VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELGAUM-590014

An Internship Report

"Student Record System – C++ Mini Project"

Submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Engineering in Computer Science and Engineering

Submitted by:

Sahil Showkat – 1DT21CS126

Mohin Khan – 1DT21CS089

Preyuhsi Abrol – 1DT21CS116

DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT

Udayapura, Kanakapura Road, Bangalore-560082

(Affiliated to Visvesvaraya Technological University, Belagavi, Approved by AICTE, New Delhi)

Department of Computer Science and Engineering

Accredited by NBA, NAAC A+, New Delhi

2022-2023

DAYANANDA SAGAR ACADEMY OF TECHNOLOGY & MANAGEMENT

Opp. Art of Living, Udayapura, Kanakapura Road, Bangalore- 560082

Affiliated to Visvesvaraya Technological University, Belagavi and Approved by AICTE, New Delhi

CSE, ISE, ECE, EEE, ME, CE Branches Accredited by NBA, New Delhi

NAAC Accredited with A+ Grade

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

Certified that the Internship work entitled “**Student Record System”** carried out by **Sahil Showkat** bearing USN: **1DT21CS126**, **Mohin Khan** bearing USN: **1DT21CS089** & **Preyushi Abrol** bearing USN: **(1DT21CS116)** are bonafide students of **Dayananda Sagar Academy of Technology and Management** in partial fulfilment for the award of Bachelor of Engineering in **Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022-23. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The Internship report has been approved as it satisfies the academic requirements in respect of Internship work prescribed for the said Degree.

Internship coordinator   Signature of HOD

Roles & Responsibilities

Name of the member	Role	Responsibilities
Sahil Showkat	Team Lead	Coding, Logic Building, Rep–ort Work
Preyushi Abrol	Member	Research & Report
Mohin Khan	Member	Presentation & Research

Contents

1.1 Introduction to object-oriented programming features

Object-Oriented Programming (OOP) is a programming model that uses classes and objects. It's utilized to break down a software program into reusable code blueprints (called classes) that you may use to build specific instances of things. Object-oriented programming languages include JavaScript, C++, Java, and Python, to name a few.

Top Features of OOP

Inheritance

In layman's terms, the attributes that you inherit from your parents are a simple illustration of inheritance. Classes may inherit characteristics from other classes thanks to inheritance. Parent classes, in other words, extend properties and behaviors to child classes. Reusability is aided via inheritance. Prototyping is another name for inheritance in JavaScript. A prototype object serves as a base from which another object may derive its features and actions. Thus, you may use multiple prototype object templates to form a prototype chain. Inheritance is passed down from one generation to the next. parent

Consider the application Polygon, which represents several Shapes. We're expected to make two distinct sorts of polygons: a Rectangle and a Triangle.

Encapsulation

Encapsulation is the process of enclosing all critical information inside an object and only revealing a subset of it to the outside world. For example, code inside the class template defines attributes and behaviors.

The data and methods are then enclosed in the object when it is created from the class. Inside a class, encapsulation conceals the underlying software code implementation and the internal data of the objects. Encapsulation necessitates designating certain fields as private while others are made public.

Methods and attributes only available from other methods in the same class make up the private/internal interface.

Methods and attributes that are available from outside the class are known as the public / external interface.

Encapsulation Demonstration in Real-Time

One of the most practical examples of encapsulation is a school bag. Our books, pencils, and other items may be kept in our school bag.

The following are some of the advantages of encapsulation:

Data Hiding: In this case, the user will be unaware of the class’s internal implementation. Even the user will have no idea how the class stores data in variables. He or she will only be aware that the values are sent to a setter method and that variables are initialised with that value.

Increased Flexibility: Depending on our needs, we may make the variables of the class read-only or write-only. If you want to make the variables read-only, remove the setter methods like setName(), setAge(), and so on from the above programme. If you want to make the variables write-only, remove the get methods like getName(), getAge(), and so on from the above programme.

It also promotes reusability and makes it simple to alter to meet new needs.

Abstraction

Abstraction refers to the user’s interaction with just a subset of an object’s characteristics and operations. To access a complicated item, abstraction uses simpler, high-level techniques.

Simple items are used to show complexity.

Keep complicated information hidden from the user.

Simple classes are used to indicate complexity in abstraction. Encapsulation is an extension of abstraction.

A Real-Life Example of Abstraction

Abstraction reveals just the most significant facts to the user while hiding the underlying intricacies. For example, when we ride a bike, we only know how to ride it but not how it works. We also have no idea how a bike works on the inside.

Advantages of Abstraction

It simplifies the process of seeing things in their entirety.

Code duplication is avoided, and reusability is increased.

Because just the most necessary information is shown to the user, it helps to enhance the security of an application or software.

Polymorphism

Polymorphism refers to the creation of items that have similar behavior. For example, objects may override common parent behaviors with particular child behaviors through inheritance. Method overriding and method overloading are two ways that polymorphism enables the same method to perform various actions.

Examine how Polymorphism and the actual world are interconnected with examples.

Take, for example, your mobile phone. It has the capability of storing your Contacts. Consider the following scenario: you wish to store two numbers for one individual. You may do this by storing the second number under the same name as the first.

Consider the following scenario: you wish to store two numbers for the same individual in an object-oriented language such as Java. Create a function that will accept as arguments two integers and the name of the individual to some function void createContact that will be defined later (String name, int number1, int number2).

Method Overriding

Method overriding is used in runtime polymorphism. When a child class overrides a parent class’s method, the child class might offer an alternative implementation.

Consider a family of three, consisting of the father, mother, and son. The father makes the decision to teach his kid to shoot. As a result, he brings him to the range with his favorite rifle and teaches him how to aim and fire at targets. The father, on the other hand, is right-handed, while the kid is left-handed. So they each have their own way of handling the pistol! Because of their differing orientations, the father was concerned that he may not be able to teach his son how to shoot.

The son, on the other hand, was astute and chose to flip his father’s hands, putting his dominant hand on the trigger rather than the father’s. Specifically, the right hand. By significantly changing the learning process, the son was able to grasp the skill of shooting!

Method overriding is the term used in programming to describe this idea.

Method Overloading

Method overloading is used in Compile Time Polymorphism. Although two methods or functions may have the same name, the number of arguments given into the method call may vary. Therefore, depending on the number of parameters entered, you may obtain different results.

With the help of a simple example, it may be comprehended in simple words. A class addition contains two add() methods, one with arguments int a and int b and the other with three integer parameters, int a, int b, and int c. As a result, the add() function is considered overloaded.

The amount of arguments given in the method calling statement determines which method is performed. For example, add(20,30) calls the two-parameter add() function, whereas add(10,20,30) calls the three-parameter add method

Objects

An object is a self-contained segment with the attributes and processes needed to make data usable in programming terms. From an object-oriented perspective, objects are the main building pieces of programs. In each application you create, you may employ a variety of objects of various sorts. Each kind of object is derived from a specific class of that type. Consider an object to be a sculpt of the real-world perceptions, processes, or objects that are important to the application you’re designing.

A variable, function, or data structure may all be considered an object. The term “object” in object-oriented programming refers to a specific instance of a class. Objects are used in software development to combine data components with methods that alter them, allowing for the usage of abstract data structures. Objects in object-oriented programming are answers to the idea of inheritance, resulting in improved program dependability, simpler software maintenance, library administration, and task division in programmer teams. Of basic terms, “Objects” are the fundamental data types in object-oriented programming languages and are used to build object-oriented programming.

Classes

In the oops concept, a class is a construct that is used to describe an individual type. The class is instantiated into instances of itself – referred to as class instances or simply objects. A class defines ingredient members that allow its instances to have position and behavior. Member variables or instance variables facilitate a class instance to maintain its position. On the other hand, other kinds of members, especially methods, allow the behavior of class instances. Simply classes consequently define the type of their instances. A class usually represents a person, place or thing, or something.

For example, a "Bird" class would symbolize the properties and functionality of birds. A single, particular bird would be an instance of the "Bird" class, an object of the type "Bird". There is a set of access specifiers in classes. private (or class-private) specifiers restrict the entrance to the class itself. Only the methods that are elements of a similar class only can access private members. protected (or class-protected) specifies enables the class itself and all classes under it (sub-classes) to access the member and public means that member can be accessed by its name using any code.

Constructors and Destructors

Constructors in most object-oriented languages have the same name as the class and are public. Constructors may be overloaded, which means that multiple argument lists can be used with the same name. The function Object() { [native code] } in PHP 5.0 is the function _construct (). Normally, attribute values would be initialised in a function Object() { [native code] }. The _destruct() method is optional, although it might be used to implement code that cleans up once an object is destroyed, such as shutting files or database connections.

OOP Advantages

- Complex things are modeled as repeatable, basic structures in OOP.
- Thus, OOP objects are reusable and may be utilized in several applications.
- Modularity for easier troubleshooting.
- Classes are easier to debug since they generally include all relevant information.
- Reuse of code through inheritance.

1.2 Objectives of the project

This project enables the user to store information about students with different attributes. The goal was to implement core C++ concepts like handling loops, control statements, file handling, and much more.

The project is a console application and the logic is written in C++.

The program makes use of different functions to perform its functions which include storing, fetching, and deleting records. All the data is stored in a text file and each entry is stored in the following line.

The program also verifies the user based on a username and password combination. The program will cease to execute if the credentials are incorrect.

The program involves best use of the command line printing statements to provides the best user experience, such as using tabs and spaces and clearing the screen whenever necessary and providing helpful error messages.

The core objectives behind this project were:

1. Build a user-friendly application.
2. Provide best user experience.
3. Store and retrieve data efficiently.
4. Write the data to file so that it will not be lost after the session is over.
5. Using well defined functions.
6. Implementing good coding practices.
7. Store, retrieve and delete data using well defined functions.
8. Implementing reusability of code.

1.3 Applications of the project

Some key applications of this project include:

1. Ideal to be used to store information of students in a school.
2. The project can be used in libraries to store information of the students.
3. As the project is lightweight and has very less hardware and software requirements thus can be run on a basic setup as well.
4. The project can be ideally used to store information of students present in a class and the data can be altered in real time.

2.1 Hardware Requirements

1. Intel Core Celeron or above.
2. 512 MB RAM or above.
3. Onboard Graphics is sufficient.
4. Standard PS/2 keyboard.

Other basic hardware requirements.

2.1 Software Requirements

1. Windows 7 Operating system or above.
2. C++ Compiler (mingw)
3. A terminal. (cmd, powershell)
4. Basic display adapters.

3.1 Algorithm

1. Display the menu.
2. Let the user choose one option
3. In loop
    a. Add
    b. Search
    c. Display All
    d. Delete
    e. Exit
4. Scan and switch on basis of operation selected.
    a. Add():

        Display the form and take the input using the following input fields.

        USN, first name, last name, branch & class.

    b. Search():

        Print the message to enter the usn.

        Scan the usn, locate the line in the text file and diplay its details.

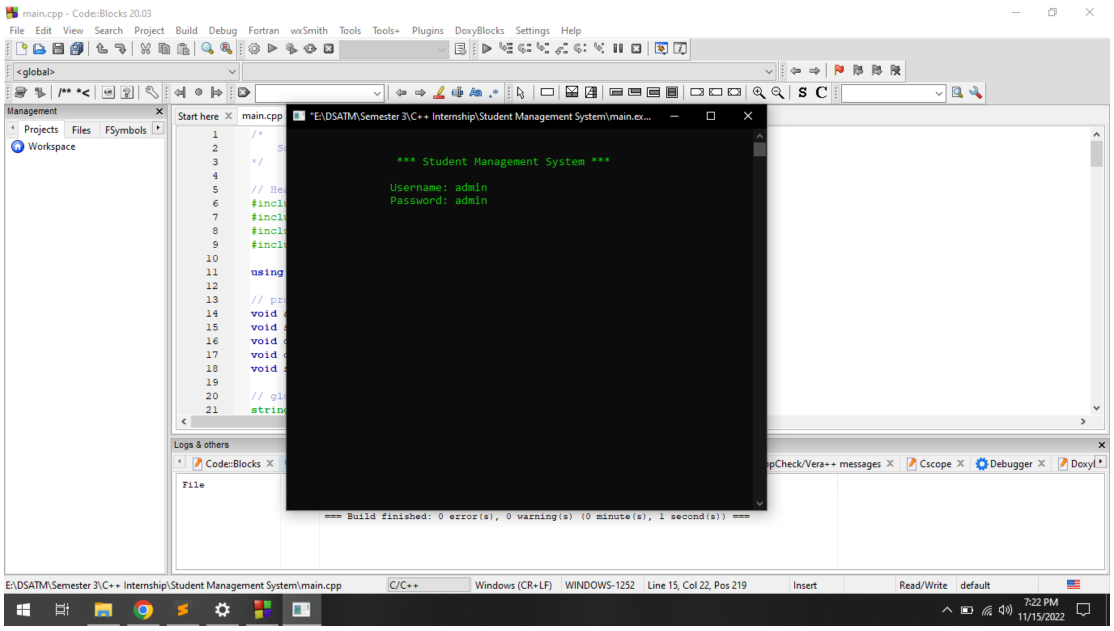
    c. Display All():

        Run a loop to display all the lines present in the text file.

    d. Delete():
        Display a message to alert the user to enter usn.
        Scan usn
        Locate the usn and its line copy every line except this one to a new temp file and rename the temp file and delete the original one.
    e. Exit
        Exit the program.

4.1 Results

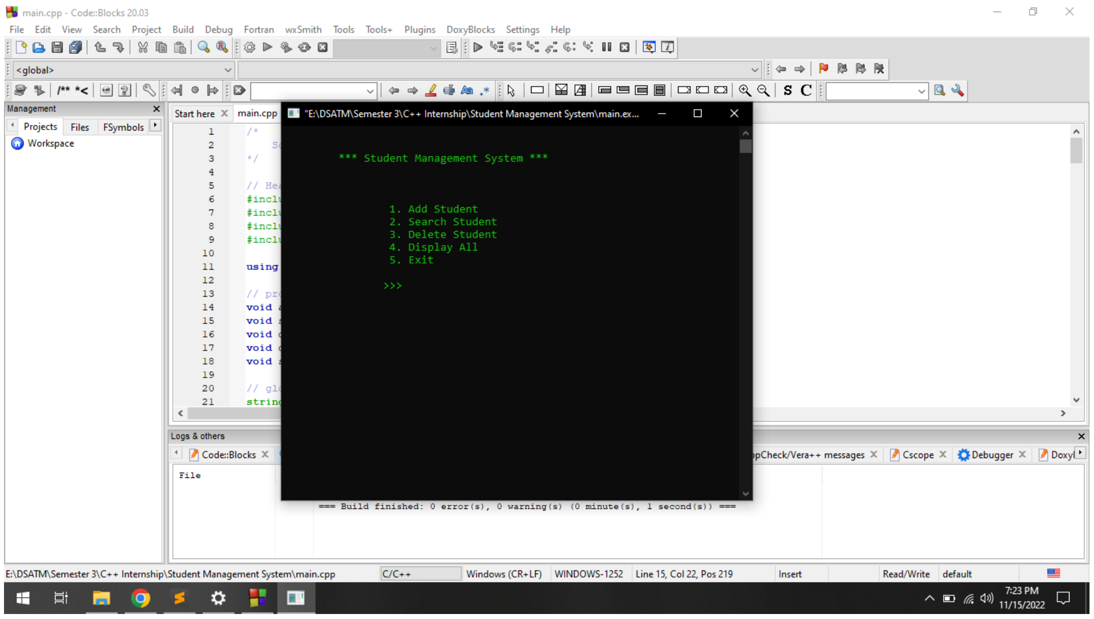
1. Login screen



```
*** Student Management System ***

Username: admin
Password: admin
```

```
=== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 1 second(s)) ===
```

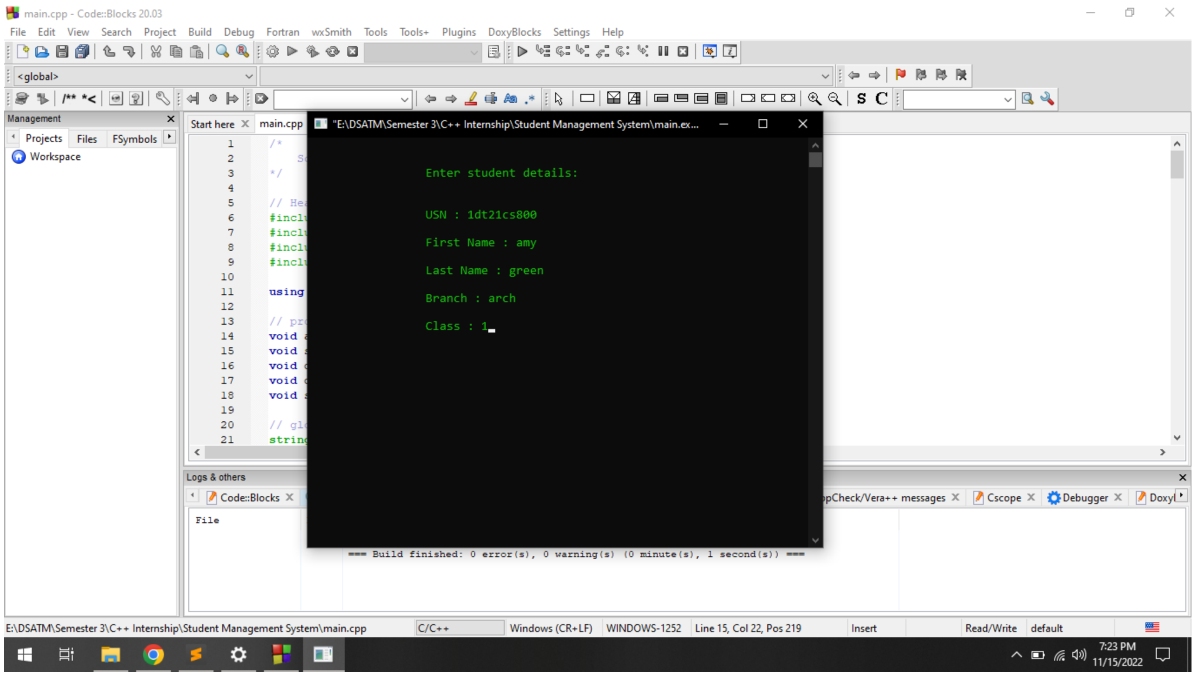
2. Main menu



```
*** Student Management System ***


    1. Add Student
    2. Search Student
    3. Delete Student
    4. Display All
    5. Exit

   >>>
```

```
=== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 1 second(s)) ===
```

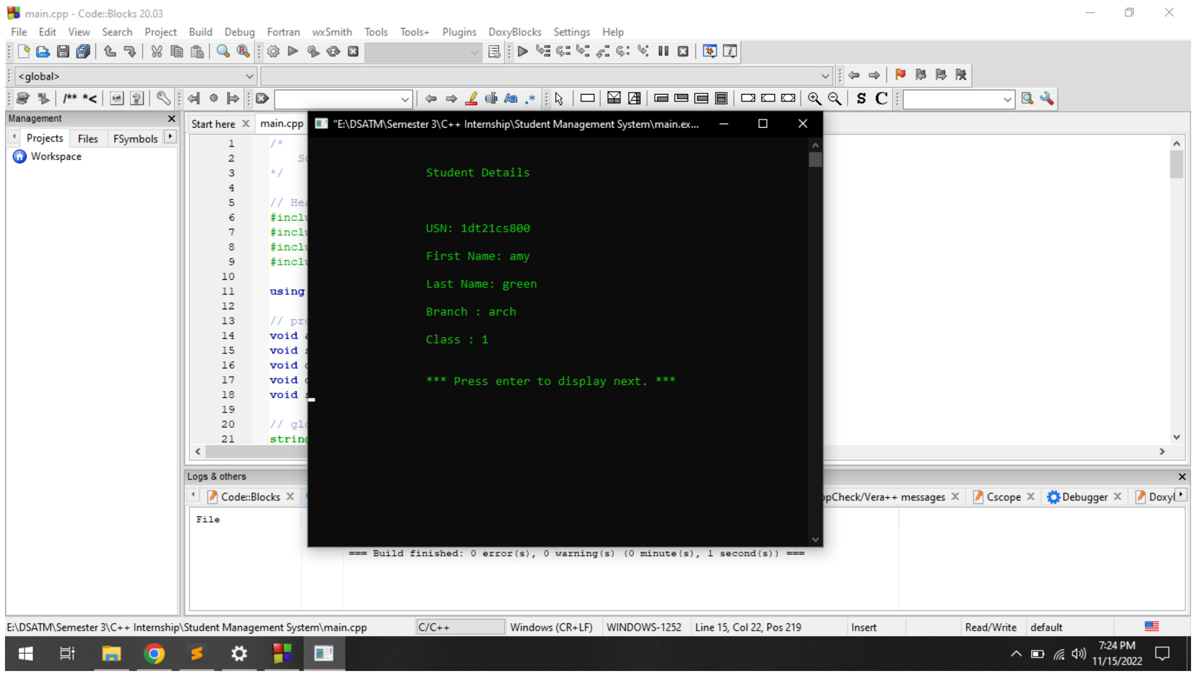
3. Add student screen.



```
Enter student details:

USN : 1dt21cs800

First Name : amy

Last Name : green

Branch : arch

Class : 1_
```

```
=== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 1 second(s)) ===
```

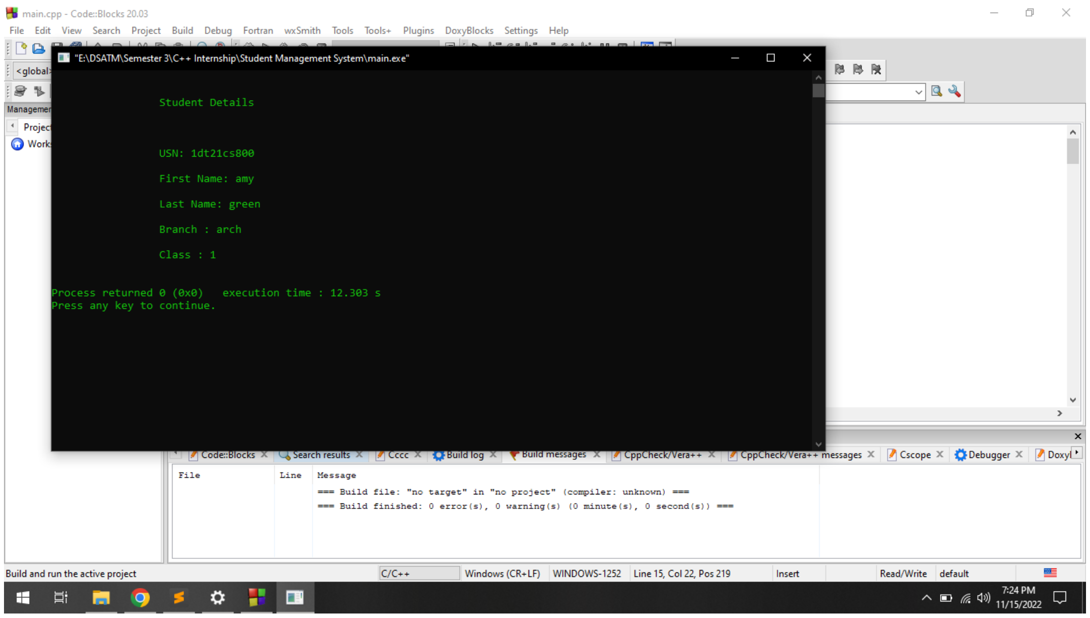
4. Display Student details:



```
Student Details


USN: 1dt21cs800

First Name: amy

Last Name: green

Branch : arch

Class : 1

*** Press enter to display next. ***
```

```
=== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 1 second(s)) ===
```

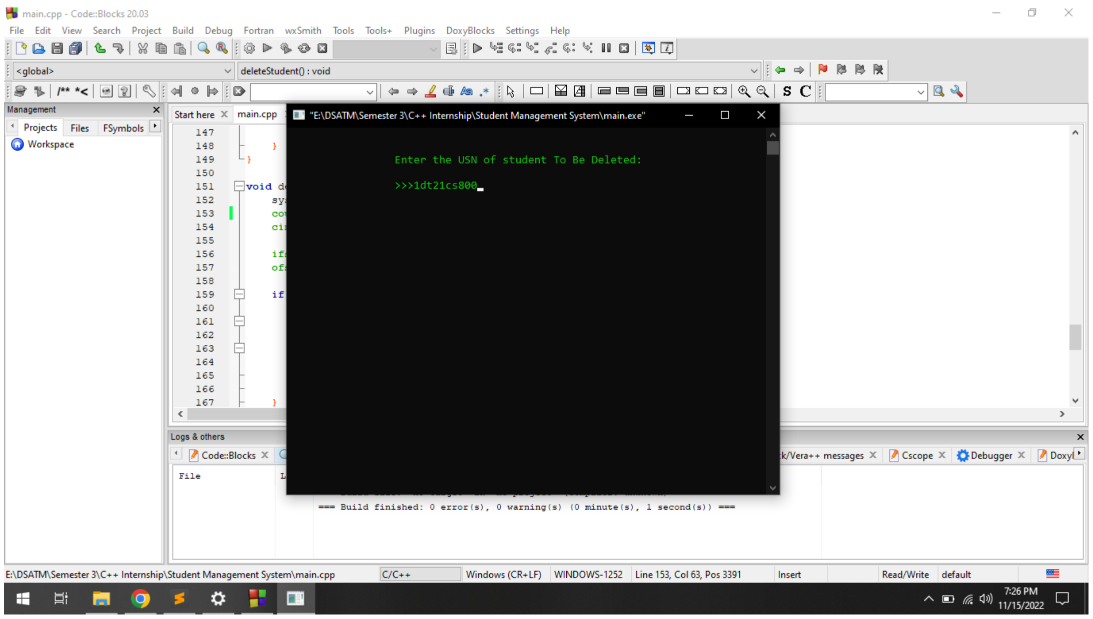
5. Looping till last entry



```
"E:\DSATM\Semester 3\C++ Internship\Student Management System\main.exe"

            Student Details


            USN: 1dt21cs800

            First Name: amy

            Last Name: green

            Branch : arch

            Class : 1


Process returned 0 (0x0)   execution time : 12.303 s
Press any key to continue.
```

```
File        Line   Message
                   === Build file: "no target" in "no project" (compiler: unknown) ===
                   === Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```

6. Deleting a user



```
"E:\DSATM\Semester 3\C++ Internship\Student Management System\main.exe"

            Enter the USN of student To Be Deleted:

            >>>1dt21cs800_
```

```
=== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 1 second(s)) ===
```

5. References

Softwares used:

1. Codeblocks
2. Sublime Text 3
3. Windows 10 with x64 bit architecture
4. Microsoft Word 2016

Websites referred:

1. StackOverflow
2. Geeks for Geeks
3. Javapoint
4. W3Schools