 12;
    int intIndex = linearSearch(intArray, intKey);
    printSearchResult(intIndex, intKey);

    // Test with a float array of size 4
    float floatArray[4] = {3.14, 2.71, 1.62, 0.57};
    float floatKey = 1.62;
    int floatIndex = linearSearch(floatArray, floatKey);
    printSearchResult(floatIndex, floatKey);

    // Test with a string array of size 4
    string stringArray[4] = {"apple", "orange", "banana", "grape"};
    string stringKey = "banana";
    int stringIndex = linearSearch(stringArray, stringKey);
    printSearchResult(stringIndex, stringKey);

    return 0;
}
```

Task 4

Binary search is another searching algorithm used to search a specific value (or index of value) from the *sorted array*. In binary search, we first compare the *value to be searched* with the item in the middle position of the array. If there's a match, we can return immediately. If the key is less than the middle key, then the item sought must lie in the lower half of the array; if it's greater than the item sought must lie in the upper half of the array. So we repeat the procedure on the lower (or upper) half of the array. The pseudocode of the task is given below:

```
Input: A Sorted array, A of N elements and value to be searched
Output: Index of searched element or -1 if not found

low = 0, high = N-1;
While (low <= high)
   mid = (low + high)/2;
   If ( A[mid] == Value )
     Return mid;
   Else-if ( A[mid] < Value )
     low = mid + 1;
   Else
     high = mid – 1;
   End-if
End-While
return -1;
```

Implement the code for binary search using templates. Test your code using the main function below and upload it to your repository: **Lab-1-DSA.** The name of the code on your repository should be **Task_4.cpp**

```
int main() {
    // Test with an integer array (sorted) of size 5
    int intArray[5] = {11, 12, 22, 25, 64};
    int intKey = 22;
    int intIndex = binarySearch(intArray, intKey);
    printSearchResult(intIndex, intKey);

    // Test with a float array (sorted) of size 4
    float floatArray[4] = {0.57, 1.62, 2.71, 3.14};
    float floatKey = 2.71;
    int floatIndex = binarySearch(floatArray, floatKey);
    printSearchResult(floatIndex, floatKey);

    // Test with a string array (sorted) of size 4
    string stringArray[4] = {"apple", "banana", "grape", "orange"};
    string stringKey = "grape";
    int stringIndex = binarySearch(stringArray, stringKey);
    printSearchResult(stringIndex, stringKey);

    return 0;
}
```