

Programming

INTERNATIONAL SCHOOL OF MANAGEMENT AND TECHNOLOGY



Bisesh Shrestha (HND/ First Semester)

Contents

[Introduction 4](#_Toc93670472)

[What is algorithm? 4](#_Toc93670473)

[**FEATURES of algorithm** 4](#_Toc93670474)

[**ADVANTAGES OF Algorithm** 4](#_Toc93670475)

[**DISADVANTAGES OF Algorithm** 4](#_Toc93670476)

[**Examples of algorithm** 4](#_Toc93670477)

[Development environment. 6](#_Toc93670478)

[**Advantages of Microsoft visual studio.** 7](#_Toc93670479)

[**Disadvantage of Microsoft visual studio** 7](#_Toc93670480)

[Programming language 7](#_Toc93670481)

[Types of programming language 7](#_Toc93670482)

[**Features of C#** 8](#_Toc93670483)

[Platform environment 8](#_Toc93670484)

[Design Architecture 8](#_Toc93670485)

[**How it works** 9](#_Toc93670486)

[Screenshot of all program form 10](#_Toc93670487)

[Data dictionary 14](#_Toc93670488)

[Algorithm and flowchart of the program 16](#_Toc93670489)

[(P2)Introduction 18](#_Toc93670490)

[Procedural language 18](#_Toc93670491)

[Characteristic and features of procedural programming language (PPL) 18](#_Toc93670492)

[Features of PPL 18](#_Toc93670493)

[Limitation of PPL 19](#_Toc93670494)

[Example: 19](#_Toc93670495)

[Basic IO 19](#_Toc93670496)

[Control statement. 20](#_Toc93670497)

[Looping 24](#_Toc93670498)

[Array 27](#_Toc93670499)

[Structure 28](#_Toc93670500)

[Pointer 28](#_Toc93670501)

[String 29](#_Toc93670502)

[Function 30](#_Toc93670503)

[File Handling 31](#_Toc93670504)

[Object oriented programming language 31](#_Toc93670505)

[Event driven programming language 35](#_Toc93670506)

[Features 35](#_Toc93670507)

[Advantage and Disadvantage 35](#_Toc93670508)

[Relationship between PPL, OOPL, event driven programming language. 35](#_Toc93670509)

[Conclusion 36](#_Toc93670510)

[(P3)Implementation of Application 36](#_Toc93670511)

[(P4)Introduction 48](#_Toc93670512)

[Debugging 48](#_Toc93670513)

[Debugging process with diagram 48](#_Toc93670514)

[Debugging facilities in IDE explain with snapshot 49](#_Toc93670515)

[**Break point** 49](#_Toc93670516)

[**Tracer** 49](#_Toc93670517)

[**Watch point** 49](#_Toc93670518)

[**Step command** 50](#_Toc93670519)

[Coding standard 50](#_Toc93670520)

[**Advantages of the coding standard** 50](#_Toc93670521)

[Coding standard I have used 51](#_Toc93670522)

[A. Indentation 51](#_Toc93670523)

[B. Commenting code. 51](#_Toc93670524)

[C. Variable declaration 52](#_Toc93670525)

[**Naming convention** 52](#_Toc93670526)

[**E. Camel Case** 53](#_Toc93670527)

[**F. Pascal Case** 53](#_Toc93670528)

# Introduction

In this task, I am going to describe about what is algorithm and some advantages and disadvantages. I am also describing about flow diagram of software development life cycle. I am going to describe why I have used Microsoft visual studio and c#.

# What is algorithm?

Algorithm is a logical sequence of statement or steps to solve a particular problem. It includes calculations, reasoning and data processing. Also, it is used to manipulating data in different way such as inserting and searching for a data.

## **FEATURES of algorithm**

Algorithms are there to help humans to understand the solution. So it is easy to understand because it is normal language. It have to provide input and produce output according to given input. Instruction must be clear. Algorithms can also be done with the help of pencil and paper and after some step, process should be stopped.

## **ADVANTAGES OF Algorithm**

It is a stepwise representation of a solution so it is easy to understand. It uses a definite procedure. So it is easy to debug. It is used to breakdown problem into smaller pieces or steps so that it became easier for programmer to convert into an actual program.

## **DISADVANTAGES OF Algorithm**

It is time-consuming process. Showing branching and looping in algorithms is difficult and Inserting Big tasks are difficult to put.

## **Examples of algorithm**

* Printing from 1 to 20:

1. Start
2. initialize X as 0.
3. increment X by 1
4. print X.
5. if X is less than 20 then go back to step 2.
6. Stop.

* Converting temperature from Fahrenheit to Celsius.

1. Start
2. Declare variable c and Fahrenheit
3. Input temperature in Fahrenheit
4. calculate temperature using formula(c=5/9\*(f-32))
5. print c.
6. stop.

**Flow diagram of software development life cycle.**

Software development life cycle (SDLC) is the structure which is used in defining tasks performing in desiging of software in every step of software decvelopment process. SDLC is a detail plan that describe how particular software can be develop, maintain and replace. SDLC shows various phases of the design process

Figure 1: flow diagram of SDLC

There are seven phase in SDLC. Which are described below

1. Planning

Planning is the first phase in SDLC, which is a conceptual plan to modify or improve a system. This phase is meant to determine the scope of the problem and identify its solution. In this phase cost, time, resources, benefit are such items, which should be considered. Usually the most skilled and experienced software engineers do the most perform this phase.

1. System Analysis

System analysis is second phase in SDLC, In which system analysist determine what is needed as well as how it can be fulfilled in a software.

1. System Design

Design is the third phase in SDLC, where a designer design the software according to client and make it as sample and present it to client.

1. Development

Development is the forth and most real working phase in SDLC, In which a programmer do the real project developing a software

1. Testing

After development of a software, testing is the fifth phase where software which is created in development phase is tested such as code testing, security testing, performance testing. Bugs and defects are searched in this phase.

1. Implementation

After development and testing phase of SDLC, implementation is sixth phase where software is implemented in real field

1. Maintenance

Maintenance is the last phase of SDLC in which software is updated or improved time to time.

# Development environment.

It is a procedure for developing, testing and debugging the software. Developing, staging, and production are normally three tiers of development environment. Testing and checking weather software run smoothly or not using the code is developing tier. Software then moves on staging tier where it check that the software does not fails then the software is moved on production tier after the approval. After approval is done, then the software will be the part of the development environment.

There are three types of development environment

1. Language centered environment: it is built by only one specific language.
2. Structure oriented environment: user are allowed to structure directly in this environment.
3. Toolkit environment: this provide tools that includes a configuration management and version control.

Glade, adobe flex builder, Microsoft visual studio, are some example of development environment. Among them Microsoft visual studio is one of the development environment which I have used. I have preferred Microsoft visual studio because it is used to develop computer programs and other different application. It is also a widely spread application. It also supports syntax highlighting and code for loops, variable, methods, function and queries. It provide feedback about syntax and compiles errors. Source-level debugger and machine-level debugger are included in visual studio. Natural code and managed code can be used for debugging application. It uses true object oriented language.

## **Advantages of Microsoft visual studio.**

Visual studio is easy to install and is faster and it can be easily controlled which helps in developing graphical user interface and to connect them. Language used in it is more flexible then other development environment. In Microsoft visual studio codes can be easily written. It increase productivity in daily development work. Color code are generated such as comments in blue and keyword in blue, which help user to understand it clearly.

## **Disadvantage of Microsoft visual studio**

Program created from Microsoft visual studio cannot be transferred easily to other operating system. It need higher capacity processor laptops.

# Programming language

A programming language is a set of instruction used in instructing computer to perform certain task. Every programming language uses a set of keywords and special syntax. These programming languages allows computers to process large and complex part of information quickly.

## Types of programming language

There are three types of programming language, which are described below:

1. Low-level language: this is computer’s most understandable language used for performing task. Machine language and assembly language is two categories of low-level language. Machine language uses binary number (0s and 1s) which is directly understood by the processor and in assembly language symbols are used just as ‘ADD’ for additional.
2. High-level language: this programming language is easy to learn because it use human language (English) which is translated in machine level language using interpreter or compiler. Procedural-oriented language and problem-oriented language are two categories. Procedural-oriented language are designed for expressing the procedure of a problem using languages such as Pascal, COB0L, c, etc. in problem-oriented language it allows user to specify what the output should be. Visual basic, C#, PHP, etc. are some example of problem-oriented language.
3. Natural language: this language is still in research phase where we can write statement in normal sentence. This is much more interactive and interesting language.

Among all programing, I have used C#, which is pronounced as ‘c sharp’. It is an object-oriented language, which is also a hybrid of C and C++. It uses Boolean data.

## **Features of C#**

It has features that makes it simpler, faster and is easier to develop solution. It is simple, modern, object-oriented language, with the aim of combining productivity of visual basic and power of C++. It also support different language such as VB, VC++, JSCRIPT, etc.

# Platform environment

The phase on which computer program can operate is Platform environment. It execute a piece of software that may be hardware or OS. If you want to create window desktop app then the first decision will be which platform should we use. Universal Window Platform, WPF(.NET), Windows Form(.NET), Win32 are four platform provided by windows. All those platform helps in creating desktop apps.

**Requirements in Visual Studio:**

* It need Windows 7 or above Operating System.
* It need Minimum 2 GB and Maximum 4GB or more memory.
* It need Processor of 1.8 GHz or faster processor.
* It need Hard Disk - (Minimum – 50 GB) & (Maximum – at least 130 GB depending on features installed.)

**Requirements in SQL Server:**

* It need Windows 7 or above Operating System.
* It need 1GB Minimum and 4GB Maximum Memory.
* It need Processor of x86 Processor – 1.0 GHz, x64 Processor – 1.4 GHz, Recommended – 2.0 GHz or faster.
* It need Hard Disk of minimum of 6 GB space for better performance.

# Design Architecture

Assignment management system is the application I have designed. In this application, it is especially designed for assignment manager who managed and recorded the assignment of all students. It record about the status of assignment which by whom to whom, when is assignment is submitted.

## **How it works**

At 1st I have designed login page where we login using email and password. After login is successfully done then dashboard form appears where we can see different button( teacher, student, subject, faculty, assignment submission and status). Button, which we select, opens the form in which we have to insert information for record. In my application, we don’t need to return to dashboard for selecting another window form we can directly change from the same window form. there are 6 windows form which are described below:

1. Teacher form: In this form, we record about teachers detail such as name, contact number, address, and email.
2. Student form: In this form, we can record about student detail such as name, address, class, section, faculty etc.
3. Faculty form: In this form, we can record about faculty.
4. Subject form: In this form, we can record about subject
5. Assignment submission: In this form, we record about the status of assignments such as by whom to whom, at what time is it submitted, etc.
6. Status form: In this form, we have status such as submitted or not, completed or not etc.

TEACHER

LOG IN

STUDENT

DASHBOARD

FACULTY

SUBJECT

LOG OUT

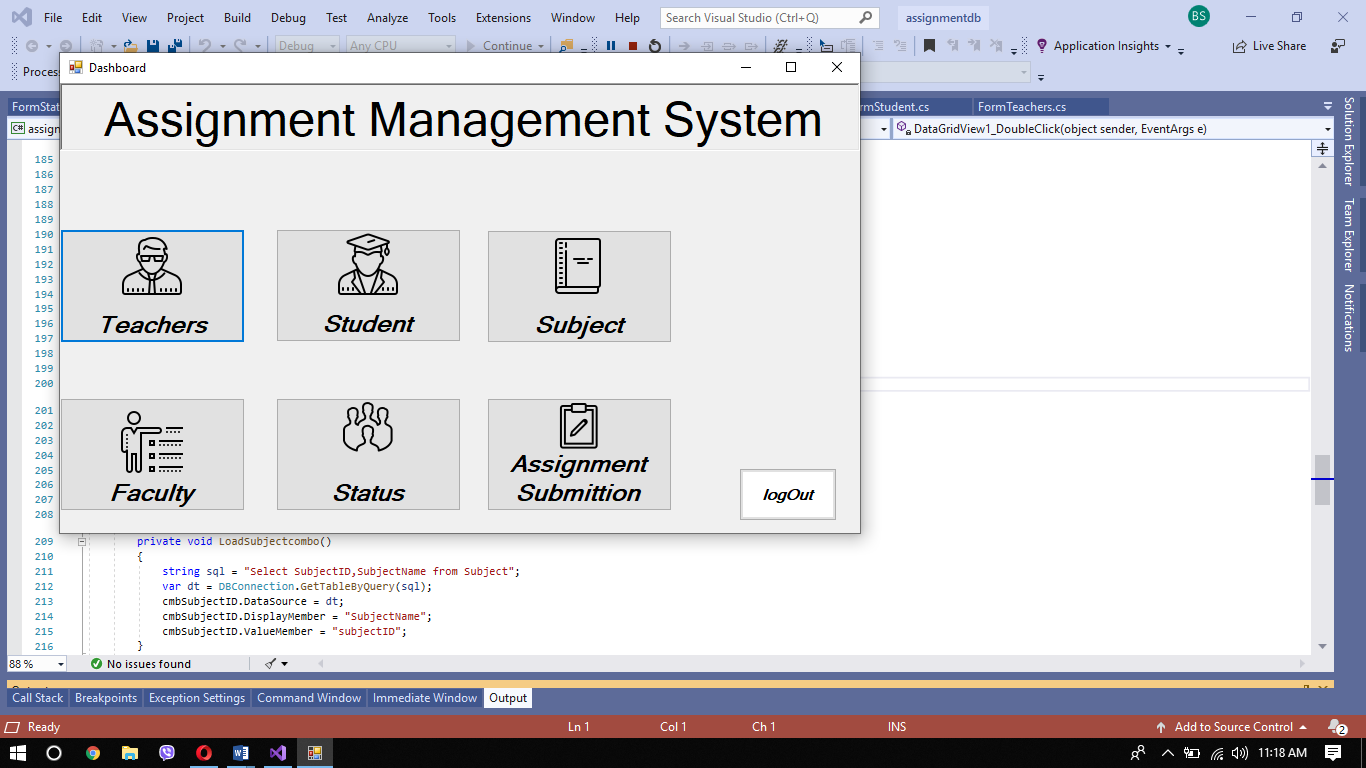
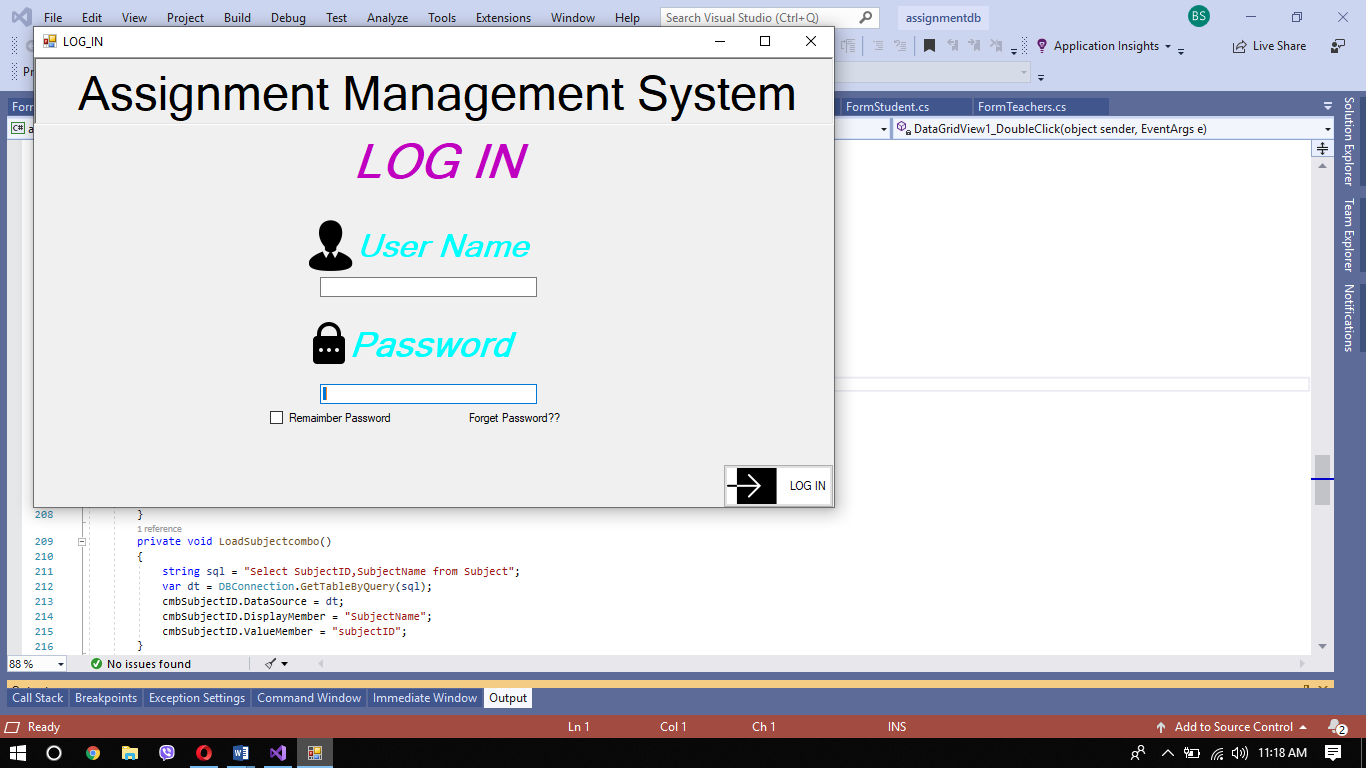
ASSIGNMENT SUBISSION

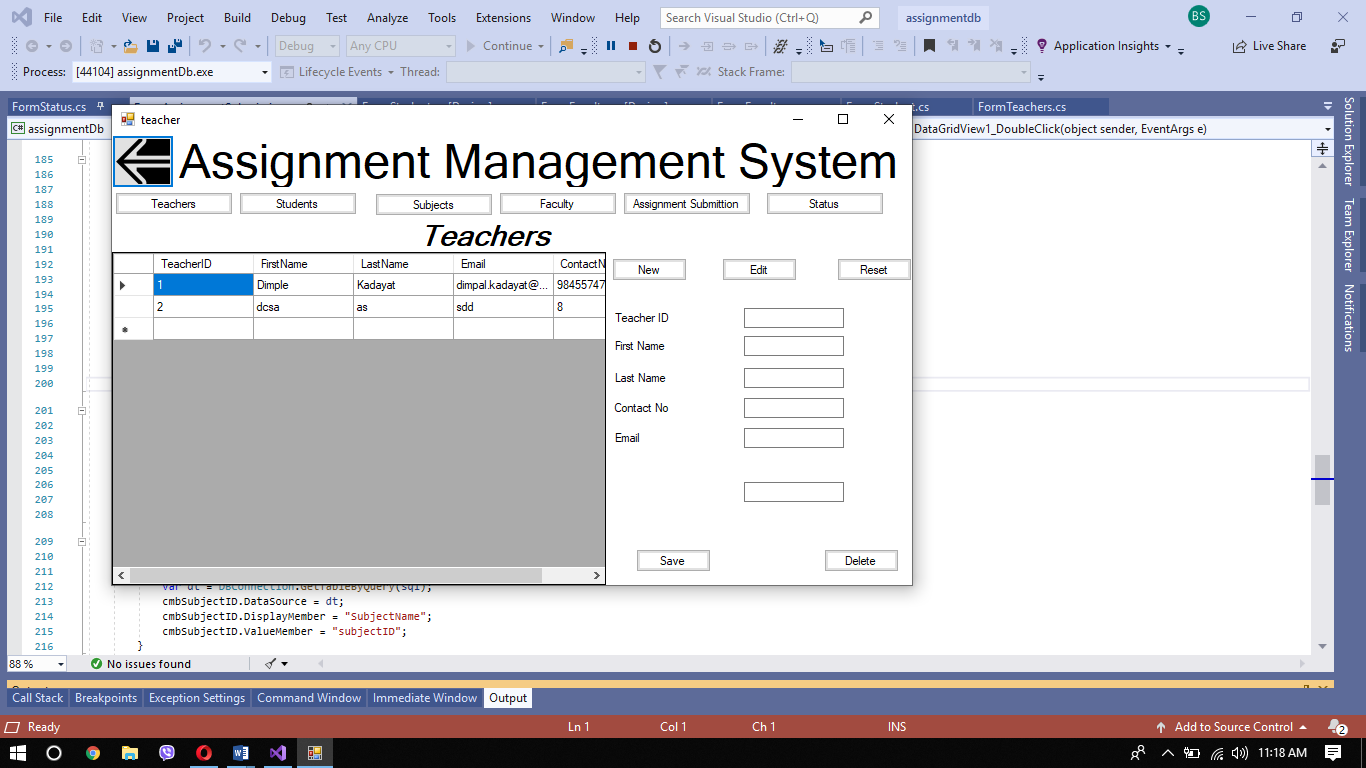
S

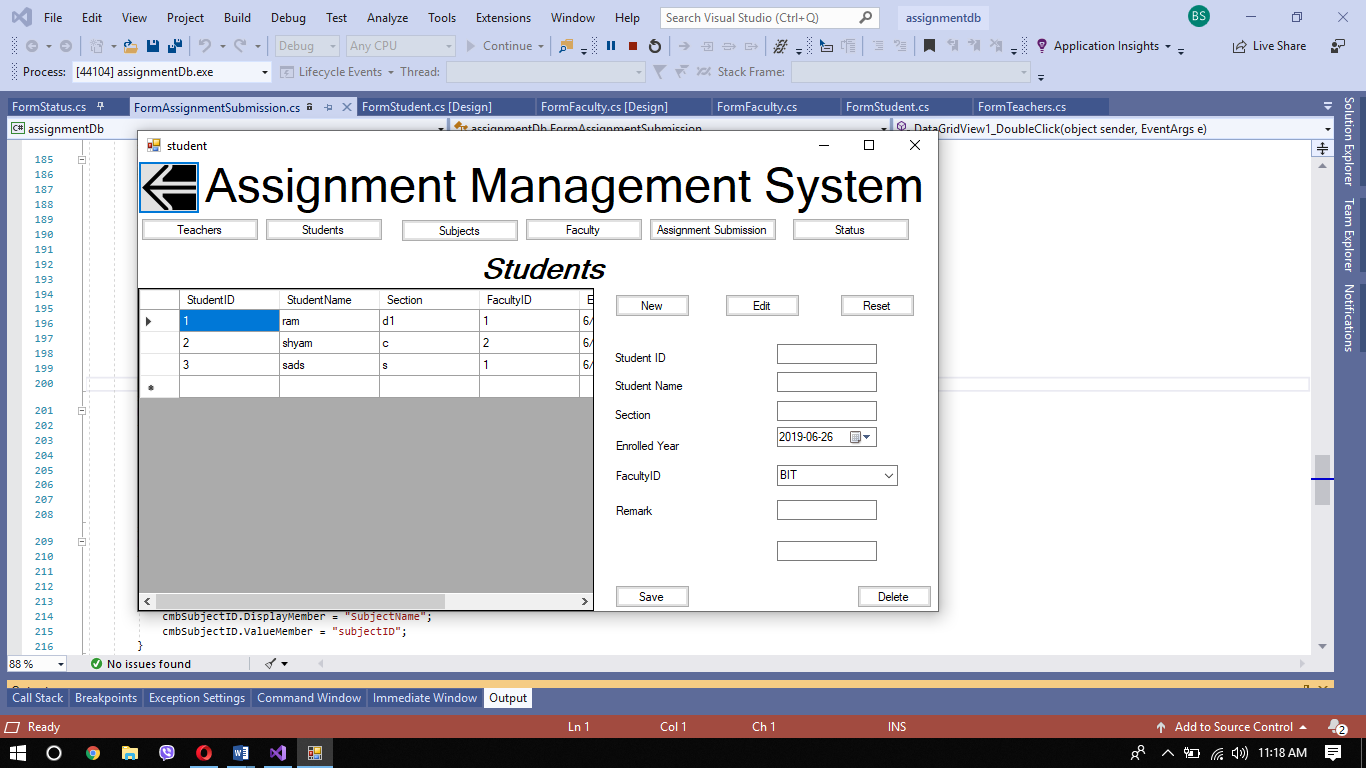
STATUS

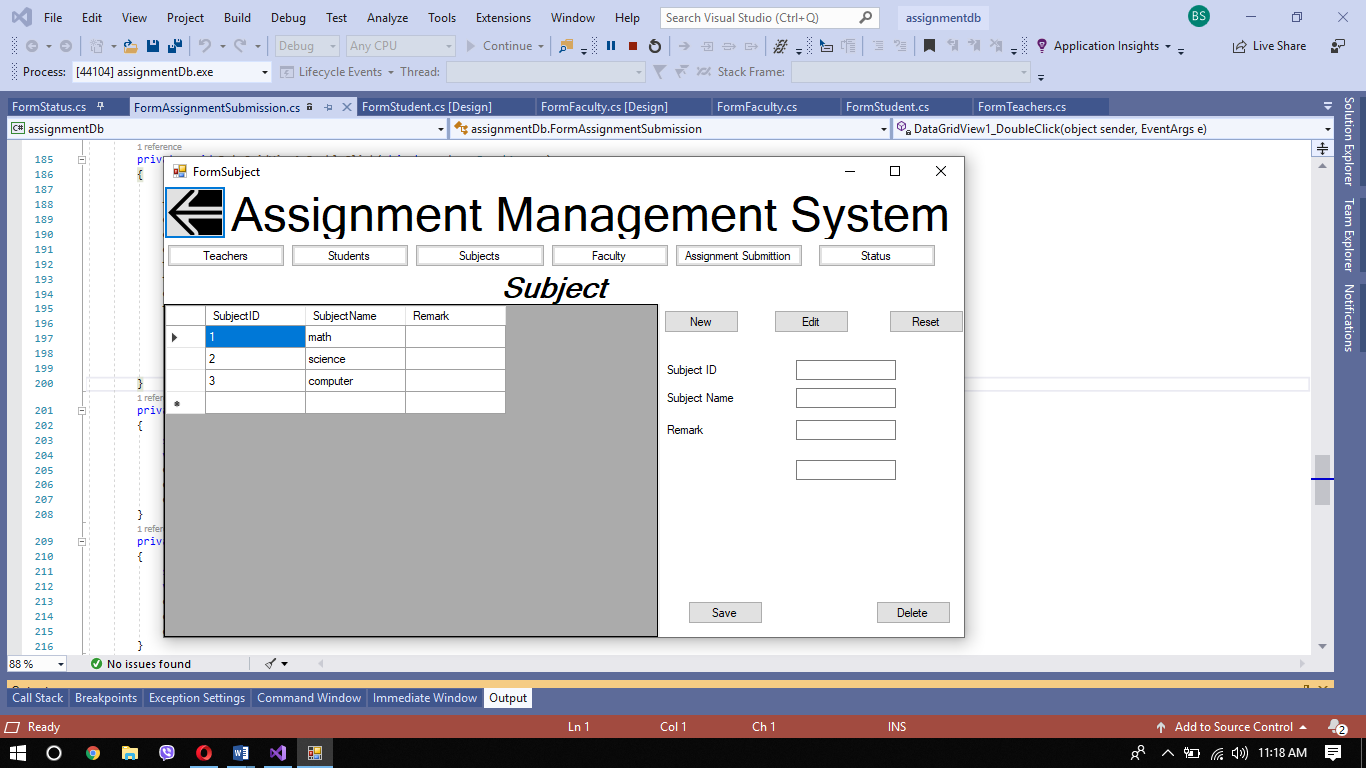
Diagram: Assignment Management System

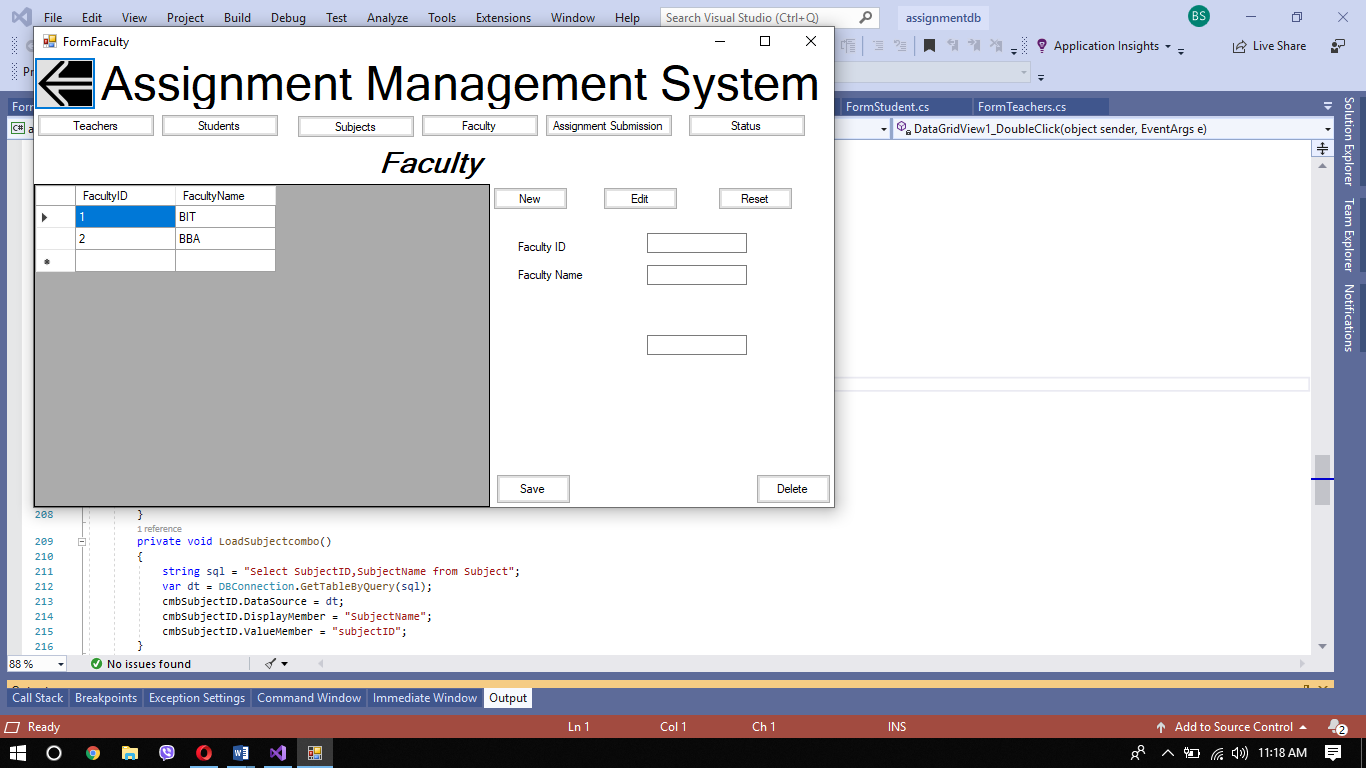
Screenshot of all program form I have created

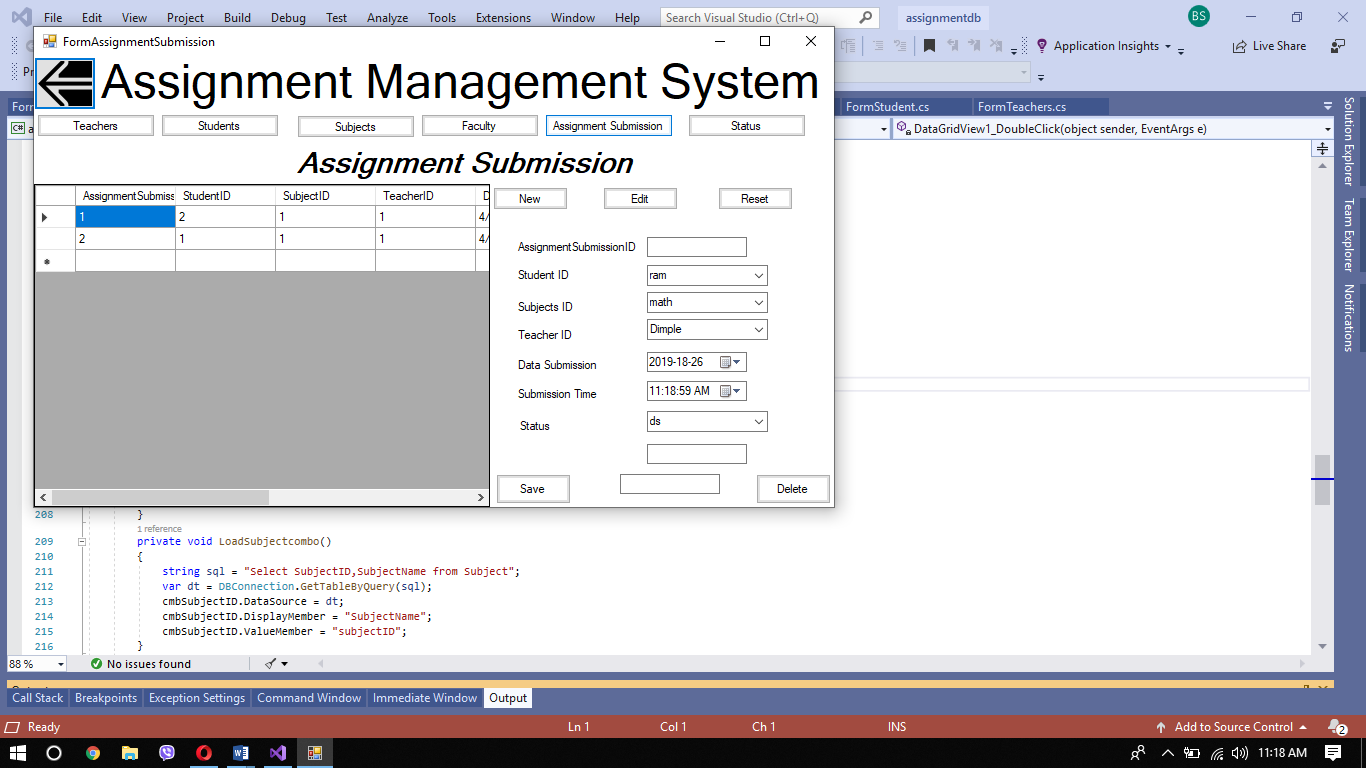


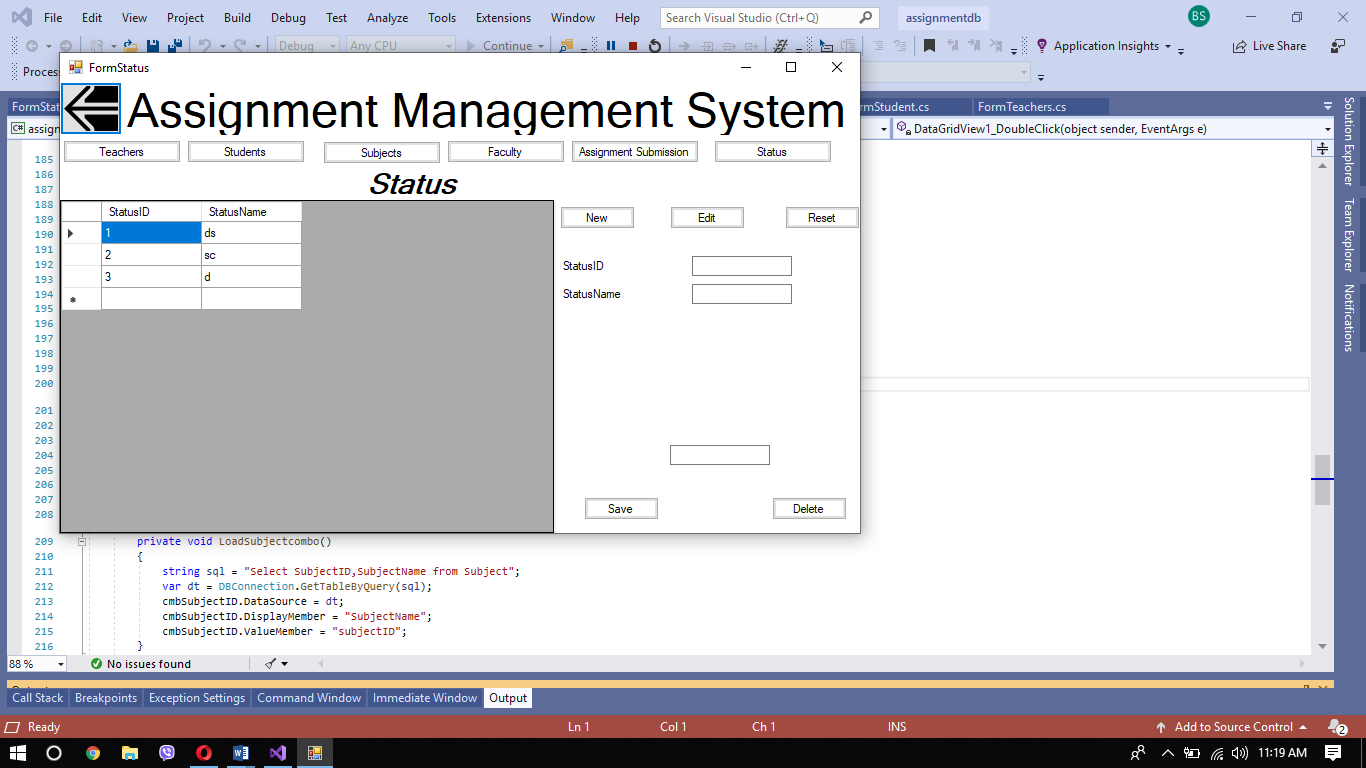






****

****

****

# Data dictionary

Data dictionary is a collection of data or item benefit for programmer. In data dictionary types of data such as text or image or binary values is described. By using data dictionary programmer can understand what and where data item fits in application. It consist of tables containing different column such as column name, data type, constrain etc which stores input values given by the user.

Table 1: teacher

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Column Name** | **Data Type** | **Constraint** | **Remark** |
| **1** | TeacherID | int | PK, Not null, auto increment |  |
| **2** | FirstName | Varchar(50) | Not null | Stores teacher first name |
| **3** | LastName | Varchar(50) | Allow null | Stores teacher last name |
| **4** | Email | Varchar(50) | Allow null | Stores teacher email |
| **5** | ContactNo | Varchar(10) | Not null | Stores teacher contact number |

Table 2: Student

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Column Name** | **Data Type** | **Constraint** | **Remark** |
| **1** | StudentID | Int | PK, Not Null, auto increment |  |
| **2** | StudentName | Varchar(50) | Not Null | Stores student name |
| **3** | FacultyID | Int | FK, Not Null | Stores student Faculty id number |
| **4** | EnrolledYear | Date | Not Null | Stores student enrolled year |
| **5** | Section | Varchar(50) | Allow Null | Stores student section |
| **6** | Remark | Varchar(250) | Allow Null | Stores remarks |

Table 3: Faculty

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Column Name** | **Data Type** | **Constraint** | **Remark** |
| **1** | FacultyID | Int | PK, Not Null, auto increment |  |
| **2** | FacultyName | Varchar(50) | Not Null | Stores faculty name |

Table 4: Subject

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Column Name** | **Data Type** | **Constraint** | **Remark** |
| **1** | SubjectID | Int | PK, Not Null, auto increment |  |
| **2** | SubjectName | Varchar(50) | Not Null | Stores subject name |
| **3** | Remark | Varchar(250) | Allow Null | Stores remark |

Table 5: Teacher Dispatch

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Column Name** | **Data Type** | **Constraint** | **Remark** |
| **1** | TeacherDispatchID | Int | PK, Not Null, auto increment |  |
| **2** | AssignmentID | int | FK,Not Null | Stores faculty name |
| **3** | DateOfDispatch | Date | Not Null | Stores date of assignment dispatch |
| **4** | ReturnTime | Time | Allow Null | Stores return time of assignment |
| **5** | Remark | Varchar(150) | Allow Null | Stores remark |

Table 6: Assignment Submission

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Column Name** | **Data Type** | **Constraint** | **Remark** |
| **1** | AssignmentSubmittionID | int | PK, not null, auto increment |  |
| **2** | StudentID | int | FK, not null |  |
| **3** | SubjectID | int | FK, not null |  |
| **4** | TeacherID | int | FK, not null |  |
| **5** | DateSubmission | Date | allow null | Stores date of assignment submission |
| **6** | SubmissionTime | Time | allow null | Stores time of assignment submission |
| **7** | Status | Varchar(50) | allow null | Stores status |
| **8** | Comment | Varchar(250) | allow null | Stores comments |

# Algorithm and flowchart of the program

For LOGIN

|  |  |
| --- | --- |
| **Algorithm** | **flowchart** |
| 1. Open application | Error  Enter email and password  If  true  Dashboard  false |
| 1. Enter email and password |
| 1. Enter to dashboard page if entered email and password is true with the record enter to dashboard. Else display error |
| 1. Log out |

For save

|  |  |
| --- | --- |
| **Algorithm** | **flowchart** |
| 1. Start | Error  Enter all data  If  true  Record  false |
| 1. Enter all data |
| 1. If true record the data else display error |
| 1. Stop |

For delete

|  |  |
| --- | --- |
| **Algorithm** | **flowchart** |
| 1. Start | Choose data  If data display    True  Record  false  Error |
| 1. Choose data |
| 1. Check data is displayed in text box |
| 1. If true delete the data   else display error |
| 1. stop |

# (P2)

# Introduction

In this part I am going describe about the procedural programming language(PPL). I am going to talk about characteristic, features, limitation of PPL. I am also going to describe about Different types of programming language I have used.

# Procedural language

A procedural language is a language of a computer programming that follows a set of commands in order. Basic, c, FORTRAN, java, and Pascal. Script and software programmers uses some common procedural language. A programming editor can be used to create a program using a procedural language. These editors help users develop programming code using one or more procedural languages, test the code, and fix bugs in the code.

## Characteristic and features of procedural programming language (PPL)

Instead of information, it focus on processing or doing stuff. The whole programmer, based on behavior such as reading, calculating and printing is introduced. Many features are therefore written to fix a program from top to bottom. All features frequently share global information. Data shifts from function to function openly.

## Features of PPL

1. Predefine function: Examples of predefined feature like "System.out.println" can be used as a feature that is already within a programming language, which allows programmers to work easily.
2. local variable: Local variables are a variable that can only be accessed within the specific chunk ,block of code in which it was written, and not through the entire code script (like a global variable) declared a local variable to override the same variable name in the larger scope.
3. Global variable: A global variable is a variable that can be viewed by any other procedure throughout the program, and is accessible by any other task running in the program. A global variable is mostly a static variable, the extent of which is the program's entire runtime.
4. Parameter passing: The passing of parameters enables the transmission of variable values to the program that will manage it with a procedure.
5. Modularity: Modularity is a software technique that demonstrates that it is possible to separate the functionality into individual, interchangeable modules, each allowing it to perform the particular thing it is intended to do. These all combine to accomplish an general objective as distinct duties.
6. Procedures: If you have programming processes, the program that has them in will systematically follow the processes systematically. The program does precisely what the programmer says to do in the order set by the programmer

## Limitation of PPL

Although procedural-oriented programs are extremely powerful, they do have some limitations. What you can do with procedural programming is limited. You don't have a lot of flexibility with it when using procedural programming. Procedural programming is packed tightly. What I mean is that the program won't function without one line of code. If you want to modify portion of the program, you need to modify other components to operate properly. Another limitation is that procedural programming eventually turns into lots of lines of code that can be confusing to go through. If there is a bug in the program, it will take some time to find the problem by going through the lines of code.

## Example:

Example of C coding

int main()

{

int count;

for(count=0; count<10; count++)

print("This is a test that will be printed everytime the number is under 10");

return(0);

}

example of pascal coding

program Loops;

var

i:integer;

s: String;

begin i = 0;

repeat

i = i+1

Write('enter a number');

ReadIn(s);

until (i = 10) or (s=0);

end.

## Basic IO

# Control statement.

A control statement is a statement that determines the execution of other statements. Control statements allow us to specify program policy flow i.e. the order in which the program directions are to be executed. They allow decision-making, constantly performing assignments, or jumping from one portion of code to another. The types of control statement are:

1. Decision making statement

* If statement

|  |
| --- |
| Statement just below if  If Body  If condition |

An if statement is a Boolean expression followed by one or more statements. If the Boolean expression is evaluated to true, then the code block will be performed within the ' if ' declaration. If the Boolean expression fails, the first set of code will be executed after the end of the ' if ' statement (after the curly brace closing).

Syntax

{If (Boolean expression)

Statement

}

* If else statement

The if-else statement is used to perform a logical test and then perform one of two possible behavior depending on the test result (i.e. whether the test result is true or incorrect). If the condition indicated in the if statement is valid, the statements inside the if-block are performed and immediately after the if-block the control is passed to the statement. Even if the condition is false and no other block is present, immediately after the if-block, control is transferred to the statement.

|  |
| --- |
| Statement just below if  If Body  If condition  false  true  Else body |

Syntax:

If(condition)

{

Statement

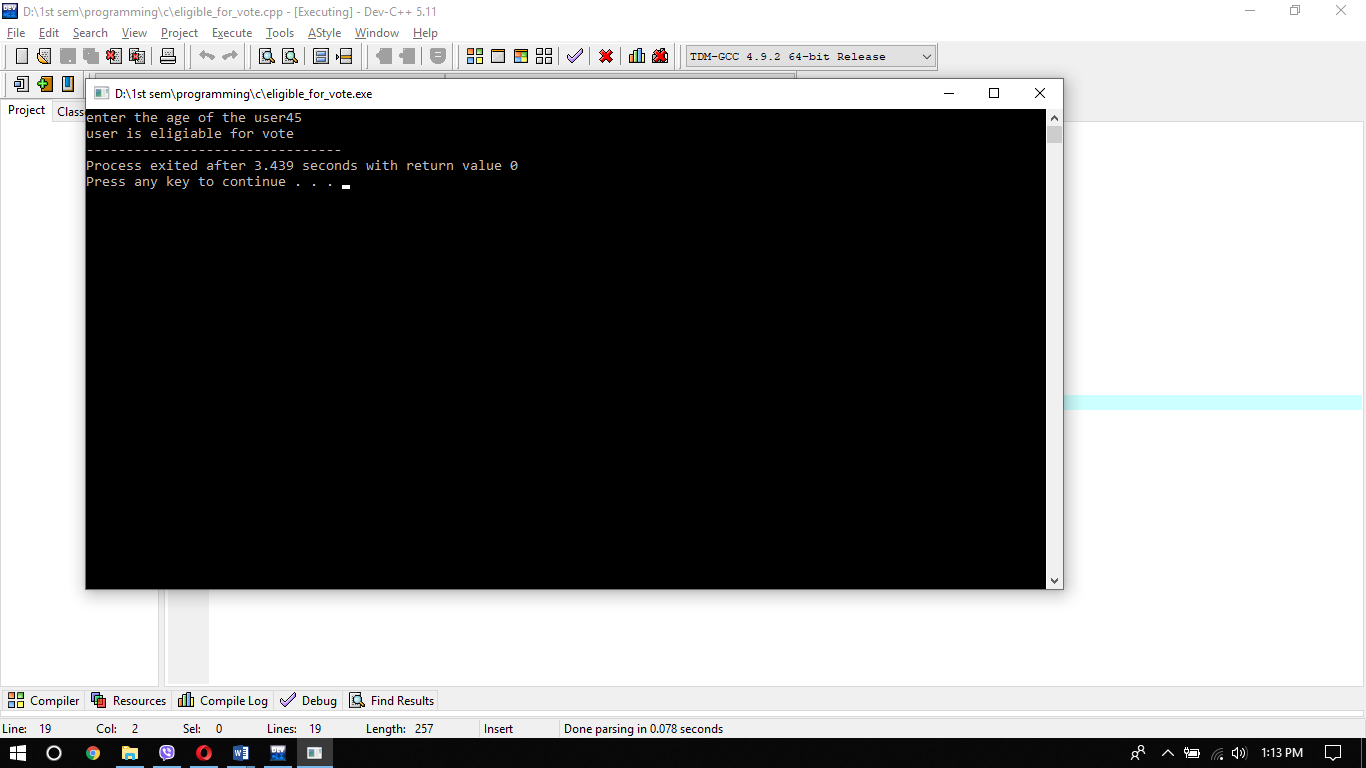
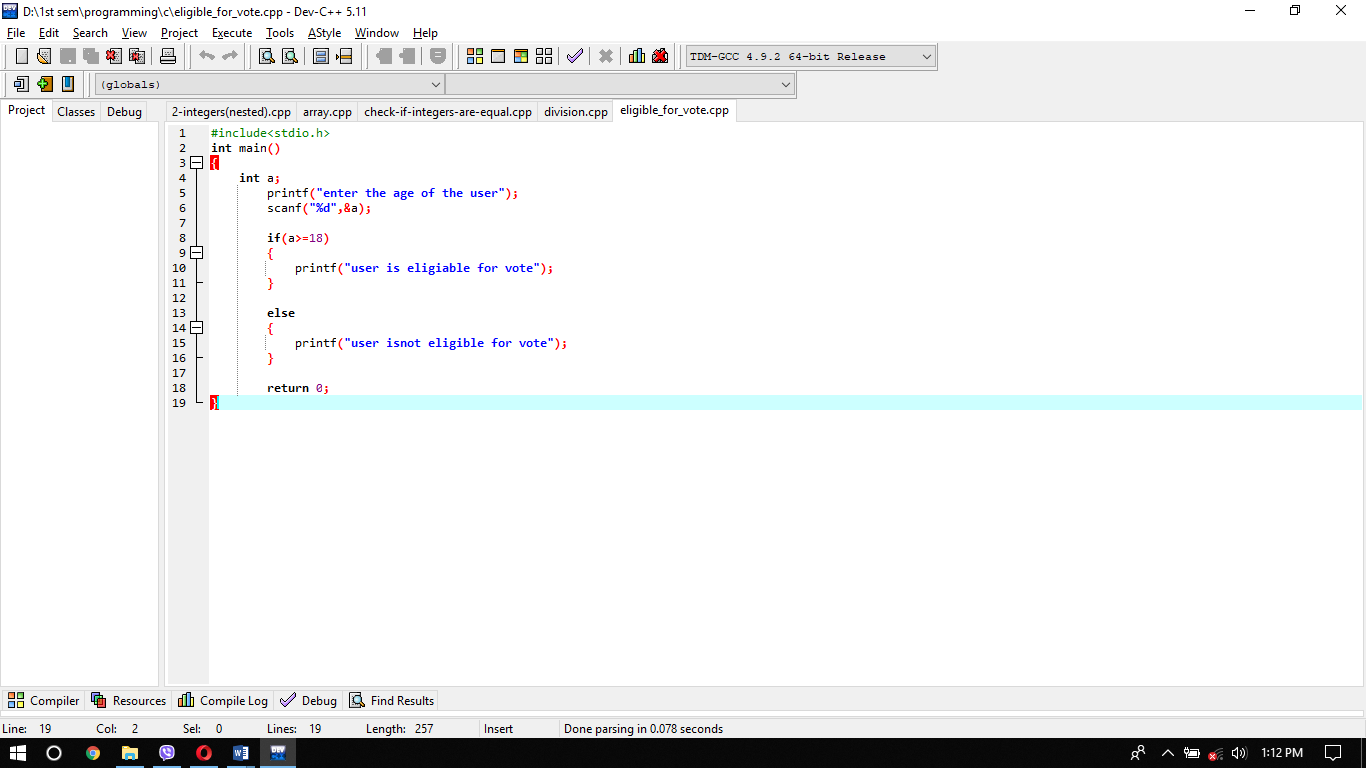
}

Else

{

Statement

}



* Nested If statement

A nested if an if statement is another's target if statement. Nested if statements means a statement in another if statement in another statement.

Syntax

If(condition1)

{

If(condition2)

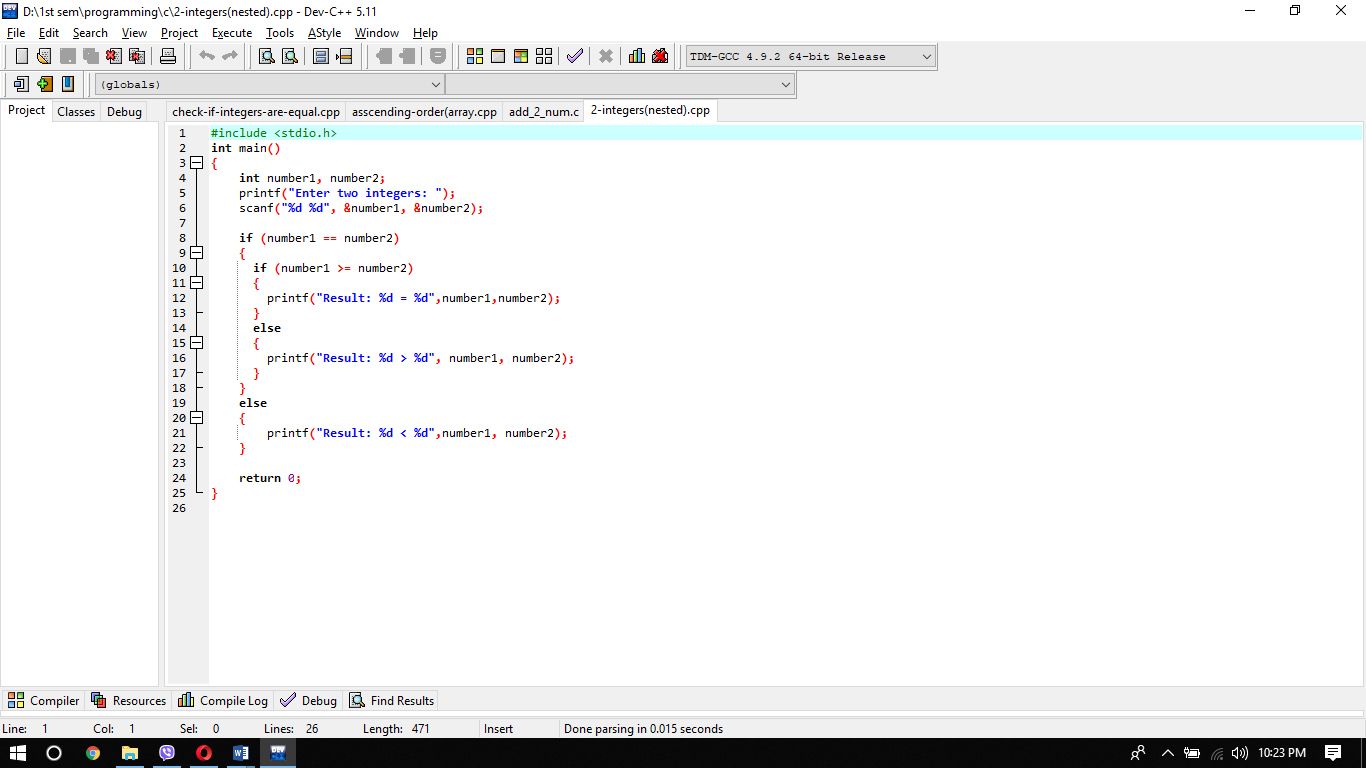
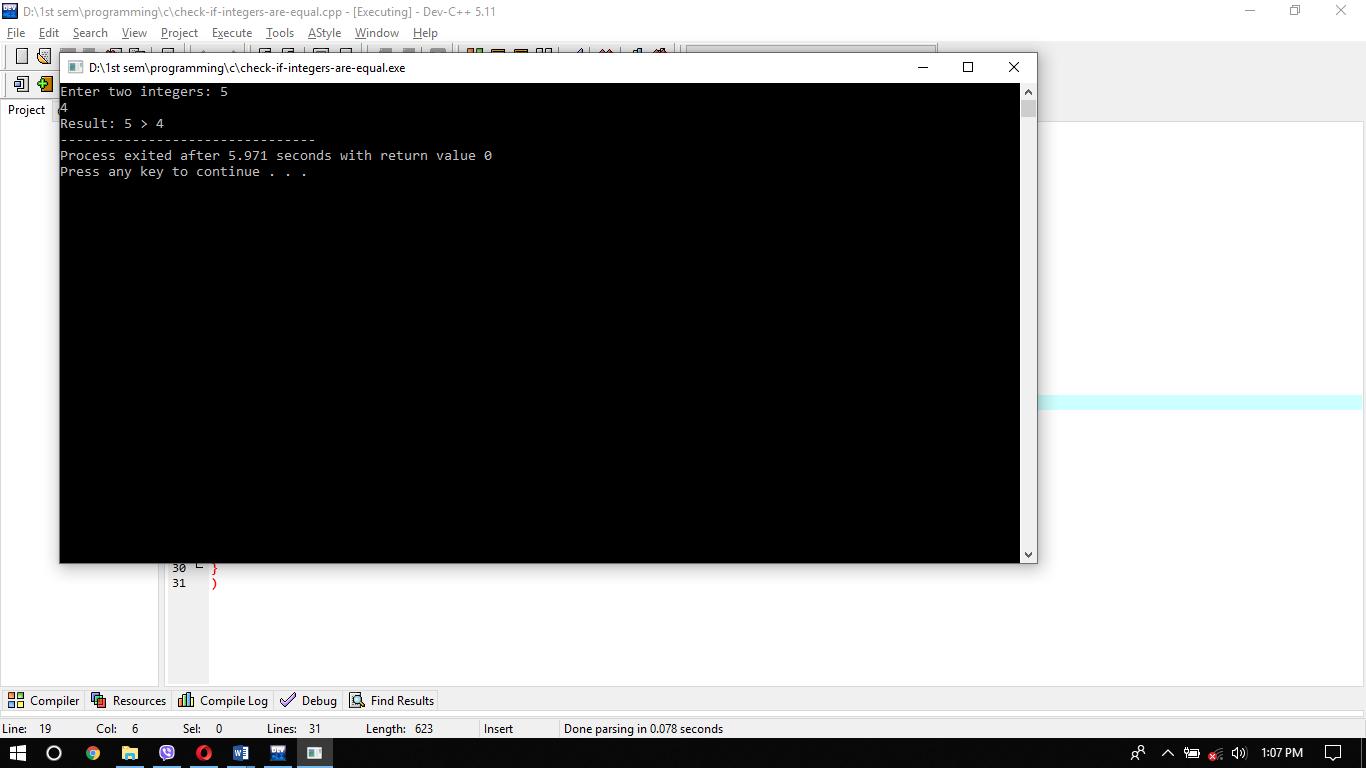
{

Statement

}

}

|  |
| --- |
| Statement just below if  If Body  If condition  Nested If condition  false  true true  Else body  Else body |



1. Selection statement

* Switch statement For various way choices, a switch statement is used that branches into different code segments based on a variable or expression value. It provides an easy way to dispatch execution based on the value of the expression to distinct components of code. Switch is a control declaration allowing a value to alter implementation control.

Syntax

Switch(n)

{

Case1:

Break;

Case2:

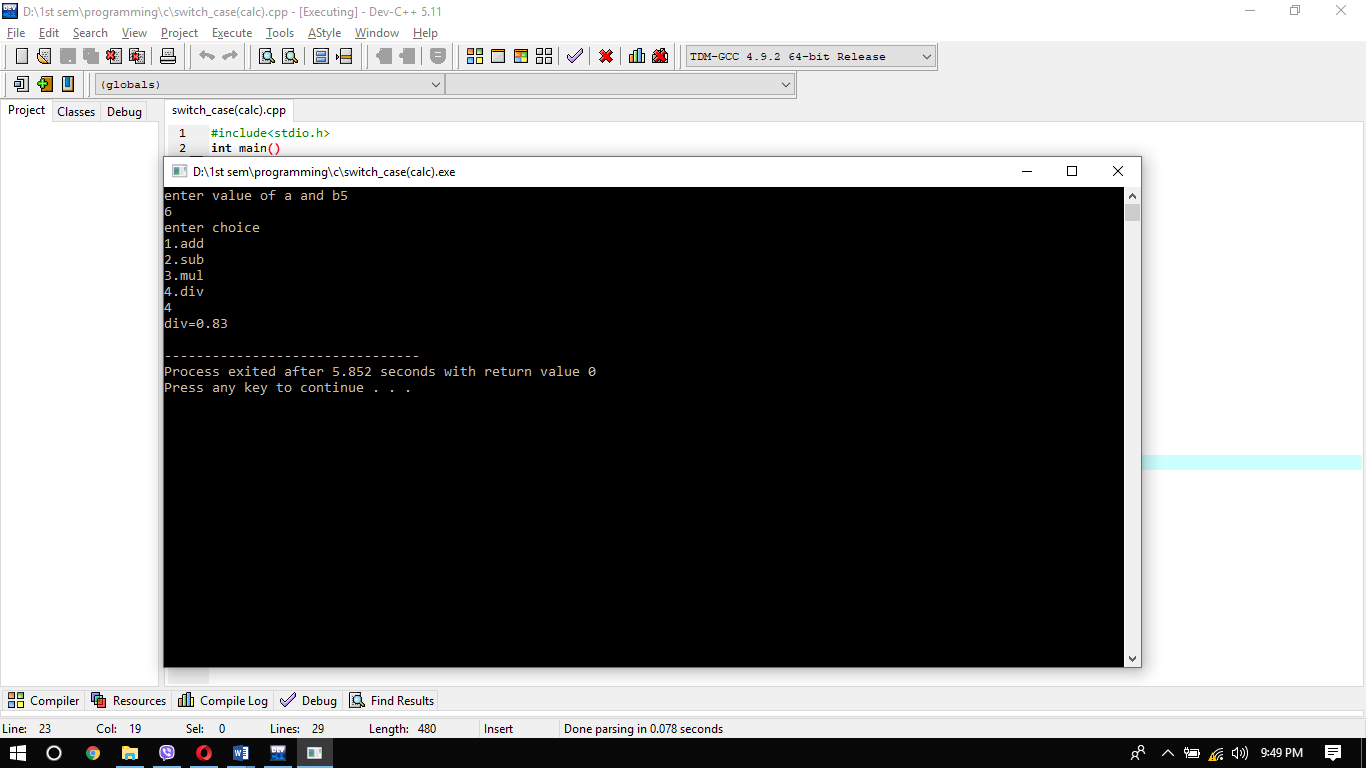
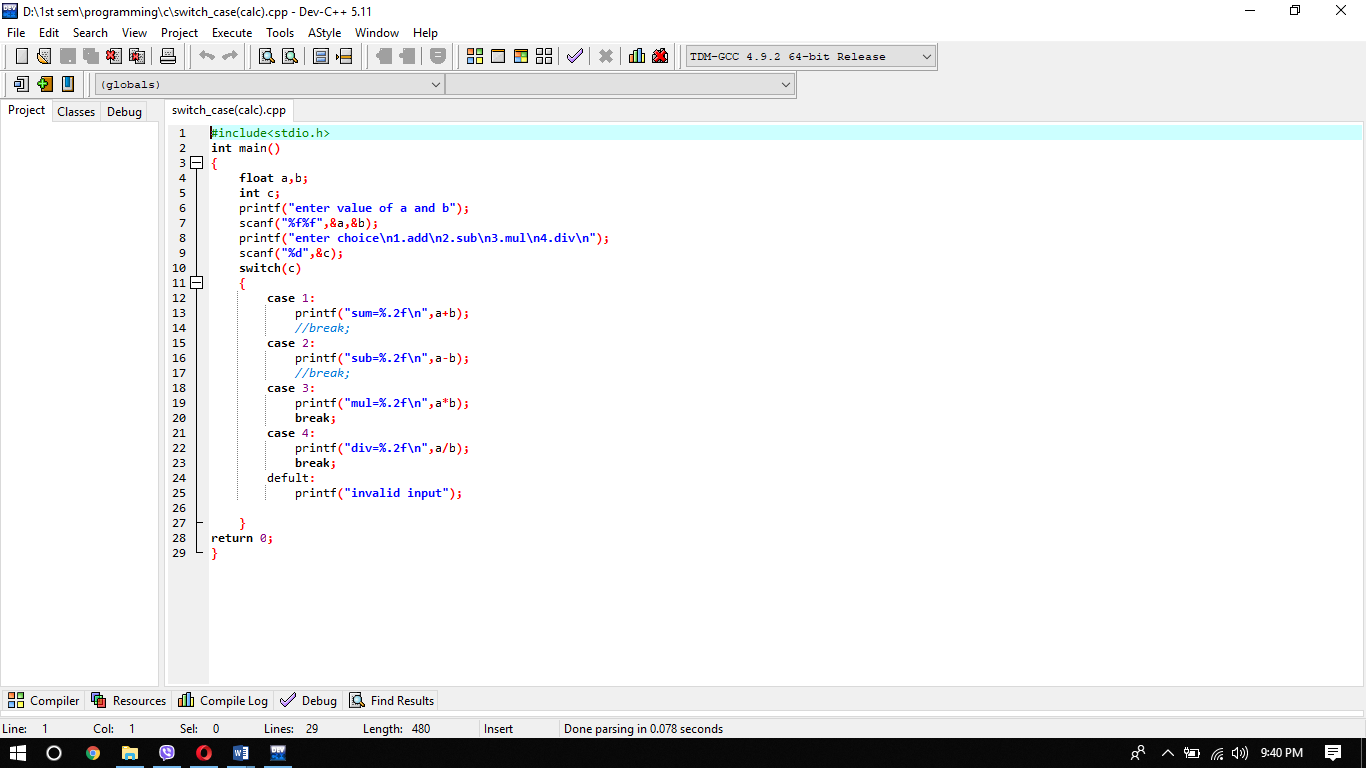
Break;

Default:

Break;

}

|  |
| --- |
| Switch  Case11+  Break  Case21+  Break  Break  Break  Case3+3    default+ |



# Looping

A loop is used to repeatedly execute a block of statements until false returns a given condition. There are 3 types of loop

* While loop

The while loop loops as long as a defined condition is true through a block of code.

Syntax

While(condition)

{

Statement;

}

|  |
| --- |
| condition  false  true  Statement |

* Do while loop

A do ... while a loop is comparable to a while loop, except that at least one time is guaranteed to be executed.

|  |
| --- |
| Statement  false  true  condition  true |

Syntax

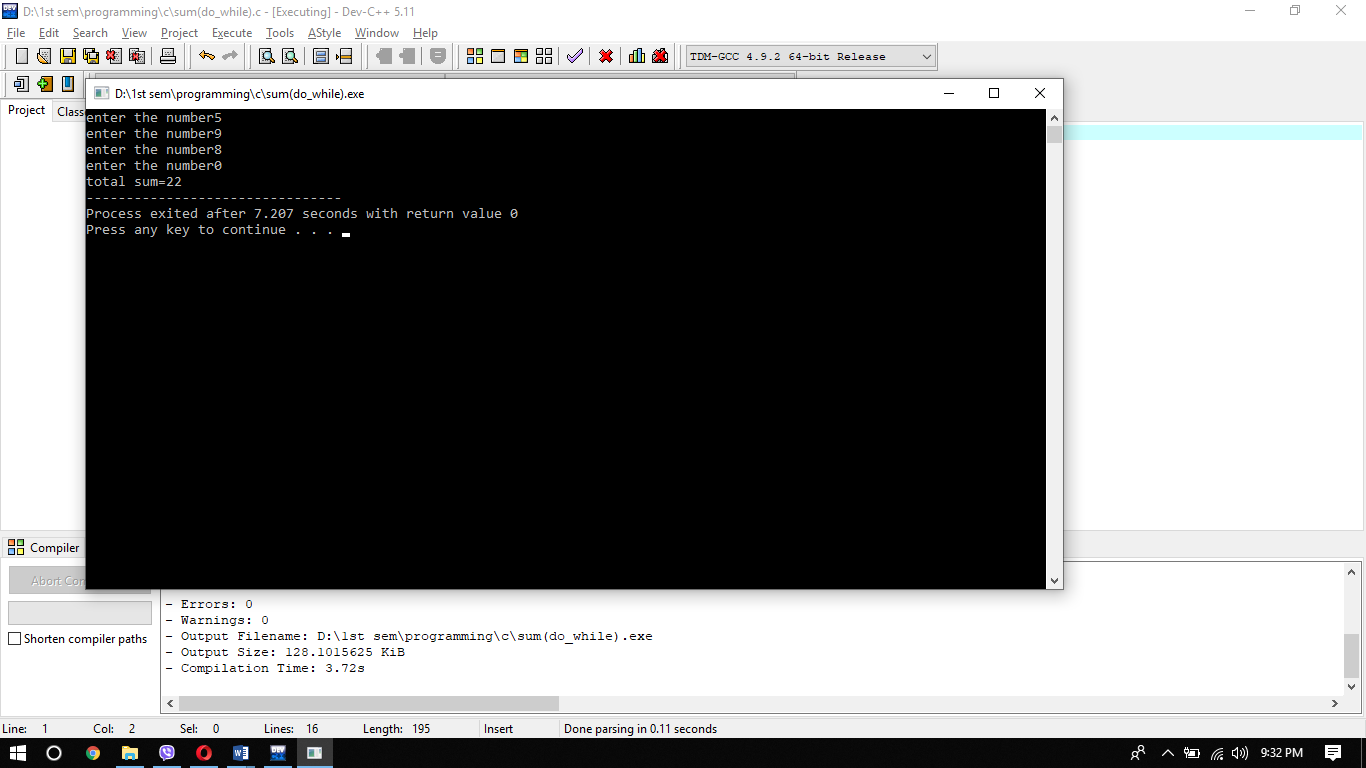
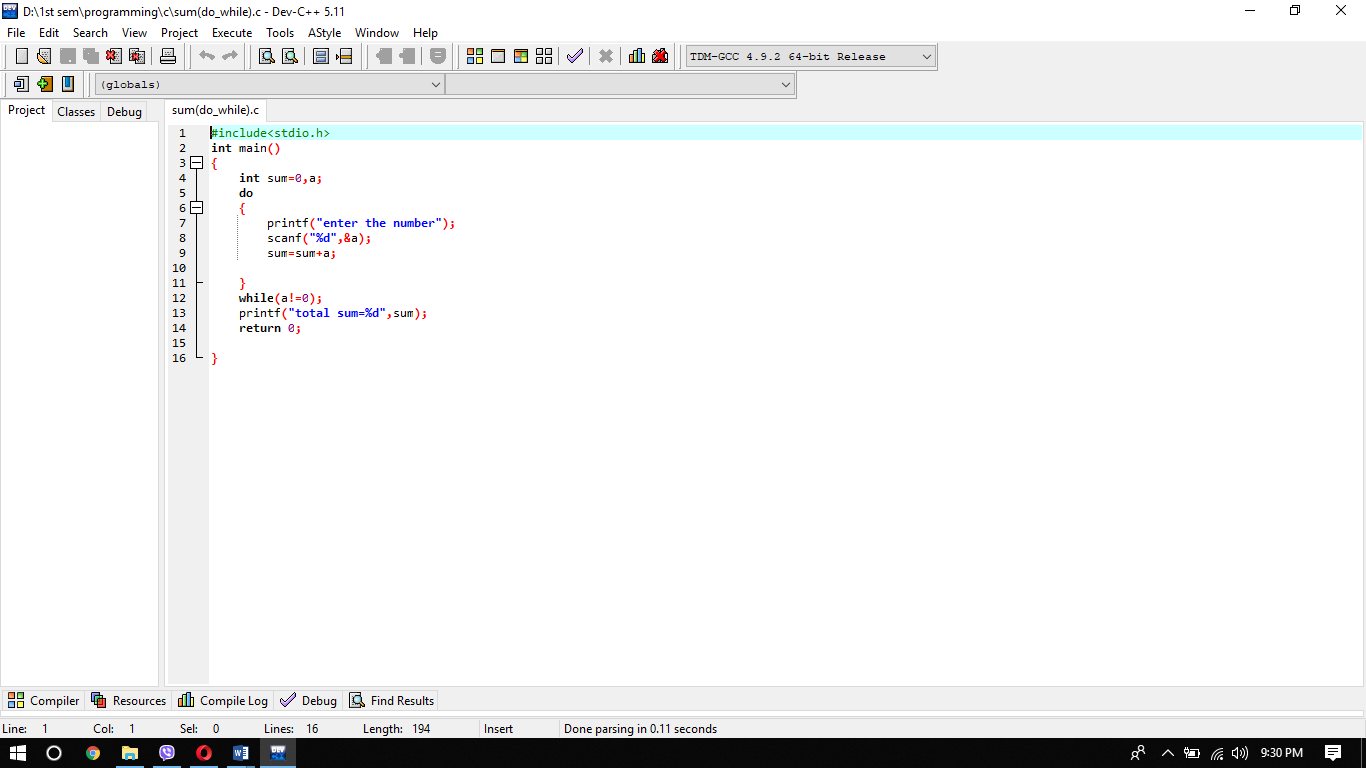
Do

{

Statement;

}

While (condition);



* For loop

The for loop is used multiple times to iterate the statements or a portion of the program. It is often used to cross the constructions of information such as the array and the linked list.

|  |
| --- |
| condition  Initialization  true  Statement  Increment decrement  false |

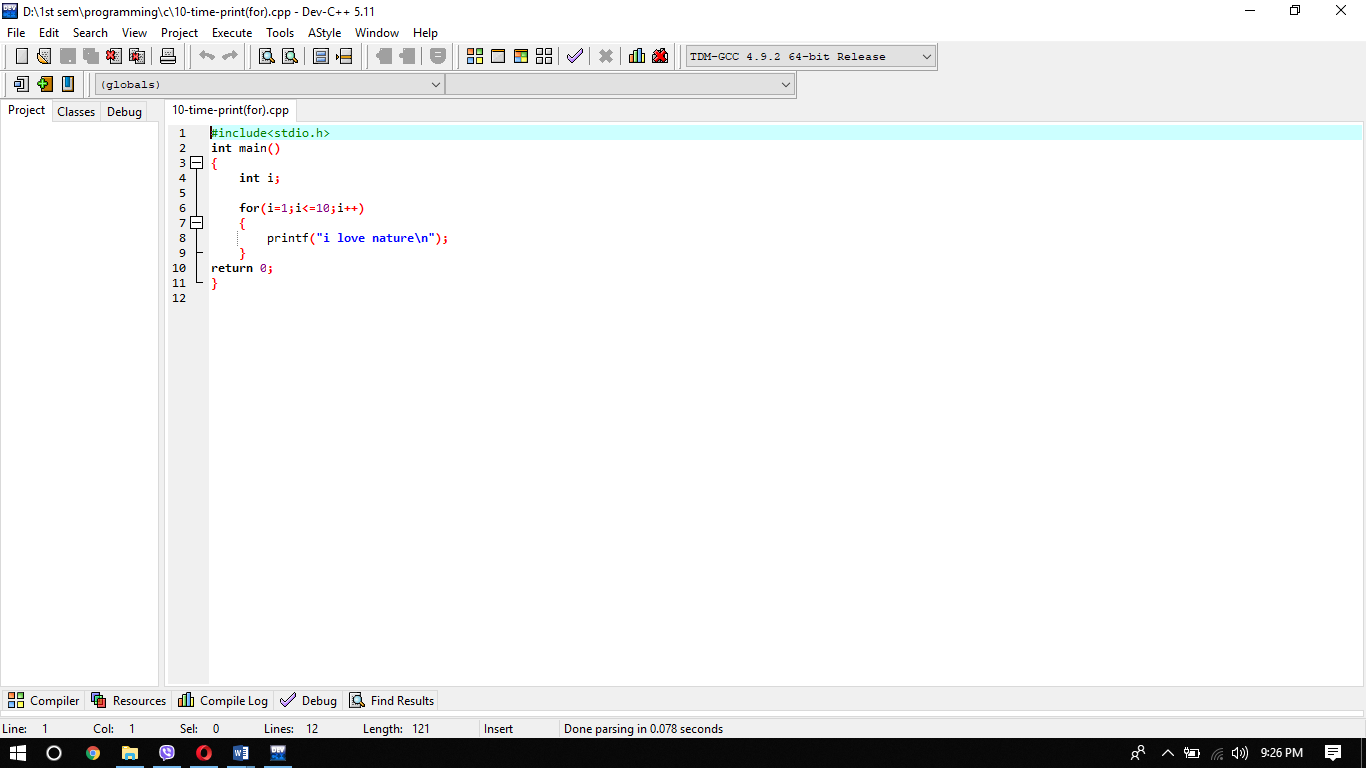
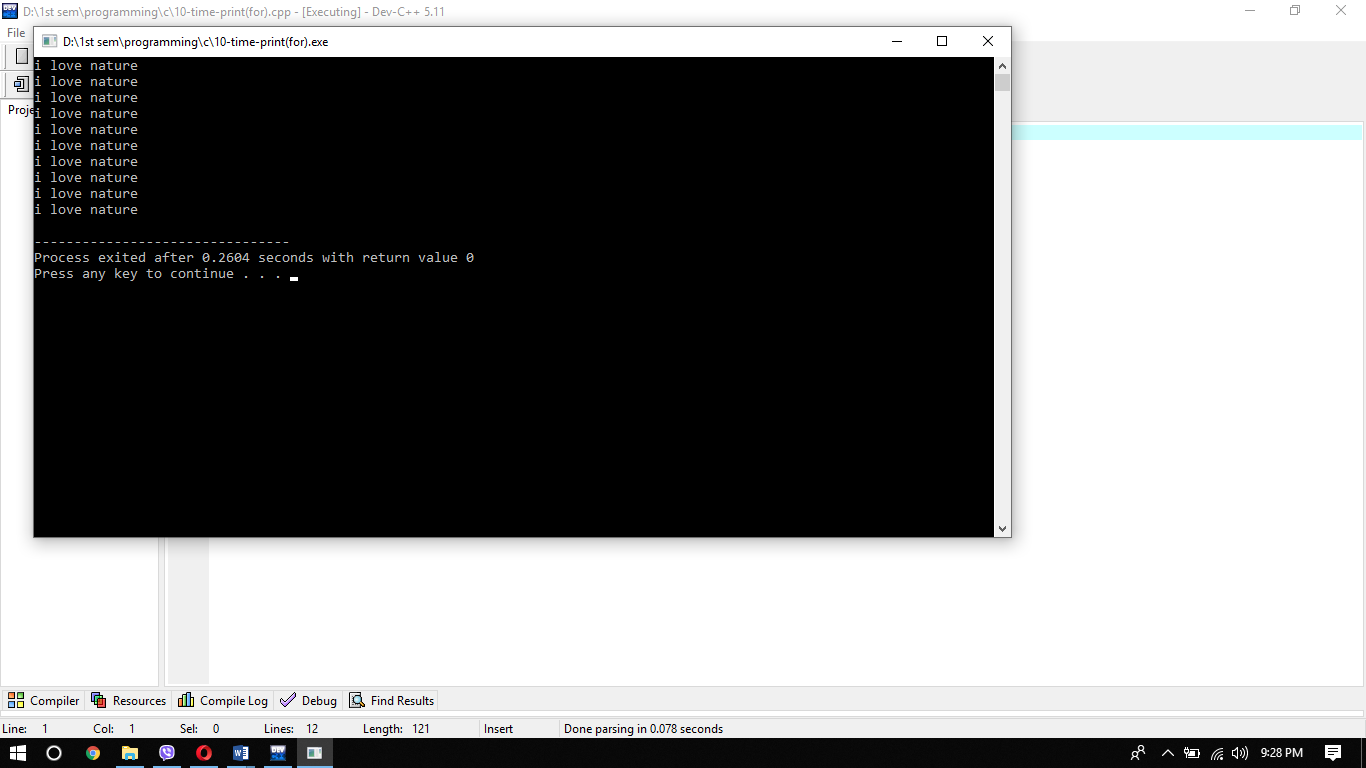
Syntax

For (expression1;expression2;expression3)

{

Statement

}



# Array

An array is defined as collecting similar data items stored in adjacent memory locations. Arrays are the derived data type that can store the primitive data type (int, char, float, etc. and derived data type (pointer, structure) in C programming language. The array is the simplest data structure where by using its index number each data element can be accessed randomly. Array index always start with 0.

Example of array;

Int a[4]; // integer array

Char a[4]; // character array

Types of arrays

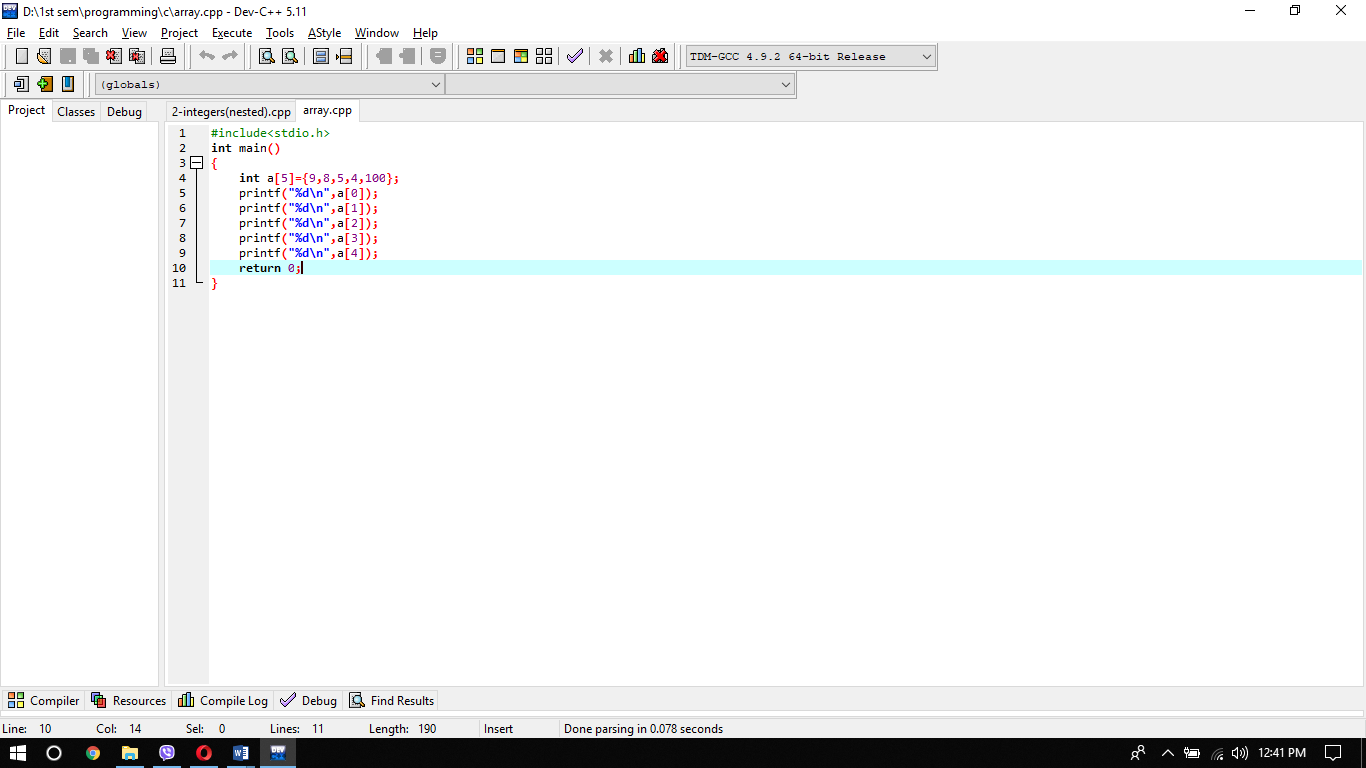
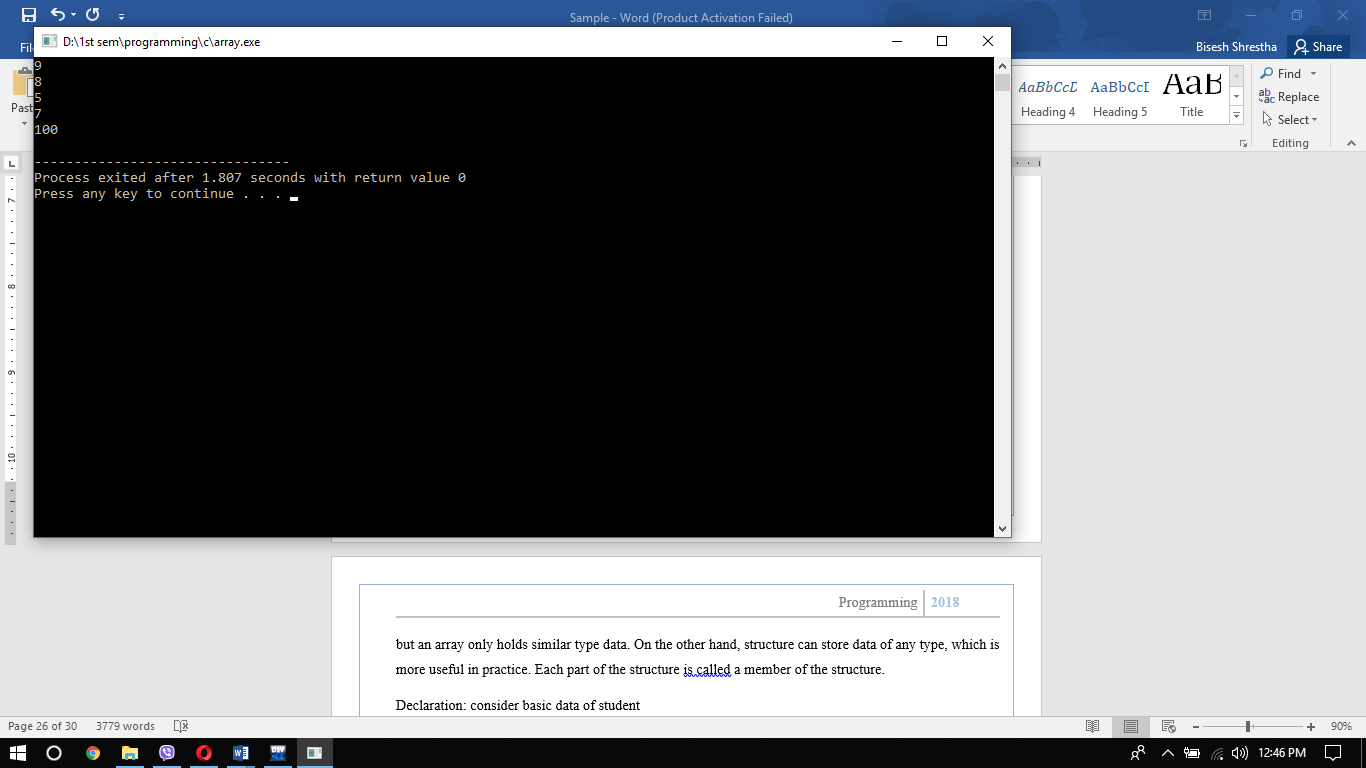
1. One-dimensional array: The variable allows one value to be stored at a time.

For e.g. roll no[5];

1. Two-dimensional array: Multiple parts of data are tracked in linear order, a one-dimensional list.

For e.g.: int[2][2];

1. Multi-dimensional array: Array with more than one dimension. It allow programmer to create arrays of arrays. For eg : float [2][3][4];



# Structure

Structure is a user-defined C-language datatype that enables us to combine information of various kinds. Structure helps to build a more significant, complicated information form. It is a bit similar to an Array, but an array only holds similar type data. On the other hand, structure can store data of any type, which is more useful in practice. Each part of the structure is called a member of the structure.

Declaration: consider basic data of student

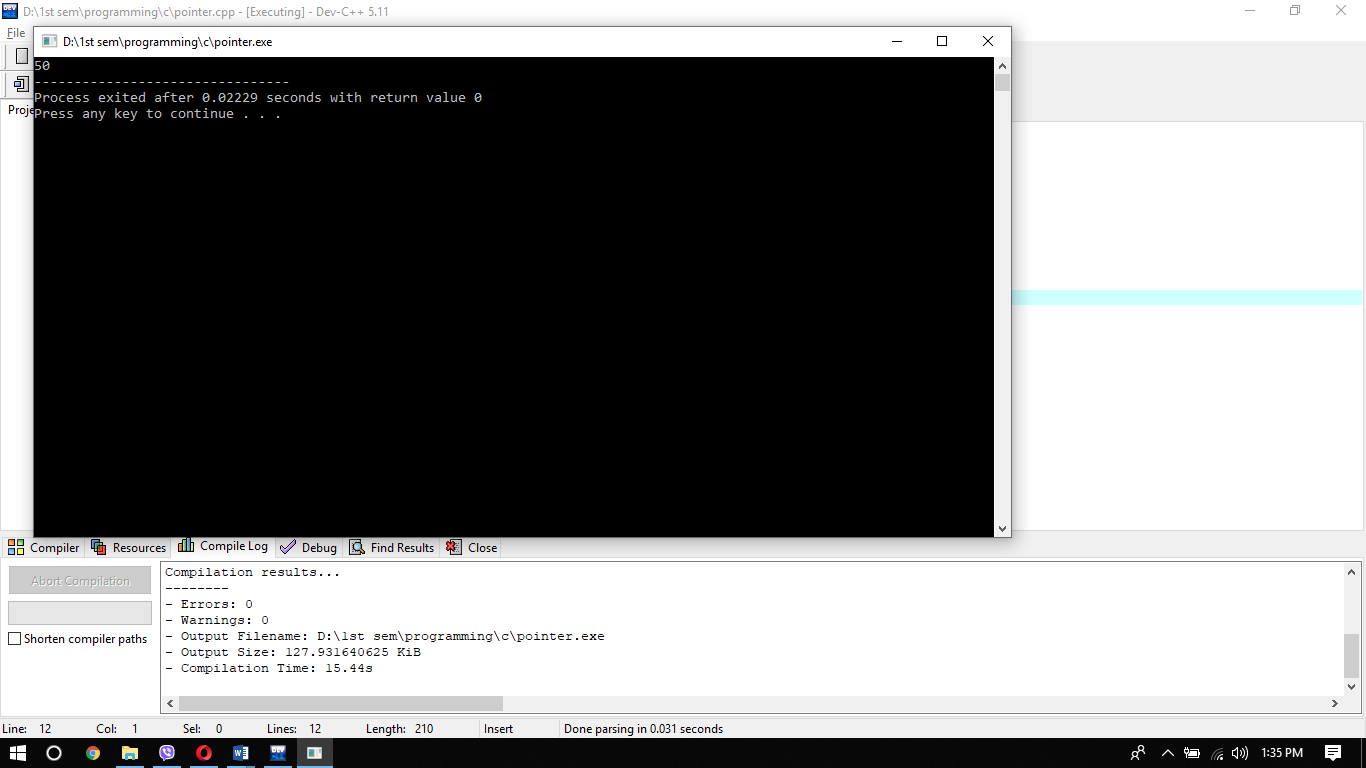
Initialization: values are listed inside braces

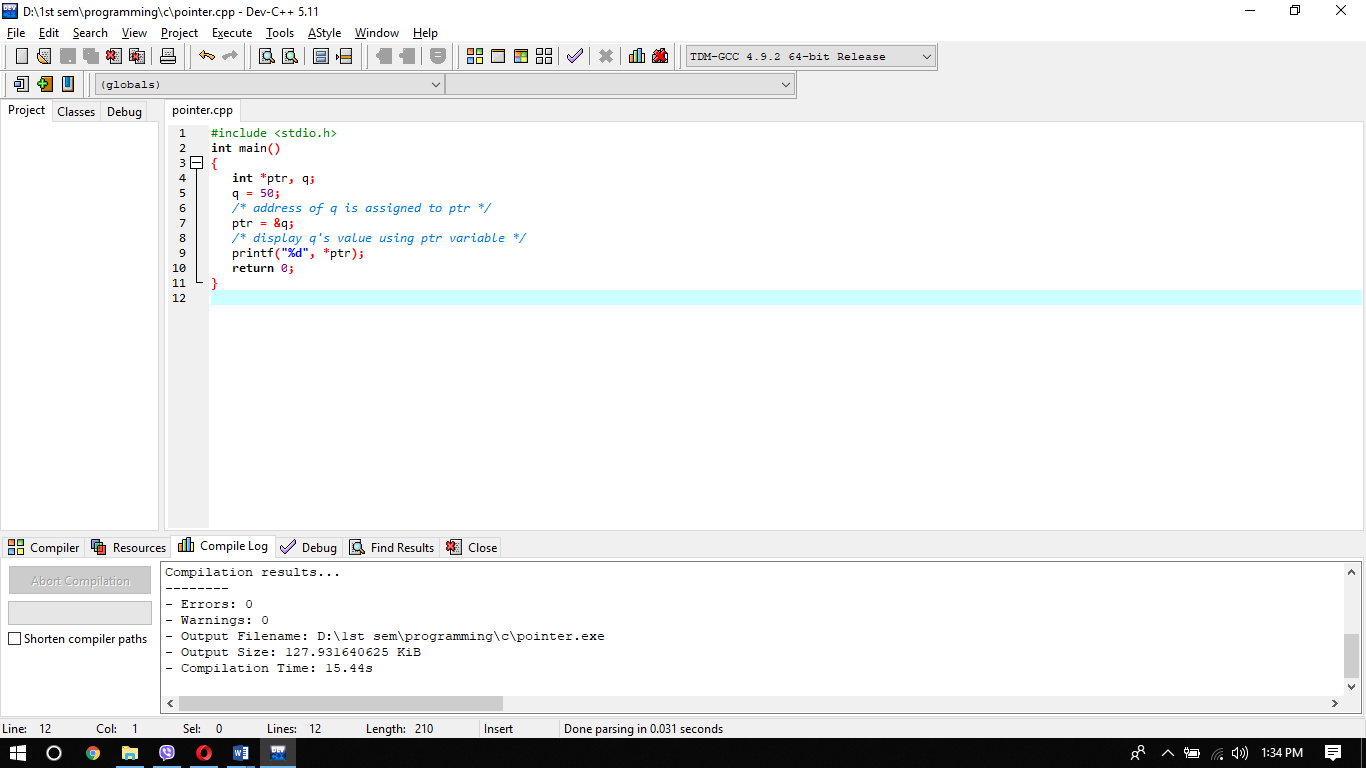
# Pointer

The C language pointer is a variable that stores another variable's address. This variable may include type int, char, array, function, or any other pointer. Size of a pointer depends on the architecture. It helps in improving your program’s efficiency and even allow to handle unlimited amount of data. For e.g : int\*a; character\*a; where ,\* is used to denote that “a” is a pointer

Key point to remember

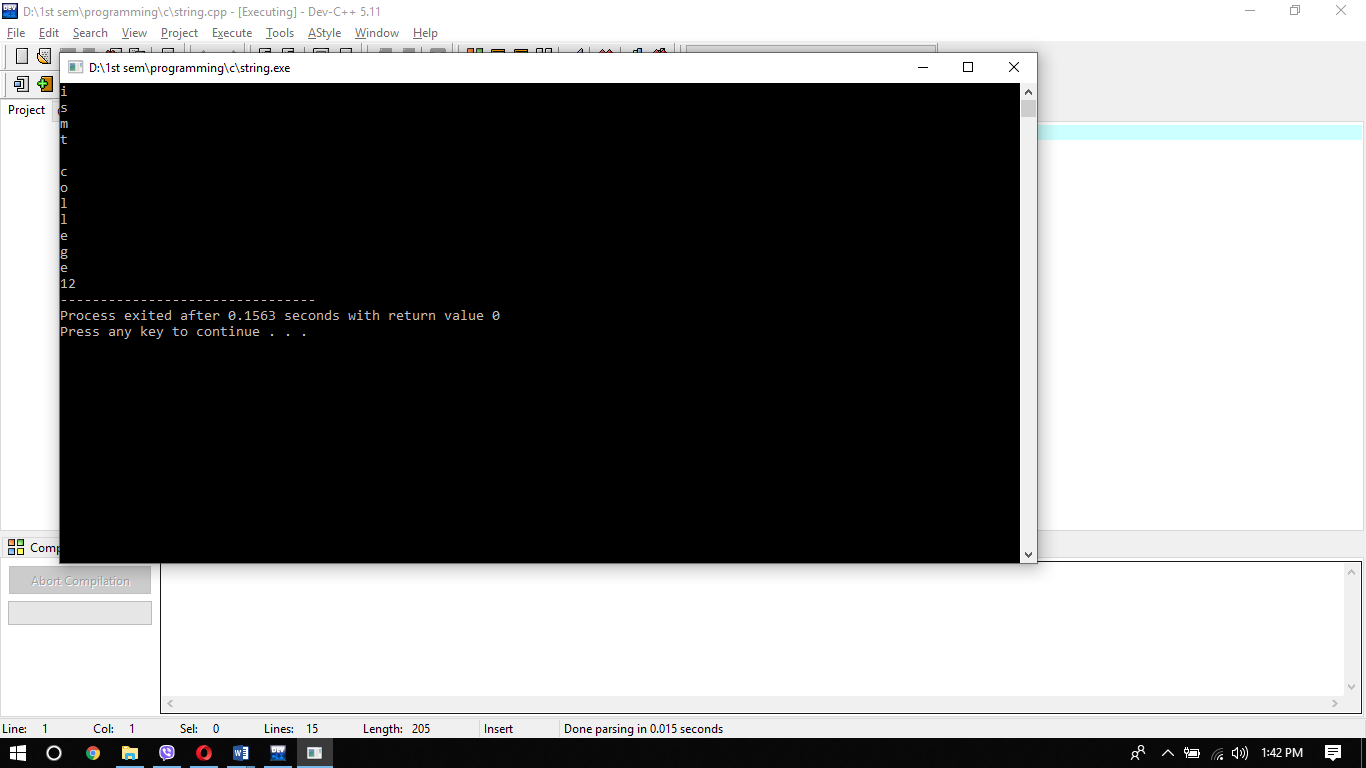
* The normal variable stores the value while the variable pointer stores the variable's address.
* C pointer is always initialized to zero, i.e. int \* p = null.
* & Symbol is used to obtain the variable's address.
* \* This symbol is used to get the value of the pointing variable.
* If a C-pointer is allocated to NULL, it implies nothing
* To understand how many elements are available between these two pointers, two pointers can be subtracted.

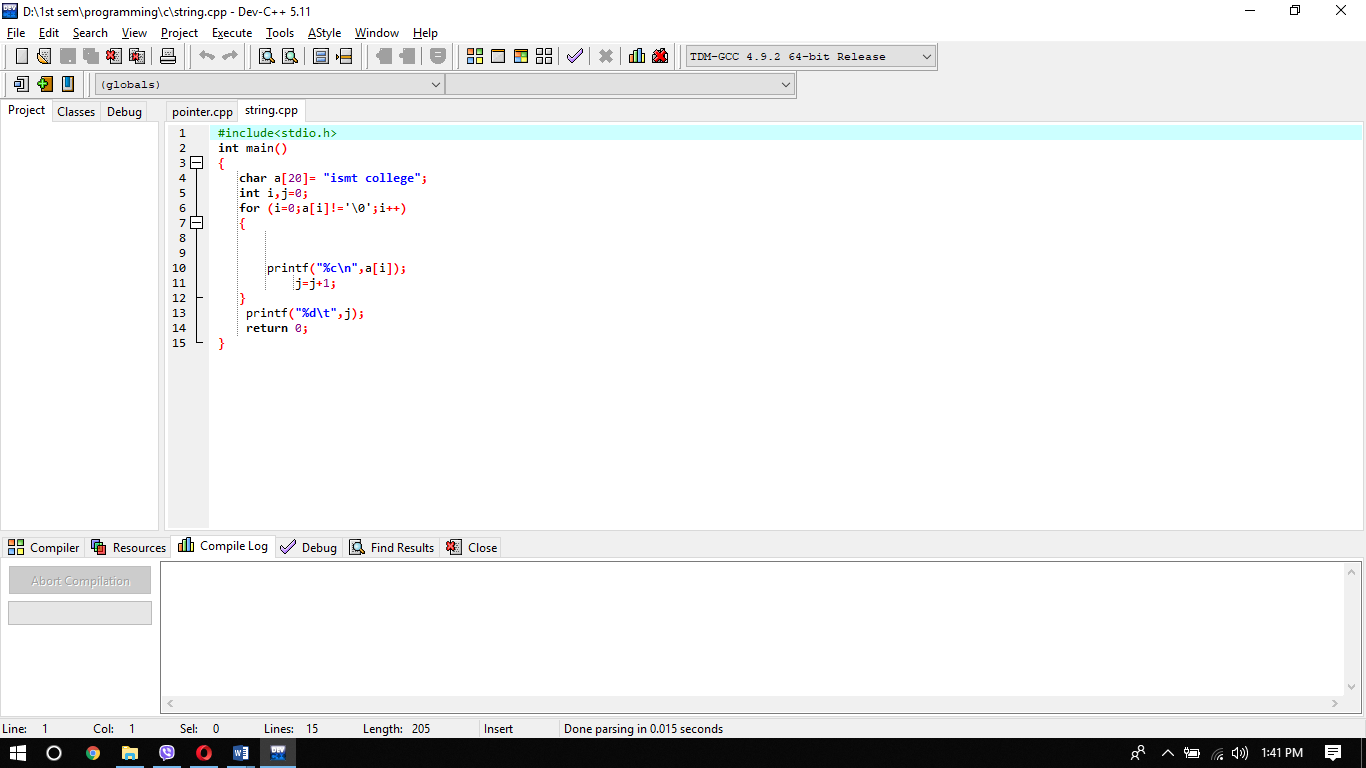




# String

A string is a simply an array of characters which is terminated by a null character which shows the end of the string. It reflects the character sequence. Comparison, concatenation, transformation etc. are the operations that can be conducted on strings. The one-dimensional or single-dimensional character array is known as Strings.





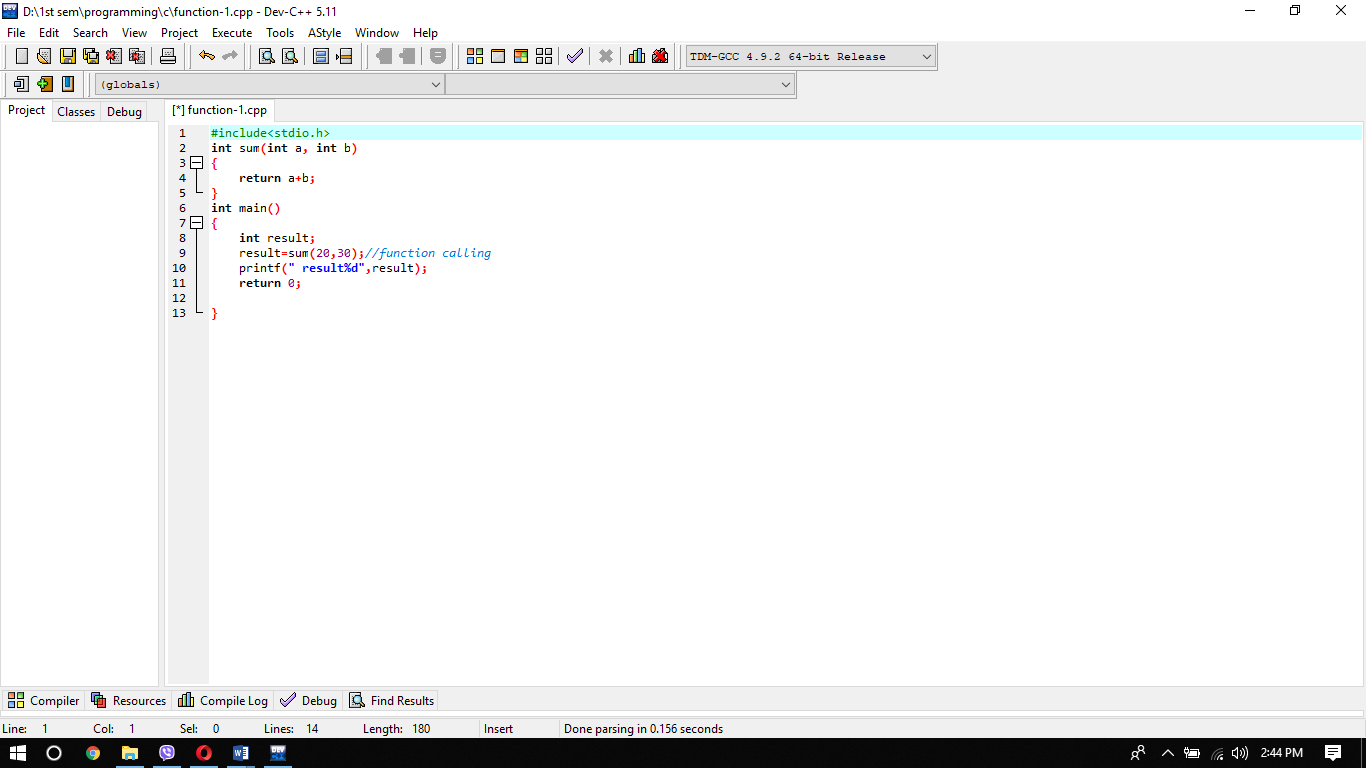
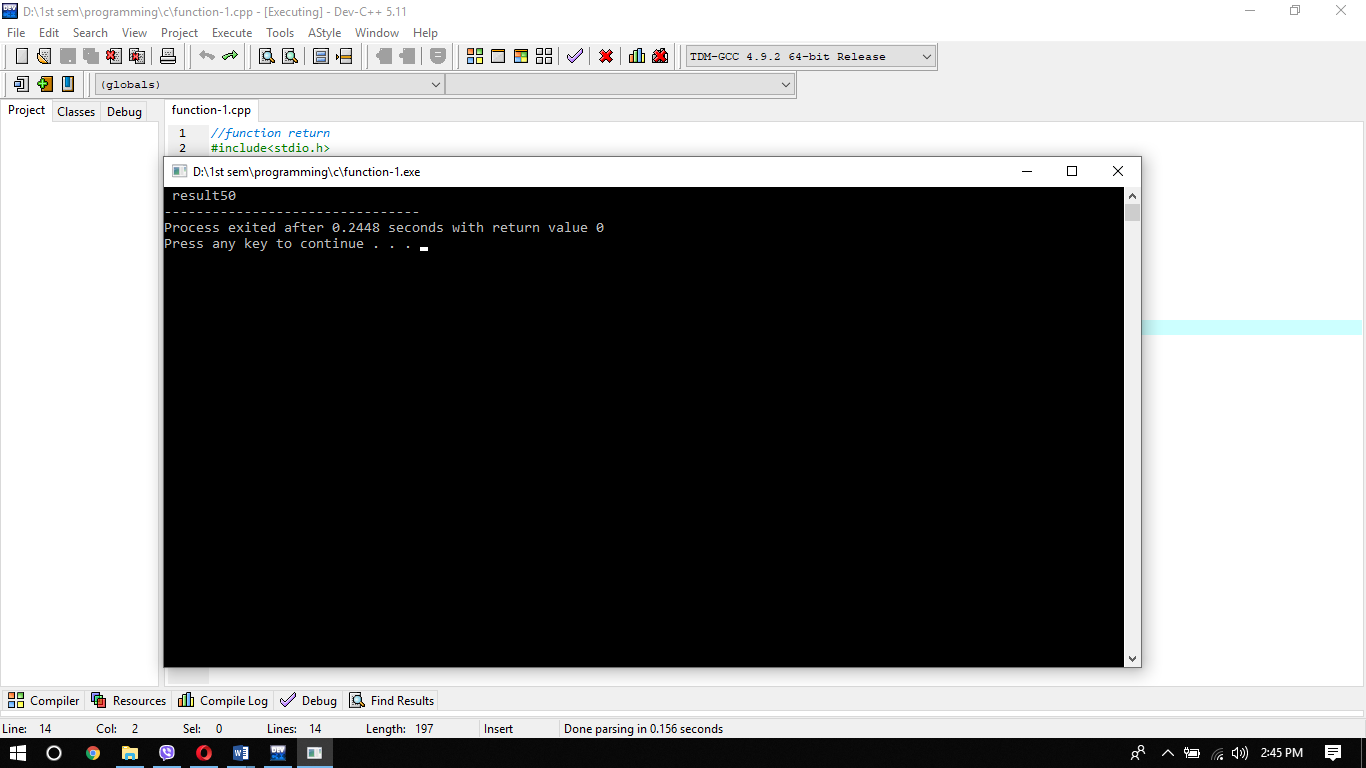
There are some string functions

1. Strncpy( ) : copies str2 into str1
2. Strlen ( ) : gives the length of str1
3. Strchr( ) : returns pointer to first occurrence of char in str1.
4. Strdup( ) : duplicates the strings
5. Strlwr( ) : converts string to lowercase

# Function

Function is a collection of statements or statements that work together to execute the task. Each program is comprised of a single main ( ) feature. The functions take inputs, execute the task, and provide output on those inputs. It provides a program's standard and we use functions while producing a program that make it simpler to understand, edit, and verify mistakes, etc. there are two types of function:

1. Library function: These functions are the built-in features that a programmer who writes C++ program provides as it saves the programmer's time. Programmers can immediately bring up the functions and do not need to rewrite them. strcpy( ), strcat( ), sqrt( ), etc are some example of library functiom
2. User-defined function: those functions, which are defined by the user. It enables extra tasks to be performed in addition to the built-in function. add( ), multiply( ), divide( ) etc. are some example of user-defined function.



# File Handling

C file handling enables us, through our C program, to generate, update, read and delete records stored on the local file system.

Functions for file handling are :

fopen() create a new file or open an existing file

fclose() closes a file

getc() reads a character from a file

putc() writes a character to a file

fscanf() reads a set of data from a file

fprintf() writes a set of data to a file

getw() reads a integer from a file

putw() writes a integer to a file

fseek() set the position to desire point

ftell() gives current position in the file

rewind() set the position to the beginning point

# Object oriented programming language

Object-oriented programming system (OOPs) is a programming paradigm based on an "object" concept containing data and methods. Object-oriented programming's main aim is to improve program flexibility and sustainability. Object-oriented programming brings together information and make it simpler to understand how a program operates in a single object with its behavior (methods).

Advantages of OOPL

Since everything is treated as objects, OOP allows us to model a real-world concept. OOPL offers a definite modular framework for programs that enables the definition of abstract data types in which details of execution are concealed. It makes maintenance of software simpler. It also provides a good framework for code libraries where the programmer can easily adapt and modify the supplied software components.

Disadvantages of OOPL

Design a program in OOPL is quite difficult because everything is treated in OOP as objects, programmers need proper skills such as design skills, programming skills, object thinking, etc. The size of the OOP programs is greater than the procedural approach Because of the bigger size, this implies more instruction to be performed, resulting in slower program execution.

Features of OOPL.

1. Data encapsulation

Encapsulation is a process in which data members and functions are combined in a single unit called a class. This is to avoid direct access to the information, providing access to it through the class's features. Data abstraction refers to providing only important data to the outside world and hiding its background details, i.e. representing the data required in the program without the details being presented.

1. Class and object

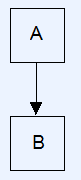
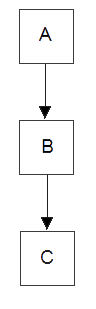
Class is a user-defined data type that contains its own data members and member functions that can be accessed and used by creating a class instance. Class is a blueprint to generate as many objects as you like.

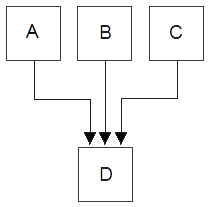
Object is a information set and its behavior(often referred to as methods). An object is an example of a class

1. Inheritance

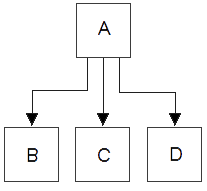
A class's ability to derive properties and features from another class is called inheritance. Inheritance is one of Object Oriented Programming's most significant features. Inheritance enables us to identify a class as a different class, making it simpler to generate and retain an application. This also offers an chance to reuse the functionality of the software and quick execution time.

There are 6 type of inheritance

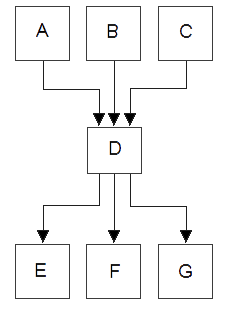
1. Single Inheritance: Single inheritance is called a derived class with only one base class.
2. Multilevel Inheritance: A derived class with one base class is called multilevel inheritance, and that base class is a derived class of another.
3. Multiple Inheritance: Multiple inheritance is called a derived class with multiple base class.



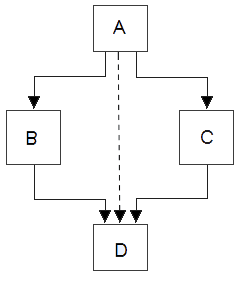
1. Hierarchical Inheritance: Multiple derived classes are called hierarchical inheritance with the same base class.



1. Hybrid Inheritance: Combination of multiple and hierarchical inheritance is called hybrid inheritance.



1. Multipath Inheritance: A derived class with two base classes is called multipath inheritance and these two base classes have one common base class.



1. Polymorphism

The term polymorphism means multiple forms. Polymorphism means that a call to a member function causes a different function to be performed depending on the type of object invoking the function. Polymorphism typically happens when class hierarchy exists and they are inheritance-related.

1. Abstraction

OOP language objects provide an abstraction that hides information of the inner execution. Abstraction selects information from a bigger pool to only display the object's appropriate information. It enables to decrease the complexity and effort of programming.

# Event driven programming language

Event-driven programming is a programming paradigm in which events such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs or threads are determined to flow the program. It is the dominant paradigm for graphical user interfaces and other apps. You can write event-driven programs in any programming language,

## Features

* Service oriented: Service oriented is a programming paradigm used to write services-based programs.
* Time driven: It's code running on a time trigger, it could be a piece of code running at a particular moment that could be once a week or when a program is running, that implies it's a pre-set job.
* Event holder: Event handler is a function or method that occurs when an event occurs and is executed in response to an event that occurs,
* Trigger function: Trigger functions decide which code is running when a particular event occurs, they are used to select when event handlers are running for the event that occurs, most applications have a trigger function for each event that is likely to occur.

## Advantage and Disadvantage

One of the greatest benefits of using event driven programming is the language it utilizes is called Visual Basic.net, this programming code helps you as you code and provides you suggestions on what to give you, and it is also written in an understandable (English) language. The primary benefit of programming driven by events is that it is simpler than other programming kinds. It is primarily based on the user / person triggering the action with event-driven programming. Event driven programming is flexible which enables the coder to be flexible in the order in which the code is executed.

EDP's primary disadvantage as an event loop in an EDP is that it runs continuously in the background of the implementation and hog loads of system resources. EDP also utilizes GUI to provide the interface and would require a greater spec machine. But as most machines can run EDP apps, this is no longer the case. It is also very hard to translate into another language from EDP

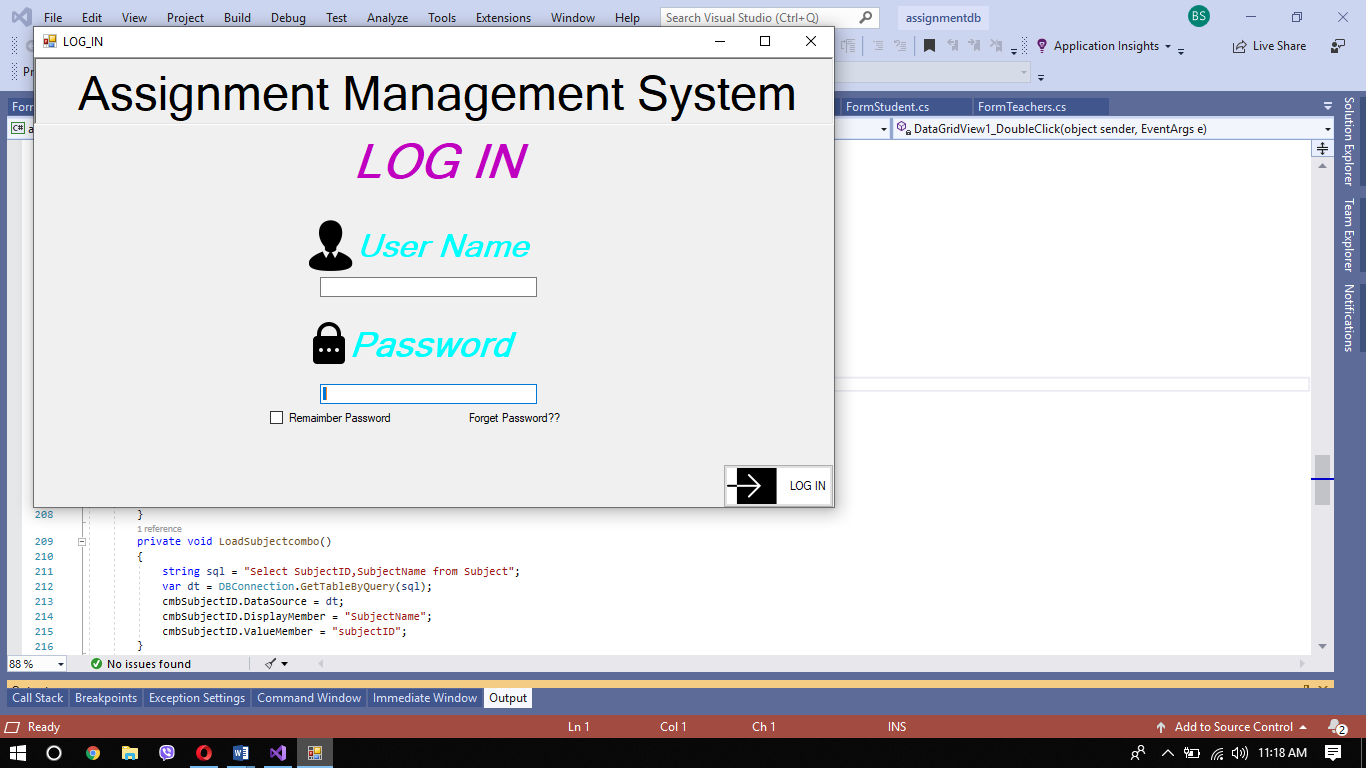
# Relationship between PPL, OOPL, event driven programming language.

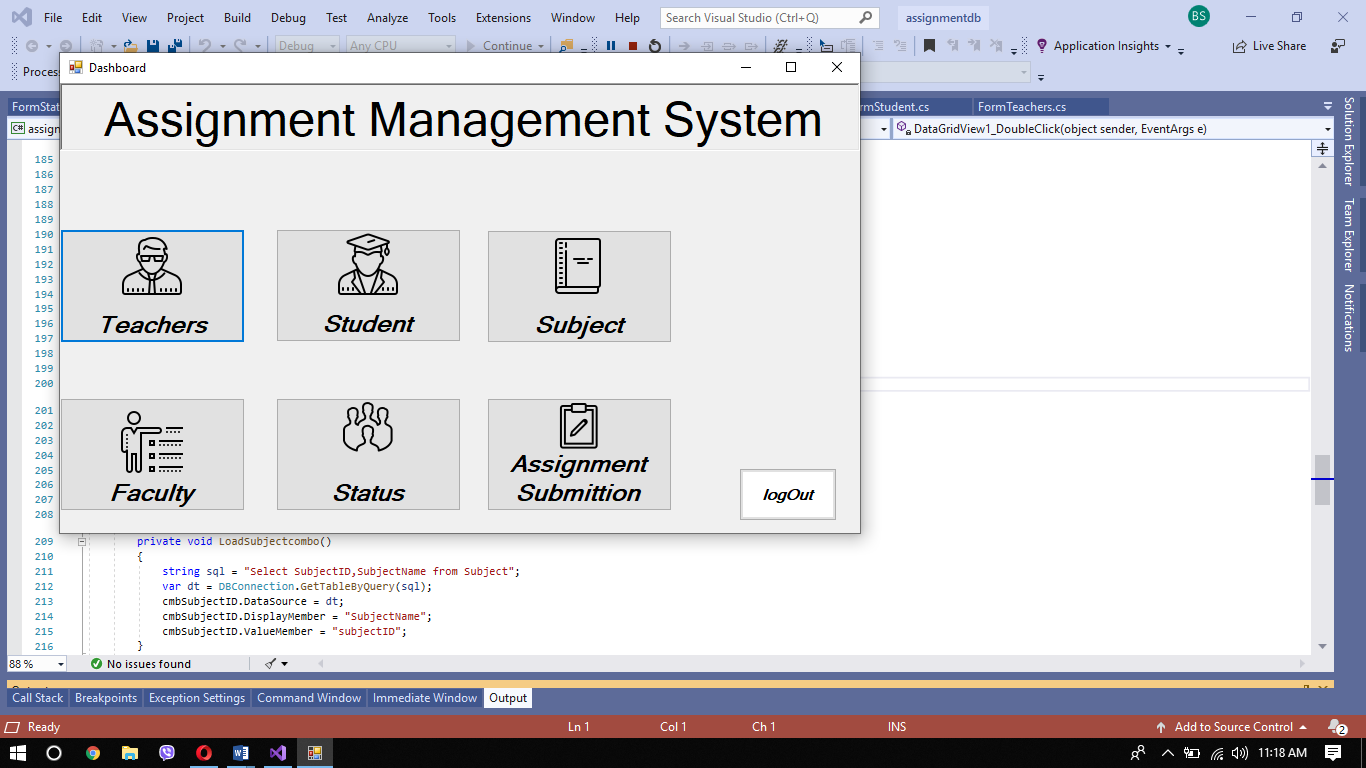
Procedural programming implies that as a sequence of steps you define a program and its sub-programs. Object-oriented programming is a way to organize code around encapsulation, inheritance, substitution, interface programming, and so on. Event-based programming is about writing procedures for event handling and having the underlying system provide the core event loop.

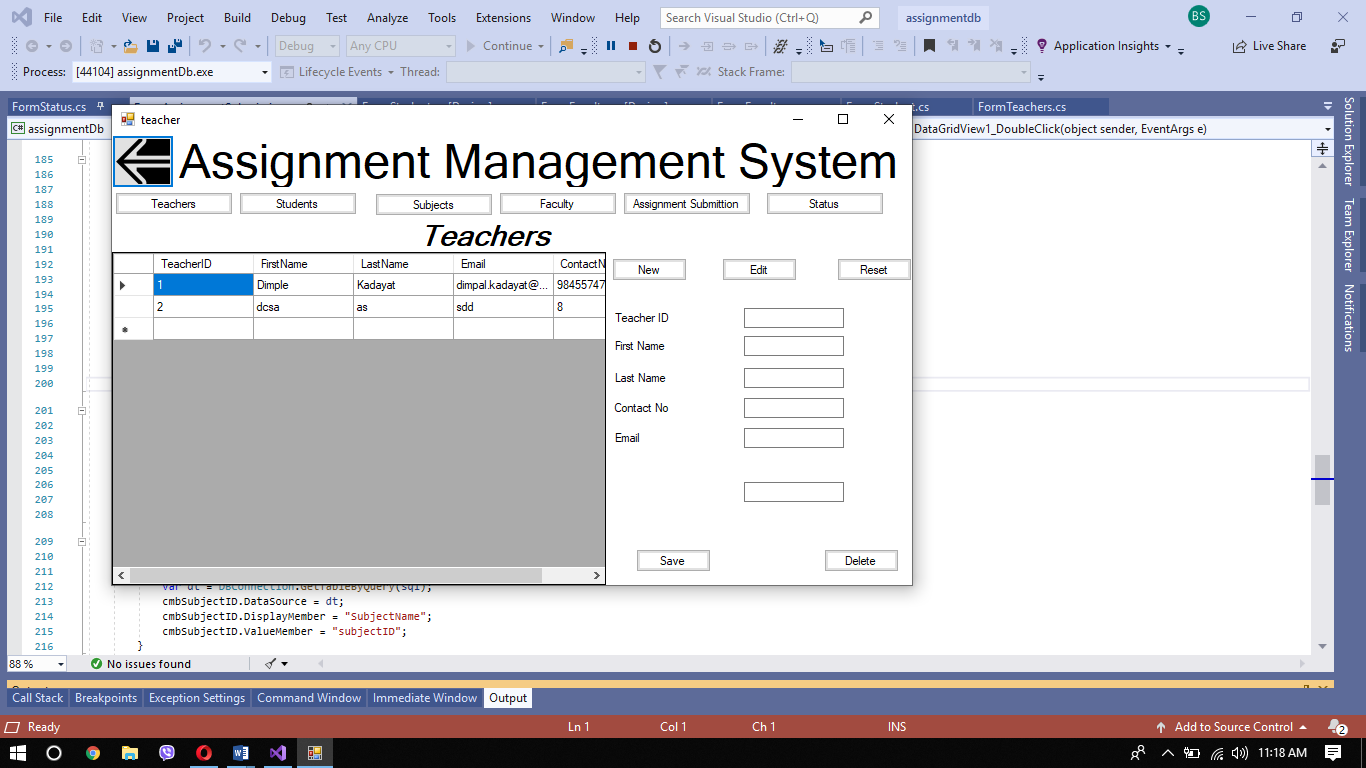
|  |  |  |
| --- | --- | --- |
| PPL | OOPL | Event-driven programming |
| It is based on how the task is to be done | It is data security based, only objects are allowed to access a class's entities. | It is based on the event-determined flow of program execution |
| The programmes, i.e. functions, are split into units. | It divides the entire program into objects. | EDP can be written in any programming language |
| It follows the strategy from top-down. | It follows the strategy of the button-top. |
| C, VB, FORTRAN, Pascal. Etc are some example | C++, JAVA, VB.NET, C#.NET. are some example | Visual Studio, Visual C++, Java etc. are some example |

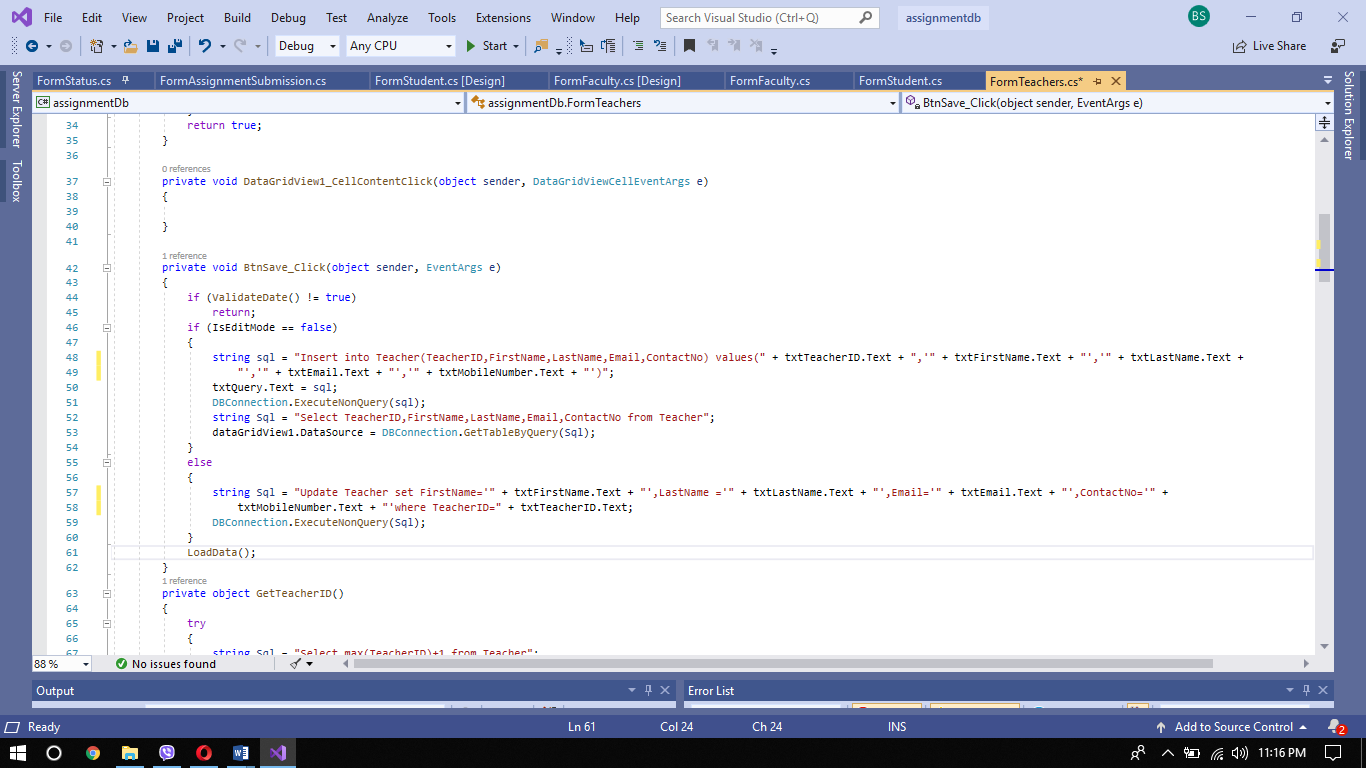
# Conclusion

# (P3)Implementation of Application

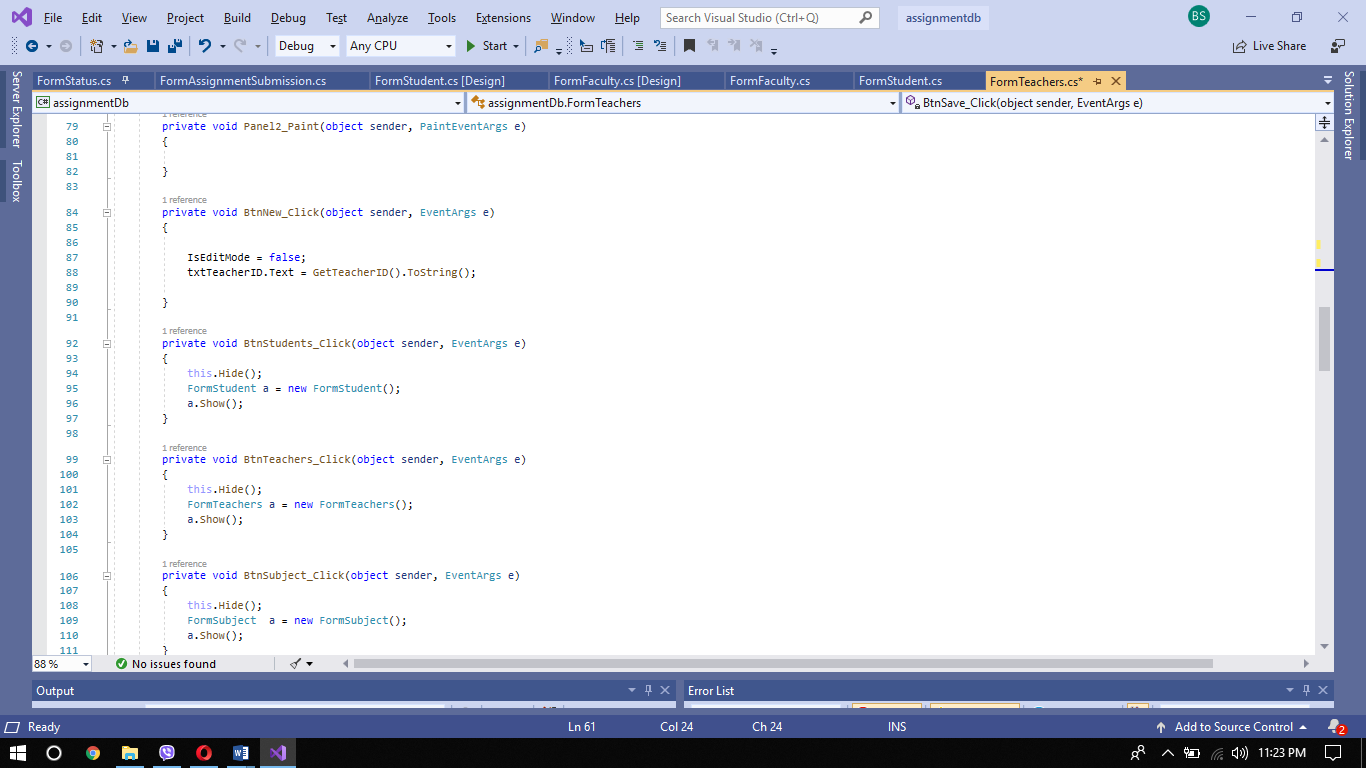




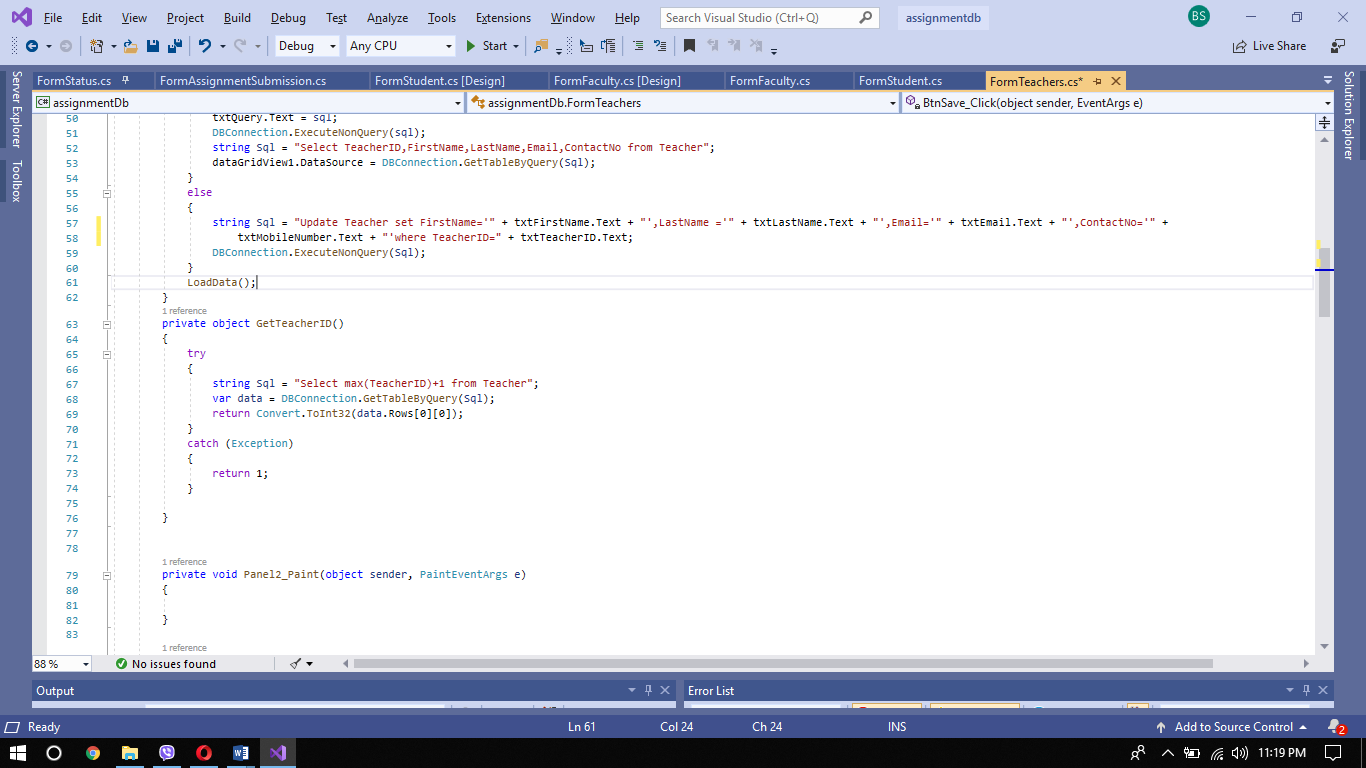


for save button I have used following codes to save given input in all forms

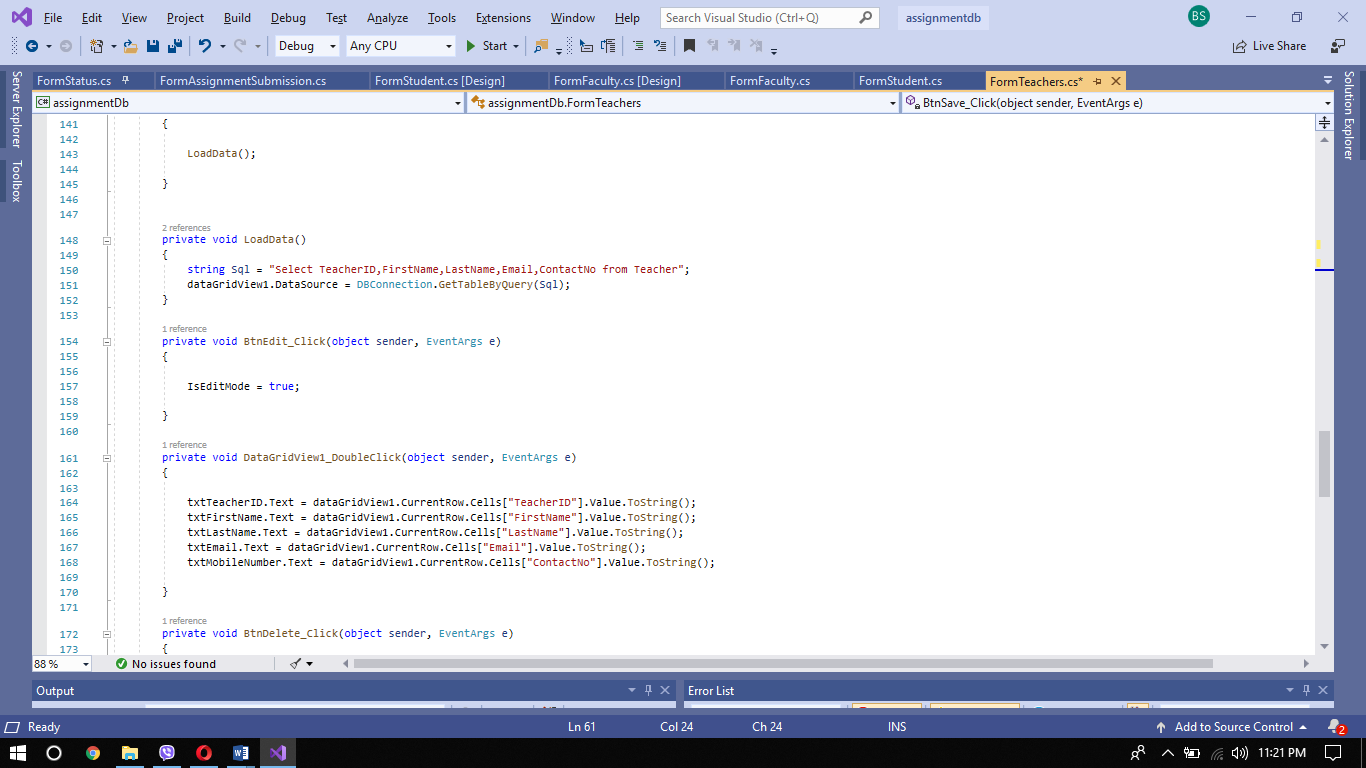
for new button I have used following codes



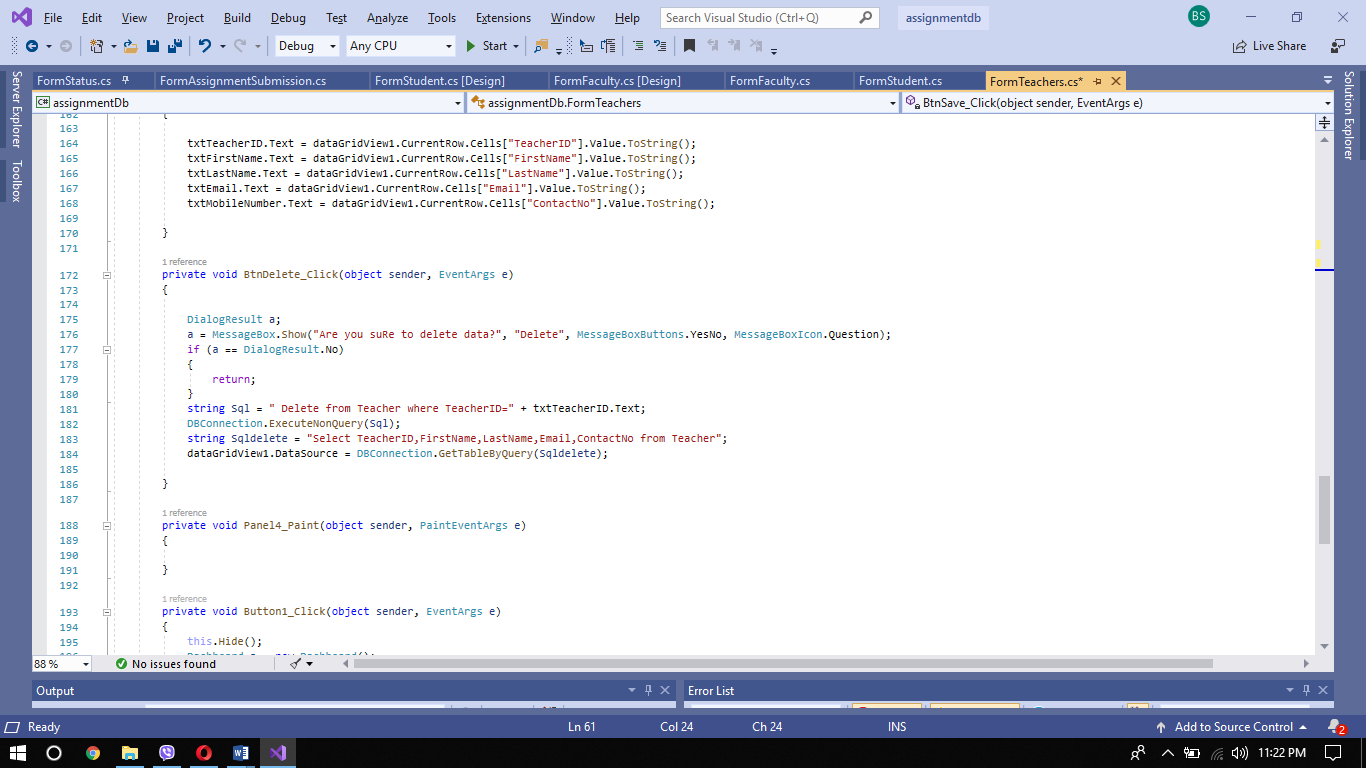
for increasing primaryID I have used following codes which auto increase ID number

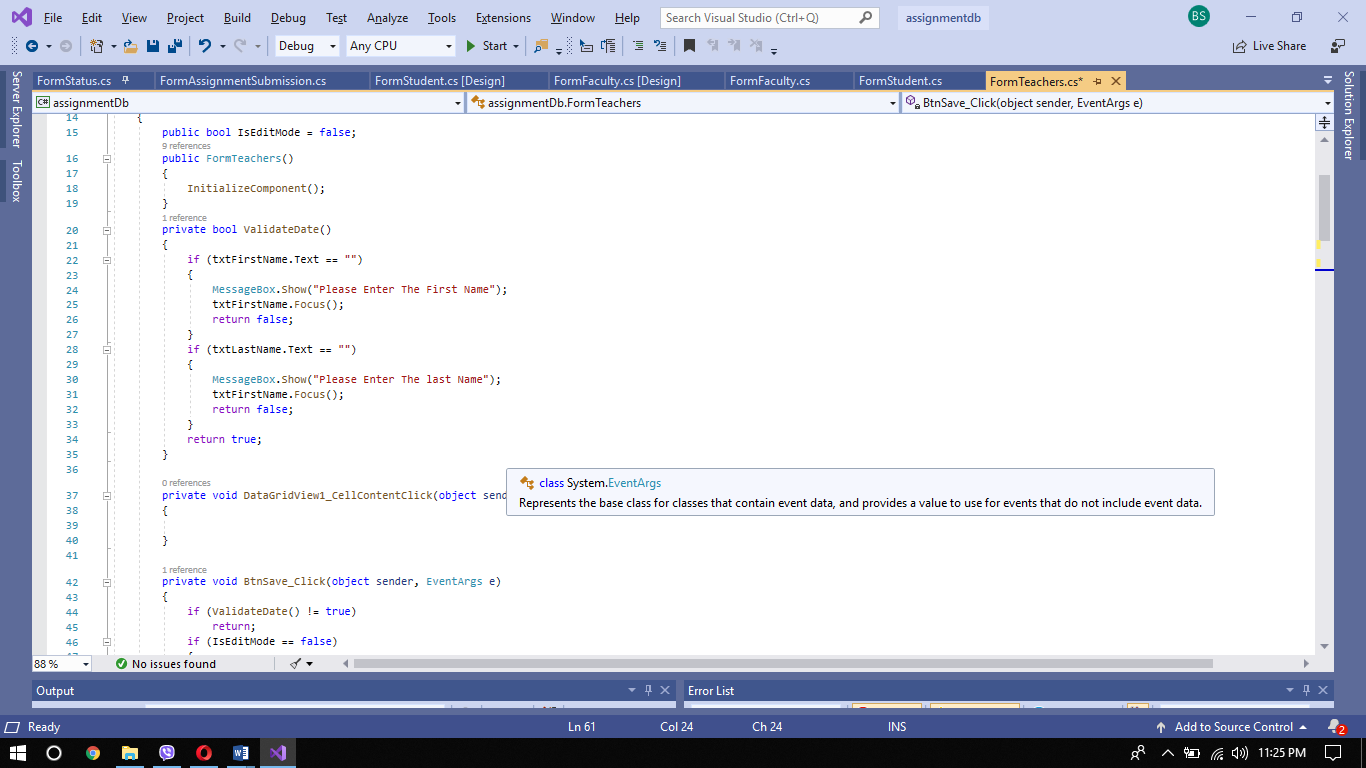


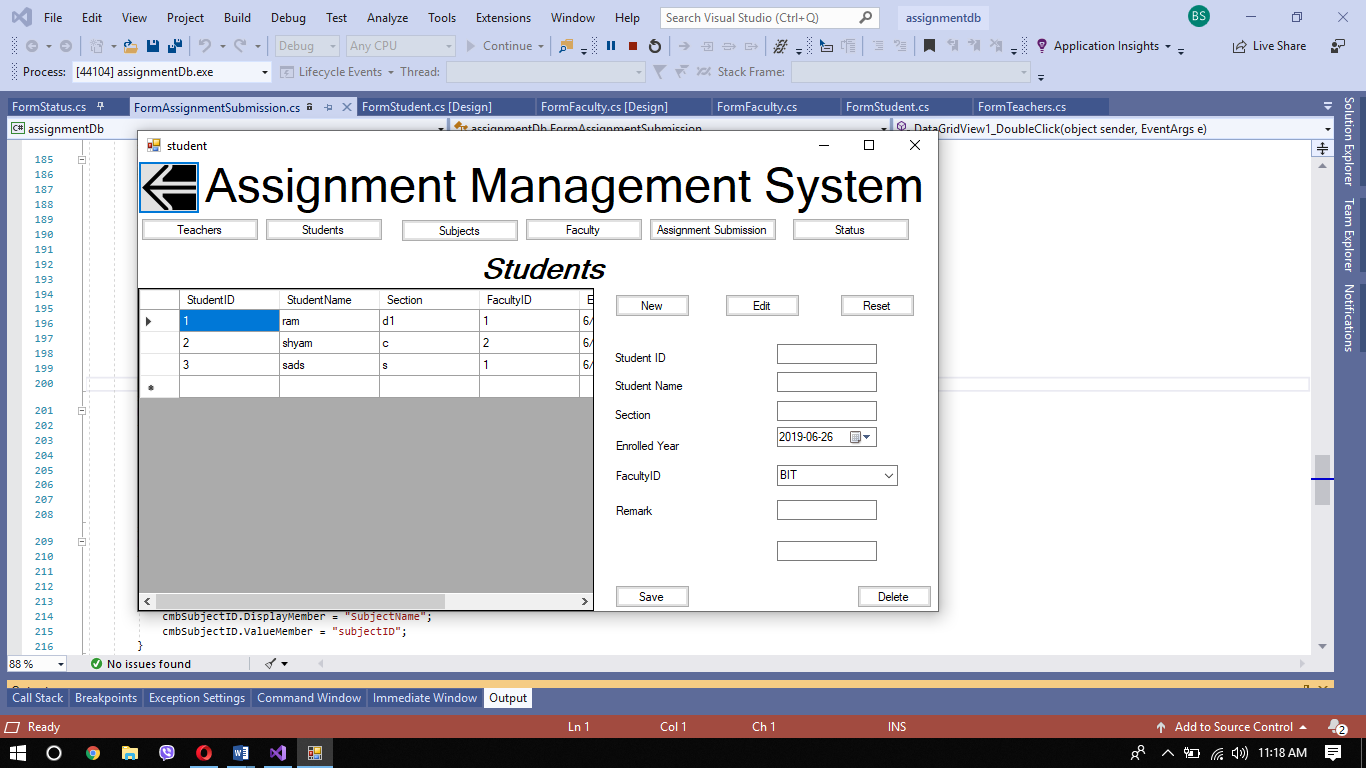
For loading data n for edit button I have used follwing codes



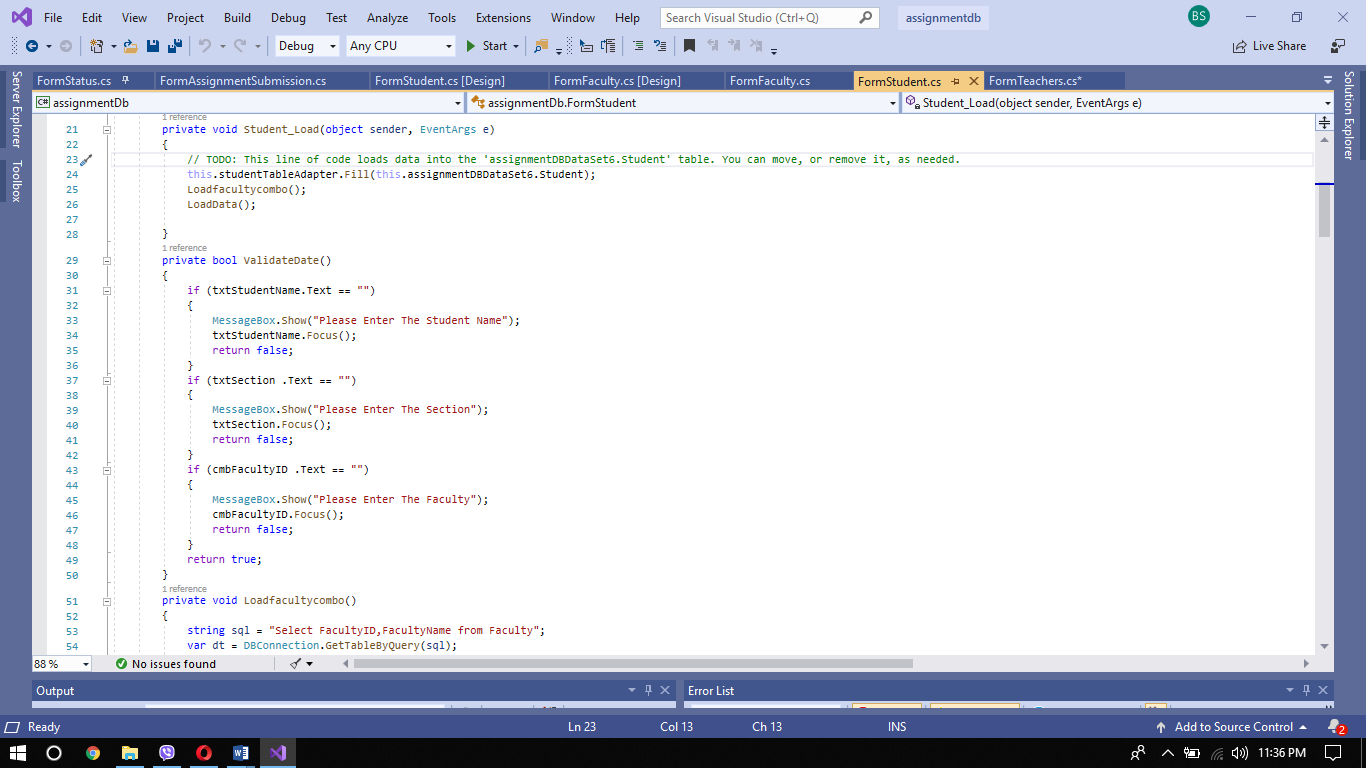
For delete button I have used following codes so that it can delete the selected data.

For inputing compulsary input I have used following codes so that if input is not given then it will give an error

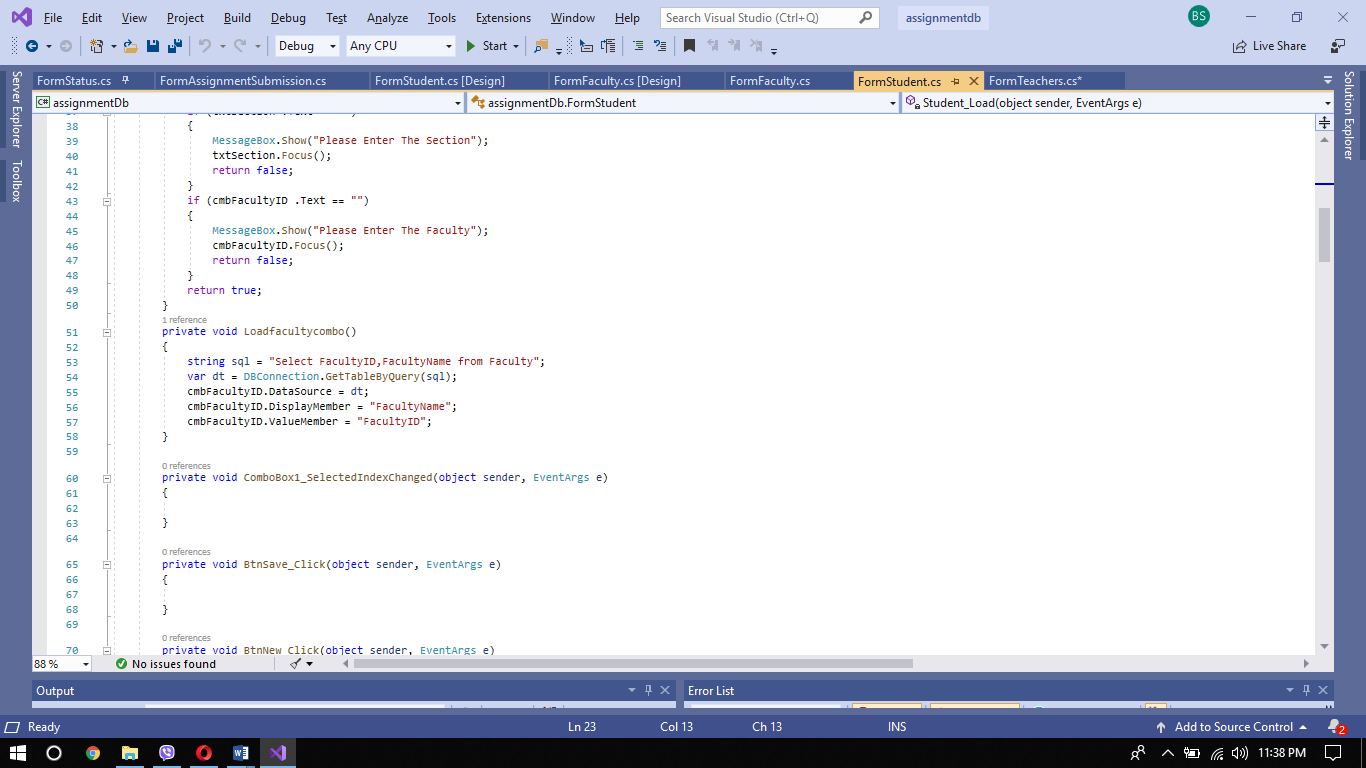


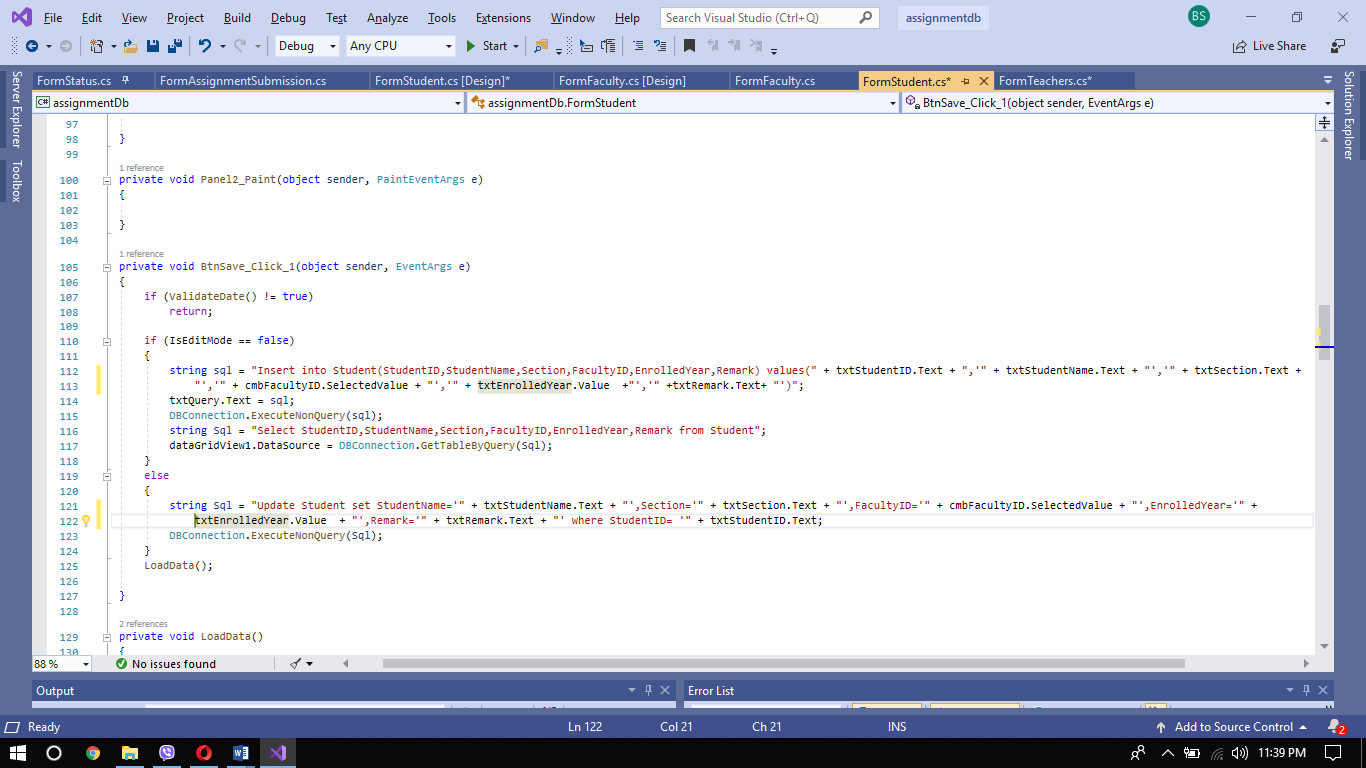


For taking compulasary inputs

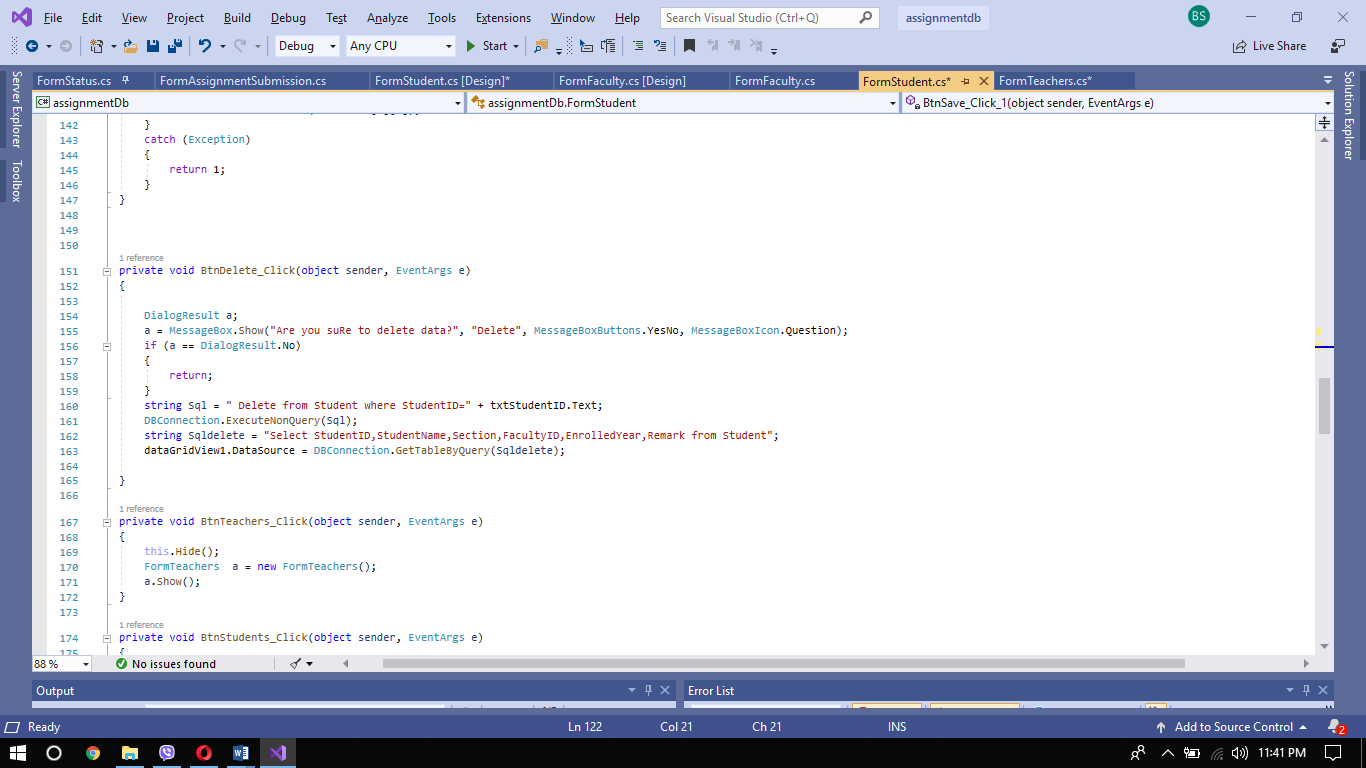


For loading faculty combo box

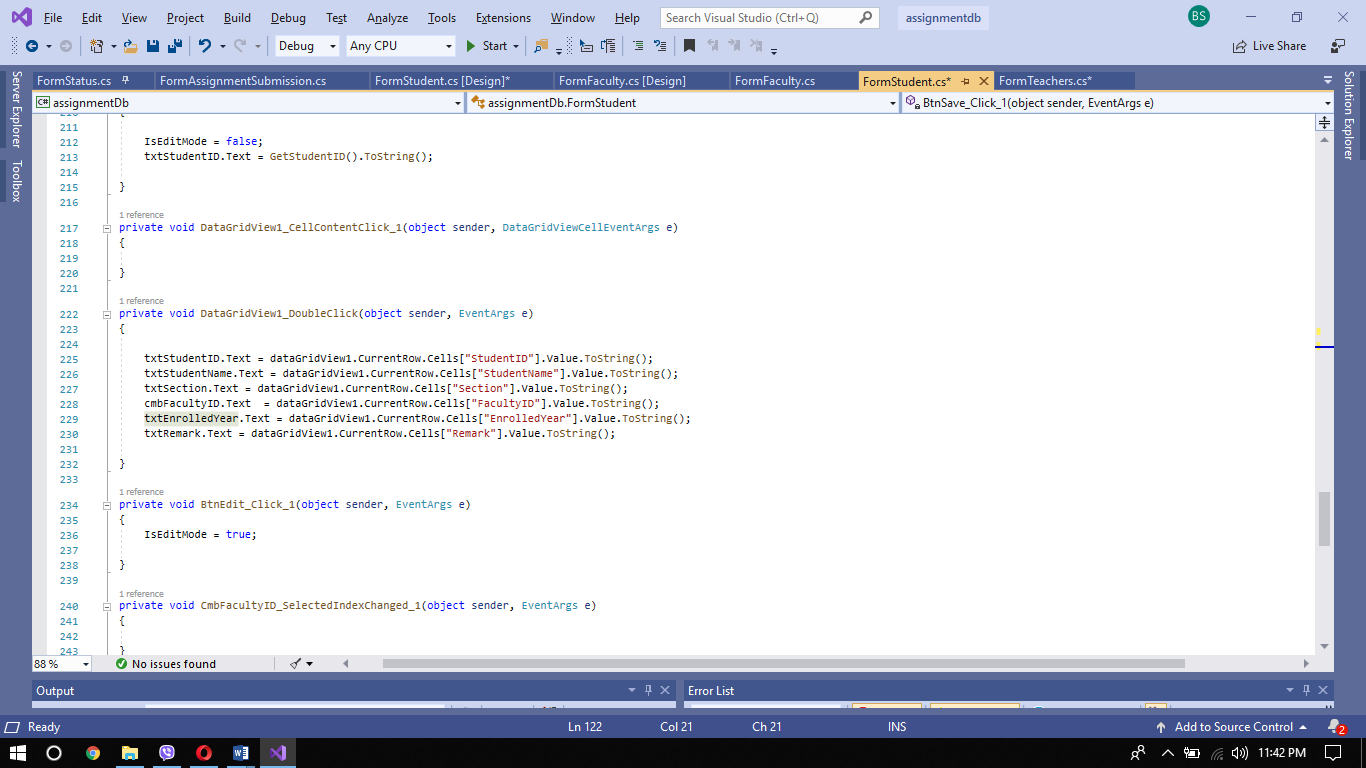


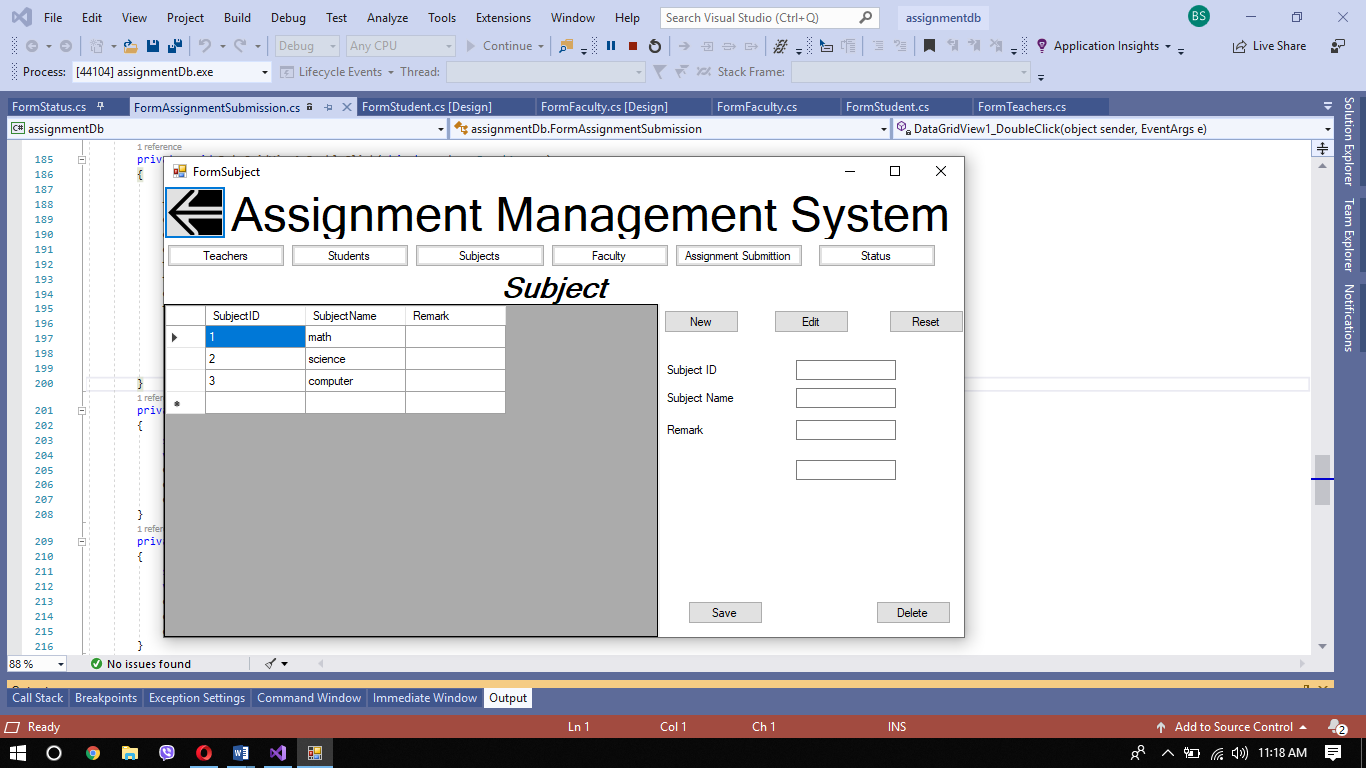
For save button

For delete button

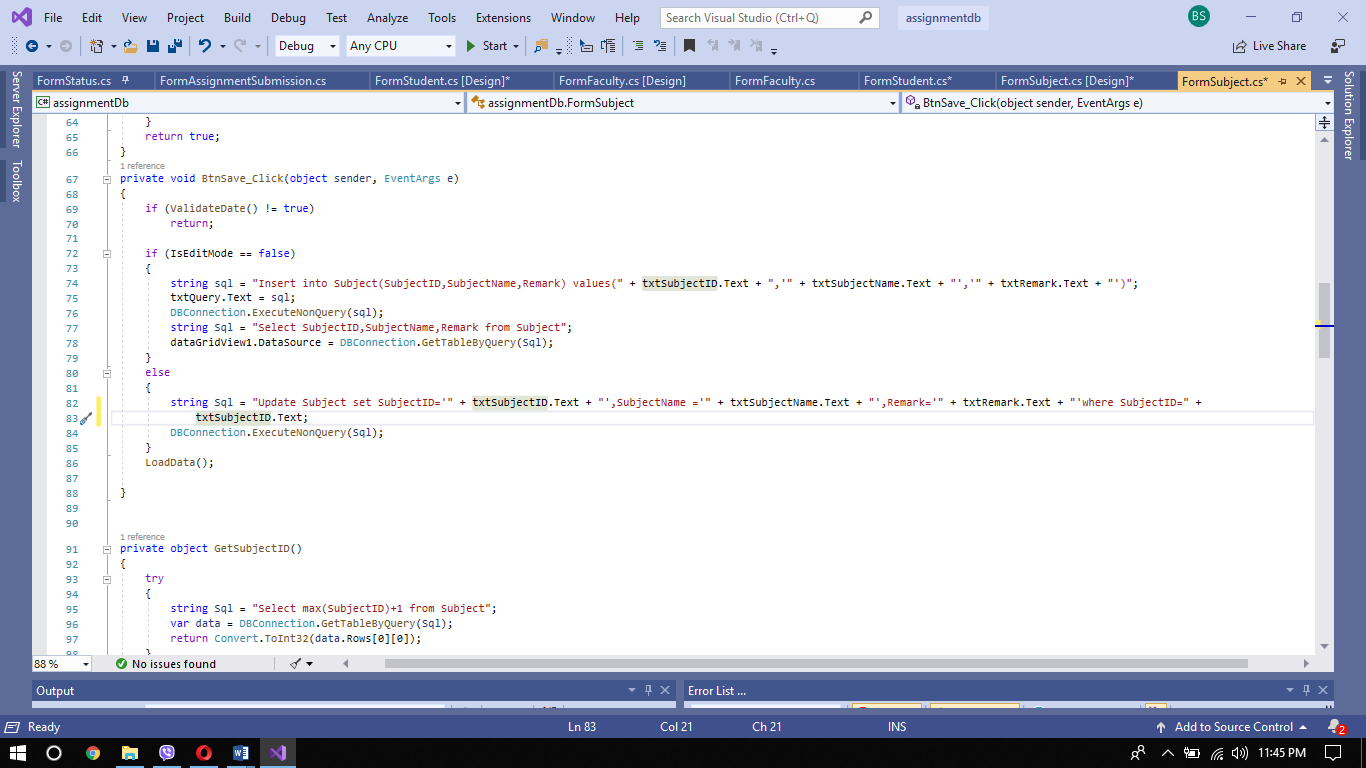


For loading data ni datagrid box.

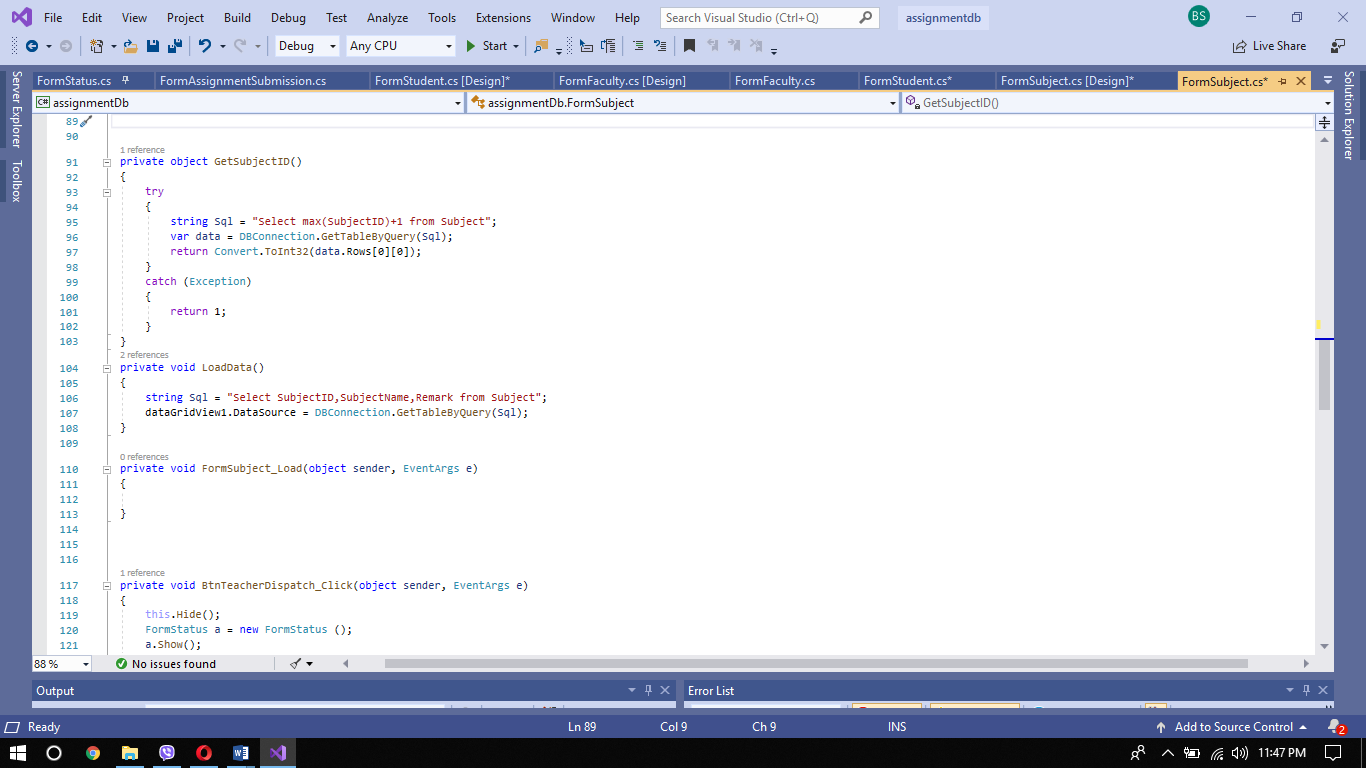




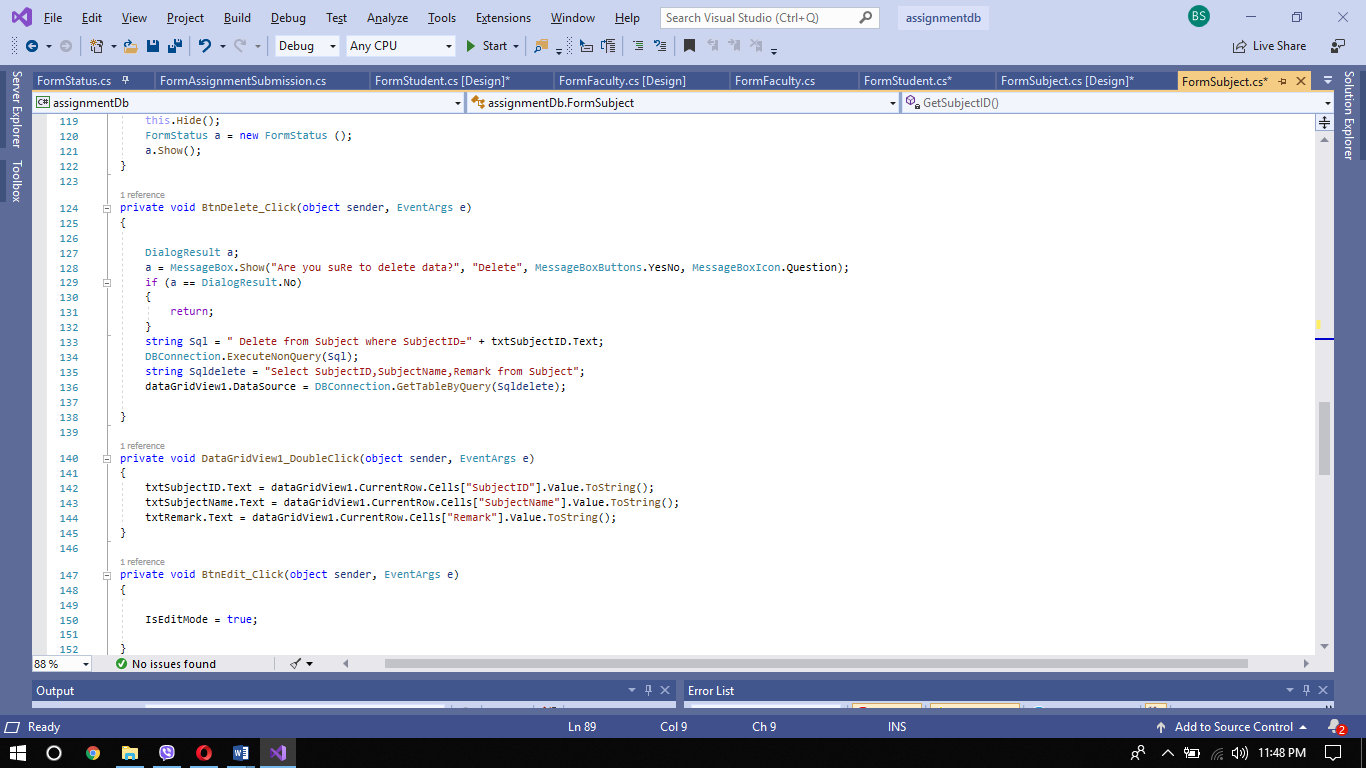
For save



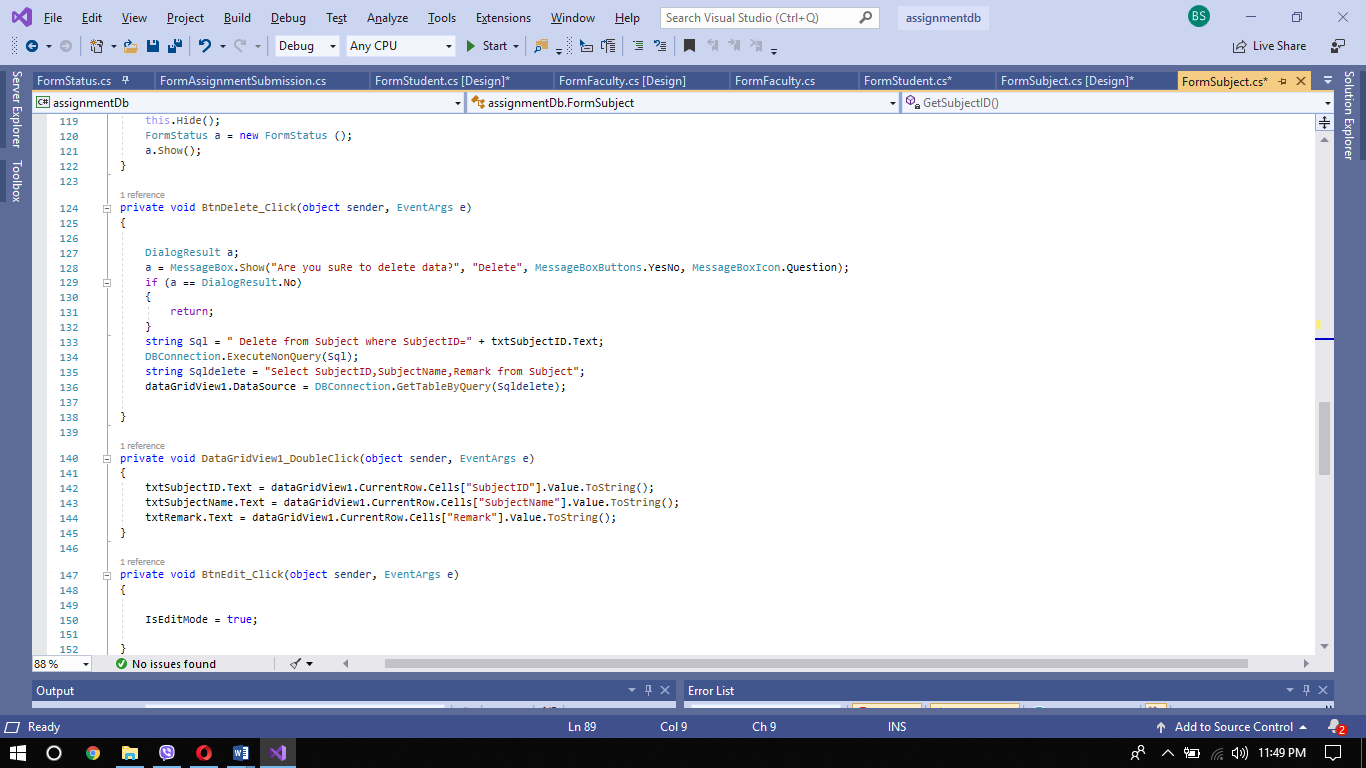
for getting subject id and loading data

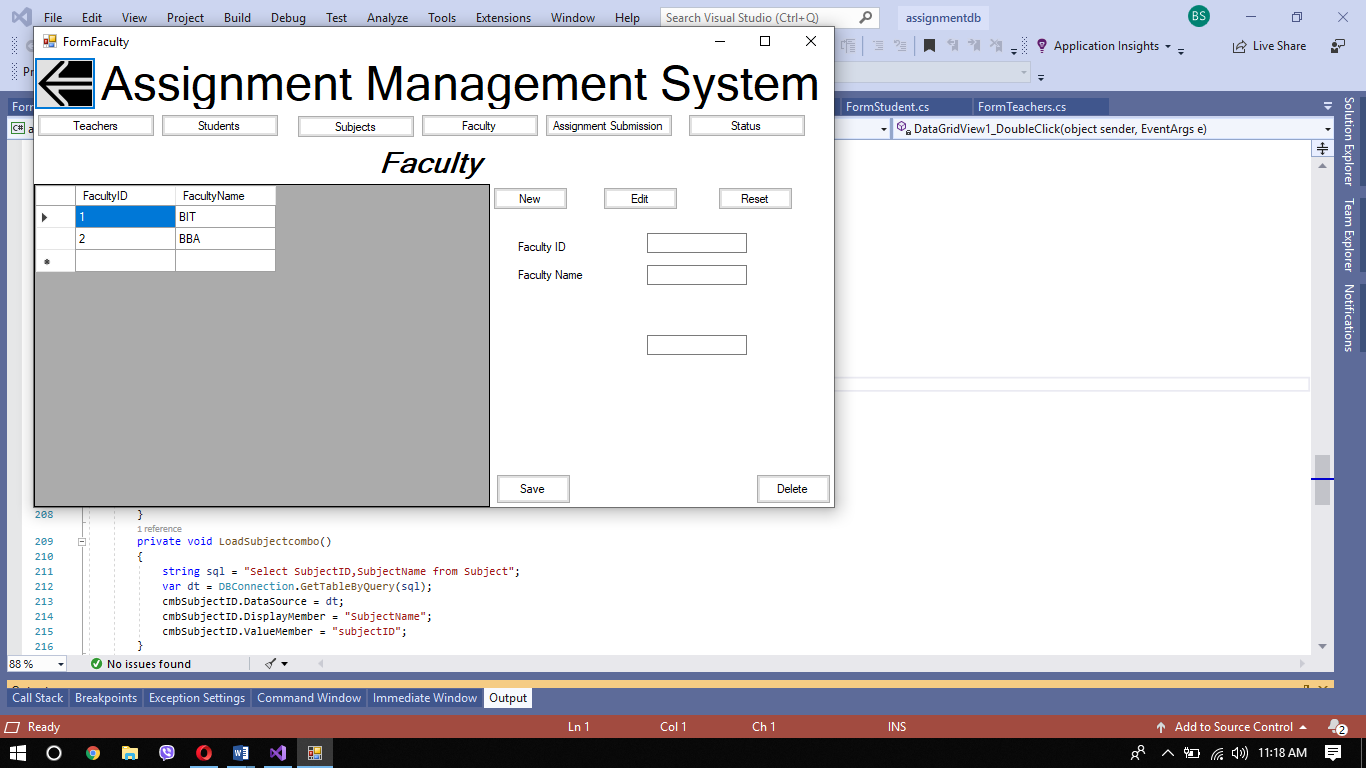


For delete

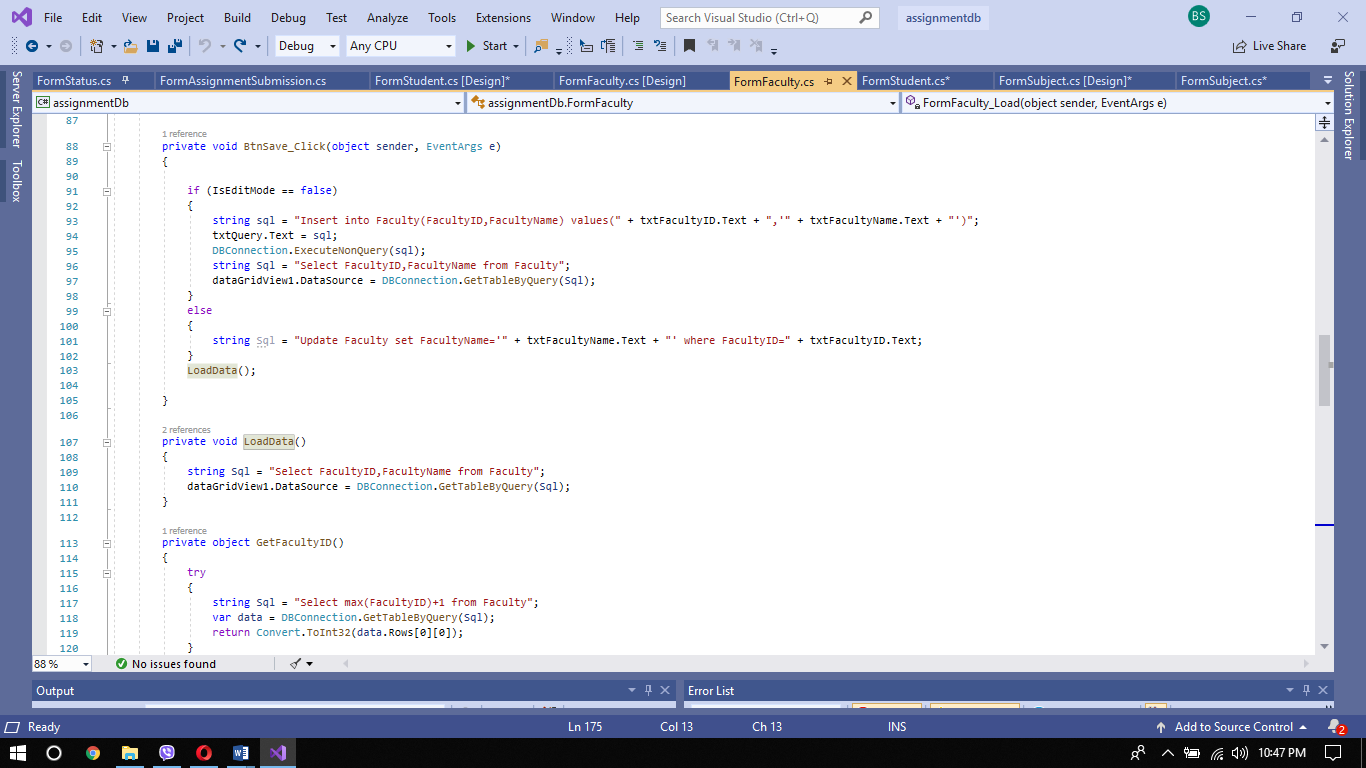


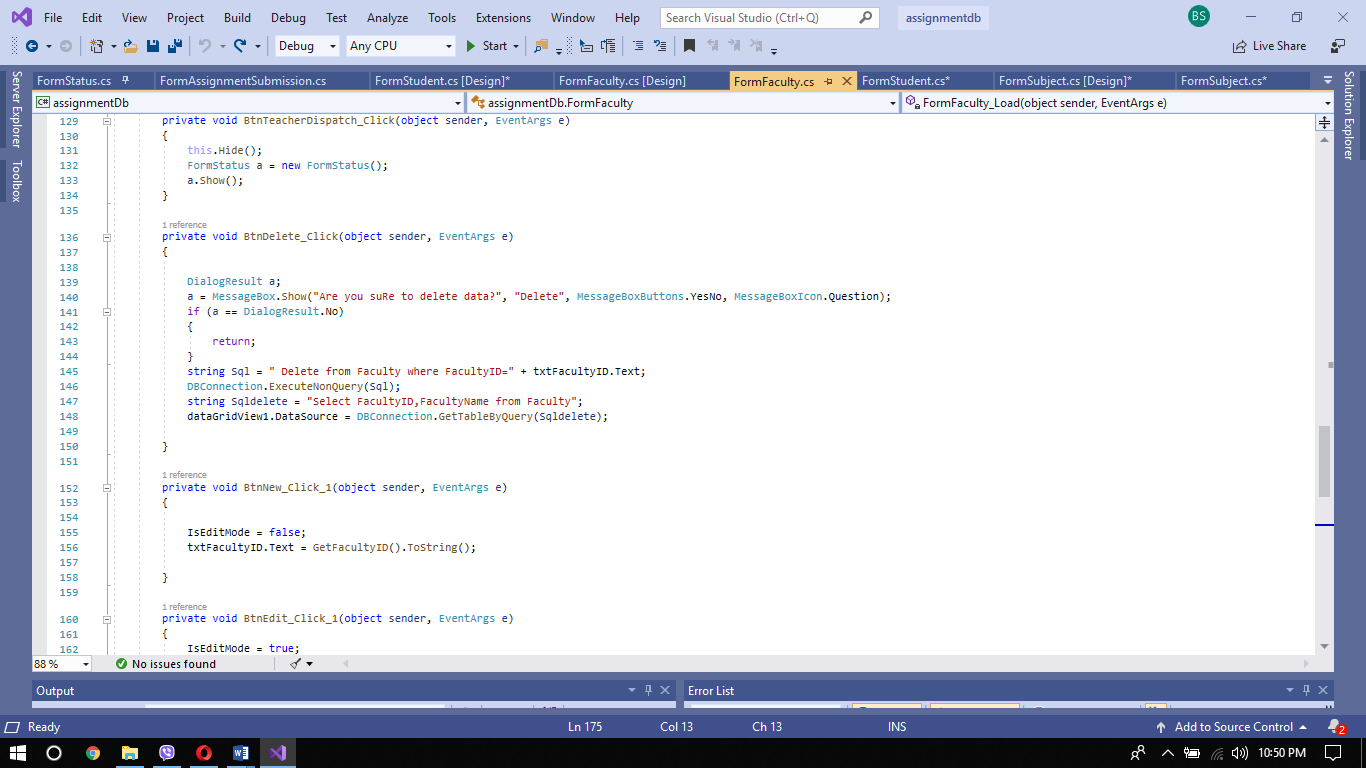
for data grid view



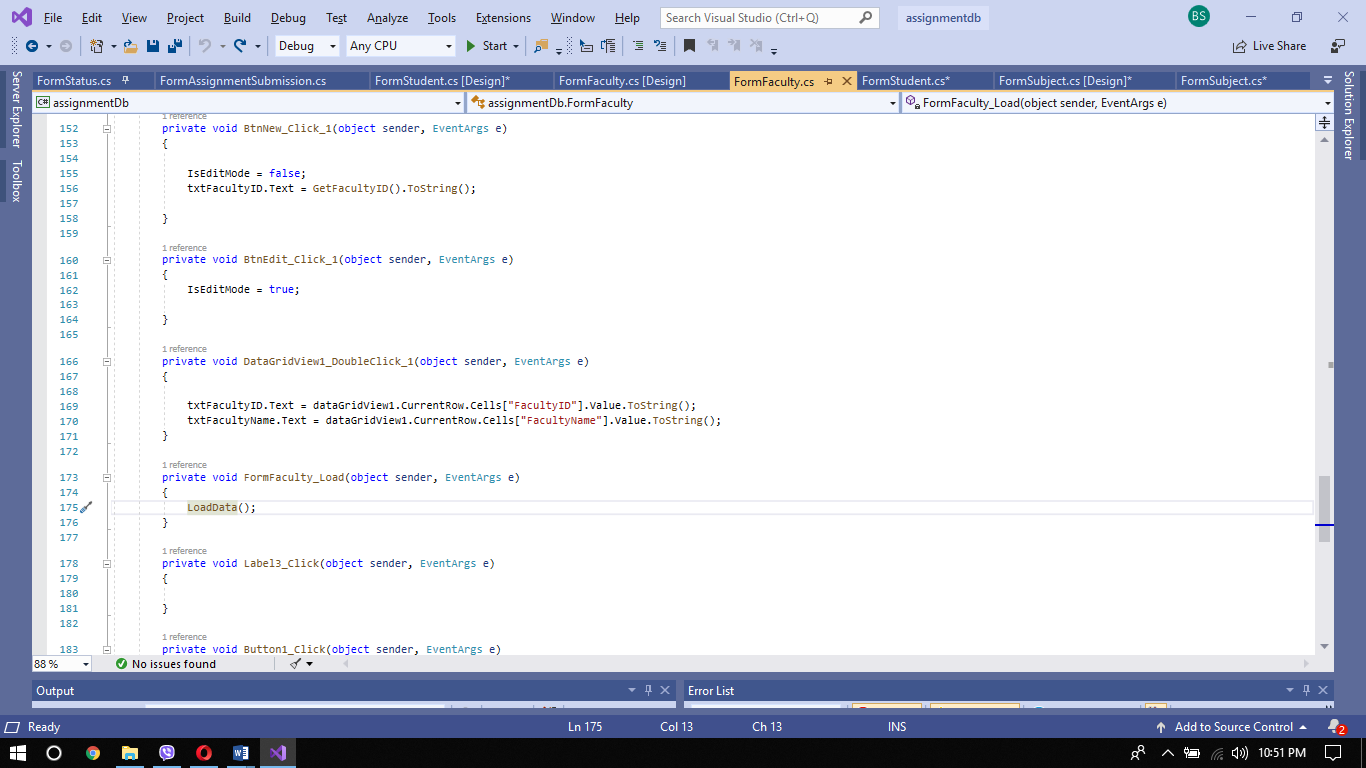
****

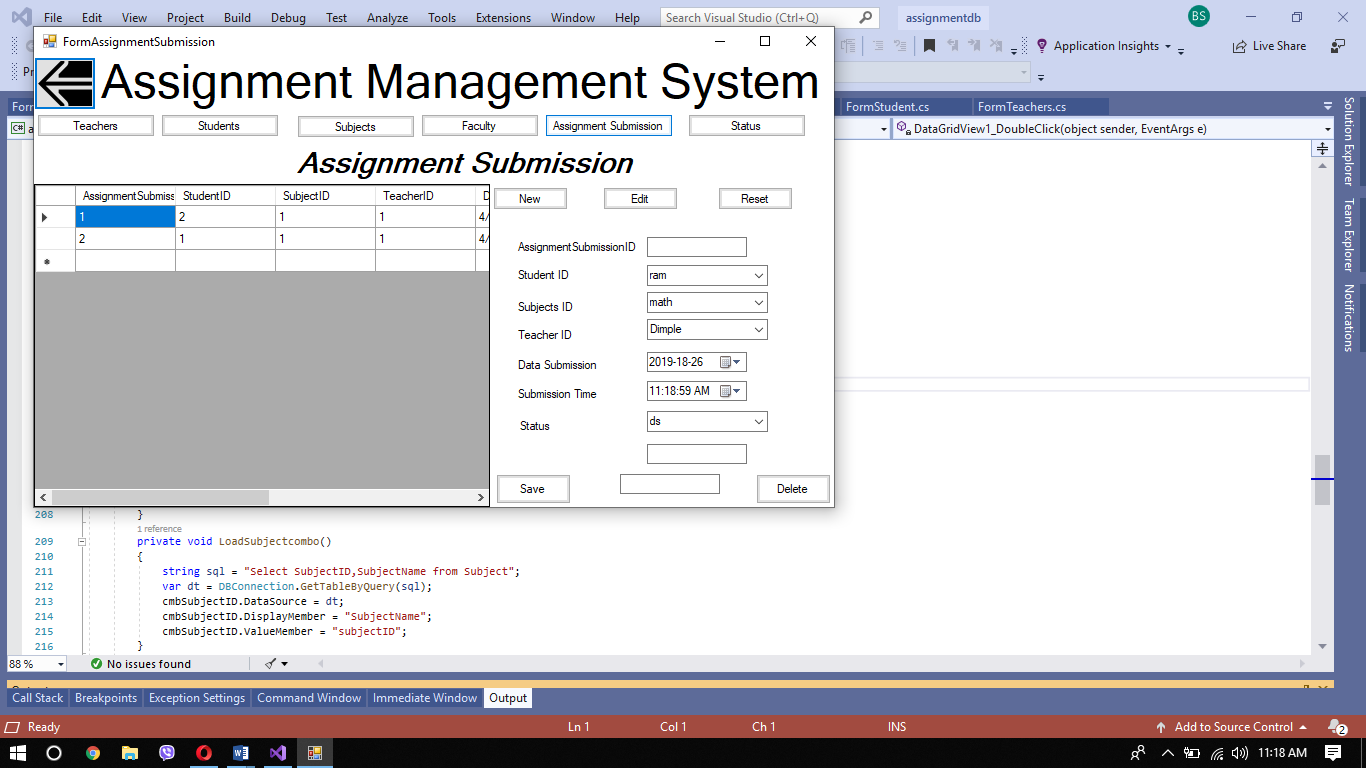
for save

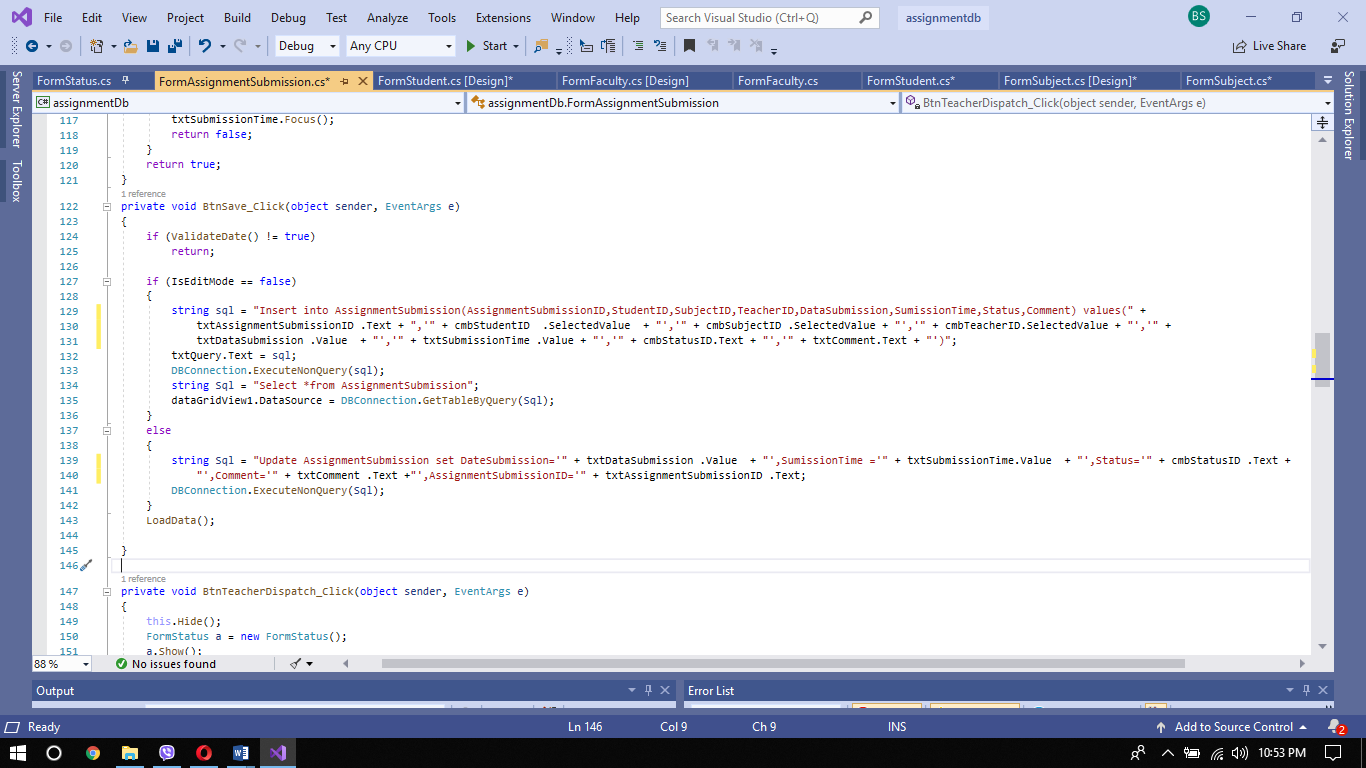


For delete

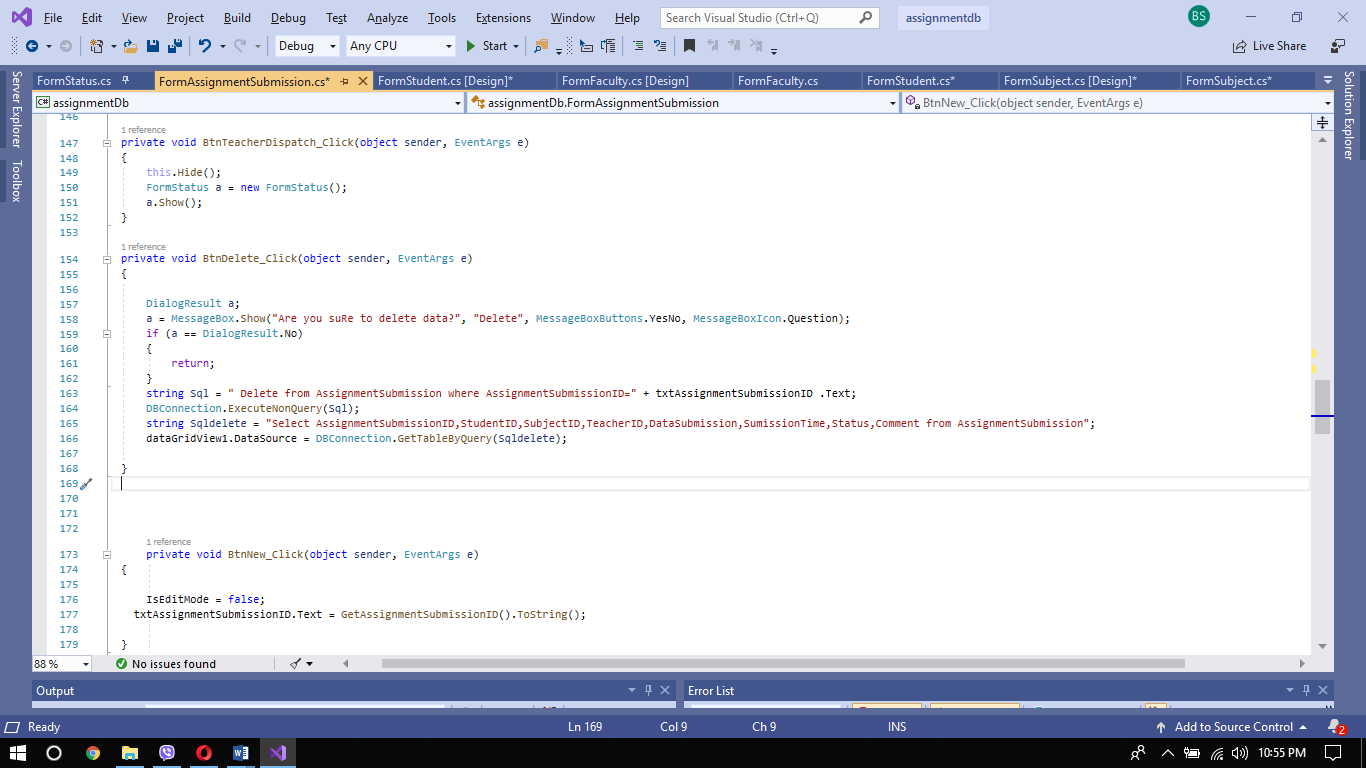
For data gridview



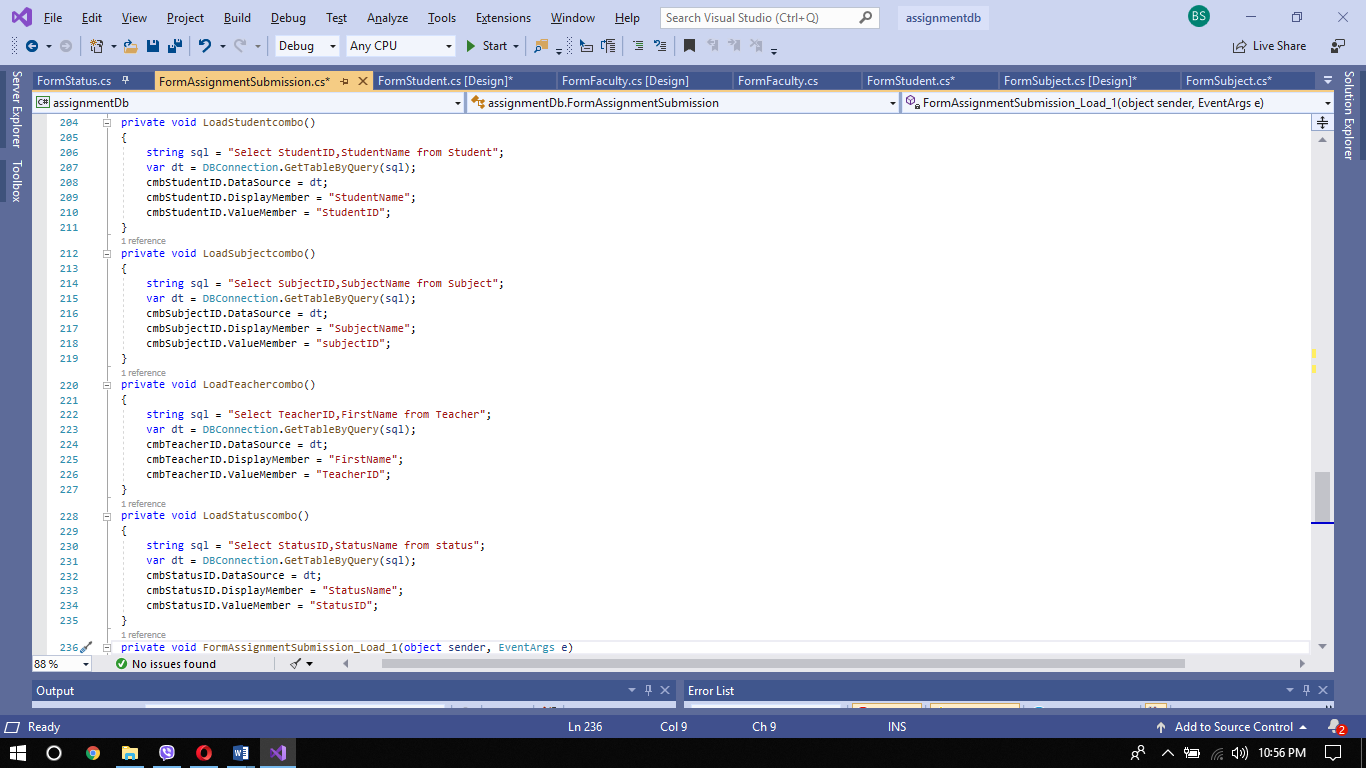
****

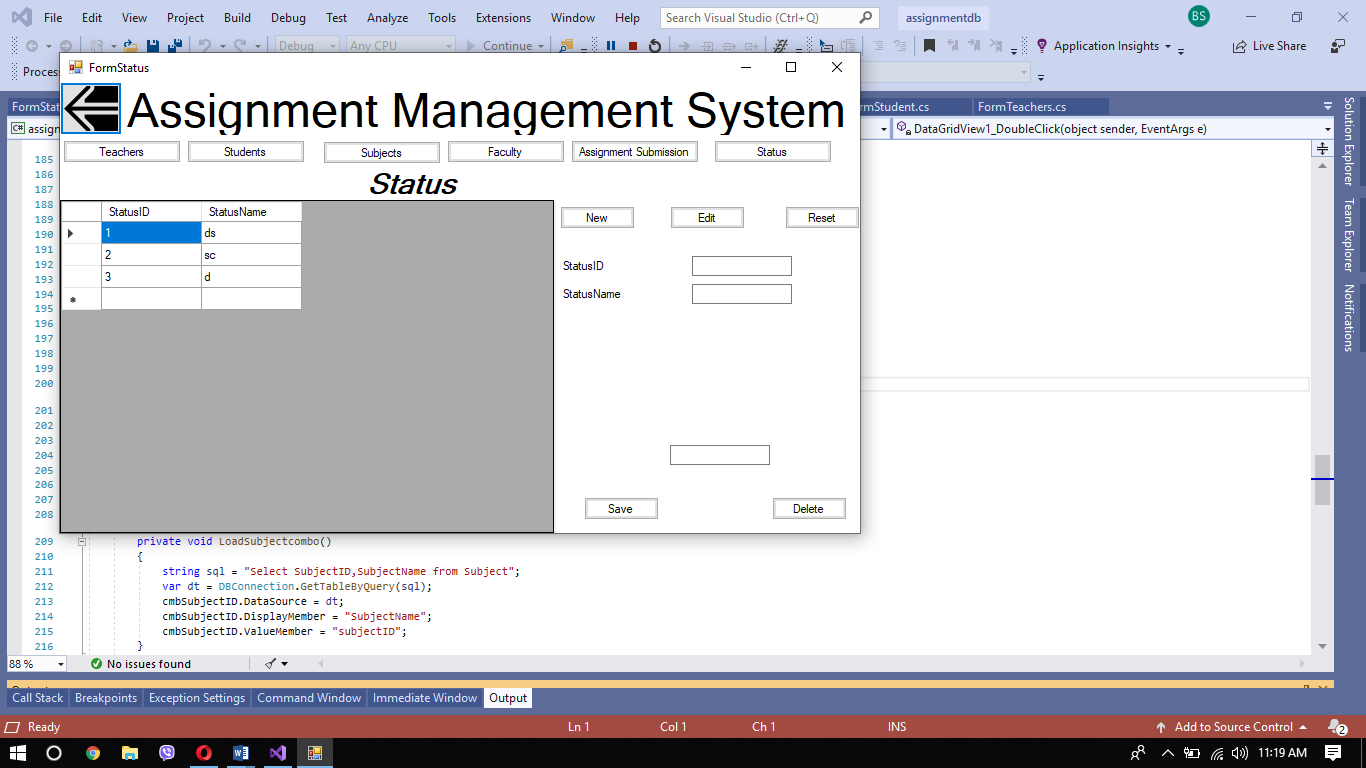
For save

For delete

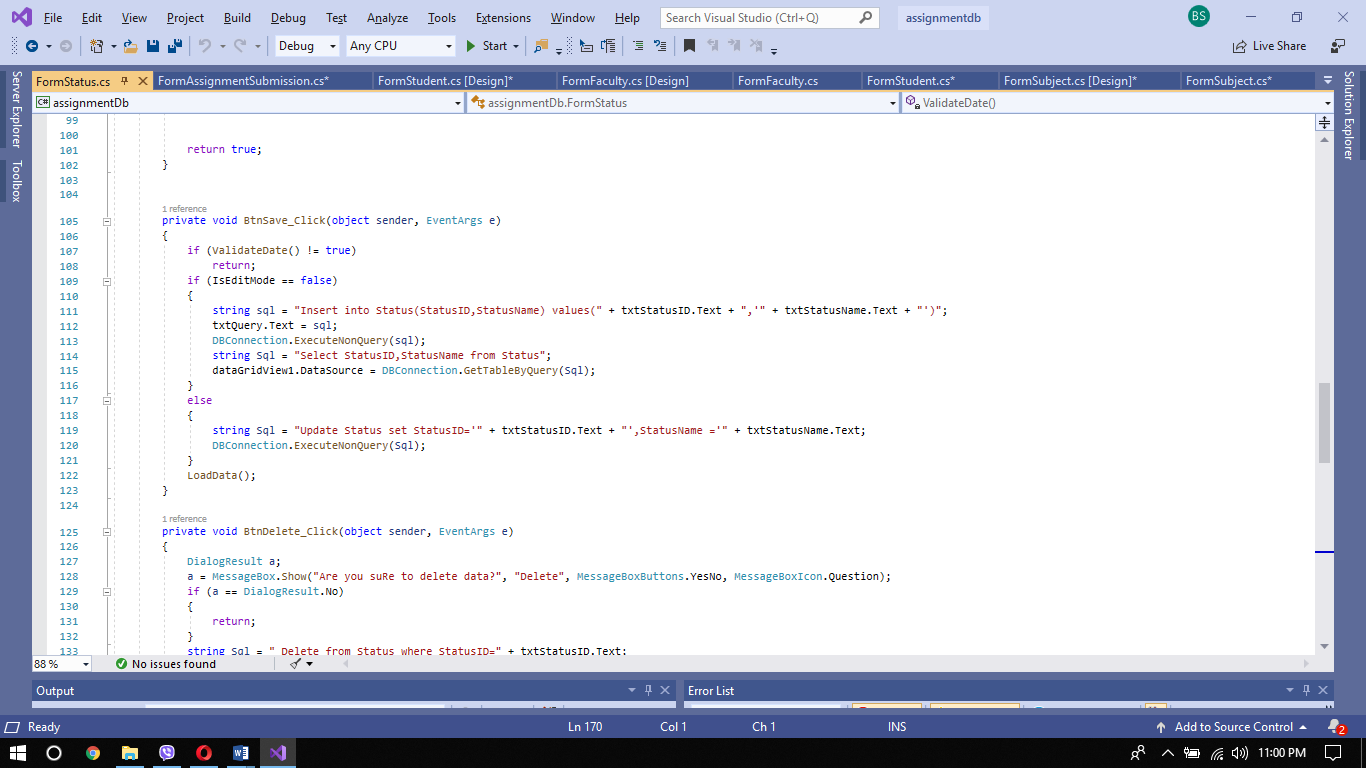


For loading combo box

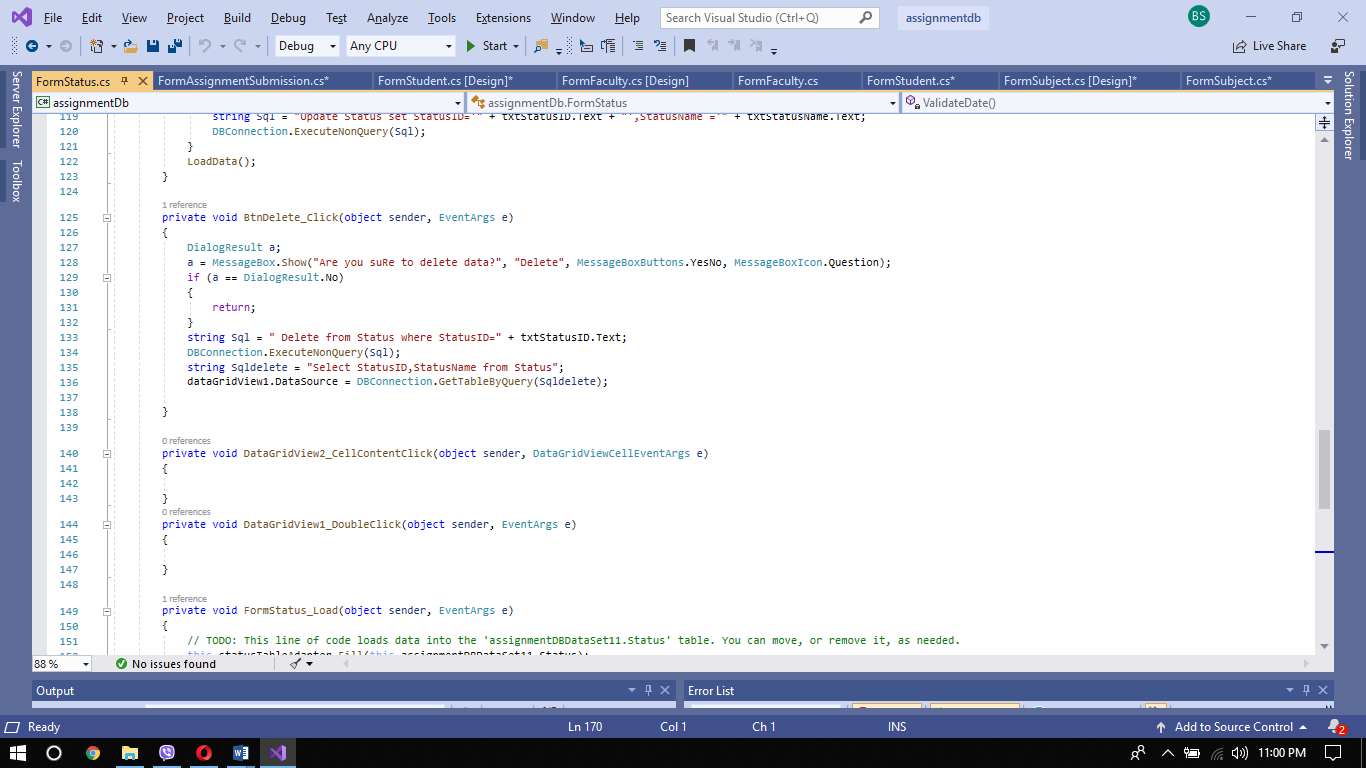


****

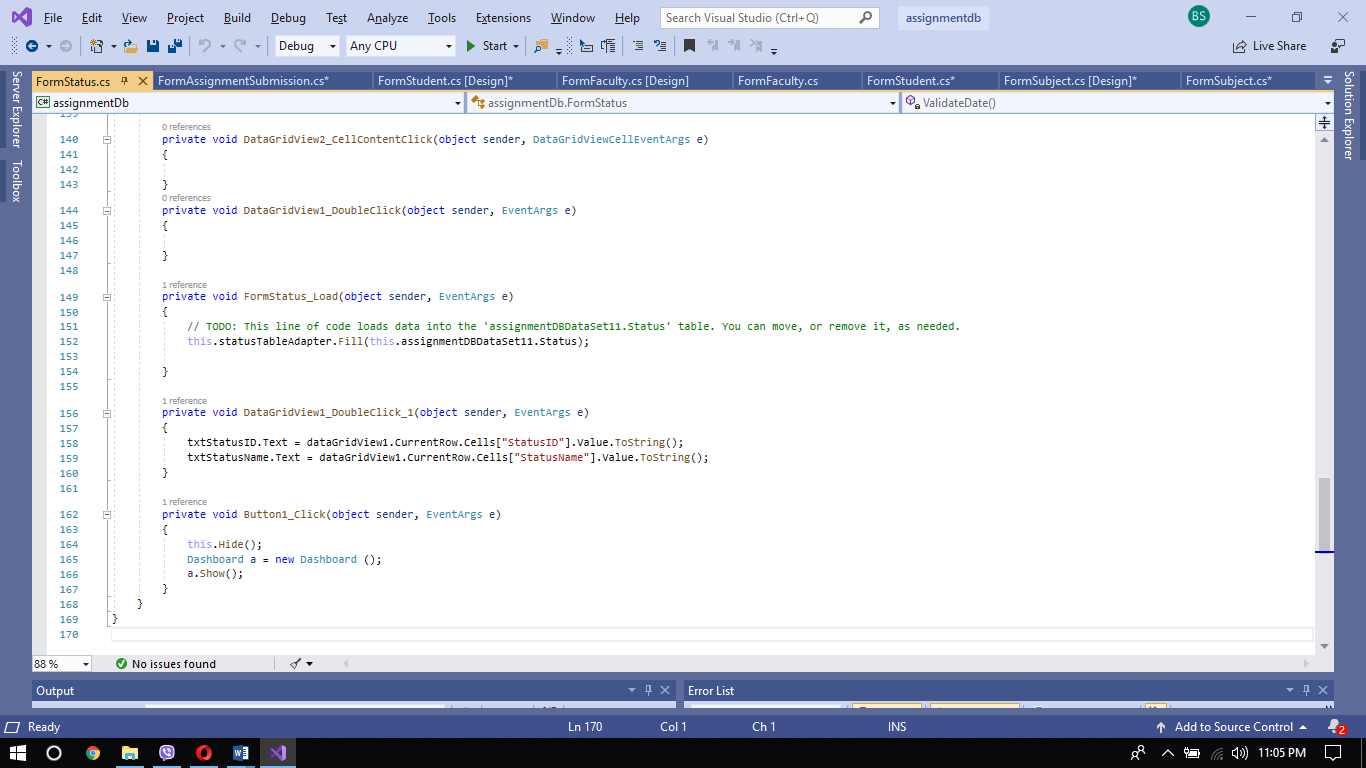
For save



For delete



For data grid view



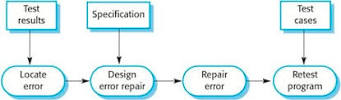
# (P4)Introduction

In this part, I am going to describe about debugging and its process. I am going to describe it using diagram also.

# Debugging

Debugging is the process of detecting and removing existing and potential errors (also referred to as ' bugs ') in a software code that may cause it to act or crash unexpectedly. It is used to discover and fix bugs or faults to avoid improper operation of a software or system. It won't give confidence that the system will fully meet its requirements, but testing gives confidence.

Debugging process with diagram



Debugging is part of the testing process for software and is an integral part of the entire lifecycle of software development. It includes the location and correction of code mistakes in a computer program in software development. The debugging method begins when code is written. Once an error has been recognized, the mistake in the code must be found. After the code is detected it design how to repair the error then it repair it.

## Debugging facilities in IDE explain with snapshot

## **Break point**

In the context of C #, a breakpoint is an intentional stop marked in an application code where debugging pauses for execution. This enables the programmer to inspect the application's inner state at that point. It helps speed up a large program's debugging process by allowing execution to continue to a desired point before debugging starts. This is more effective than a line-by-line step through the code. Adjustment to the result of the program by viewing and modifying the variable values. On the source code line or on a function, a breakpoint can be set with the ability to enable / disable, edit, and delete it. A breakpoint can be set in a single step for all the same-name functions (both overloaded methods and functions that occur in multiple projects).

## **Tracer**

Tracking is a method used by quality assurance staff and programmers to monitor software issues and solutions. Usually a tracking system is set up to store recorded bug data. This type of tracking system provides a clear, centralized overview of the requests for development and their respective states. It allows users to directly log and track bug reports into a system. Tracking systems include a database that tracks bug-related facts.

## **Watch point**

A watch point is a conditional breakpoint given only by the Advanced Business Application Programming (ABAP) Debugger. It is one of the runtime utility parts supplied for debugging service advertising protocol(SAP) application programs and serves as an indication for interrupting further program processing from the given stage or relationship to the ABAP runtime processor.

## **Step command**

There are three step command

* + - 1. Step over command: This command allows the next action to be executed by the implementation. If the action includes a call to an activity, it will not enter its execution (it will instead step over it).
      2. Step into command: This command allows the next action to be executed by the implementation. If the intervention includes a call to an procedure, it will step into its implementation and break the process on that implementation's first action.
      3. Step out command: This command allows the application to run until the procedure implementation that is presently being performed is restored.

# Coding standard

Coding standards are a collection of rules, best practices, modes of programming, and conventions that designers conform to when composing a project source code. In a more general context, a coding standard does not generally deal with the wrong or the correct.

**Common Aspects of a Coding Standard:**  
• Naming Conventions  
• File Naming and Organization  
• Formatting and Indentation  
• Comments and Documentation  
• Classes, Functions and Interfaces  
• Pointer and Reference Usage  
• Testing

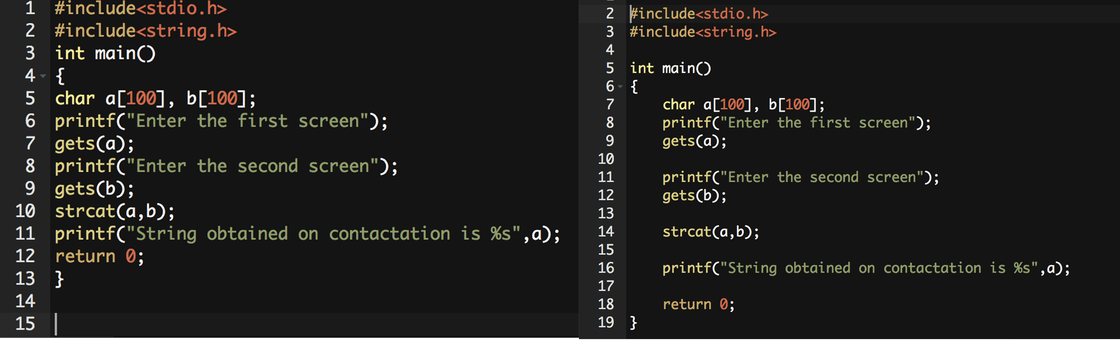
**Advantages of the coding standard**  
Readability, sustainability and compatibility are the advantages of coding standards. Some other advantages are listed below

• Code Integration  
• Team Member Integration  
• Maintenance  
• Uniform Problem Solving  
• Minimizes Communication  
• Minimizes Performance Pitfalls  
• Saves Money Due to Less Man Hours

Coding standard I have used

### Indentation

In any programming domain, indentation is one of the most significant characteristics. Many initiatives become hard to manage due to poor code indentation and incorrect programming methods. Indentation enables the readers to communicate a better program structure. It is used to explain the connection between constructs of control flows such as conditions or loops and code placed within and outside them.  It is used instead of using braces or keywords to determine the structure this is called the off-side rule



### Commenting code.

Commenting on code is the method of sprinkling brief, usually single-line notes across your software. These notes are known as remarks. They clarify how your program functions, and behind it are your plans. Comments have no impact on your program, but are invaluable to read your software for individuals.

 /\* and end with \*/ is multi line comment

// comments :: is single line comment.

/\*\* documentation \*/ :: is document comment

### Variable declaration

In C programming it is necessary to declare variables that are to be used subsequently in separate components of the tasks. Variable declaration tells the compiler two things: variable name, variable type of data. User Defined Type Declaration, Primary Type Declaration are two types of variable declaration.

1. **Primary Type Declaration**: Any data type can be stored in C programming by a variable. The variable's name has nothing to do with its type. The variables must be separated by commas when the variables are declared in a separate row. All statements of declarations must finish with a semi-colon (;). For e.g:

int age;

float weight;

char gender;

1. **User Defined Type Declaration**: A feature known as "type definition" is available in C programming which allows a programmer to define an identifier representing an existing type of data. The user-defined identifier may be used to declare variables later in the program. For e.g:

typedef int age;

typedef float weight;

## **Naming convention**

A naming convention in computer programming is a set of rules for selecting the sequence of characters to be used in identifiers that denote variables, types, functions, and other entities in source code and documentation. There are some reason for using naming convention.

* To decrease the effort required for reading and understanding source code.
* To allow code reviews to focus on problems that are more important than arguing about syntax and naming standards
* To allow tools for code quality assessment to focus their reporting on important problems other than preferences for syntax and type.

**Package**: The names of the packages should be all inferior. If multiple words are needed, they should be separated by an underscore. It is usually preferable to stick to 1 word names

**Class**: Class names should be nouns, the first letter of each inner word should be capitalized in a mixed case. The title of the buttons and the names of the classes should be capitalized. Use whole sentences and avoid acronyms and abbreviations.

## **E. Camel Case**

Camel Case is a naming convention in which a compound word capitalizes the first element of each phrase. The word "Camel Case" itself includes the protocol for naming Camel Case. Although Camel Case has many applications, it is most commonly used in programming languages and website names in computing. Most programming languages do not allow the names of functions, variables, or other entities to use spaces. CamelCase is therefore often used by programmers to describe items within the source code.

## **F. Pascal Case**

Pascal is a high-level, general-purpose language established as a systematic discipline for learning programming and developing reliable and effective programs. It provides several kinds of information and constructions for programming. The Pascal programs are simple to understand and maintain. It also enables programmers to identify complicated organized data types and to construct dynamic and recursive data structures such as lists, trees and graphs. It provides functions such as records, enumerations, sub-ranges, variables dynamically assigned with related pointers and sets.

**References**

<https://www.innovativearchitects.com>

<https://notes.tyrocity.com/types-of-programming-languages/>

<https://www.computerhope.com/jargon/p/proclang.htmx>

<https://docs.microsoft.com/en-us/dotnet/standard/managed-execution-process>

<https://sites.google.com/site/mattysassignments/home/p1>

<https://www.geeksforgeeks.org/>

<https://www.javatpoint.com>

<https://fresh2refresh.com>

<https://www.valid-computing.com/benefits-of-coding-standards.html>