



Alliance with **FPT** Education

ĐẠI HỌC GREENWICH
(VIỆT NAM)

Liên kết với Tổ chức Giáo dục **FPT**

ASSIGNMENT PROGRAMMING REPORT 1

Student performance: TRAN QUANG HUY

ID: GCD18457

Class: GCD0821

Teacher: HOANG HUU DUC

ASSIGNMENT 1 FRONT SHEET

Qualification	BTEC Level 5 HND Diploma in Computing		
Unit number and title	Unit 1: Programming		
Submission date		Date Received 1st submission	
Re-submission Date		Date Received 2nd submission	
Student Name	TRAN QUANG HUY	Student ID	GCD18457
Class	GCD0821	Assessor name	
Student declaration I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.			
		Student's signature	HUY

Grading grid

P1	M1	D1
----	----	----

--	--	--

☐ **Summative Feedback:**

☐ **Resubmission Feedback:**

Grade:

Assessor Signature:

Date:

Lecturer Signature:

INTRODUCTION

Programming is important for speeding up the input and output processes in a machine. Programming is important to automate, collect, manage, calculate, analyze processing of data and information accurately.

Programming is important to create software and applications that help computer and mobile users in daily life. Due to all these reasons, it's important to learn how to use programming languages in our daily life.

Java, javascript, C#, C++, PHP, Python, Swift, SQL, Ruby etc. programming languages are the reasons behind the innovations in information technologies. If today we are seeing robots, artificial intelligence, machine learning, bitcoins, the blockchain, IOT (Internet of Things), Cloud Computing etc. new technology and products in IT industry, then it's because of programming languages.

It's also important to program a world that helps to reduce the corruption, improves transparency and happiness in everyone's life. Programming languages are now a part of our life. Learning can never be wrong, but purposeful learning helps in a great way for students, businesses and all human beings. It's should be done to support nature not to disturb it.

That's it for today!

Contents

I. Algorithm	1
1. Definition of algorithm and characteristics of algorithm.	1
1.1. Definition of algorithm.	1
1.2. Characteristics of algorithm.	2
2. Criteria to measure a good algorithm.....	4
3. Steps to build an application.....	5
3.1. Defining or Analyzing the problem:	6
3.2. Design (Algorithm):	6
3.3. Coding:	6
3.4. Documenting the program:	7
3.5. Compiling and running the program:.....	7
3.6. Testing and debugging:	7
3.7. Maintenance:	7
4. Steps from writing code to execution:	8
4.1. In General	8
4.2 In C#	8
5. The relationship between the written algorithm and the code variant	9
5.1. Code variant:	9
5.2. The difference between algorithm and code variant:	9
5.3. The relationship between algorithm and code variant:.....	9
II.Programming paradigm.....	10
1. Procedural programming.....	12
1.1. Introduction:	12
1.2. Characteristics:	12
2. Object-oriented programming.....	12
2.1. Introduction:	12
2.2. Characteristics:	12

3. Event-driven paradigm.....	13
3.1. Introduction:	15
3.2. Characteristics:	15
4. The source code of an application which implements the programming paradigms, in terms of the code structure and characteristics.	16
III.Integrated Development Environment.....	19
1. Definition:.....	19
2. Features:	19
3. Some specific IDE:	19
3.1. Visual Studio:	19
3.2. NetBeans:	19
References.....	22

TABLE OF PICTURES

Picture 1 - Example coding to sum 2 number	2
Picture 2 - Example An algorithm. (creately, n.d.)	5
Picture 3 - Example console run in C#	18
Picture 4 - Visual studio's interface	20
Picture 5 - Option in visual studio	21
Picture 6 - Create a project with visual studio.....	22
Picture 7 - Writing program in visual studio	23

TABLE OF FIGURES

Figure 1 - Example Flowchart of an Algorithm. (visual-paradigm, n.d.).....	1
Figure 2 - Example steps to cook rice	3
Figure 3 - 7 steps to build an application	6
Figure 4 - Writing code to execution in general	8
Figure 5 - Writing code to execution in C#.....	8
Figure 6 - Flowchart to find a number is even or odd.....	10

I. Algorithm

1. Definition of algorithm and characteristics of algorithm.

1.1. Definition of algorithm.

An algorithm is step by step method of solving a class of problem. It is a well-defined procedure that allows a computer to solve a math or some problem. Algorithms can perform calculation, data processing and automated reasoning tasks.

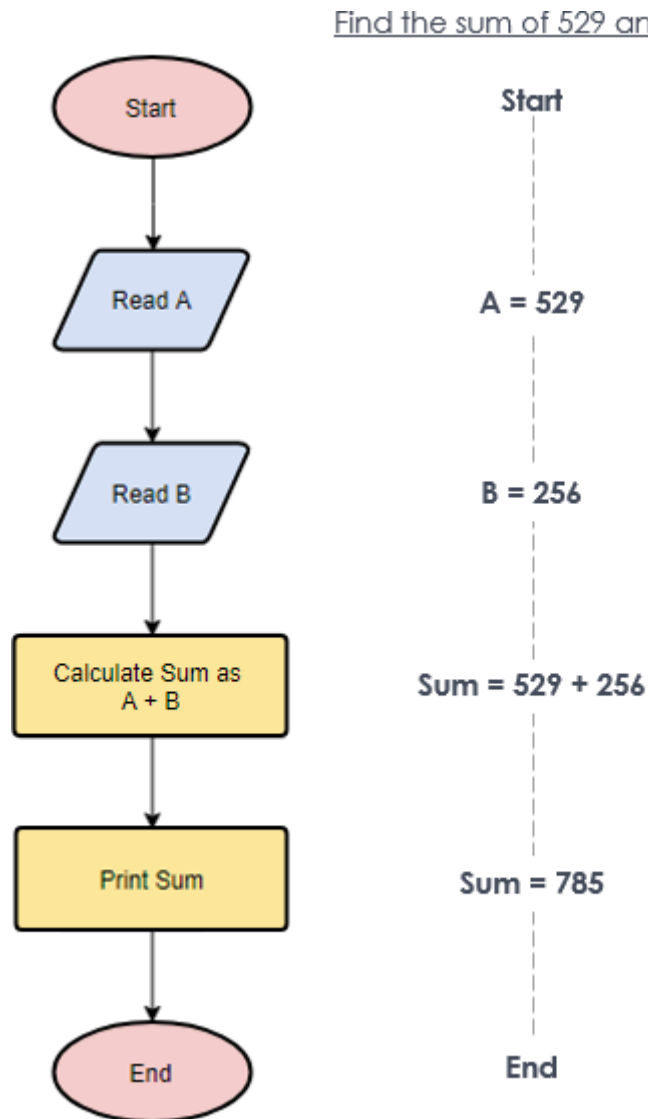


Figure 1 - Example Flowchart of an Algorithm. (visual-paradigm, n.d.)

Example: Write a function that returns the sum of two numbers.

For A = 529 and B = 256, the output should be Sum(A, B) = 785.

```
1 int Sum(int A, int B) {  
2     return A+B;  
3 }
```

Picture 1 - Example coding to sum 2 number

1.2. Characteristics of algorithm.

- There are five important characteristics of an algorithm that should be considered while designing any algorithm for any problem:
 - **Finiteness:** An algorithm should terminate infinite number of steps and each step must finish in finite amount of time.
 - **Definiteness:** Each step of algorithm should be clearly and precise define and there should not be any ambiguity.
 - Example: A program fragment is given below:
 - $X \rightarrow 1$. Toss a coin, if result is head then $X \rightarrow 3$ else $x \rightarrow 4$. In the above program, all the steps would be carried out effectively but there is no definiteness since there are two possible values of X 1 and $\frac{3}{4}$.
 - **Inputs:** An algorithm must have zero or more but must be finite number of inputs. Example of zero input algorithm. Print the ASCII code of each of the letter in the alphabet of computer system.
 - **Output:** An algorithm must have at-least one desirable outcome
 - **Effectiveness:** An algorithm should be effective. Effective means that each step should be referred as principle and should be executing in finite time. (owlgen, n.d.)
 - **Example:** Cook perfectly fluffy rice.
(<https://www.realsimple.com>)

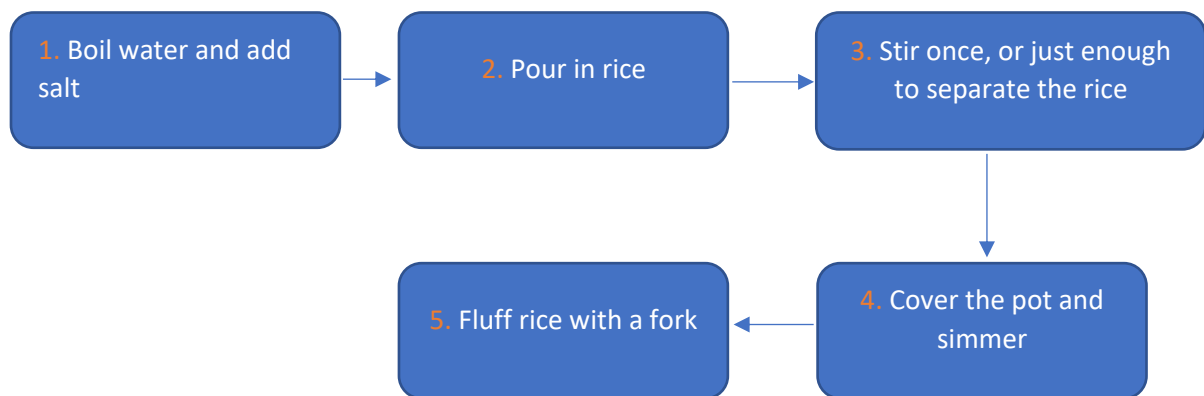


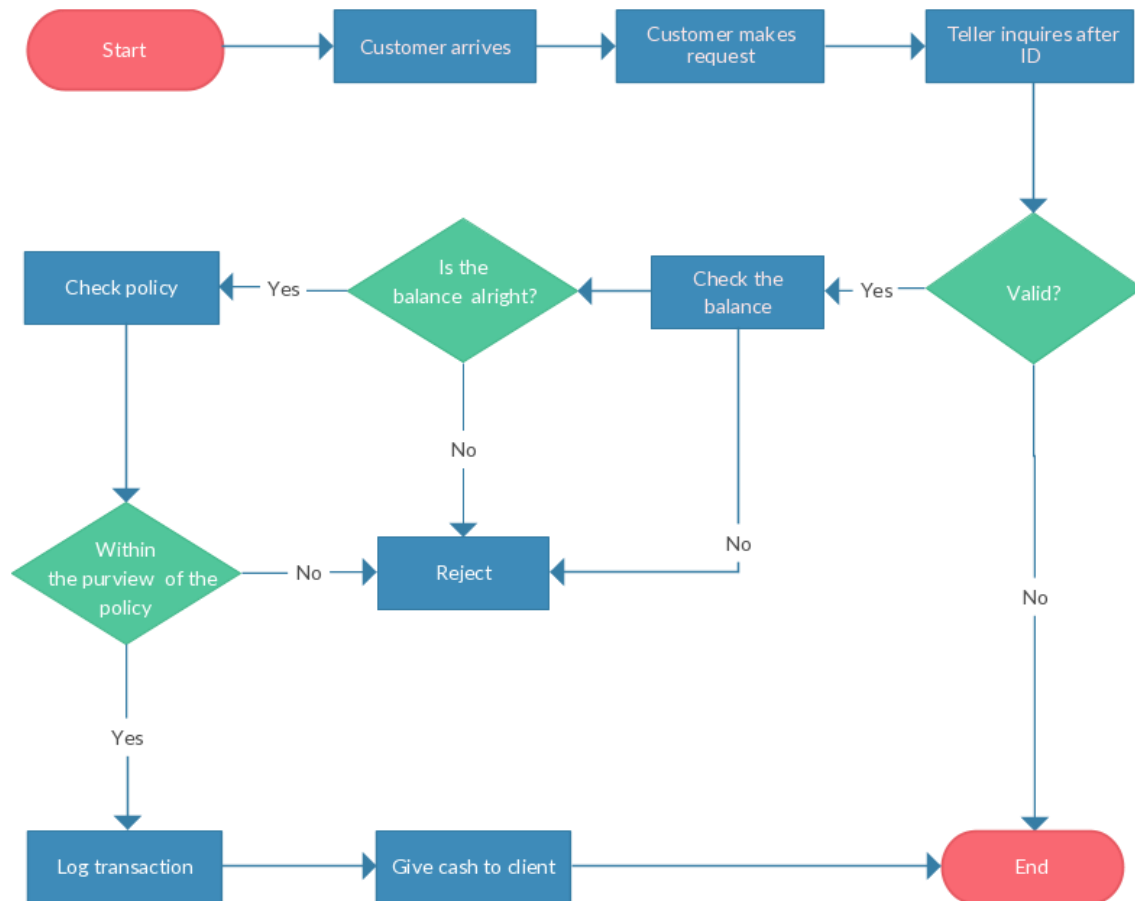
Figure 2 - Example steps to cook rice

2. Criteria to measure a good algorithm.

Good algorithm is one type of algorithm for solving a problem from a specific case.

- A good algorithm has these criteria:
 - **Input:** There are zero or more quantities which are externally supplied;
 - **Output:** at least one quantity is produced;
 - **Definiteness:** Each instruction must be clear and unambiguous;
 - **Finiteness:** If we trace out the instructions of an algorithm, then for all cases the algorithm will terminate after a finite number of steps;
 - **Effectiveness:** Every instruction must be sufficiently basic that its principle be carried out by a person using only pencil and paper.
 - **Execution time:** An algorithm may have good time efficiency but bad space efficiency or an algorithm may have good space efficiency but bad time efficiency. Time efficiency is the time that the algorithm requires to solve a problem and size/space efficiency is the memory size (allocated for variables) that the algorithm requires solve the given problem. (Adhikari, 2018)
 - **Size of input data:** Generally, algorithms are efficient when small inputs are given, but their efficiency gradually decreases when the size of input is increased. For evaluating the efficiency of your algorithm, you must check the time required for processing by your algorithm for different size of inputs given. (Chandel, 2017)
- It is not enough for any kind of problem, but it must also be making an algorithm become good.

- **Example:** Find factorial number.
 - Input: Positive integer number.
 - Output: Factorial of that number.
 - Process: Solution technique which transforms input to output. Factory of a number can be calculated by the formula $n! = 1 * 2 * 3 * \dots * n$



Picture 2 - Example An algorithm. (creately, n.d.)

3. Steps to build an application.

There are 7 steps to build an application:

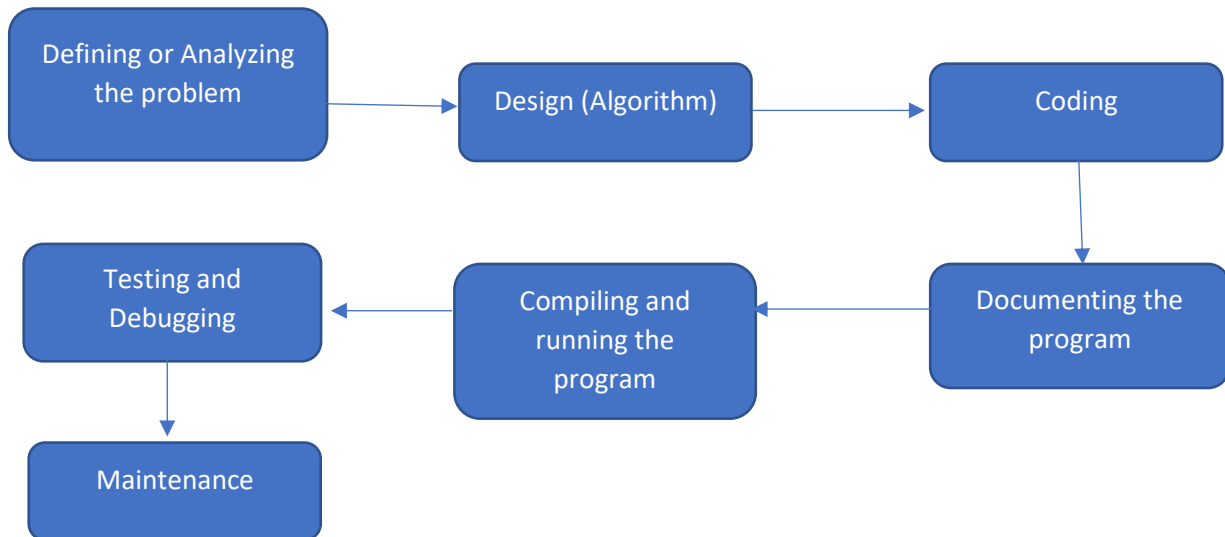


Figure 3 - 7 steps to build an application

3.1. Defining or Analyzing the problem:

Firstly, in this case, the problem must be understood before the program will be written. Then the problem must be analyzed to determine the input and output of the problem solution. This step is very important to make a successful of a project.

3.2. Design (Algorithm):

With the problem was determined in the first step, there are a lot of way to solve it, but we should analyze and chose the best suitable solution that we are going to design into algorithm. To make the problem unambiguous and understandability, flow charts are drawn with the algorithms.

3.3. Coding:

In programming language, all the logic and solution have been developed with the algorithm.

3.4. Documenting the program:

Documenting the program is the information that describes the product to its users. For a user, the documentation helps them know how the program works and how to use it. For a developer, it helps them know all aspect of an application such as troubleshooting, database, server environment, etc. With using programing languages like C#, C++, Java, etc.

Documentation often comes in two form:

- **External documentation:** including Reference manuals, Algorithm descriptions, Flowcharts and Project Workbooks.
- **Internal documentation:** including a part of the source code with declarations, statements and comments to provide an easier understanding.

3.5. Compiling and running the program:

Compiling is a process of translating source code into binary machine code, which make computer can understand and work.

3.6. Testing and debugging:

- Testing: is process of exploring the system to find defects present in the software, this process must locate the defects and define what will happen once these defects occur. After this phase, it will report to the developer to debug. Testing means detecting errors.

- Debugging: When received the report from the testing, it will start debugging to locate the bug and rids the software of it. Debugging means diagnosing and correcting the root causes.

3.7. Maintenance:

Program maintenance is the process of modifying a software or program after delivery to achieve any of these outcomes:

- Correct errors
- Improve performance
- Add functionalities
- Remove obsolete portions

Maintenance also keep pace with changing requirements and technologies.

4. Steps from writing code to execution:

4.1. In General

- Coding: This step is translating the design into program which is written in program language.
- Compiling: This is a process make the program written in program language into machine code made up with 0 and 1 that computer will understand and work. The computer works only understand binary machine code.
- Debugging: This is a process of finding and fixing the bug in the program.

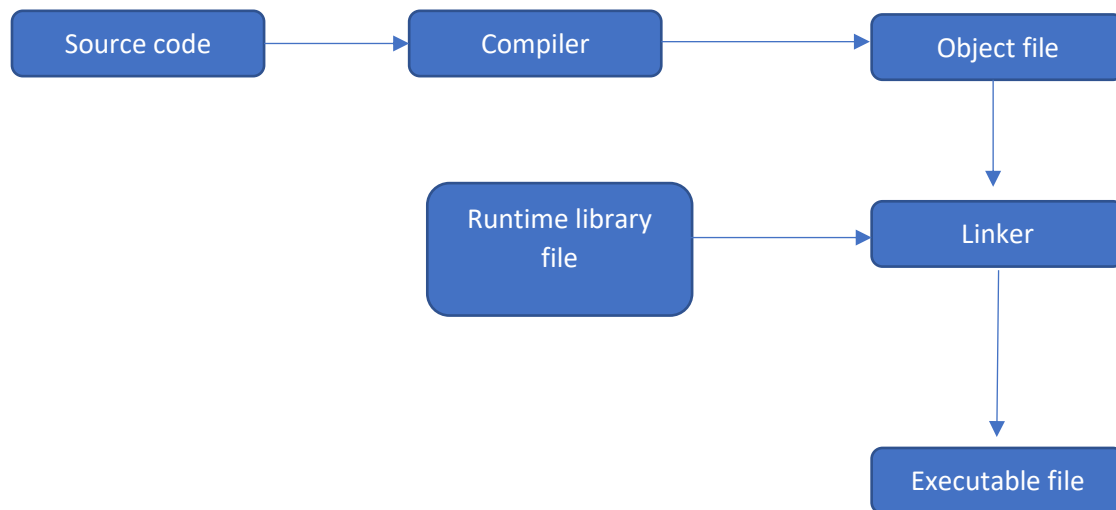


Figure 4 - Writing code to execution in general

4.2 In C#

- Source code will be parsed to ensure you comply with the Microsoft Intermediate Language (MSIL) code.

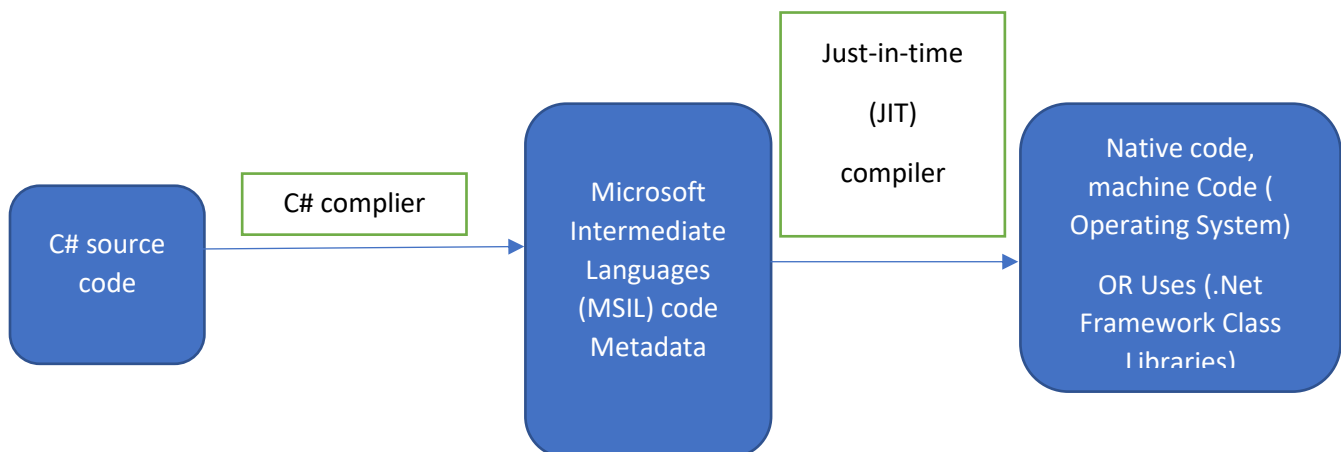


Figure 5 - Writing code to execution in C#

5. The relationship between the written algorithm and the code variant

5.1. Code variant:

- Variant is a data type in certain programming languages, particularly Visual Basic, Ocaml, Delphi and C++ when using the component object model. Variant helps programmer develop their algorithms and with this code variant are essential for machine understand about the algorithm.
- While the use of not explicitly declared variants is not recommended, they can be of use when the needed data type can only be known at runtime, when the data type is expected to vary, or when optional parameters and parameter arrays are desired. In fact, languages with dynamic type system often have variant as the only available type for variables.

5.2. The difference between algorithm and code variant:

- The algorithm is used to provide a solution to a problem in form of well-defined steps. Whenever people use a computer to solve a problem and the algorithms can be expressed using natural language, flowcharts, etc. But variant does not use specific programming language syntax and therefor could be understood by programmers with different programming languages.

5.3. The relationship between algorithm and code variant:

- The algorithms can be written in normal human language, allowing experts who are not programmers to provide input into the program's development phase. Code variant allows programmers who specialize in different programming languages to pool their knowledge and create the most efficient code, resulting in fast, compact programs. Each kind of "language" gives the programmer an advantage at the development stage when express their algorithms into code variant to solve solution.

Example: Find Whether a Number is Even or Odd

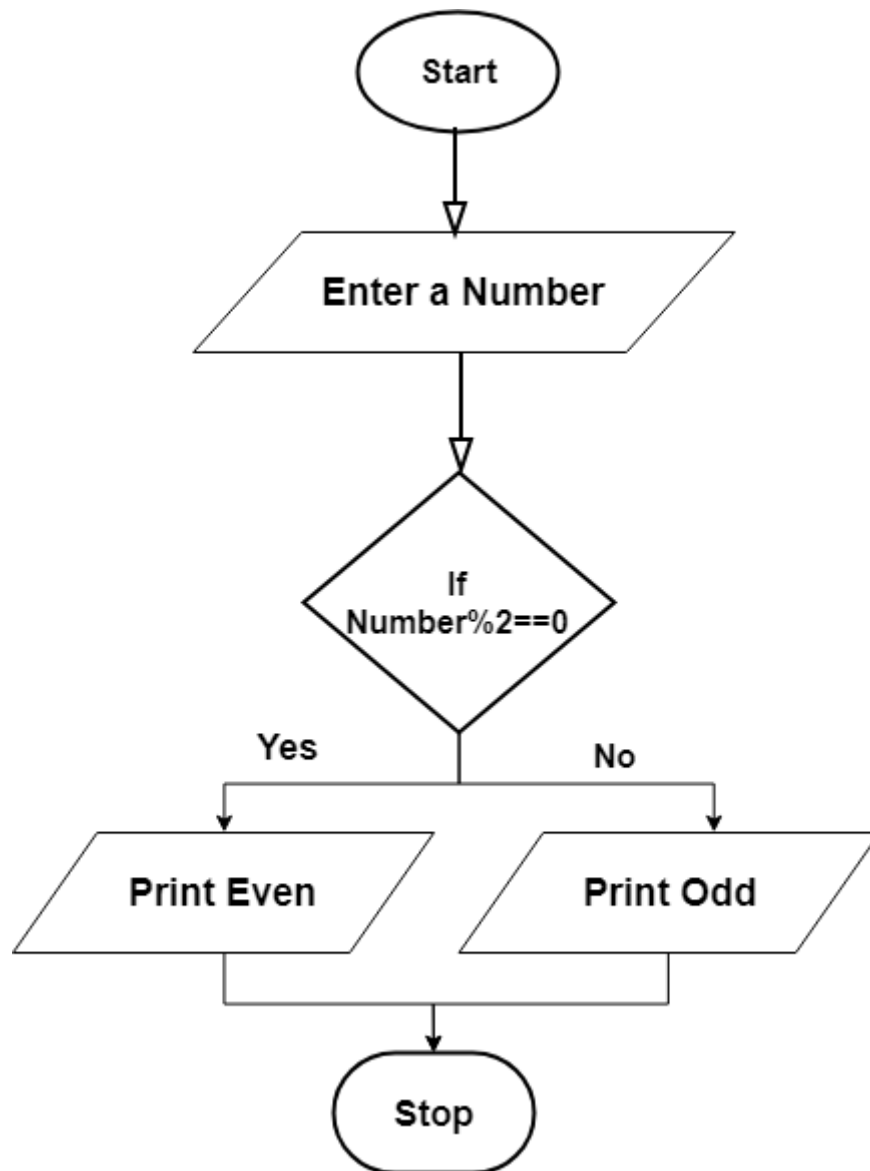


Figure 6 - Flowchart to find a number is even or odd

- Writing program in Pascal:

```
public class whilettest {  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        int I=1500;  
        while (I>=-1500)  
        {  
            if (I%2==0)  
                System.out.println(I+ " : Even");  
            else  
                System.out.println(I+ " : Odd");  
            I--;  
        }  
    }  
}
```

- Writing program in C#:

```
# Program.cs x  
1  using System;  
2  
3  namespace dotnet_sample  
4  {  
5      class Program  
6      {  
7          static void Main(string[] args)  
8          {  
9              int n;  
10             Console.WriteLine("Enter a number to check : ");  
11  
12             n = int.Parse(Console.ReadLine());  
13  
14             if(n % 2 == 0)  
15             {  
16                 Console.WriteLine(n + " is an even number");  
17             }  
18             else  
19             {  
20                 Console.WriteLine(n + " is an odd number");  
21             }  
22         }  
23     }  
24 }
```

II. Programming paradigm

1. Procedural programming

1.1. Introduction:

Procedural programming is a programming paradigm, derived from structured programming, based upon the concept of the procedure call. Procedural programming is a list of instructions to tell a computer how to run from the code. Procedures also known as routines, subroutines, or functions, simply contain a series of computational steps to be carried out.

- Example: Pascal and C are two of the procedural programming languages.

1.2. Characteristics:

- Global data is often shared by all function.
- Data moves openly from function to function.
- Focus on process or doing things rather than data.
- Using top-down approach in program design.
- Functions transfers data from one form to another.
- Data move openly around the system from function to function.
- Large problems are divided into smaller programs known as function.

2. Object-oriented programming

2.1. Introduction:

Object-oriented programming (OOP) is a programming paradigm based on the concept of “objects” which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures often known as methods. OOP packs data together, so an “object” is an instance of a class, operates on its “own” data structure.

2.2. Characteristics:

- **Encapsulation:** Encapsulation keeps data safety and securely from outside interfaces.
- **Inheritance:** This is a process which a class can be received from a base class with all features of base class and some of its own. This process is very useful and helps us to prevent from reusing code that will waste time and memory.
- **Polymorphism:** Polymorphism is the ability to exist in various forms. That means objects of classes that inherited from a common base class may possess functions having the same name, but each having different behaviors.
- **Abstraction:** Abstraction is a concept which facilitates the easy conceptualization of real-world objects into the software program. It refers to providing only essential information to the outside world and hiding their background details or eliminating the unnecessary details.
- **Objects:** The instances of a class which are used in real functionality. It's variables and operations.

- **Class definitions:** Basic building blocks OOP and a single entity which has data and operations on data together.
- **Generic classes:** Class definitions for unspecified data. They are known as container classes and flexible and reusable.
- **Class libraries:** Built-in-language specific classes.

Example: Object-Oriented Programming

```

1. using System;
2. namespace oops
3. {
4.     //Base Class
5.     public class Father
6.     {
7.
8.         //constructor
9.         public Father()
10.        {
11.            Console.WriteLine("Father class constructor");
12.        }
13.
14.        public void FatherMethod()
15.        {
16.            Console.WriteLine("this property belong to Father");
17.        }
18.    }
19.
20.    //Derived class
21.    public class Child : Father
22.    {
23.        public Child()
24.            : base()
25.        {
26.            Console.WriteLine("child class constructor");
27.        }
28.        public void ChildMethod()
29.        {
30.            Console.WriteLine("this property belong to Child");
31.        }
32.    }
33.    class Inheritance
34.    {
35.        //Entry point
36.        static void Main(string[] args)
37.        {
38.            //Here Child object can access both class methods
39.            Child cObj = new Child();
40.            cObj.FatherMethod();
41.            cObj.ChildMethod();
42.            Console.ReadKey();
43.        }
44.    }
45. }

```

3. Event-driven paradigm

3.1. Introduction:

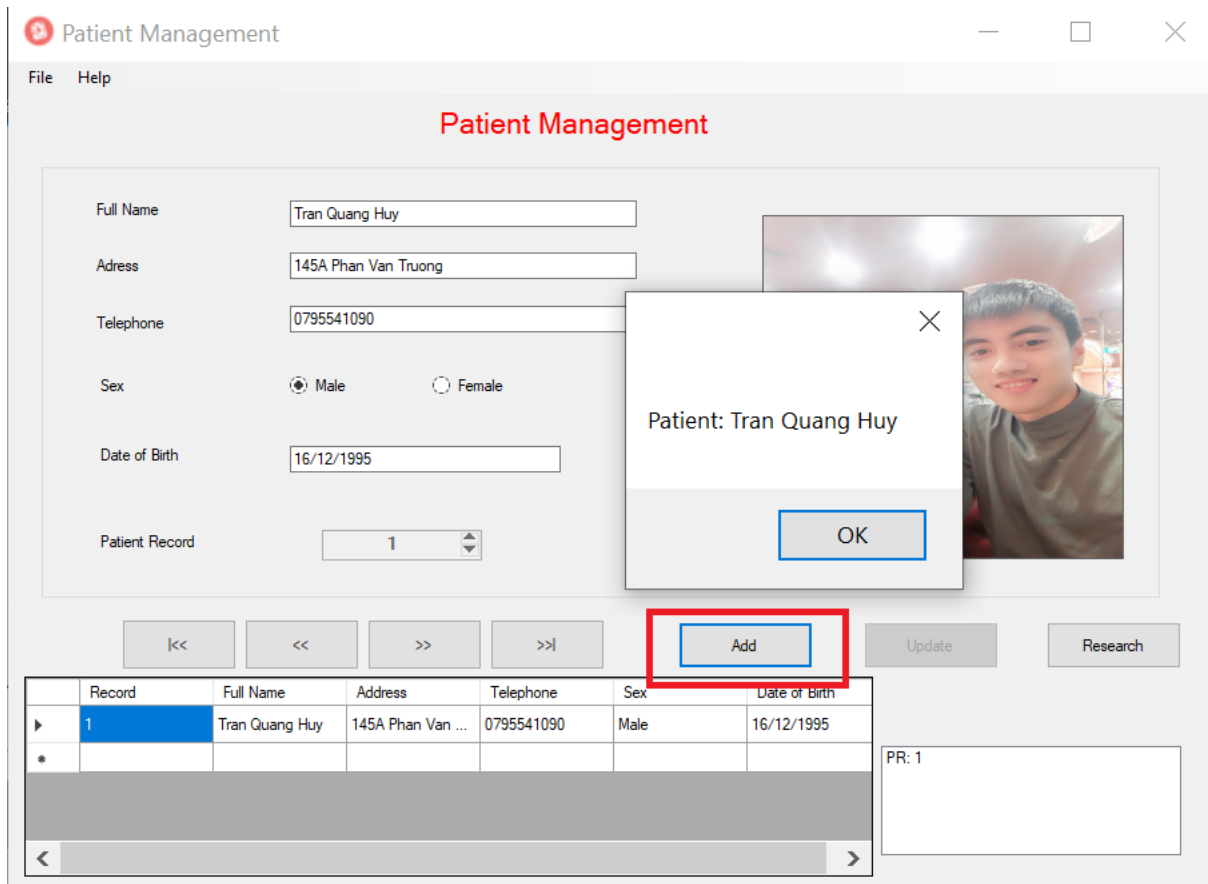
- Event-driven programming is a programming paradigm in which the control flow of program execution is determined by events (user's mouse clicks, key presses, messages from other programs). In an event-driven application, there is always have an event listener (the main loop) that detects for event that just occurred, then it runs an event "handler" (callback function). This handler deals with event by responding to it with statements.
- Event-driven programs can be supported by any programming languages, but in some languages that provide high-level abstractions, such as closures will be easier to implement.

3.2. Characteristics:

- Time driven: often used in Real-time computing and in which the program is driven by a timer. The developer would set the amount of time for the application to do their works. The programmer had to set the event that would happen after a finite time.
- Service oriented: Everything on the computer screen provide a service to the user. Service orientation inherits some of principles object-oriented paradigm including: Multiple use, non-context specific, Composable, Encapsulated.
- Trigger function: When the event just occurred, a trigger function has called. A trigger function needs to decide what kind of event has happened and manage the actions to respond to an event for the event handler.
- Event handler: is the callback function that will execute when event listener detects an event from the user. It will respond to the user's action with its code.
- 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.
- Flexibility: Event driven programs are very convenient for the user if they need to change something. They just simple change the screen adding or change some event handlers.
- Suitable for Graphical User Interface: These programs rely on user 's action like dragging objects on the screens onto the form. It's friendly and more comfortable for programmer to double click on the objects add the code to make it work.

3.3. Example Event-Driven programming

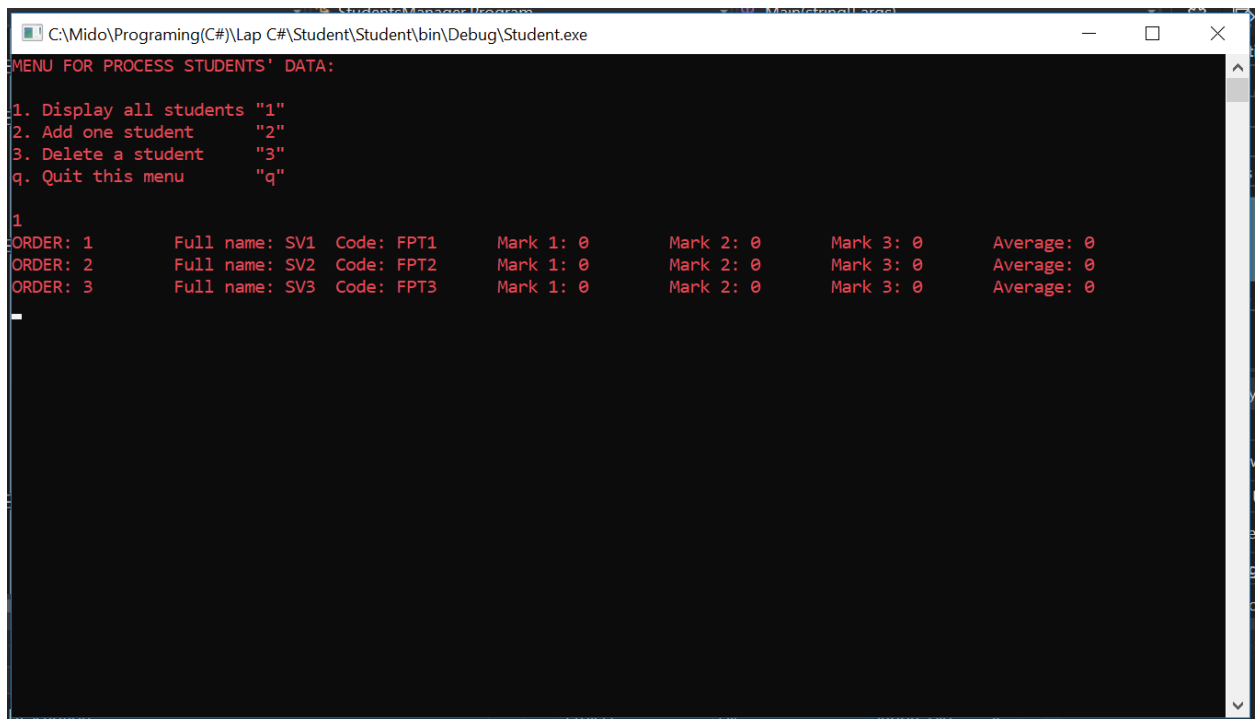
- This button will add patient information: When user click this button, all information that user fills in program will be added into database.



```
private void pictureBoxAva_Click(object sender, EventArgs e)
{
    OpenFileDialog OpenIMG = new OpenFileDialog();
    if (OpenIMG.ShowDialog() == DialogResult.OK)
    {
        Bitmap imgOpen = new Bitmap(OpenIMG.FileName);
        pictureBoxAva.Image = imgOpen;
    }
}
```

4. The source code of an application which implements the programming paradigms, in terms of the code structures and characteristics.

- The application I selected that I have developed is “Programming to manage students”
 - Data type (The role of constants/variables): I have used “string” to Name of student and student ID; Integral type to Mark of students, Number of students, and Double point type to Average of Mark for each student; etc.
 - Methods: Calling a method on an object is like accessing a field. After the object name, add a period, the name of the method, and parentheses. Arguments are listed within the parentheses and are separated by commas. The methods of the Student class can therefore be called as in the program. Specially, I used the “get” and “set” accessor consists of a single statement that returns a value for Name of students, Student ID, Mark, etc.
 - Control structures:
 - Iteration constructs: “For loops” allows me to specify the number of times to repeat a block of code to input Information about students such as Name, ID and Mark and also help me to display all information of each students.
 - Decision Constructs: “If...else statement” - This is an extension to the standard if statement. It allows to specify the code that should run, if the condition in the first if statement evaluates to false. “Switch Statement” - A switch statement allows use to execute code based on a predefined set choice, in this programming to manage students that help us make a menu have 4 option (1. Display all students; 2. Add one more student; 3. Delete one student; 4. Quit).
 - Variable Scopes: Variables that are defined at the class level are available to any non-static method within the class. In this article the full extent of class scopes is not discussed as the object-oriented programming techniques involved are beyond the scope of a beginner's tutorial.
 - Classes: I have used 2 class on this program to manage students. The first class is “Program” to do important action and main of program, the second class is “Student” to store and make the Data type has means.



```
C:\Mido\Programing(C#)\Lap C#\Student\Student\bin\Debug\Student.exe
MENU FOR PROCESS STUDENTS' DATA:
1. Display all students "1"
2. Add one student      "2"
3. Delete a student     "3"
q. Quit this menu       "q"

1
ORDER: 1      Full name: SV1  Code: FPT1      Mark 1: 0      Mark 2: 0      Mark 3: 0      Average: 0
ORDER: 2      Full name: SV2  Code: FPT2      Mark 1: 0      Mark 2: 0      Mark 3: 0      Average: 0
ORDER: 3      Full name: SV3  Code: FPT3      Mark 1: 0      Mark 2: 0      Mark 3: 0      Average: 0
```

Picture 3 - Example console run in C#

III. Integrated Development Environment

1. Definition:

- An Integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development.
- An IDE normally consists of source code editor, build automation tools, and a debugger. Most modern IDEs have intelligent code completion.

2. Features:

- Source editor: The source editor is a modern programmer's text editor that optimized for writing and editing source code. The source editor also provides color based on what kind of code is.
- Compiler: This tool helps us transform source code written by people language into a form that computer can understand and execute.
- Debugger: This tool is used during testing to find out which bug in programs.
- Build automation tools: These tools automate common developer tasks.

3. Some specific IDE:

3.1. Visual Studio:

- Auto hide: Allows us to minimize the document to the edges of the user interface and pops-up docs when you mouse-over them.
- Annotations reporting: Help us to extract C# annotation them in HTML or store it in XML documents.
- Cascading Style Sheet (CSS): is a style sheet language used for describing the presentation of a document written in a mark-up language like HTML.
- Web browser: Allows us to display Web pages.
- Command Windows: Allows us to invoke command in IDE.
- XML: Edit XML with syntax directed highlighting.
- Tabbed Documents: View multiple windows in one part of the interface.
- Editor features: Auto-indent, color highlighting, auto-complete, clipboard rings, document navigation, etc.
- Macro: Play macro directly in the user interface.

3.2. NetBeans:

- Quick Search: Instead of finding by yourself, this feature is a centralized location where you can do searches through all Java types in the JDK, actions in the IDE
- Action Items: This feature auto scans your code and list commented lines and containing words such as "TODO" or "FIXME", and also line with compilation errors, quick fixes, and style warning.
- Plugin Manager: While using NetBeans you can go to Plugin Manager to manage your plugin such as add, remove, or update the set of plugins and there are a wide variety of plugins are available for all type of development.
- Databases and Services: NetBeans provides your ability to access many resources, such as databases, services, web services, and issue tracker. Also,

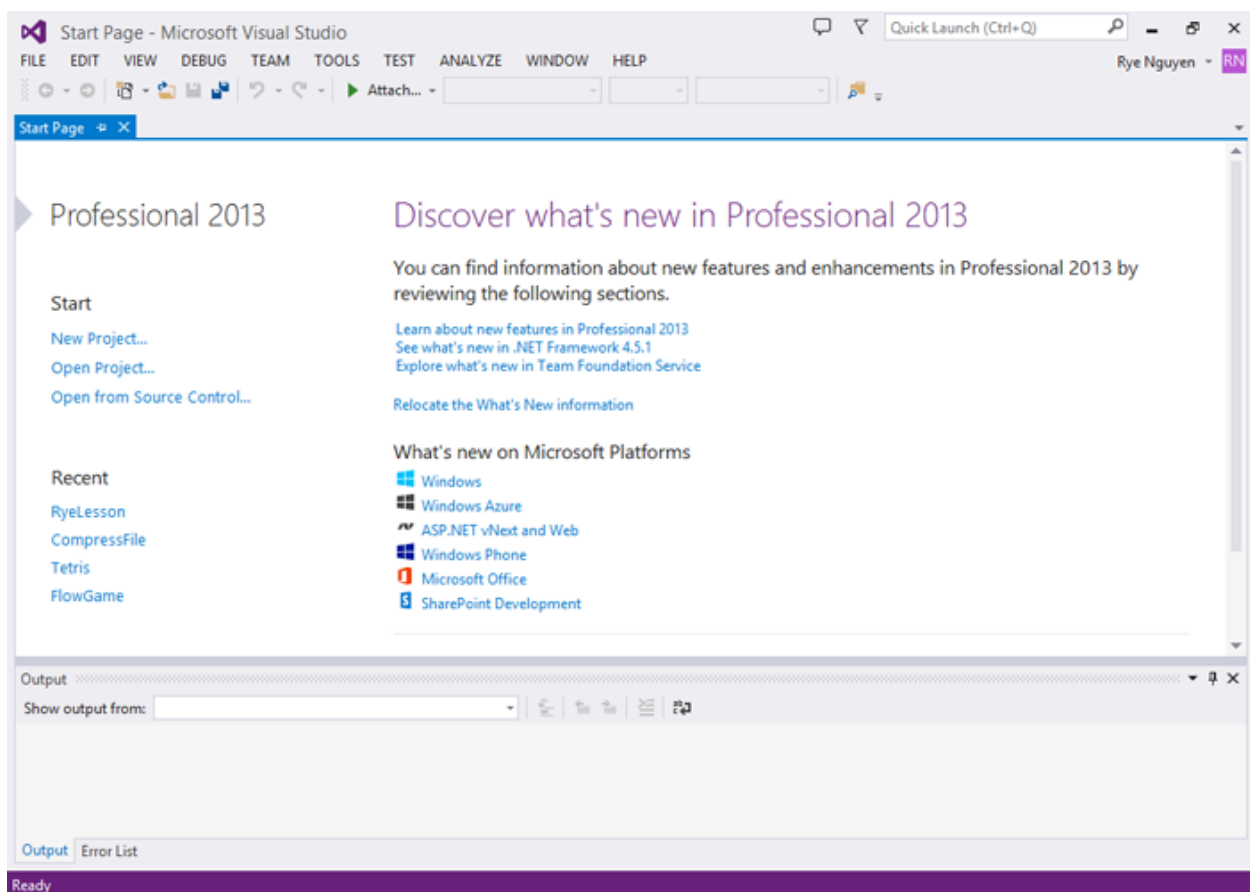
you can start and stop servers directly in the IDE. When you work with databases, you can add, remove, modify it in the IDE. This is very comfortable for programmers.

- **Templates and Sample Applications:** NetBeans gives you applications in the form of project template for all technologies it supports. The IDE provides templates and sample applications that help you create Java SE application, Java EE applications, Java ME applications, HTML5 applications, NetBeans Platform application and C/C++ applications.

4. Some function in visual studio:

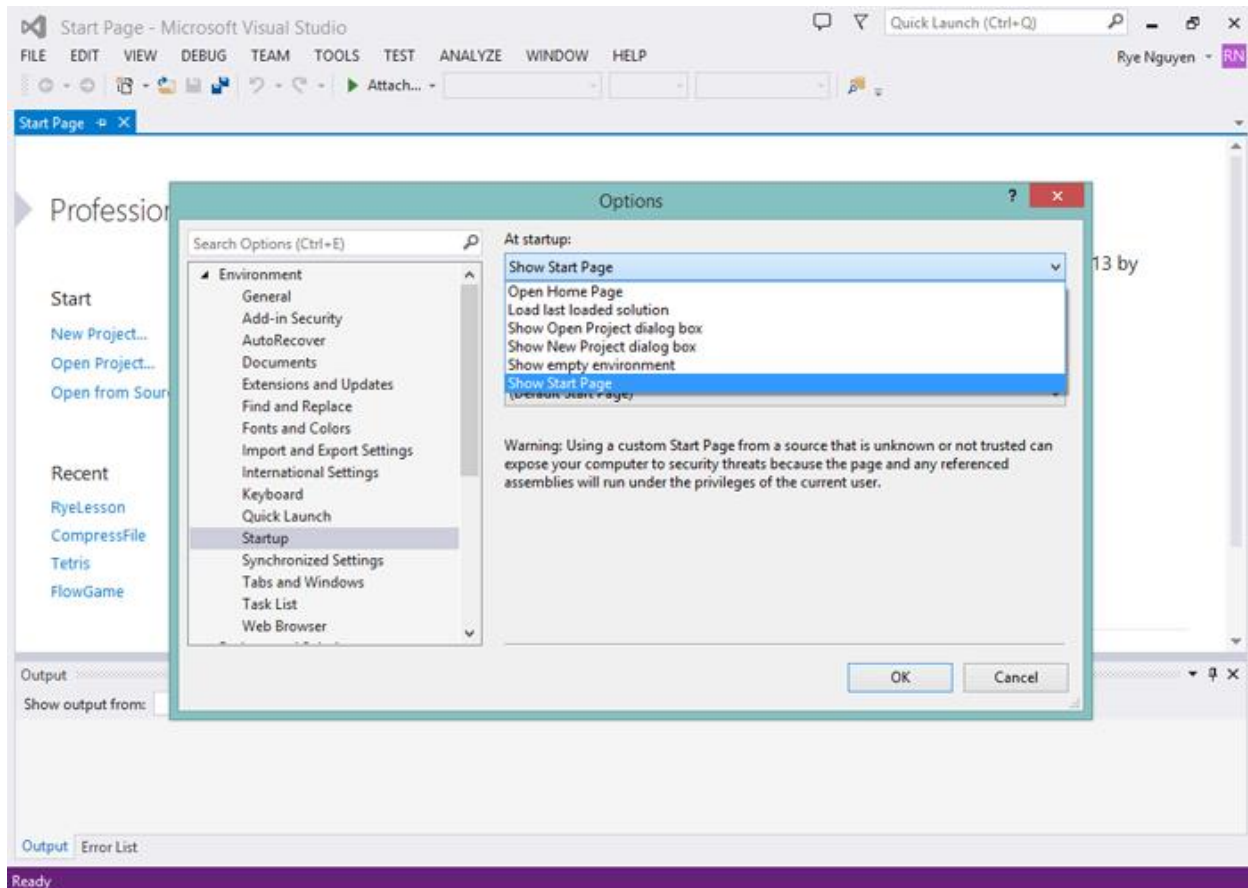
4.1. Interface of Visual Studio:

After booting, the program's working window will look like this:



Picture 4 - Visual studio's interface

As you can see, the program interface is relatively simple and intuitive. The Start section includes some project operations, the Recent section displays the projects you have worked on previously. Both of these items are in the Start Page tab of Visual Studio. Note that each version of Visual Studio will be slightly different in Start Page. If you don't show Start Page, do the following: Go to menu Tool → Option. Select Environment tab → Startup. Click on the At startup dialog box and select Show Start Page.



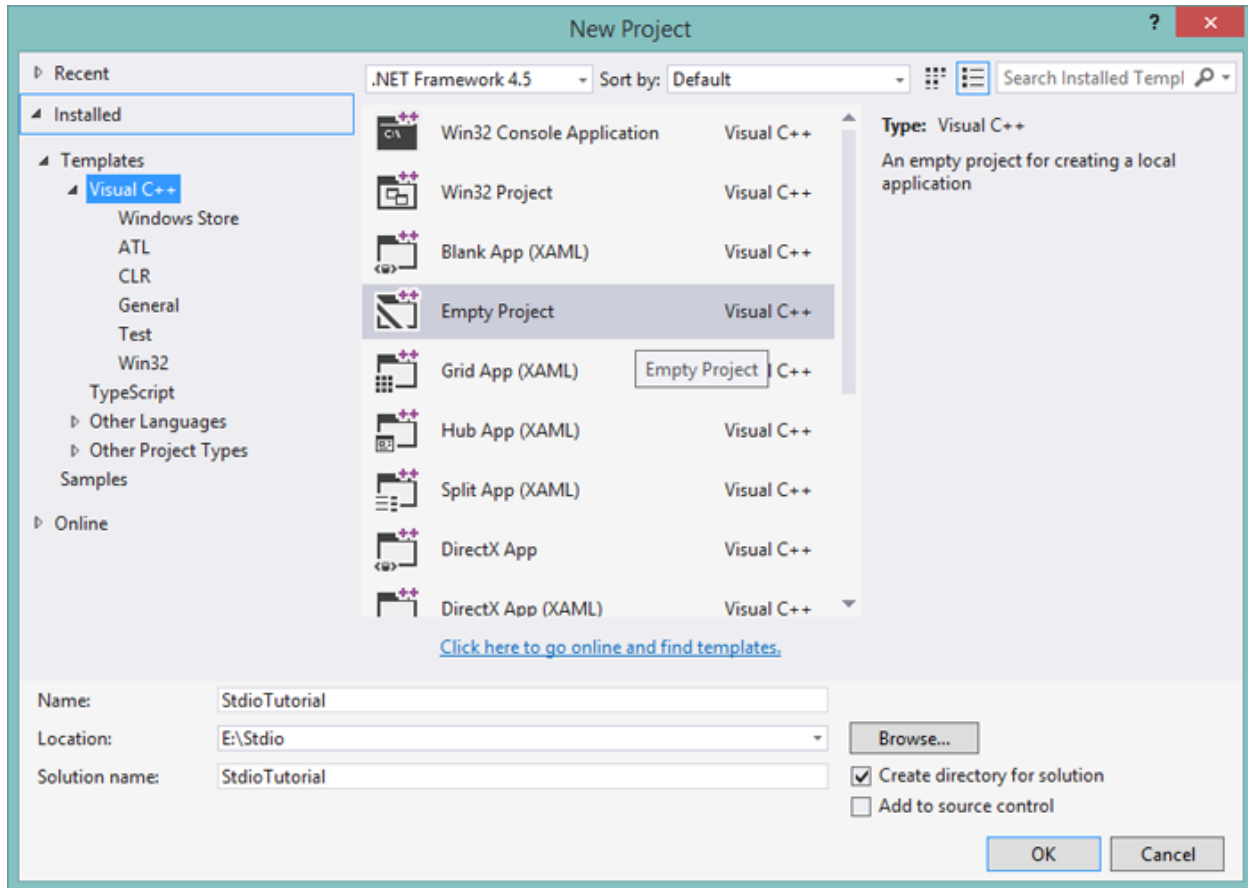
Picture 5 - Option in visual studio

4.2. Create the first Project C ++ with Visual Studio:

To create a new project, you can do it in one of four ways:

- Go to the File → New → Project menu.
- Press Ctrl + Shift + N.
- Click the New icon on the toolbar
- Click on the New Project line in the Start section in Start Page.

A window will appear as follows:

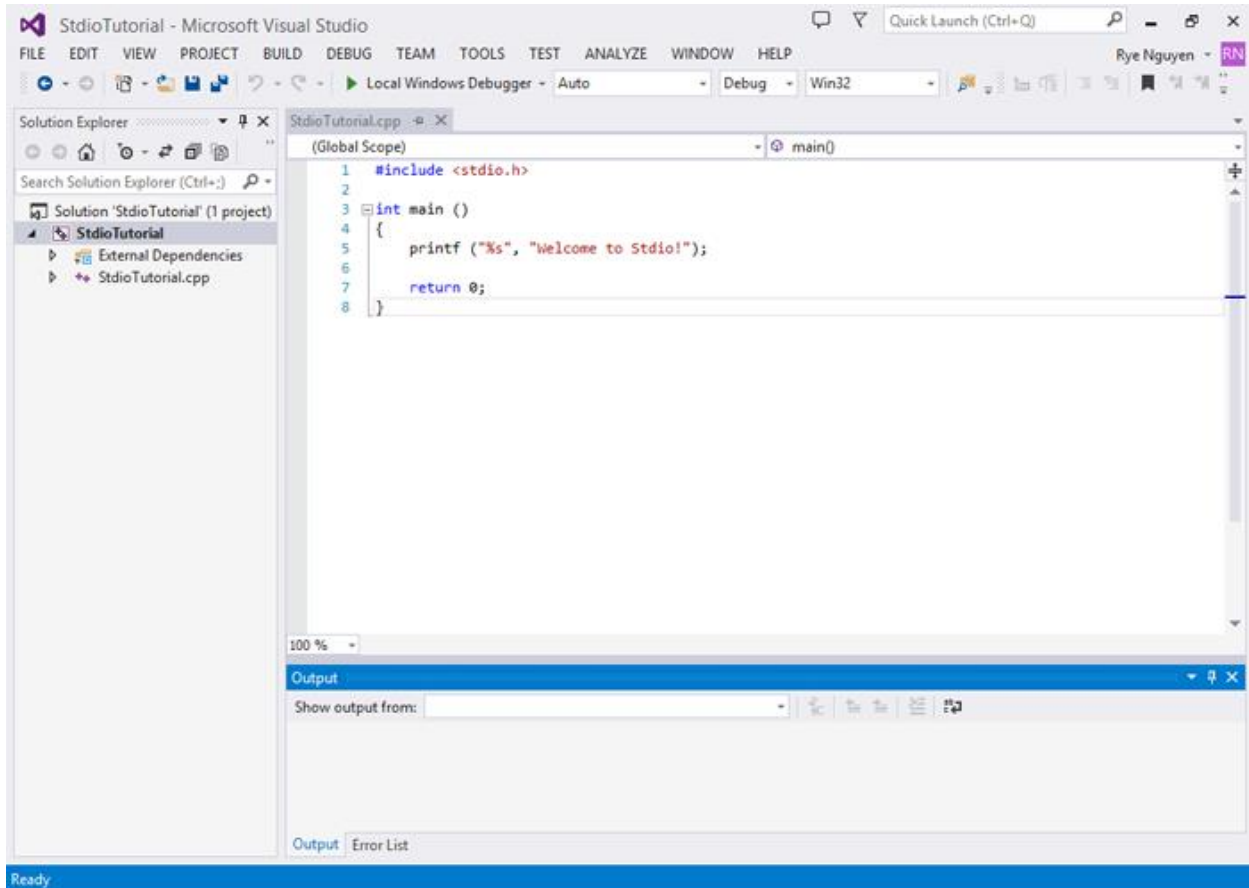


Picture 6 - Create a project with visual studio

4.3. Write the first program

We can do it in one of two ways:

- Right-click on the Project / Filter name select Add → New Item.
- Select the appropriate Project / Filter name and press Ctrl + Shift + A.



Picture 7 - Writing program in visual studio

References

- Adhikari, B. (2018). Retrieved from <https://www.quora.com/How-is-a-good-algorithm-Explain-how-to-measure-it-with-example>
- Chandel, R. S. (2017). Retrieved from <https://www.quora.com/What-criteria-are-used-to-tell-a-good-algorithm>
- creately*. (n.d.). Retrieved from *creately*: <https://creately.com/blog/diagrams/flowchart-guide-flowchart-tutorial/>
- owlgen*. (n.d.). Retrieved from *owlgen*: <https://www.owlgen.com/question/what-is-an-algorithm-explain-characteristics-of-an-algorithm-with-the-help-of-an-example>
- visual-paradigm*. (n.d.). Retrieved from *visual-paradigm*: <https://www.visual-paradigm.com/tutorials/flowchart-tutorial/>
- What is event-driven programming? (2017). Retrieved January 17, 2017, from <http://www.computerhope.com/jargon/e/event-driven-prog.htm>
- 1010, zakkcuthbert. (2014, November 7). Key features of event driven programs. Retrieved January 17, 2017, from <https://zakkcuthbertunit14eventdrivenprogramming.wordpress.com/2014/11/07/key-features-of-event-driven-programs/>
- Event-driven programming (2017). . In *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/Event-driven_programming#Common_uses
- Comparison of programming paradigms (2017). . In *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/Comparison_of_programming_paradigms
- Studytonight. (2016). Object oriented programming concepts in C++. Retrieved January 17, 2017, from <http://www.studytonight.com/cpp/cpp-and-oops-concepts>
- The Difference Between Algorithms, Pseudocode & Programming Languages form <https://www.techwalla.com/articles/the-difference-between-algorithms-pseudocode-programming-languages>
- Object-Oriented programming from https://en.wikipedia.org/wiki/Object-oriented_programming
- Programming paradigm from https://en.wikipedia.org/wiki/Programming_paradigm
- Algorithm from <https://en.wikipedia.org/wiki/Algorithm>

Program Maintenance from

https://www.tutorialspoint.com/programming_methodologies/programming_methodologies_program_maintenance.htm

Testing and Debugging from <https://dzone.com/articles/the-differences-between-testing-and-debugging>

Data Structures from

https://www.tutorialspoint.com/data_structures_algorithms/algorithms_basics.htm

Variant type from https://en.wikipedia.org/wiki/Variant_type

Integrated development environment from

https://en.wikipedia.org/wiki/Integrated_development_environment

Control structures loops if and switch from

<https://www.dreamincode.net/forums/topic/220356-control-structures-loops-ifs-and-switch/>