**SUBMITTED BY**

**FULL NAME:** Gaurav Shakya

**COURSE:** BTEC HND in Computing

**SEMESTER:** First Semester

**SECTION:** C

**YEAR OF ENROLLMENT:** 2018

**SUBMITTED TO**

Mr. Shiva Poudel

**Contents**

[Part I 4](#_Toc7042979)

[Introduction 4](#_Toc7042980)

[Algorithm 4](#_Toc7042981)

[Features of Algorithm 4](#_Toc7042982)

[Advantages of Algorithm 5](#_Toc7042983)

[Disadvantages of Algorithm 5](#_Toc7042984)

[Examples of Algorithm 5](#_Toc7042985)

[Development Environment 6](#_Toc7042986)

[Platform Environment 8](#_Toc7042987)

[Design Architecture of application 9](#_Toc7042988)

[Data Dictionary 10](#_Toc7042989)

[Flowchart for program 13](#_Toc7042990)

[Code execution process 18](#_Toc7042991)

[Conclusion 22](#_Toc7042992)

[Part II 23](#_Toc7042993)

[Introduction 23](#_Toc7042994)

[Procedural Programming Language (PPL) 24](#_Toc7042995)

[Features of PPL 24](#_Toc7042996)

[Advantages of PPL 25](#_Toc7042997)

[Drawbacks of PPL 25](#_Toc7042998)

[Limitations of PPL 25](#_Toc7042999)

[Control Statement 26](#_Toc7043000)

[LOOP 33](#_Toc7043001)

[Array 37](#_Toc7043002)

[Structure 38](#_Toc7043003)

[Pointers 41](#_Toc7043004)

[String 42](#_Toc7043005)

[Function 44](#_Toc7043006)

[Object Oriented Programming Language (OOP) 47](#_Toc7043007)

[Advantage of OPP 47](#_Toc7043008)

[Drawbacks of OOP 48](#_Toc7043009)

[Features of OOP 48](#_Toc7043010)

[Event-Driven Programming Language 52](#_Toc7043011)

[Advantages and disadvantages 53](#_Toc7043012)

[Key features and characteristics of event-driven programming 53](#_Toc7043013)

[Integrated Development Environment (IDE) 56](#_Toc7043014)

[Benefits of IDEs 57](#_Toc7043015)

[Tools available in IDE 57](#_Toc7043016)

[Conclusion 62](#_Toc7043017)

[Part III 63](#_Toc7043018)

[Introduction 63](#_Toc7043019)

[Design and Code 64](#_Toc7043020)

[Integrated Development Environment (IDE) 76](#_Toc7043021)

[Benefits of IDEs 76](#_Toc7043022)

[Tools available in IDE 76](#_Toc7043023)

[Debugging 78](#_Toc7043024)

[Coding Standard 82](#_Toc7043025)

[Conclusion 86](#_Toc7043026)

[Bibliography 87](#_Toc7043027)

# Part I

**Present a mini research project both in the form of a report and as a presentation. You will need to explain what an algorithm is, with examples of their use, comparing their efficiency against brute forcing, and how it will relate to the application development process, down to the implementation in a suitable language. Research could be conducted on your choice of algorithms, including sorting and searching data or encryption/decryption, compression/decompression.**

## Introduction

An algorithm is important in optimizing a computer program according to the available resources. Algorithm plays a vital role in understanding the programmed code. I will be discussing about algorithm more including with the features, benefits, constraints, and some examples of it.

I will also be discussing about my project, design, data dictionary, examining, implementing and evaluation of application. Some snapshots of the required details.

## Algorithm

An Algorithm is a process of solving a problem in finite number of steps. In other words, an Algorithm is set of rules a machine follows to achieve a particular goal. Example: There is a certain algorithm (steps) to make pizza.

## Features of Algorithm

There are some important features that should be include in algorithm. First of all, there should be precision (accurate) in the algorithm i.e. all the steps should be defined in precise way. In Algorithm, finite number of steps are must. Results of each step are uniquely defined and only depend on the input and the result of the preceding steps.

## Advantages of Algorithm

Algorithm is step by step process of solution, which simply makes it easy to understand. It also uses definite procedure. Algorithm is easy to understand even by common people because it is not dependent on programming language. Debugging may be easy in it.

## Disadvantages of Algorithm

Algorithm is time consuming as we need to provide the steps. Difficult to show branching as well as looping in it. Complex tasks are difficult to put in Algorithms.

## Examples of Algorithm

Algorithm to write and algorithm to multiply two numbers given by user.

Step 1: Start

Step 2: Declare variables a, b and multiply.

Step 3: Read values a, b.

Step 4: multiply = a \* b

Step 5: Display multiply

Step 6: Stop

Algorithm to find the greatest number among three different numbers given by the user.

Step 1: Start

Step 2: Declare variables a, b and c.

Step 3: Read values a, b and c.

Step 4: If a>b

If a>c

Print ‘a’ is the greatest number.

Else if b>a

If b>c

Print ‘b’ is the greatest number.

Else

Print ‘c’ is the greatest.

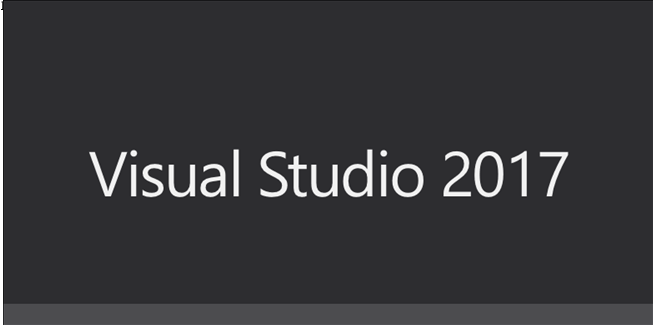
Step 5: Display Sum

Step 6: Stop

## Development Environment

I will be using .NET software i.e. Visual Studio. Code might be written in Visual C++, Visual F# and JavaScript, but we have used C# in it. Computer programs for Microsoft Windows are developed using this. It is perfect for everyone with any OS (Windows, Mac or Linux). The main reason behind why we used Visual Studio is that it is easy to Edit, Build and Debug. As I have been using this tool, I found that it provides auto-option, bracket-matching, box-selection and many more. There is also a easy Keyboard shortcuts, which helps us in code, also consumes less time. Using Visual Studio, also webpages, web applications and web services are developed. It include different components like Windows API, Windows Forms, and many more. Similarly, when the coding got tough, it got debugging. Breakpoints where you can pause debugger execution as you need.

Similarly, I have used SQL server –express edition for Relational Database Management System because SQL Server organizes all objects, such as tables, views, and procedures by database names, also, it can be used for small projects as well as large applications and can handle millions of transactions per day.





Platform Environment**:-**

We have used Visual Studio of desktop based, which is only for Windows. There are some Computing platform where the software is executed.

System requirements defines the configuration that a system must have for a hardware or software, application to run smoothly and efficiently. Incase these requirements are not met, there might be failure of installing the software or performance problems. Performance can differ from different computers, laptops according to their System. As I used SQL-Server 2014 for the database storage and for IDE I used Visual Studio 2017, here are the following requirements for it.

**Requirements in SQL Server:**

* Operating System – Windows 7 or above.
* Memory – (Minimum – 1 GB) & (Maximum – at least 4GB)
* Processor – (x86 Processor – 1.0 GHz), (x64 Processor – 1.4 GHz), (Recommended – 2.0 GHz or faster.
* Hard Disk – SQL server 2014 requires minimum of 6 GB space. For better performance use more than 20 GB of Hard disk.

**Requirements in Visual Studio:**

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

Note: Always use computers on which there is NTFs file format. It is most secured file system. Also, use windows 8 or above for better performance.

## Design Architecture of application



Here, this is the design architecture of our application.

Data Dictionary**:-**

|  |  |
| --- | --- |
| **Customer table** includes Customer Id, Customer Name, Address, Telephone, Remarks of the Customers with data types; INT, varchar, varchar, INT and varchar respectively. | **Product Table** includes Product Id, Product Name, Unit, Selling Price, Cost Price and Stock Quantity of Product with data types INT, varchar, INT, INT, INT, INT respectively. |
|  |  |

|  |  |
| --- | --- |
| **Purchase table** includes Purchase Id, Invoice Number, Supplier Id, Dare Of Purchase, Amount Of Invoice, Remarks with data types; INT, INT, INT, Date, INT and varchar respectively. | **Purchase Details Table** includes Purchase Id, Product Id, Purchase Rate, Quantity and Purchase Details Id with data types INT, INT, Decimal, INT, INT respectively. |
|  |  |

|  |  |
| --- | --- |
| **Sales table** includes Sales Id, Customer Id, Date Of Sale, Total Amount, Remarks with data types; INT, INT, Date time, Decimal and varchar respectively. | **Sales Details Table** includes Sales Details Id, Sales Id, Product Id, Sales Rate, and Quantity with data types INT, INT, INT, Decimal and INT respectively. |
|  |  |

|  |  |
| --- | --- |
| **Staff table** includes Staff Id, Staff Name, Staff Address, Staff Phone Number, Username, Password, Remarks and Is Admin with data types; INT, varchar, varchar, INT and varchar, varchar, varchar, varchar respectively. | **Supplier Table** includes Supplier Id, Supplier Name, Address, Telephone and Remarks with data types INT, varchar, varchar, INT and varchar respectively. |
|  |  |

## Flowchart for program

|  |  |
| --- | --- |
| Flowchart for Login Form. | Algorithm for it. |
|  | Step 1- Start  Step 2- Input Username and Password.  Step 3- Check the given Username and Password is valid or not.  Step 4- If yes, open the MDI form.  Else, display error message.  Step 6- Stop. |

|  |  |
| --- | --- |
| Flowchart for Save button in every form. | Algorithm for it. |
|  | Step 1- Start  Step 2- Input data in text box.  Step 3- Check the given data are valid or not.  Step 4- If yes, save the data in the database.  Else, do not change anything and display error message.  Step 6- Stop. |

|  |  |
| --- | --- |
| Flowchart for Delete button in every form. | Algorithm for it. |
|  | Step 1- Start  Step 2- Select data from datagrid view box to respective text box.  Step 3- Check if the data are displayed in text box or not.  Step 4- If yes, delete the data in the database.  Else, do not change anything and display error message.  Step 6- Stop. |

**\*** **Determining the steps taken from writing code to execution.**

**Flowchart Algorithm**

Step I – Start

Step II – Declare and input a and b

Step III – Calculate Sum = a + b

Step IV – Print Sum

Input a & b

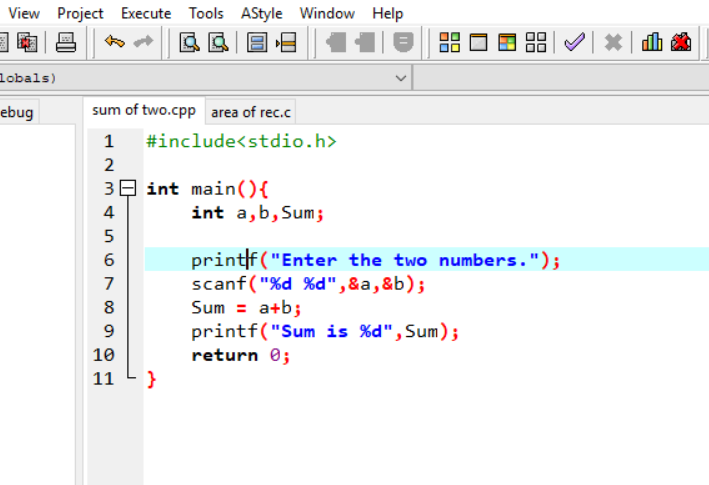
Step V – Stop

Sum = a + b

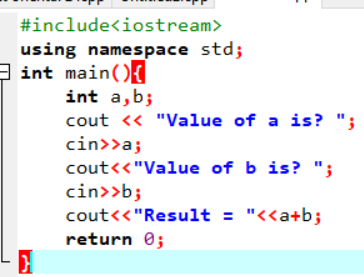
Print Sum

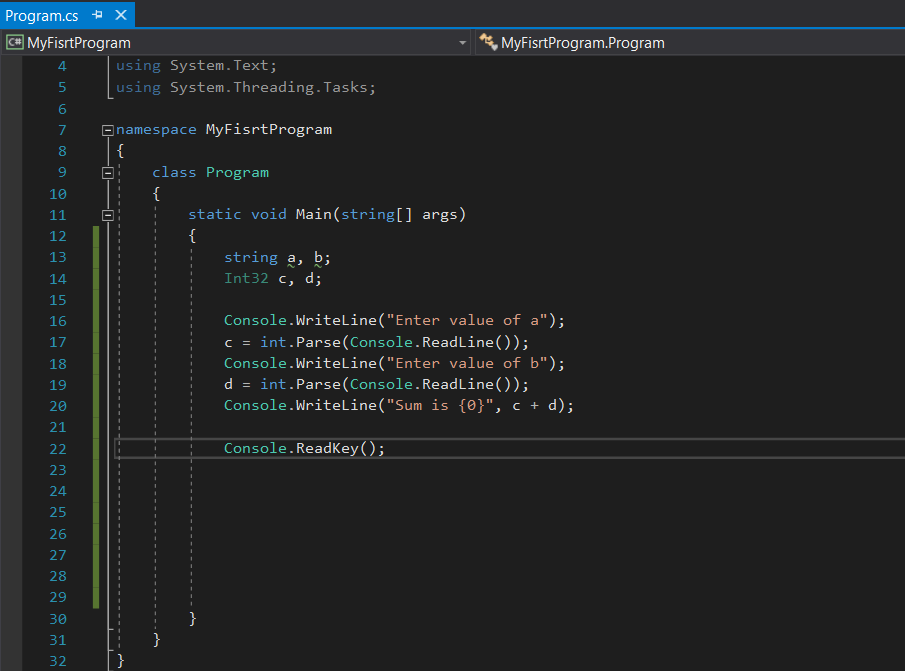
**Calculating sum of two numbers**

1. In C Programming :



1. In C++ programming :



1. ****In C# Programming :

## Code execution process

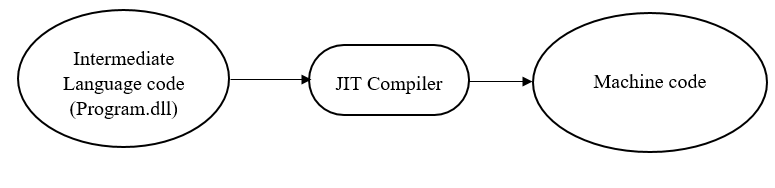
Here, the first thing to know is about Compiler. Compiler is a program that transforms source written in one programming language into machine codes. Here, the processor of the computer doesn’t know the high level languages like C, C++, C#, JAVA etc. It only understands machine level language.

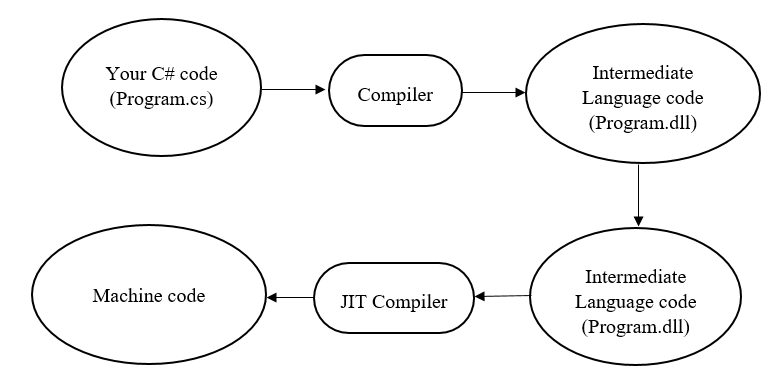
The code execution process includes the following types;

**1.** Compiler time process.

 It is the first process involved in the code execution. In visual studio, when you execute the program, it takes your code as an input, process it, and give your output in Intermediate Language (IL) code which is saved in ‘.dll’ or ‘.exe’ files. Following diagram is shows Compiler time process.

**2.** Runtime process.

 Only Compiler time process is not the complete process of the execution. Here, IL doesn’t convert the code to machine language, so Common Language Runtime (CLR) is needed to convert the IL code to machine codes. CLR is a program running on computer that manages the execution of IL code. CLR uses Just-In-Time or JIT compiler to translate the IL code into machine code. CLR program compiles IL code at the moment when the user tries to use it, example, when you open the .dll file or .exe file of compiler time process.



Here is the final code execution process. Both CLR and JIT compiles IL into a machine code that the computer is comfortable with.

**Language Processing System**

We know computer system is made of hardware and software. Only hardware understands a language, which humans cannot understand. So we write programs in high-level language, which is easier for us to write and remember which is again translated to machine code, so that machine could understand. This is called Language Processing System.

**Source Code-** Source code are those code which are written in high-level programming language by the programmers. These code are not understandable by the computer so we sent it to Preprocessing process.

**Preprocessor-** It is the first step of language processing system where it translates the source code in such a way that machine could understand it. It is done to transform your program before actual compilation. It also deals with macro-processing, augmentation, file inclusion, language extension, etc.

**Interpreter-** An interpreter, alike compiler, translates high-level language into low-level machine language. Here, a compiler reads the whole source code at once, creates tokens, generates intermediate code, and executes the whole program. Similarly, an interpreter reads a statement from the input, converts it to an intermediate code, executes it and takes the next statement in sequence. If any error, it stops execution and report it, whereas a compiler reads whole program even there is errors.

**Assembler-** An assembler translates assembly language programs into machine code. The output of a assembler is called an object file, which contains a combination of machine instruction as well as the data required to place these instructions in memory.

**Linker-** It is a program that links and merges various object file together in order to make an executable file. All these might have been compiled by separate assemblers. The major task of a linker is to search and locate referenced module in a program and to determine the memory location where these codes will be loaded.

**Loader-** Loader is a part of operating system which is responsible for loading executable files into memory and execute them. It calculates the size of a program (instructions and data) and create memory space for it.

**Memory-** At last, these code are loaded and saved here.

Here this is the flow of code execution process.

**Differentiate between Algorithm and Code variant**

|  |  |
| --- | --- |
| Algorithm | Code variant |
| Algorithm is a step by step procedure to solve a given problem. | Code is a method of writing an algorithm. |
| It is an unclear specification of how to solve a problem. | It is an informal high-level description of operating principle of a computer program. |
| It is simple procedure. | It is combined form of programming language and natural English language. |
| It is moderate and complex to understand than Code Variant. | It is simpler and easy to understand. |

## Conclusion

Hence, I have presented a mini research project both in the form of a report and as a presentation. I have explained what an algorithm is, with examples, snapshot, and mentioned their use accordingly. Also, describe about the algorithm and flowchart of our application with snapshots. Secondly, I showed the data dictionary used in our application. Also, design architecture of application and described how the program code execution process works.

# Part II

**The research and development team you work with have been tasked with further investigation into how best to build more efficient, secure software. You have been asked to look into programming paradigms and the advantages and disadvantages of using different programming language approaches.**

**You will need to create a report covering findings from research into the characteristics of different programming paradigms – procedural, object-orientated and event-driven programming.**

**Your report should include an explanation of each paradigm, an analysis of suitable IDEs, and an evaluation of source code that would be generated for an application.**

## Introduction

As I have finished my part I work, I am heading towards part II. This part is about Procedural Programming Language (PPL), I will discuss its example, characteristics, and limitations. Similarly, about Control Structure and Conditional Structure which includes if, if else, if else if statements and various types of loop.

Accordingly, I will discuss about Object Oriented Programming (OPP), features, impacts, limitations, which also includes class &objects, Data encapsulation, Inheritance and its types, Polymorphism, Abstraction and etc. Also, I will be discussing about Integrated Development Environment (IDE) and SQL-Server where we will be working our project. Also, I will be writing about Event Driven Programming Language briefly, IDE we have used for the project. I will also show how to connect the .Net app with a database.

## Procedural Programming Language (PPL)

A procedural language is a computer programming language that follows a set of commands in an order. It includes a systematic order of statements, functions and commands to complete a task or program. It is common type of language used by script and software programmer. BASIC, C, C++, FORTRAN, Java and Pascal can be the examples of it.

### Features of PPL

* **Pre-defined functions** – Examples of pre-defined functions such as ‘printf’, ‘scanf’, can be used as a function that is already within a programming language that helps in easy work.
* **Local variables** – Local variables can be accessed within the specific block of code (not through the entire script).
* **Global variables** – It is usually a variable that’s declared outside every other function in the code. Dissimilar to a local variable, it can normally be used with all other functions.
* **Parameter passing –** Parameter passing enables the passing of variable to the procedure, for example, the number 2 in ‘numb (2)’.
* **Modularity –** It is applied when two different systems both have different tasks, but are grouped together to finish one larger task.
* **Procedures -** If you have a procedure in programming, the program in procedure will follow the procedure step by step, systematically.
* **Programming libraries –** They are the collection of pre-built codes, classes, values which can be used at any time by the program and its users.
* **Procedural programming paradigm –** Normally procedural programming would use step-by-step procedures / functions / methods to inform a computer of what it’s expected to do. These steps would be taken by the program until it’s achieved its own expected state. These procedures can normally only be requested at a point during a program.

### Advantages of PPL

* Large problems are divided into smaller programs known as functions.
* It focuses on process rather than data.
* Most of the functions share global data.
* Data moves openly around the system from function to function.
* Employs top-down approach in program designing.

### Drawbacks of PPL

* Since every function has complete access to the global variables, the new programmer can corrupt the data accidentally by creating function
* Similarly, if new data is to be added, all the function needed to be modified to access the data.
* It is often difficult to design because the components function and data structure do not model the real World.

### Limitations of PPL

There are limitations to what you can do with procedural programming. When you are using procedural programming you do not have much flexibility with it. It has to be one straight programming with calculations done precisely. Procedural programming is tightly packed. What I mean is without one line of code the program will not work. If you are to change part of the program you will need to change other parts in order for it to work correctly. Another limitation is that procedural programming will eventually become lots of lines of code that can be confusing to go through. If there is a bug in the program it will take a while to go through the lines of code to find the problem.

**Note:** To overcome the limitation of Procedural language, Object oriented language were developed.

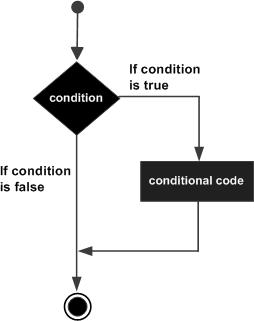
## **Control Statement**

The statement that allows the programmer to control the flow of execution of a program is known as Control Statement.

**TYPES:**

1. IF statement
2. IF ELSE statement
3. NESTED IF statement
4. ELSE IF statement**If statement:**

It takes an expression in program and a statement or block of statements. If the expression is true then the statement or block of statements gets executed otherwise these statements are skipped.

Syntax: Flowchart:

If (Boolean Expression)

{

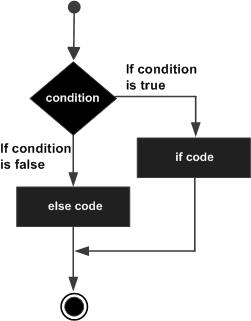
// statement(s)

}

|  |  |
| --- | --- |
| In the figure below, user input number 5, as it is more than 0, it displays message. | |
|  |  |

**IF/ ELSE statement -** The if/else statement executes a block of code if a specified condition is true. If the condition is false, another block of code can be executed.

Syntax: Flowchart:

If (Boolean Expression) {

//Statement(s) inside the body of it

}

Else {

// Statement(s) inside the body of else

}

|  |  |
| --- | --- |
| In the snapshot a below, when the user input number 5, it displayed ‘It is odd’. If the number was 6, it would display ‘It is even’. | |
|  |  |

**ELSE IF statement -** The ELSE IF statement executes two or more different codes depending upon whether the test expression is true or false. Sometimes, a choice has to be made from more than 2 possibilities. That means it allows you to check for multiple test expressions and execute different statement(s).

Syntax:

If (Boolean Expression 1) {

// Executes when Expression 1 is true

} Else if (Boolean Expression 2) {

// Executes when Expression 2 is true

} Else (Boolean Expression 3) {

|  |  |
| --- | --- |
| Here is the program of Else If statement which checks 2 or more expressions. | |
|  |  |

// Executes when none of the condition is true.

**Nested If statement -** A nested if is a statement that is the target of another if statement in another if statement.

Syntax:

If (Boolean Expression 1) {

// Executes when Expression 1 is true.

If (Boolean Expression 2) {

// Executes when Expression 2 is true.

}

}

|  |  |
| --- | --- |
| Example of Nested If to find the eldest one. | |
|  |  |

**Switch Statement -** A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each switch case.

Syntax:

Switch (n) {

Case 1:

//code to be executed if n is equal to 1;

Break;

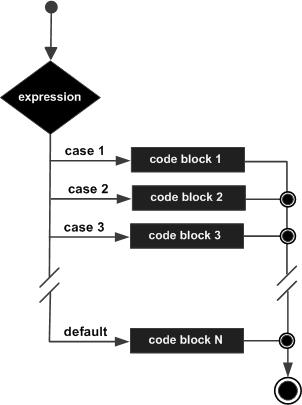
Case 2:

//code to be executed if n is equal to 2;

Break;

Default:

//code to be executed if n doesn’t match any case. }

Flowchart:

|  |
| --- |
| Example of Switch Statement to Add or Subtract. |
|  |
|  |

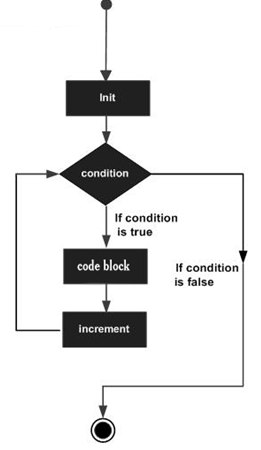
In the following example. We can do two operations; Addition and Subtraction. We choose 2 as subtraction.

## LOOP

A loop statement are those statement that execute the block of code several times according to given code.

**For Loop –** It executes the code until condition is false.

Syntax: Flowchart:

For (Initialization, Condition, Increment) {

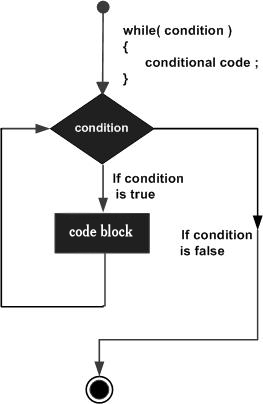
Statement(s);

}

|  |  |
| --- | --- |
| Example of For loop given below that prints 1 to until the condition is false. | |
|  |  |

**While Loop -** A **while** loop in C programming repeatedly executes a target statement as long as a given condition is true.

Syntax: Flowchart:

While (condition) {

Statement(s);

}

|  |  |
| --- | --- |
| Example of While loop to print 0 to 100. | |
|  |  |

**Do While Loop –** Unlike for and while loops, which test the loop condition at the top of the loop, do while loop checks its condition at bottom of the loop.

Syntax: Flowchart:

Do {

Statement(s);

} while (condition);

|  |  |
| --- | --- |
| Example of do while loop printing 1 to 10. | |
|  |  |

**Nested Loop –** Using one loop inside another loop is called nested loop.

Syntax:

For (Initialization; Condition; Increment) {

For (Initialization; Condition; Increment) {

Statement(s);

}

Statement(s);

}

|  |  |
| --- | --- |
| Example of Nested for loop. | |
|  |  |

## Array

An array is a collection of a fixed number of values of a single type. For example: if you want to store 100 integers in sequence, you can create an array for it.

Declaring Array:

Data\_type<Space>Array\_Name [Array\_Size];

Example - Int data [10];

Here, array size 5 means it can hold 5 integer values.

**Few key notes:**

* Arrays have 0 as the first index not 1. In this example, mark[0]
* If the size of an array is n, to access the last element, (n-1) index is used. In this example, mark[4]
* Suppose the starting address of mark [0] is 2120d. Then, the next address, a [1], will be 2124d, address of a [2] will be 2128d and so on. It's because the size of a float is 4 bytes.

Initializing an array:

It’s possible to initialize an array during declaration. Example:

Int mark [5] = {1, 2, 3, 4, 5} or Int mark [ ] = {1, 2, 3, 4, 5}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mark [0] | Mark [1] | Mark [2] | Mark [3] | Mark [4] |
| 1 | 2 | 3 | 4 | 5 |

Here, mark [0] is equal to 1, mark [1] equals o 2 and so on.

|  |  |
| --- | --- |
| Simple example of Array: | |
|  |  |

## Structure

Structure is a collection of variables (can be of different types) under a single name. Example; If you want to store information about a person; Name, Phone number, Salary. You can create different variables like Name, PhNo and Salary to store these information separately.

If you want to store information of more than one person then you can create variables for each person as Name1, PhNo1, and Salary1 etc.

Keyword **struct** is used for creating structure.

|  |  |
| --- | --- |
| Syntax of Structure: | Example: |
| Struct structure\_name  {  Data\_type member1;  Data\_type member2;  .  .  Data\_type member;  }; | Struct Person  {  Char name [100];  Int PhNo ;  Float Salary;  }; |

**Create Structure variable**

When a structure is defined, it creates a user-defined type. However, no storage or memory is allocated. To allocate memory of a given structure type and work with it, we need to create variables.

Struct Person

{

Char name [100];

Int PhNo;

Float Salary;

} Struct Person person1, person2, p [20];

Here, two variables named person1, person2 and an array variable p having 20 elements of type **struct Person** are created.

If you want to access the members of a structure. Suppose, you want to access salary of person2 you can do this: “Person2.salary”

|  |
| --- |
| Example of Structure. |
|  |
|  |

## Pointers

Pointers are easy and fun to learn. Pointers make some programming tasks easy. Pointers are powerful features of C and C++ programming.

**Address in C**

Here, if you have a variable named **var** your program, **&var** will give you its address in the memory, where & is commonly called the reference operator

Scanf (“%d”, &var);

|  |  |
| --- | --- |
| Here, you may obtain different value of address while using the code. Value 5 is stored in the memory location 6487628. **Var** is just the name given to that location. | |
|  |  |

Pointer Variables

In C, you can create a special variable that stores the address (rather than the value). This variable is called pointer variable.

Create a pointer variable:

Data\_type\* pointer\_variablename;

Int\* p;

Here, p defines the pointer variable of data type int.

**Reference operator (&) and Dereference operator (\*)**

Here, reference operator ‘**&’** represents the address of a variable and dereference operator ‘**\*’** gets you the value from the address.

**Note:** The ‘\*’ sign when declaring a pointer is not a dereference operator. It is just a similar notation that creates a pointer.

## String

String is a collection of characters. There are two types of strings used in C++ programming language.

* The standard C++ library string class.
* C-Strings (C-Style Strings)

**C-Strings**

In C, the collection of characters is stored in the form of arrays, this is also supported in C++ programming. Hence it is called C-strings. It is arrays of type **char** terminated with null character, this is \0 (ASCII value of null character is 0).

Defining C-strings – **char str [ ] = “string”;**

In above code, str holds 7 characters although ‘string’ has 6. Because null character \0 is added to the end of the string automatically. We can also define a string in following ways:

* char str [7] = “string”;
* char str [ ] = {‘s’,’t’,’r’,’i’,’n’,’g’,’\0’};
* char str [7 ] = {‘s’,’t’,’r’,’i’,’n’,’g’,’\0’};

But it is not necessary to use all the space given for the string. Example,

Char str [100] = “string”;

**Read string from the user:**

You can use the **scanf()** function to read a string.

|  |  |
| --- | --- |
| Example: | |
|  |  |

**Read a line of text:**

You can use **gets()** to read a line of string and use **puts()** to display it.

|  |  |
| --- | --- |
| Example: | |
|  |  |

## Function

A function is a block of code that performs a specific task which is reusable anytime. Suppose you are coding in C language, and in one point you need to perform a same task more than once, you can use function. We have two types of functions whether a function is predefined or created by the programmer:

1. Library Function

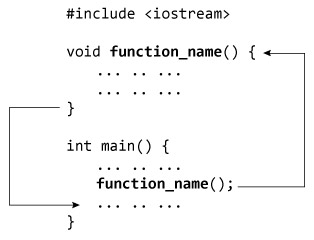
2. User-defined function.

**Library Function**

Library functions are the built in programming language where programmers can use library function directly, no need to write it by themselves. Example: the **printf ()** is a standard library function and this function is defined in ‘**studio.h’** header file. There are many other standard library functions such as **‘scanf ()’**, **‘getchar ()’, etc.**

**User-defined function**

In this function, programmer can define their own function. It code to perform a specific task and that group of code is given a name (identifier).

Working of user-defined function:

When a program start to begin, the system calls the **main()** function, that is, the execution of program begins from the **main()** function. When control of the program reaches to **function\_name()** inside main(), it moves to void **function\_name()** and all codes inside **void function\_name()** is executed. Then, control of the program moves back to the main function where the code after the call to **the function\_name()** is executed as shown in figure above. We can call the function anytime we need to.

Note: function name is an identifier and should be unique.

**Advantage of user-defined function**

1. The program will be easier to understand, maintain and debug.
2. Reusable codes that can be used in other programs.
3. A large program can be divided into smaller modules. Hence, a large project can be divided among many programmers.

**Function Prototype (declaration)**

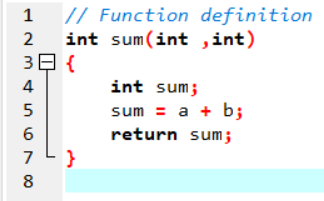
If the user-defined function is defined after **main()** function, compiler throws error. It is because compiler is unaware of user-defined function, types of argument passed to function and return type. You can declare prototype by:

**int add(int, float)**

There is no body of function in prototype. Also, only one return type of arguments but no arguments. You may also declare function prototype as below but it is your choice to:

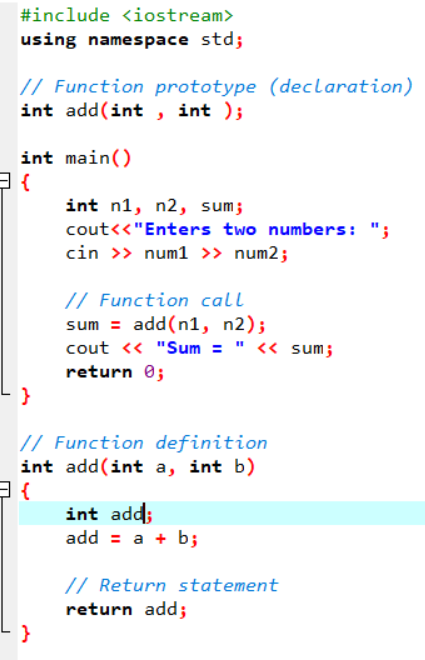
**int add(int a, float b)**

**Function Definition**

When the function is called, control is transferred to the first statement of the function body. Then then statements in function body are executed sequentially.

**Passing Arguments to Function**

Arguments (parameter) refers to the data which is passed to a function while calling it. In the following example of function, n1 and n2 are the actual arguments whereas they are initialized to variables **a** and **b**, which is called formal arguments.



Notes on passing arguments:

* The number of actual and formals arguments must be same. (Exception case in Function Overloading).
* The type of first actual argument should match the type of first formal argument. Similarly, type of second actual argument should match the type of second formal argument and so on as shown in example above. But, it is not necessary to have both arguments of same type.

|  |  |
| --- | --- |
| Example of function: | |
|  |  |

## Object Oriented Programming Language (OOP)

OOP is a programming language model organized around objects rather than actions and data rather than logic. Here, the data structure becomes an object that includes both data and functions. In addition, programmers can create relationships between one object and another. Example: objects can inherit characteristics from other objects. It is associate with class, object, inheritance, encapsulation, abstraction and polymorphism. Example of OOP includes JAVA, C++ and Smalltalk.

### Advantage of OPP

One of the advantage of Object Oriented Programming over Procedural Programming is that they enable programmers to create modules that do not need to be changed when a new object is added. This makes Object-oriented programs easy to modify. The programs written with OOP are really easy to understand. Since everything is treated as objects, so we can model a real world’s concept using OOP. OOP approach offers the reusability of classes. We can reuse the classes that are already created without writing them again and again. Since the parallel development of classes is possible in OOP concept, It results in the quick development of the complete programmers. Programs written in OOP technique are marginally easier to test, manage as well as maintain. It is a secured development technique since data is hidden and can’t be accessed by external functions.

### Drawbacks of OOP

Sometimes, the relation among the classes become artificial in nature. Designing a program in OOP concept is a little bit tricky. The programmer should have a proper planning before designing a program using OOP approach. Since everything is treated as objects in OOP, the programmers need proper skill such as design skills, programming skills, thinking in terms of objects etc. The size of programs developed with OOP is larger than the procedural approach. Since larger in size, that means more instruction to be executed, which results in the slower execution of programs.

### Features of OOP

**Class:** Class is a blueprint of data and functions or methods. Class does not take any space. A class is defined using keyword **class** followed by the name of class.

Syntax for class:

Class class\_name

{

   Private:

      //data members and member functions declarations

   Public:

      //data members and member functions declarations

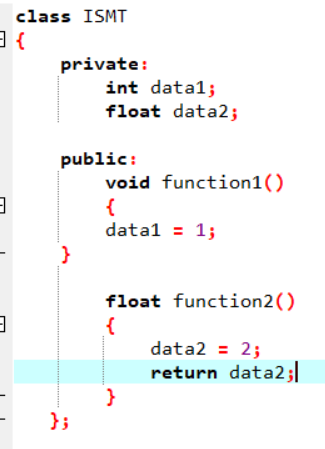
  Protected:

      //data members and member functions declarations

};

|  |
| --- |
|  |

Class is a user defined data type like structures and unions in C.

Here, we have created a class named **ISMT** which has two data members; data1 and data2, also, two member functions; function1() and function2().

Here, the **private** keyword makes data and functions private, i.e. it can only be accessed from inside the same class.

And, **public** keyword makes data and function public, i.e. it can be accessed by anyone.

If you try to access private data from outside of the class, it throws error.

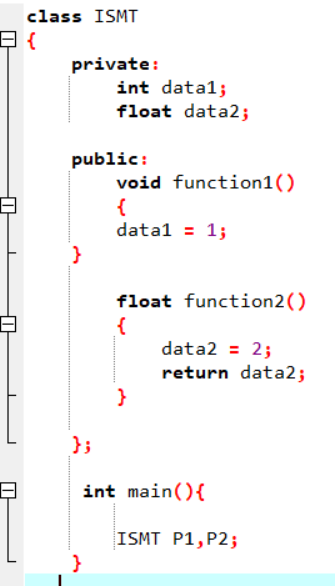
**Objects in Class**

When class in defined, only the specification for the object is defined, no memory is allocated. So, to use the data and access functions defined in the class, you need to create objects.

**Syntax for defining objects**

Class\_name ObjectVariableName;

Let us create an object of **ISMT** class defined in above example:

 Here, two objects P1 and P2 are created of class ISMT.

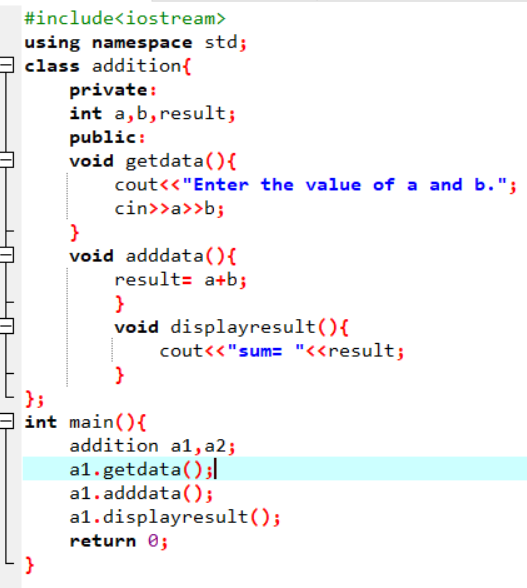
You can access the member function and by using a “. (Dot) operator”. Example,

P1.function2 ();

This will call function2 () inside the ISMT class for object P1.

Similarly, you can access the data member by,

P1.data2 = 2;   
Remember that the private members can be accessed only from inside the class.

Here, we have created a class named **addition**, where the member function are defined as **getdata()**, **adddata()** and **displayresult().** Two objects named a1 and a2 are created under class **addition** where a1 access the member function.

**Encapsulation:** Combing of data and functions into a single unit is known as encapsulation. The data is not accessible to the outside world and only those functions which are wrapping in the class can access it. This insulation of the data from direct access by the program is called data hiding or information hiding. Above snapshot can be example of Encapsulation.

**Data abstraction:** Data abstraction refers to, providing only needed information to the outside world and hiding implementation details. For example, consider a class Complex with public functions as getReal () and getImag (). We may implement the class as an array of size 2 or as two variables. The advantage of abstractions is, we can change implementation at any point, and users of Complex class won’t be affected as out method interface remains same. Had our implementation be public, we would not have been able to change it.

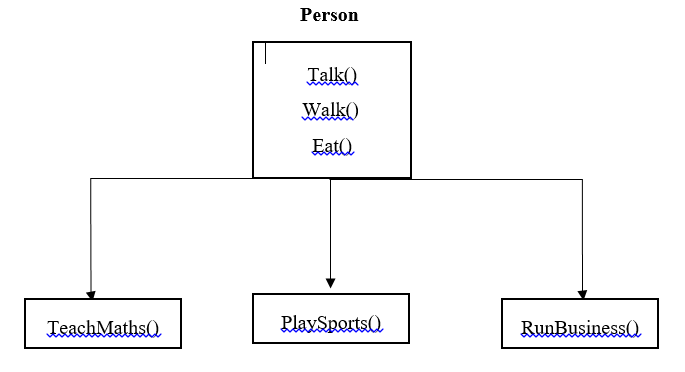
**Inheritance:** inheritance is the process by which objects of one class acquire the properties of objects of another class. It supports the concept of hierarchical classification. Inheritance provides re usability. This means that we can add additional features to an existing class without modifying it.

Suppose, in your game, you want three characters, math teacher, a sports man and a businessman. Since, all are human who can talk and walk. Here they can perform their skills as shown in below:

|  |  |  |
| --- | --- | --- |
| Math Teacher | Sport Player | Businessman |
| Talk()  Walk()  TeachMath() | Talk()  Walk()  Play() | Talk()  Walk()  Business() |

In each of the classes, you would be copying the same code for talk, walk for each character. If you just want to add the new feature – sleep (), you need to implement same code for each character.

Now, it would be easier if we had a Person class with feature like talk, walk, etc. This is done using inheritance.



**Polymorphism:** polymorphism means ability to take more than one form. An operation may exhibit different behaviors in different instances. The behavior depends upon the types of data used in the operation. C++ supports operator overloading and function overloading. The process of making an operator to exhibit different behaviors in different instances is known as operator overloading.  
Function overloading is using a single function name to perform different types of tasks.  
Polymorphism is extensively used in implementing inheritance.

## Event-Driven Programming Language

Event-Driven programming language is a programming paradigm in which the flow of program execution is determined by events. For example, a user action such as mouse click, key press, or a message from the operating system or another program. Concept of this language is an important in application development and other kinds of programming. It is also known as an event-driven application.

One of the example of Event-driven programming is Visual Studio where it reacts specific kinds of input, whether it’s a click on a button, an entry into a text box etc. Other example are Visual C++ and JAVA.

### Advantages and disadvantages

Programs that use event driven, which could be altered easily if the programmer wants something to be change. It lets the programmer to produce a form to their requirements. Also provides the programmer to be in charge. Programs which are event driven can be put together without any problems and also the program can be easily altered because if something isn't right then it will let the user to change a piece of code or moving different tools around the form. Ease of development means how straightforward the program is. Also the programmer has to manage one control at one time. The reason why the programmer has to do this is that every control which is on the form are all programmed separately. The Programmer can put different things on top of the form, which can make it simpler for the programmer. Visual basic is used to make programs because they are different tools and also it isn't just code, you can add picture boxes and text boxes. Similarly this helps simplicity in programming, ease of development, flexibility and Suitability for graphical interfaces

Talking about constraints, this programming language is often more complex than batch programming. The flow of the program is usually less logical and obvious. This programs are only useful to GUI programming. If a company trained their employees in event driven but they are unknown about GUI program then this would not be good.

### Key features and characteristics of event-driven programming

1. **Service Oriented** - Service oriented programmes are programmes that often appear in the background and have no user interface to use but all have the ability to be stopped and started at the users will. These programmes can also be set to activate immediately when your pc starts up. Examples of these sort of programmes are Bluetooth, Autodesk functions other programmes that offer customizable features that are not necessarily essential for your computer to have. Also because these programmes are constantly using your memory for background programmes which may be non-crucial to the running/ maintaining of your pc it can be seen by many as a waste of memory for some of the functions being run.
2. **Time Driven -** Time Driven programmes such as antivirus scans, task scheduler and operating system updates are important features in the maintain and running of your pc. In event driven programming, time driven is a paradigm, it’s a code that runs on a time trigger, time driven can be a specific code that runs on a specific time, which could be once an hour, once a week or once a month, this means it’s a pre-set to do task. For example, windows update is the example of time driven, which user can set when to update or when to check and download the update.
3. **Event Handlers -** Event handlers is a type of function or method that run a specific action when a specific event is triggered. For example, it could be a button that when user click it, it will display a message, and it will close the message when user click the button again, this is an event handler.
4. **Trigger functions -** Trigger functions are used to be able to designate a specific coded event with an action e.g. click the button to trigger the calculation on a calculator which works out the result based on your input numbers until the event is triggered.
5. **Events** - Events include mouse, keyboard and user interface, which events need to be triggered in the program in order to happen, that mean user have to interacts with an object in the program, for example, click a button by a mouse, use keyboard to select a button and etc.

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

|  |  |
| --- | --- |
| Procedural Programming Language | Object-oriented programming |
| It is based on how to get the task done, i.e. on the procedure or structure of a program. | It is based on data security, only objects are permitted to access the entities of a class. |
| It follows top-down approach. | It follows button-top approach. |
| Programs are divided into units, i.e. functions. | Entire program is divided into objects. |
| PPL doesn’t access specifier observed. | OOP access specifier are ‘public’, ‘private’ and ‘protected’. |
| It doesn’t overload functions nor operators. | Inheritance can be done. |
| There is no proper way of hiding the data, where the data is insecure. | Data is hidden in three modes ‘public’, ‘private’ and ‘protected’. |
| Example: C, VB, FORTRAN, Pascal. | Example: C++, JAVA, VB.NET, C#.NET. |
| Event-driven Programming |
| It is based on the flow of program execution determined by events, i.e. user action such as mouse click, key press. |
| EDP can be written in any programming language, although some language (Visual Basic) are specially designed to facilitate EDP. So, many features of PPL and OOP are include here. |
| Example: Visual Studio, Visual C++, Java etc. |

|  |  |
| --- | --- |
| Example of all 3 programming language. | |
| PPL | OOP |
|  |  |
| EDP |
|  |

## Integrated Development Environment (IDE)

IDE is an application that facilitates application development (source code editor, compiler, debugger, etc.) which is a GUI based workbench designed to help developer in building software applications. It helps maximize productivity by providing similar user interface for related components. An IDE supports single or multiple languages. Visual Studio, NetBeans, Eclipse, etc. are some example of IDE.

### Benefits of IDEs

When using IDE, it can improve the productivity of software developers. Without an IDE, developers spend time deciding what tools to use for various tasks, configuring the tools. IDE automatically checks for errors to ensure top quality code. It can maintain a smooth development cycle too. Debugging, Compiling can be useful for the programmers.

## Tools available in IDE

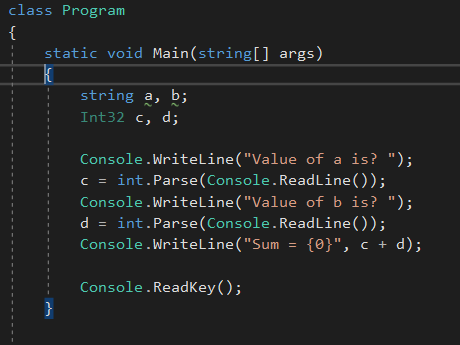
**1) Code Editor**

A code editor is a text editor program designed for editing source code of computer program. It is an independent application which is built into Integrated Development Environment (IDE). They are fundamental programming tool, which is fundamental jobs of programmer to write and edit source code.

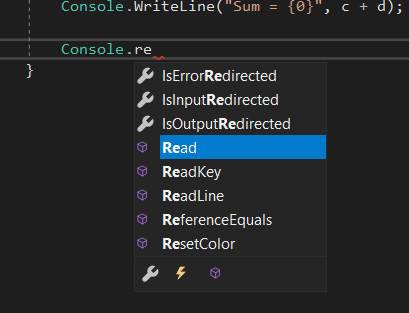
**Features of Code Editor**

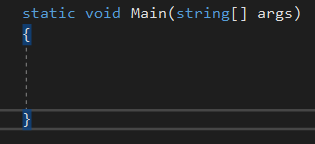
* **Syntax highlighting:** In this feature, it displays text into different colors and fonts according to the category of terms. If there is syntax error in coding, they are easily distinct by highlighting it. It makes easier to view the source code. Some editors also integrate syntax highlighting with other features like spell checking and code folding.

As you can see, visual studio has a syntax highlighting feature on it.



* **Code Completion:** When you begin your code, you can see a small drop-down list showing you a choice of how to finish what you just started typing. It will automatically suggest the code you are typing. Example:



* **Bracket Matching:** This feature shows the highlighting of brackets (square, curly) which we use in the program. It spots the improper matching, which would cause the program to not to execute. You can see the highlights in the brackets in following snapshot.

**2) Debugger**

Debugging tools helps in identifying and resolving errors within a source code. They often simulate real-world scenarios to test functionality and performance. Programmers and software engineers usually test the various group of code, identify errors and solve it before the application is released.

Note: We will be talking about it in next part.

**3) Build Automation**

Build automation is the process of automating the creation of a software build and the associated processes including compiling computer source code into binary code, packaging binary code and running automate tests.

There are two categories of tools:

**1) Build-automation utility-** It includes utilities like Make, Rake, MS build, Ant, etc. Their purpose is to generate build activities like compiling and linking source code.

**2) Build-automation servers-** It is a tool that execute build-automation utilities on a scheduled or trigger basis.

**Connecting Visual Studio and Database**

Since we have used Visual Studio and SQL-Server for our project. Here are the steps for connecting Database SQL-Server in Visual Studio.

**Step - I**

|  |
| --- |
| Open Visual Studio. Explore >>Tools and click Connect to Database as shown in snapshot. |
|  |

**Step – II**

|  |  |
| --- | --- |
| Copy this Server name from your SQL-server. | Paste it in Server name. Secondly, select your database name which you want to connect. Finally click Advanced for the Source Code. |
|  |  |

|  |  |
| --- | --- |
| Copy all the Data Source. |  |

**Step – III**

**Step – IV**

|  |  |
| --- | --- |
| Make a new Sqlconnection and paste the copied Data Source as shown in snapshot. | Don’t forget to write this code ‘using System.Data.SqlClient’ at very first. |
|  |  |

**Hence the connection is finally created between them. Now, you can code accordingly.**

## Conclusion

Hence, I have research about the programming paradigms which include procedural, object-oriented and event-driven programming including their advantages, disadvantages of using programming language approaches. This part also includes analysis of suitable IDEs, an evaluation of source code.

# Part III

The submission is in the form of five documents/files:

1. Stage 1 – Development Document

2. Stage 2 – Report (IDE Evaluation)

3. Stage 3 – Report (Debugging Evaluation)

4. Stage 4 – Report (Evaluation Report) including fully commented source code

5. An installable and executable version of your application.

You are required to make use of appropriate structure, including headings, paragraphs, subsections and illustrations as appropriate, and all work must be supported with research and referenced using the Harvard referencing system.

As part of the application process they senior management team want to see that you can implement algorithms using an IDE as required.

You are asked to create a fully working, secure application that has been developed using an IDE and adhere to coding standards.

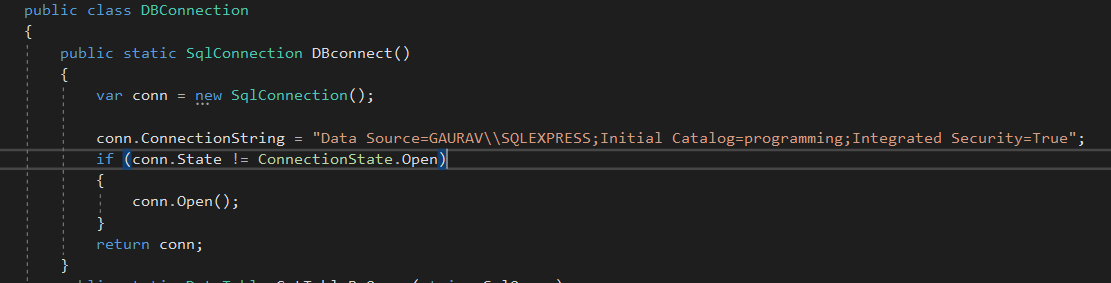
The document portfolio should include \*Evidence of how the IDE was used to manage the development of your code. \*An evaluation of developing applications using an IDE versus developing an application without using an IDE. \*An evaluation of the debugging process in the IDE used and how it helped with development. \*An evaluation of coding standards and the benefits to organizations of using them. The working application produced must also be demonstrated.

## Introduction

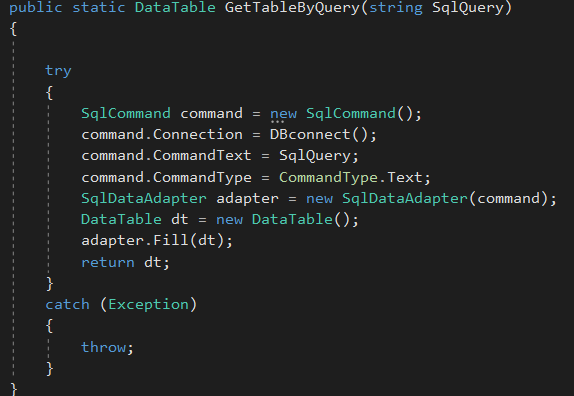
At the end, completing my part 1 and 2, this part is about how the IDE was used to manage the development of your code and design and code of our application. Secondly, an evaluation of developing applications using an IDE versus developing it without an IDE. Also, description of debugging process in IDE used and coding standards and benefits to organization of using them.

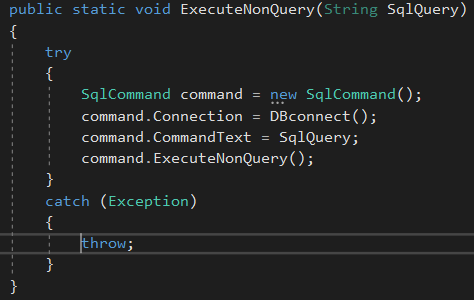
## Design and Code

In our project – Visual Studio, I made a Class named ‘Dbconnection’ so that it could be easier to call it in every form. It will be easy and fast to code.

In the following snapshot, SqlConnection represents an open connection to a SQL Server Database. It is used when I have to connect the project to SQL Database and open the Connection State.

Here, following Class is used when I have to show the data in the form of table like Data Grid. Data Table represents in one table of in-memory data. SqlCommand represents a T-SQL statement or stored procedure to execute against SQL Server Database. SqlDataAdapter represents a set of data commands and a database connection used to fill the DataSet. Try and Catch represents the error handling, we can find where the error is.

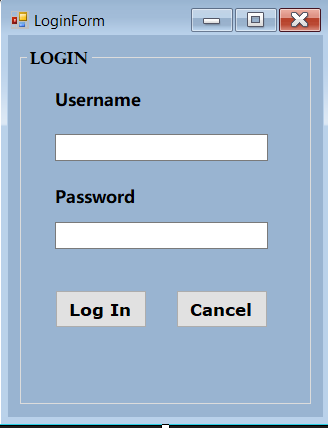


ExecuteNonQuery class is applied when we have to execute a SQL statement or stored procedure from SQL Server database.

1. **Login Page**

Login page is the gateway of main form (MDI form)

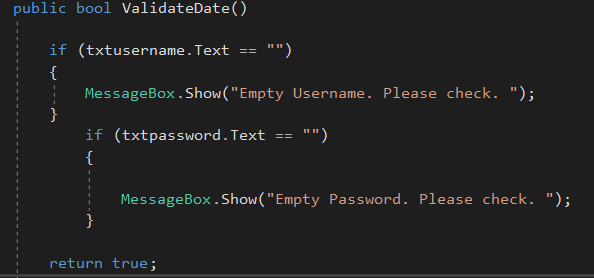
**Design:**



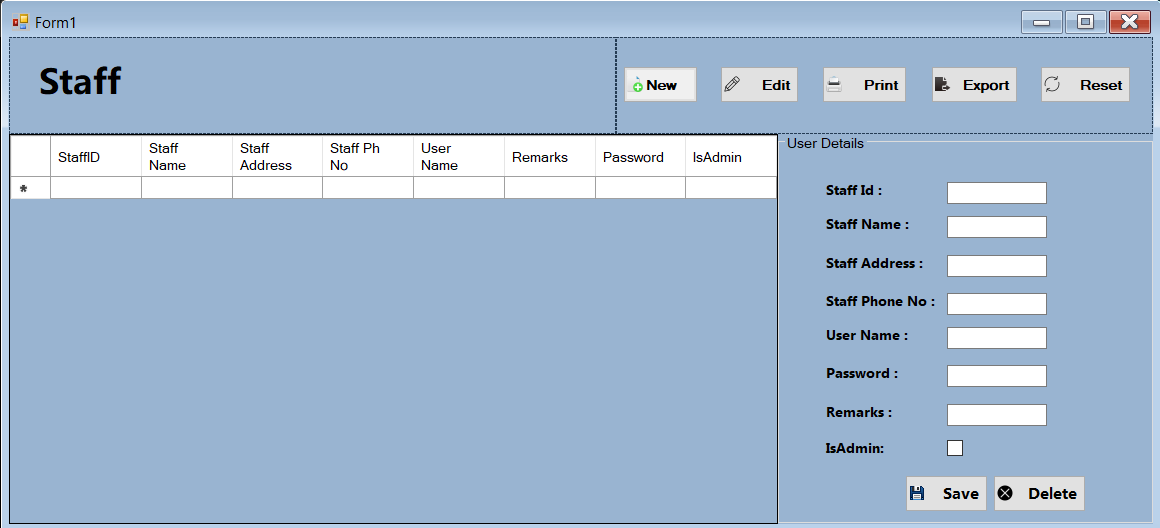
**Code:**

This is the code for Login button, where we passed parameter to prevent SQL injection. Here, the Username and Password are taken from table Staff in database.

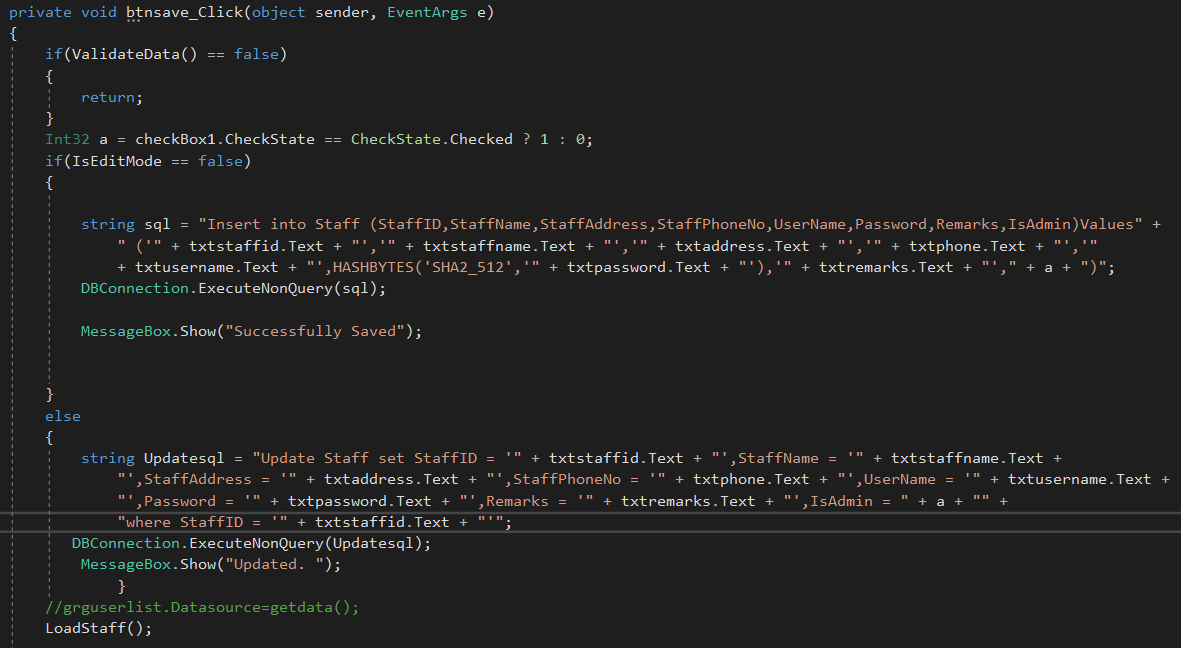


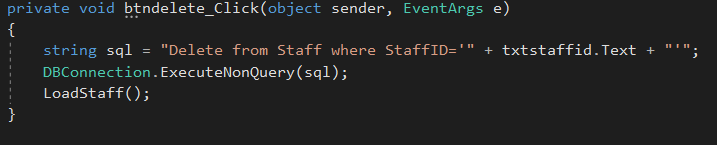
The following code are used to validate the data, which means if user inputs empty data, it notifies you.

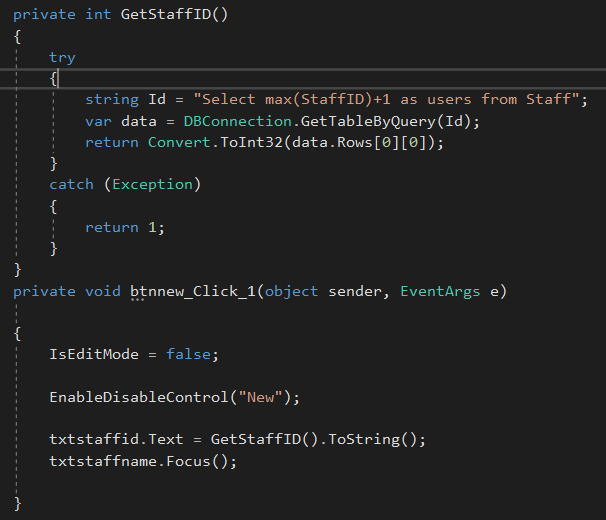
1. **Staff table**

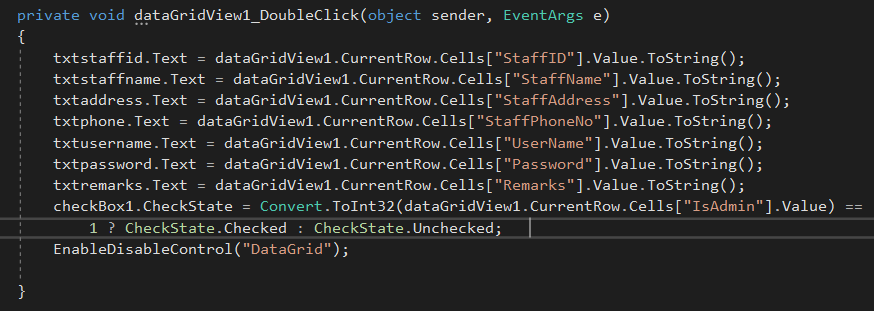
**Design:**

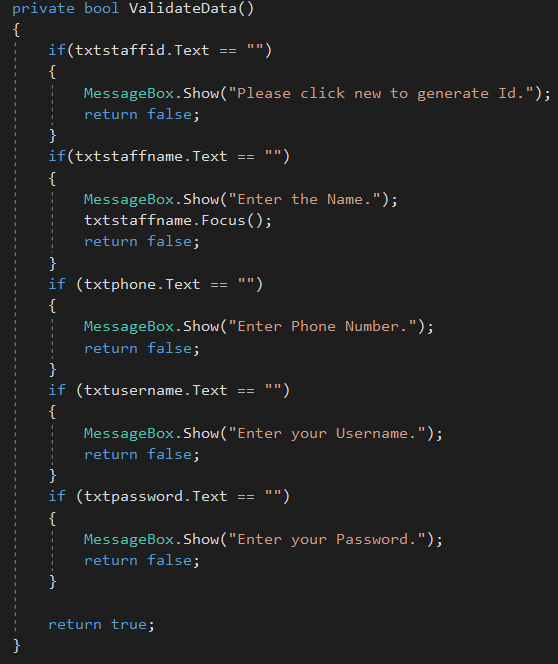
**Code:**

**For Save and Update button:**

**Delete button:**

**New button:**

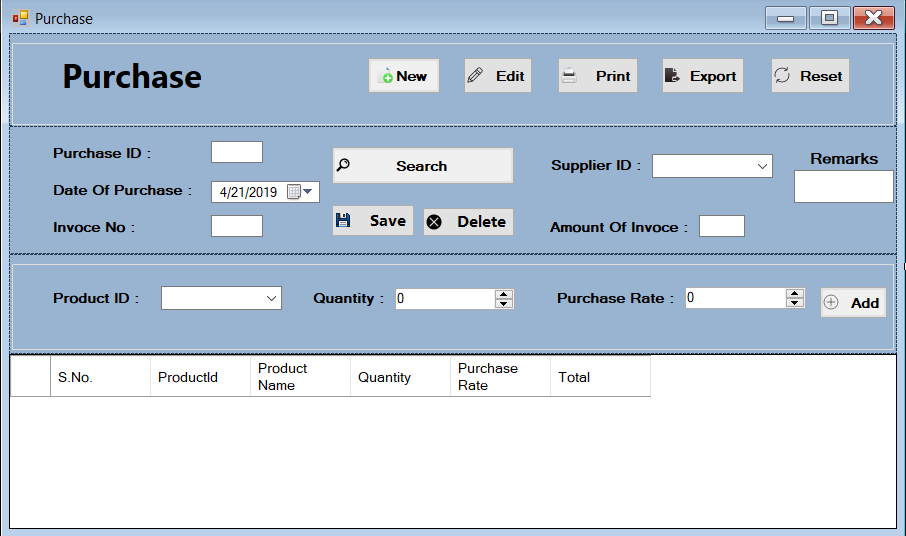
**Code to display data from data grid view to text box, used for updating and deleting the data.**

**Code for validation of data.**

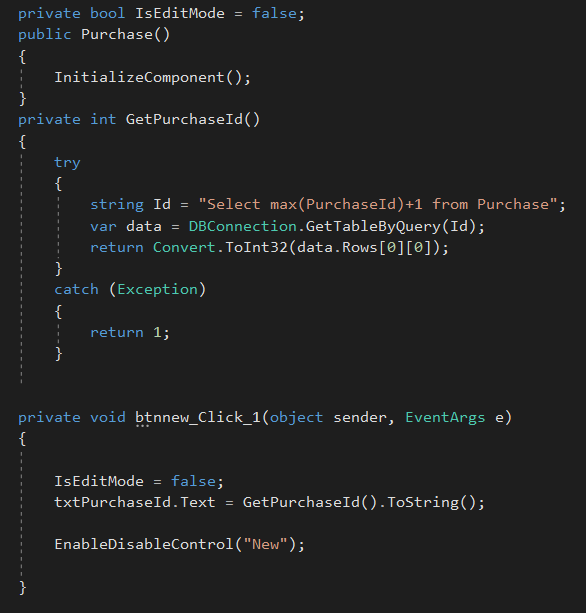
1. **Purchase**

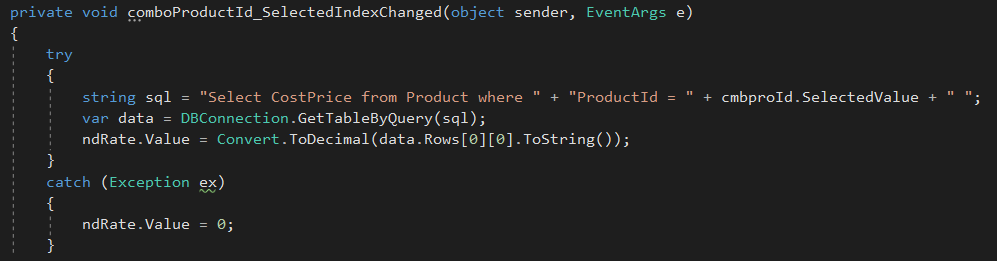
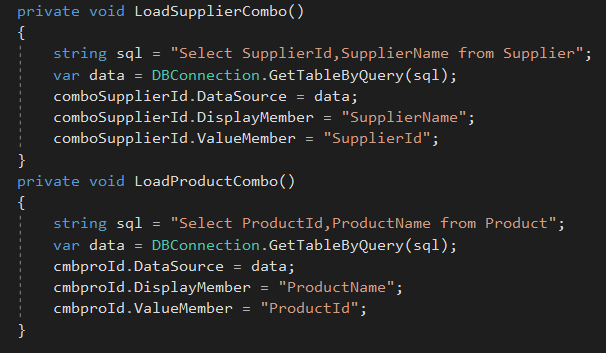
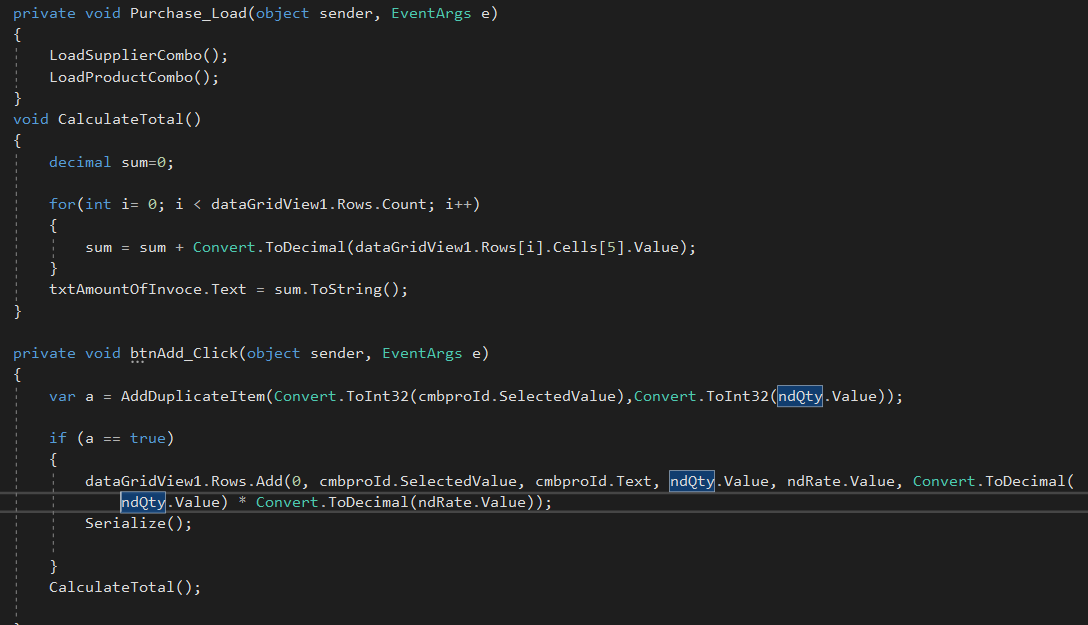
This is the form where we purchase the product.

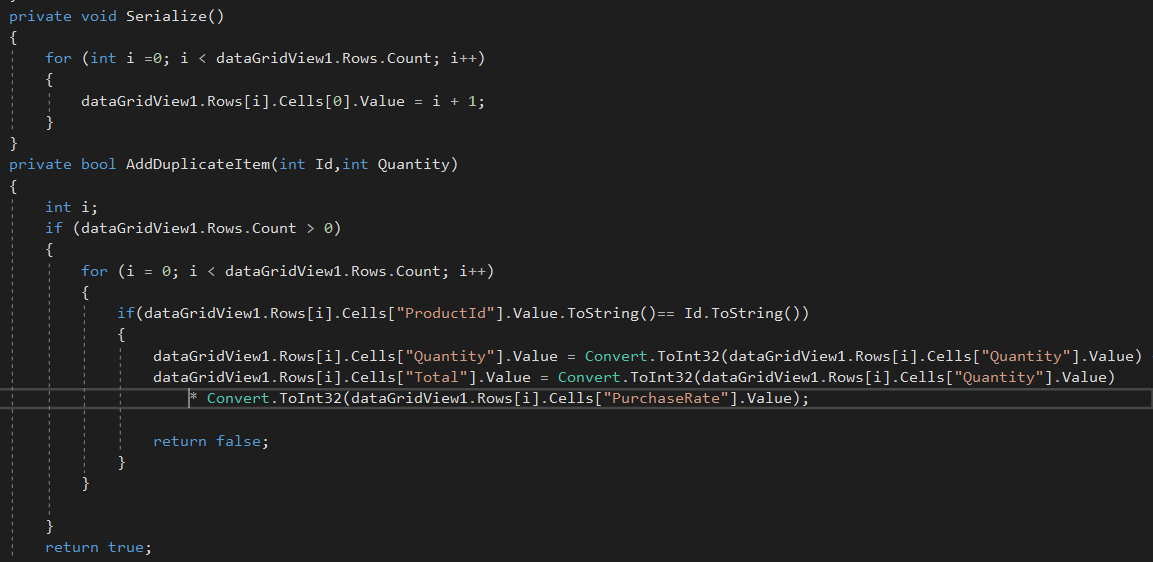
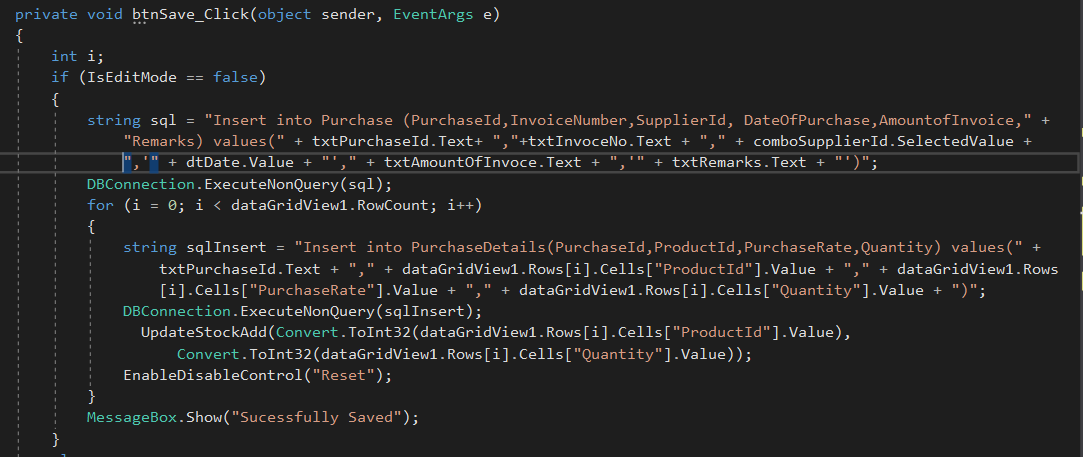
**Design:**

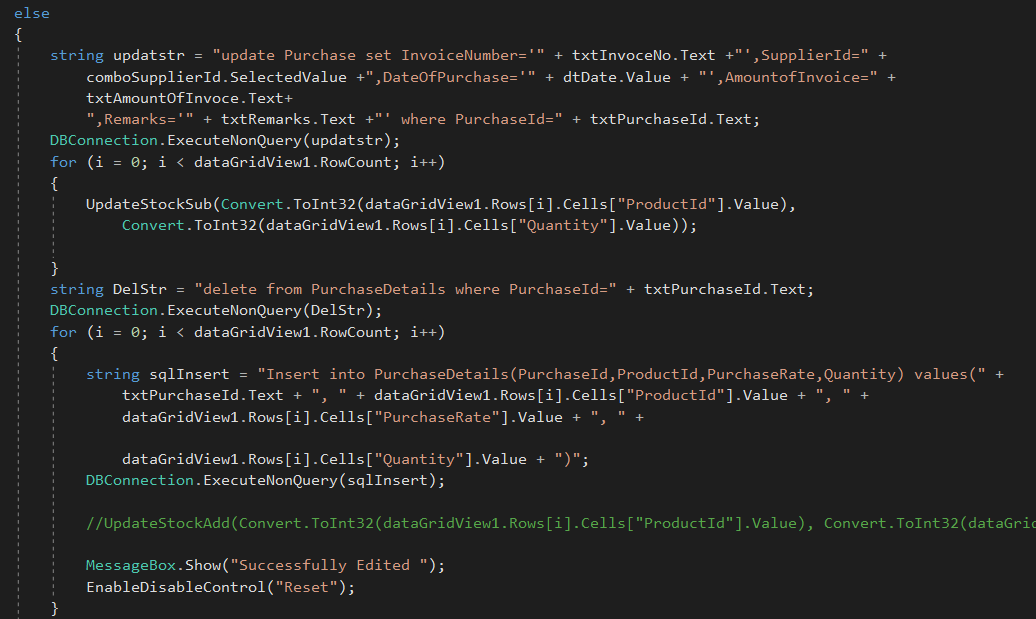
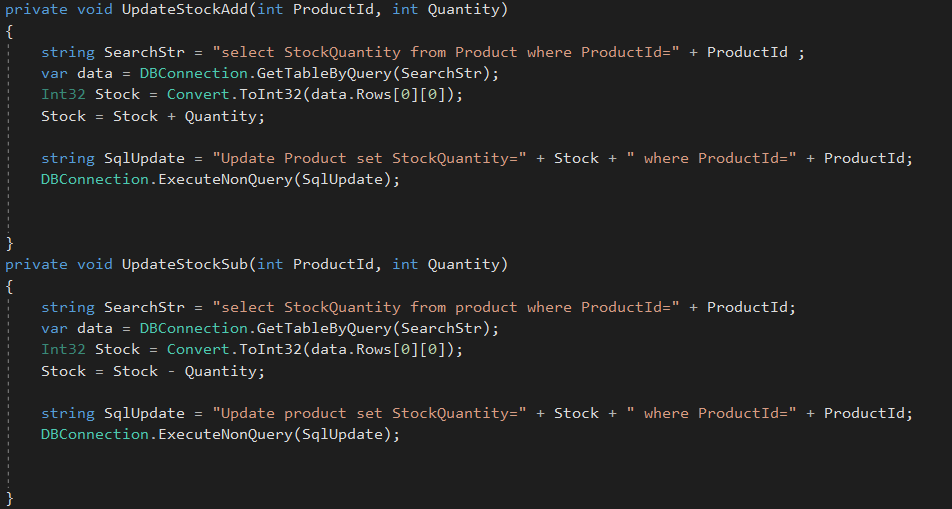


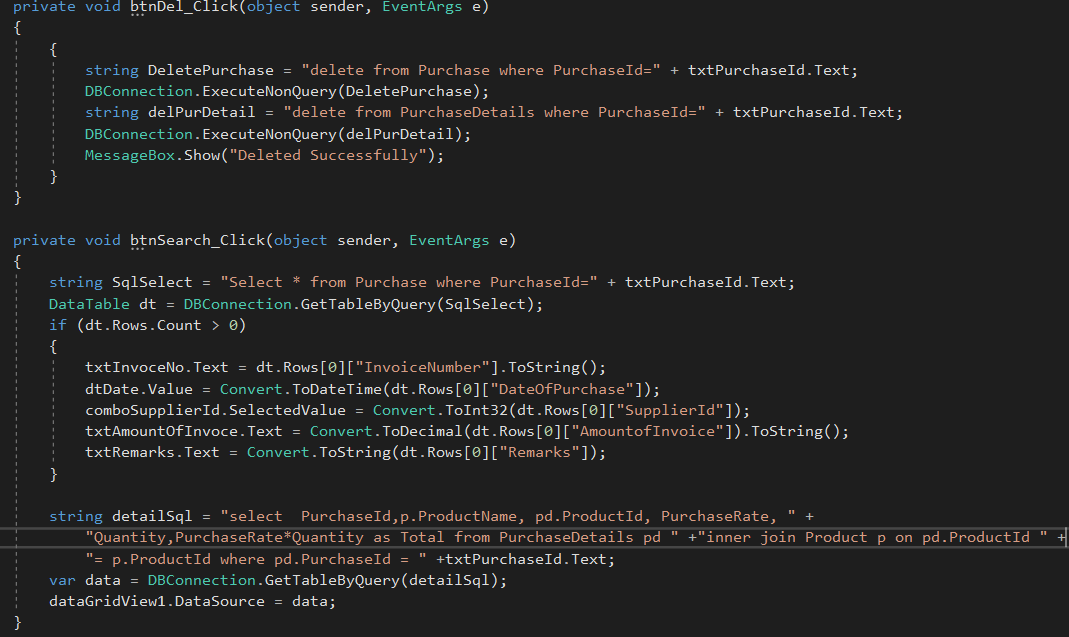
**Code:**





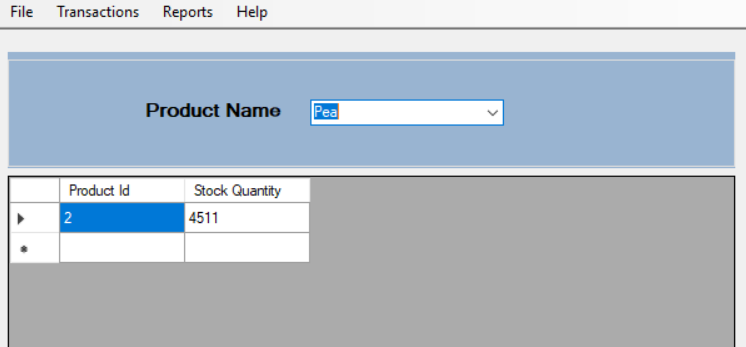


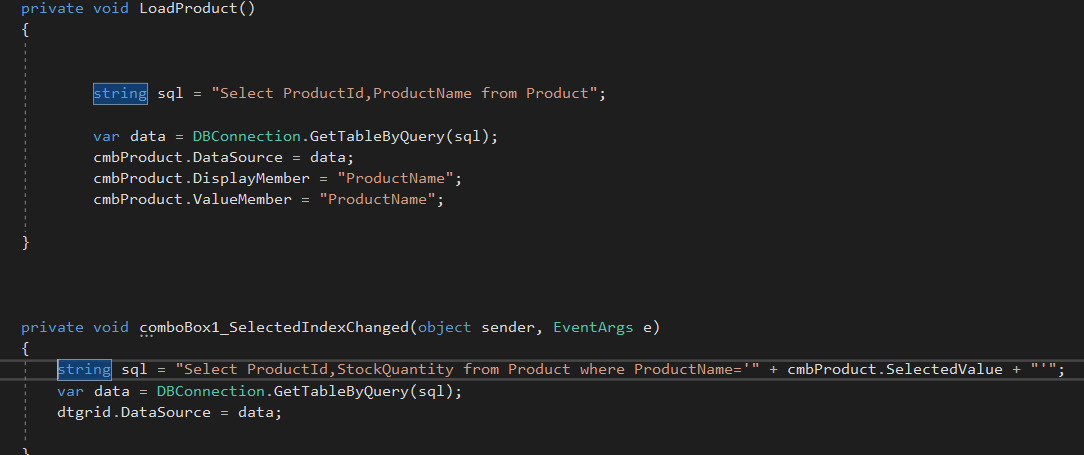




1. **Report**

Stock report with Product is shown below.

**Design:**

**Code:**

## Integrated Development Environment (IDE)

IDE is an application that facilitates application development (source code editor, compiler, debugger, etc.) which is a GUI based workbench designed to help developer in building software applications. It helps maximize productivity by providing similar user interface for related components. An IDE supports single or multiple languages. Visual Studio, NetBeans, Eclipse, etc. are some example of IDE.

### Benefits of IDEs

When using IDE, it can improve the productivity of software developers. Without an IDE, developers spend time deciding what tools to use for various tasks, configuring the tools. IDE automatically checks for errors to ensure top quality code. It can maintain a smooth development cycle too. Debugging, Compiling can be useful for the programmers.

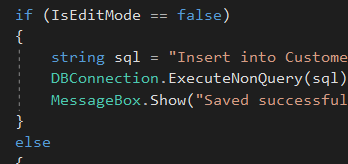
## Tools available in IDE

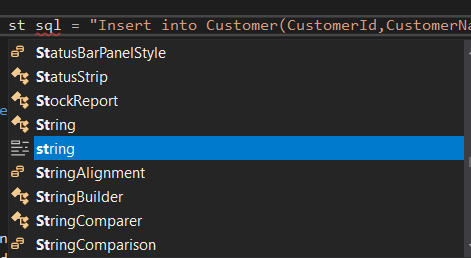
**1) Code Editor**

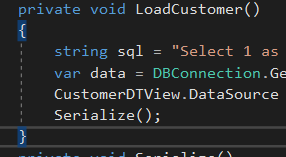
A code editor is a text editor program designed for editing source code of computer program. It is an independent application which is built into Integrated Development Environment (IDE). They are fundamental programming tool, which is fundamental jobs of programmer to write and edit source code.

**Features of Code Editor**

* **Syntax highlighting:** In this feature, it displays text into different colors and fonts according to the category of terms. If there is syntax error in coding, they are easily distinct by highlighting it. It makes easier to view the source code. Some editors also integrate syntax highlighting with other features like spell checking and code folding.

As you can see, visual studio has a syntax highlighting feature on it.

* **Code Completion:** When you begin your code, you can see a small drop-down list showing you a choice of how to finish what you just started typing. It will automatically suggest the code you are typing. Example:
* **Bracket Matching:** This feature shows the highlighting of brackets (square, curly) which we use in the program. It spots the improper matching, which would cause the program to not to execute. You can see the highlights in the brackets in following snapshot.



**2) Debugger**

Debugging tools helps in identifying and resolving errors within a source code. They often simulate real-world scenarios to test functionality and performance. Programmers and software engineers usually test the various group of code, identify errors and solve it before the application is released.

Note: We will be talking about it in next part.

**3) Build Automation**

Build automation is the process of automating the creation of a software build and the associated processes including compiling computer source code into binary code, packaging binary code and running automate tests.

There are two categories of tools:

**1) Build-automation utility-** It includes utilities like Make, Rake, MS build, Ant, etc. Their purpose is to generate build activities like compiling and linking source code.

**2) Build-automation servers-** It is a tool that execute build-automation utilities on a scheduled or trigger basis.

## Debugging

Debugging is simply defined as the process of identifying and removing errors from the computer hardware or software which is handled by software programmers via debugging tools.

**Debugging in IDE tools (Visual Studio)**

Visual studio’s debugger can help you find problems in your code efficiently. It lets you execute the program and pause it before given lines of code are run, which will show what is your program is doing. Tools of debugging are described below:

**1) Breakpoints**

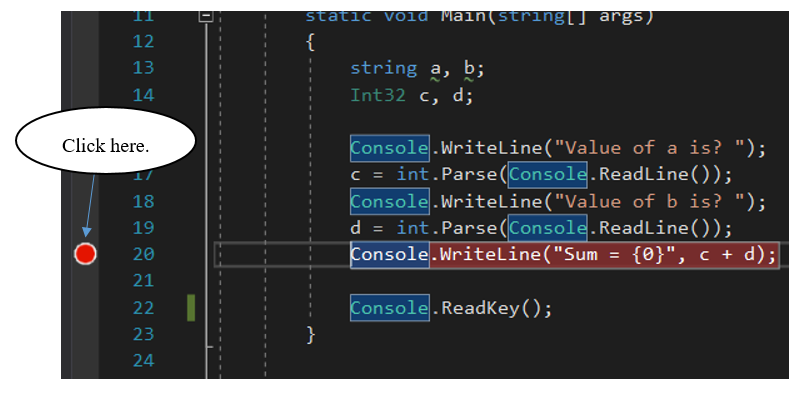
Breakpoints are one of the most used features by developers to debug application. Breakpoints break (temporarily halt) execution of the program during runtime at certain point.

**Steps to implement the breakpoint**

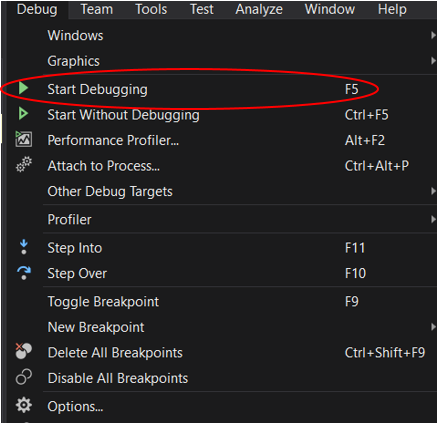
You can only set a breakpoint on any line of executable code. So, be on line of code and you can simply press **f9** where you want to set it.

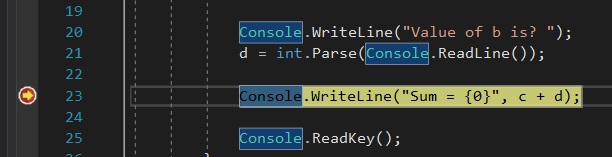
OR

Click on far left margin next to the number of lines displaying. There are more steps but this two would be enough. You will notice that the code gets marked in red, also, a red bubble comes up in next to line. You can also add multiple breakpoints



Now start to debug. Go to Debug and Click start debugging.

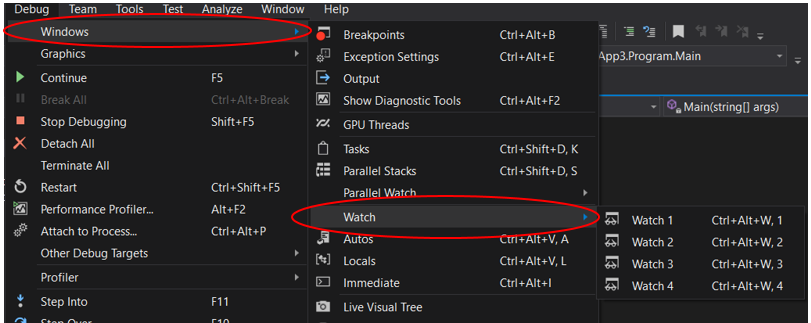


In the following snapshot, you can see the line yellowish in color highlighted which is paused before the execution. Now, if you feel the code is incorrect, the execution can be stopped and modify the code accordingly. If there is no such error, f5 will help to continue the execution.

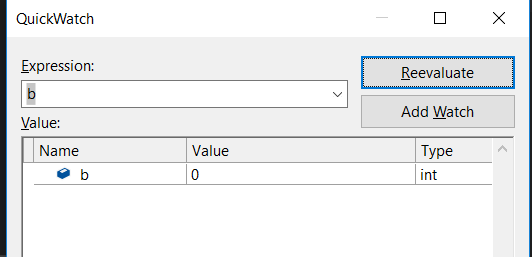
**2) Watch Point**

Watch point is a conditional breakpoint that helps application developers monitor the contents of specified variable and change their values during runtime processing.

**Steps to add a watch point**

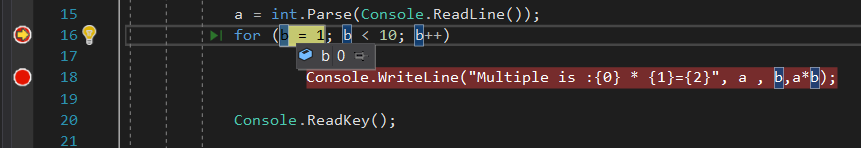
 Firstly, add a breakpoint as we already showed it above. Execute it and when it is paused, expand >>Debug, >>Windows, >>Watch, and Watch 1. Or simple press Ctrl+Alt+W,1.

Here, you can see the value of b is 0 as I input it.



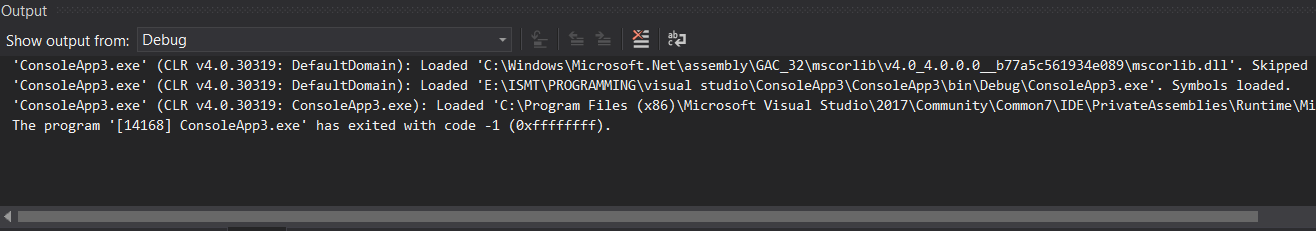
**3) Step Commands**

Step commands are used to navigate code in debugger. It is done in breakpoint. Keep a breakpoint and execute the program. To step into commands, press F11, it steps line by line in the code. This will show the flow of code in more details line by line.



You can see yellow line, keep on pressing F11, you will see the execution of code line by line.

**4) Output Window**

Output window is the window below the source code, which shows the output of the execution. It shows whether there is error or not. You can also see different kinds of information (from which location, our code is being executed, tracer, etc)

## Coding Standard

Coding standard is simply defined as a set of rules and guidelines for formatting the source code. It defines a programming style. A coding standard does not usually concern itself with wrong or right in a more abstract sense.

**Common aspects of Coding Standard**

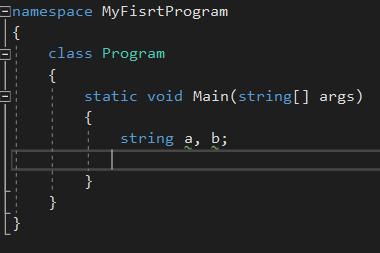
* Naming Conventions
* File Naming and Organization
* Formatting and Indentation
* Comments and Documentation
* Classes, Functions and Interfaces
* Pointer and Reference Usage
* Testing

**Coding Standard we have used are:**

**1) Indentation**

In Indentation coding standard we have used GNU style, where it puts braces on a line by themselves, indented by two spaces, except when opening a function definition. In this standard, braces are not standing out from the block. It is easier to look up the codes while coding.

One disadvantage of this standard is it might waste space and another is that the ending brace no longer lines up with statement it conceptually belongs to. Example:

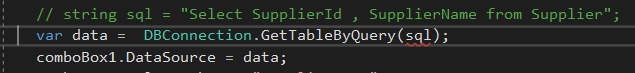


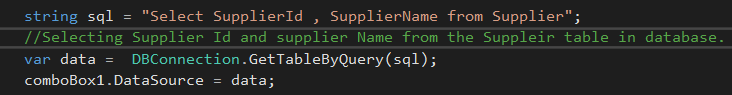
N bx

**Commenting the code**

Commenting the code is meant by the code that you write does not get compiled or execute. It simply temporarily disable the code and it is ignored by the program. Comments are done to write anything in basic language so that you can remember the process in the future or just to remind yourself that there is something left in the code.

**Line Comment:**

Commenting the specific line of code is called Line Comment.

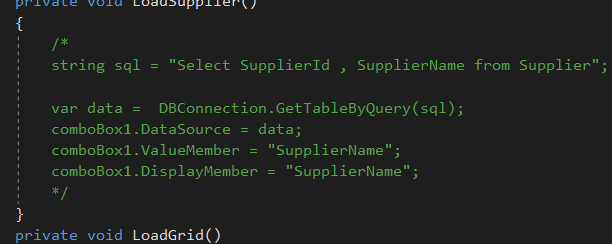


Here, the green part (code with “//”) is commented in the snapshot, program won’t execute that specific line of code. We can write anything in our own language, which can help to understand as shown in snapshot.

Simple way to comment the single line of code is add ‘//’ in the left side of code.

**Commenting Block of Code:**

Commenting the block of code is called Comment Block of Code.



Here, the block of code are commented. Use (/\*, \*/) in the code. /\* is the start of commenting the code and \*/ ends it. If you end the comment, entire code is commented.

**Naming Conversion**

Naming conversion is one of the most important elements of predictability and discoverability in a managed class library.

**Types of Naming Conversion:**

* **Pascal case-** In this case, the first letter in the identifier and the first letter of each subsequent in a series word are capitalized. Example:

4

Every first letter in a single word is capital.

We used **Pascal case** in \* Class name-

 \*Method name-

|  |  |
| --- | --- |
| Abbreviations | Standard Control |
| btn | Button |
| cb | CheckBox |
| cblist | CheckBoxList |
| img | Image |
| lbl | Label |
| pnl | Panel |
| txt | TextBox |
| dtgridview | DataGrid |

We have used following abbreviations for the name of toolbox.

* **Camel Case-** In this case, the first later in the identifier is lowercase and the first letter of each subsequent in a serial word are capitalized. Example:

**numberOfDays, isValid.**

* **Uppercase-** All the letters in the identifier are capitalized. Example: ID, PI, etc.

**Use of Coding Standard:**

Without coding standard, developers will use their own methods for coding and this can bring negative impacts.

* **Security Issues-** Main reason of program not executing are bugs, errors in the logic. Most of these problems arise due to programming errors that result from poor coding practices.
* **Site Performance Issues-** Bad coding affects the overall performance of the site. Various performance issues include: User Interaction, Server Response, Code reusability, Flow issues.

**Benefits of Coding Standards for a team as well as an individual:**

**1. Increases Efficiency:**  
 Establishing coding standards enables the team to detect problems early or prevent them entirely, which increases the efficiency over the software process.

**2. Minimize the Risk of Project Failure:**  
 Many a time IT projects fail due to software development problems. Coding standards reduce the risk of it.

**3. Reduces Complexity**  
 Higher the complexity of a code the more vulnerable it is to errors. Coding standards help develop software programs with reduced complexity thereby minimizing errors.

**4. Maintenance becomes easy**  
 If a source code is consistent/stable it can be easily maintained. This is because anyone can step in at any stage maintain it or incorporate any modifications.

**5. Correction of bugs**  
 A consistent source code makes it easy to locate and correct bugs in the software.

**6. Comprehensive view**  
 A regular source code facilitates a clearer view of how the code fits within the larger application or the company as a whole.

**7. Cost saving**  
 A constant code leads to a clear view which in turn results in the potential for more code reuse. This reduces the cost and development effort.

**Constraints of using Coding Standard:**

1. **Implement of many rules**

Standards means the rules are enforced. To enforce them, code review must be conducted. To perform code reviews, more rules will need to be established.

1. **Hard to break**

With a "standard" become a habit for quite a while, it will be hard to break away from it.

1. **Creates confusion**

When there are 2 computer languages used at the same time (C# and Sql Server is fairly common to work together), the rules may be very confusing and even against each other.

**Recommending the use of Coding Standard**

As I have used different coding standard in my project/application, I recommend to use coding standard because it was very helpful and I enjoyed to write the code. It reduced my time, cost and complexity. Increases the efficiency of my work.

## Conclusion

Hence, I completed my third part, this part included how the IDE was used to manage the development of your code and design and code of our application. Secondly, description of debugging process in IDE used and coding standards and benefits to organization of using them. Also, an evaluation of developing applications using an IDE versus developing it without an IDE.

# Bibliography

Andrew Hunt and David Thomas, The Pragmatic Programmer: From Journeyman to Master

Bjarne Stroustrup, The C++ Programming Language, 4th Edition

[https://www.programiz.com](https://www.programiz.com/)

<https://www.guru99.com/>

[https://www.geeksforgeeks.org](https://www.geeksforgeeks.org/)