Aims

- *Understanding the logic of programming and algorithm*
- *Learning C#*
- *Writing object-oriented program*

Subjects

- Basic Concepts and Algorithms
- The concept of variable and basic operations
- Control Structures
- Loop Structures
- Algorithm Examples and Analysis
- Introduction to Programming Language Variable Types and Basic Input / Output Operations
- Control and Loop Structures
- Arrays
- Multidimensional Arrays
- Functions-1
- .
- .
- .

Platforms

- C#
  - Visual Studio 2010-12-15-17 Community
  - Eclipse with C# plugin

References

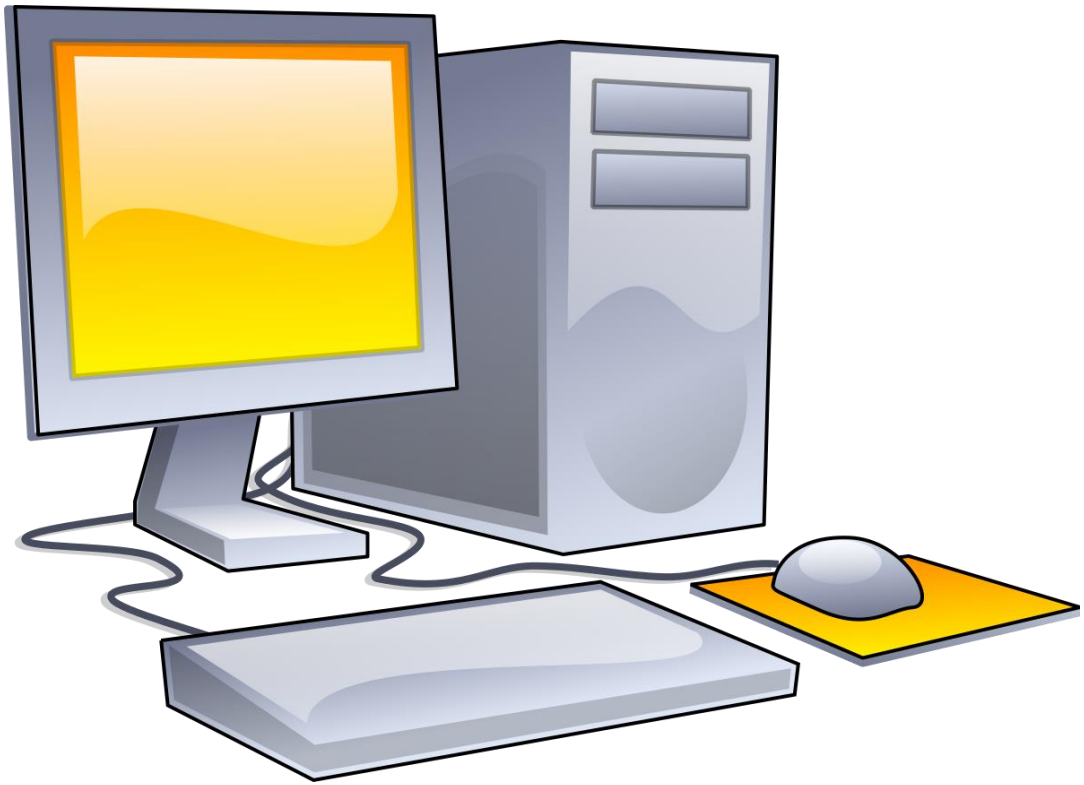
and any book from beginner level related to C#

Week 1
Introduction to Programming

Introduction

Introduction

- What is the computer?



Introduction

- **Computers** are capable of calculating and making logical decisions.

- More specifically, the computer; It is a complex electronic system that produces the desired results by processing the given input values (data) with the specified methods.

Introduction

- The data to be processed and the methods to be used are given by human.

- So, What is the data?

Introduction

- Everything we see, hear, think etc. is called data

Introduction

- As we said, these datas are given to computers by people.
- For this identification, the scripts called **“Program”** are used.
- The program refers to a specific set of sequential operations that the computer will follow when performing an operation that written by the “**programmer”**.

- *Who is programmer?*

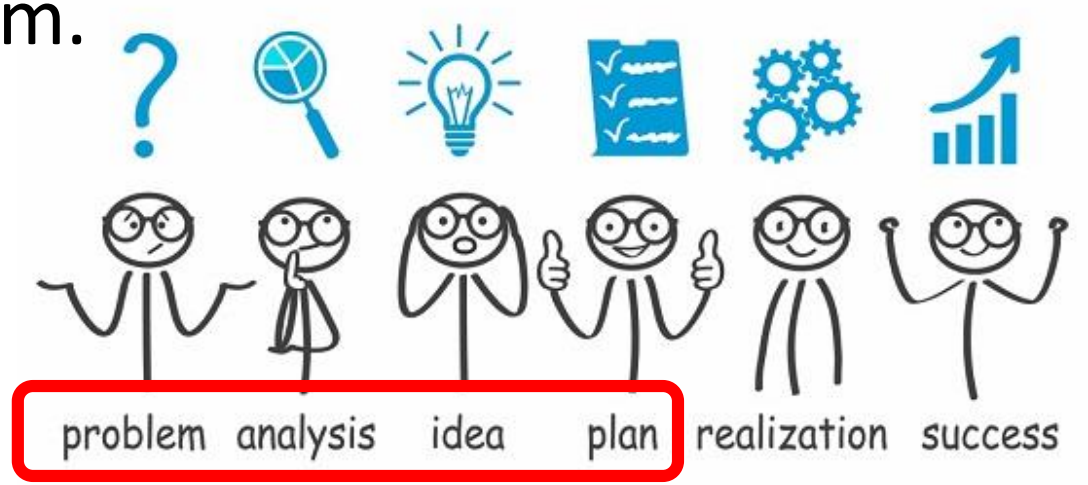
- A **programmer** is the person who creates the computer software. The programmer is also called a computer programmer, developer, encoder or software engineer.

- A Programmer must have some foreknowledge before starting to write a computer program.

- Because it is very difficult to solve a problem quickly and accurately in the computer environment without foreknowledge.

- First of all, programming is a process and does not end with the program being written. Mostly, the program continues in different ways during the period.

- For this reason, some steps must be taken before starting the program or solving the problem.



- In order to provide this and to write solid-based programs, prestudies are carried out on paper and the program is written according to the obtained data.

- However, if you are new to programming, it will be very difficult to write programs.

- First of all, you will need to get the **programming logic** and the **problem solving intuition**.

- These logic and intuition will devolep with writing program too much.

- What is **software**?

- Computers contain a variety of devices called hardware such as keyboard, mouse, screen, DVD, etc.

- Programs that allow us to use these devices and to perform the operations we want to do to the computer are called **Software**.

Problem and Problem Solving

- Problem is defined as a question, a problem or an obstacle to be solved

- The emergence of the idea of solving a scientific calculation, an automation problem or a process problem by a computer is called **problem**.

Problem and Problem Solving

- In order to solve the problem, the problem must be clearly understood by the programmer.
- All needs and requirements must be determined.
- A problem can of course have multiple solutions. In this case, the most appropriate solution should be selected using the computer.

Program

- To solve a problem, all commands given to the computer are called Programs.

- There are many different types of programs depending on their intended use and location:
  - Operating Systems
  - Utilities
  - Device Drivers

- **Operating Systems:** Each program runs on an operating system. The operating system provides the resources and environment required for the operation of other programs.

- **Utilities:** They are programs that run on the operating system and provide the functions that users need.

- **Device Drivers:** Programs that provide communication between the operating system and hardware devices. For example, the keyboard's drive program is used to detect the text written by the keyboard.

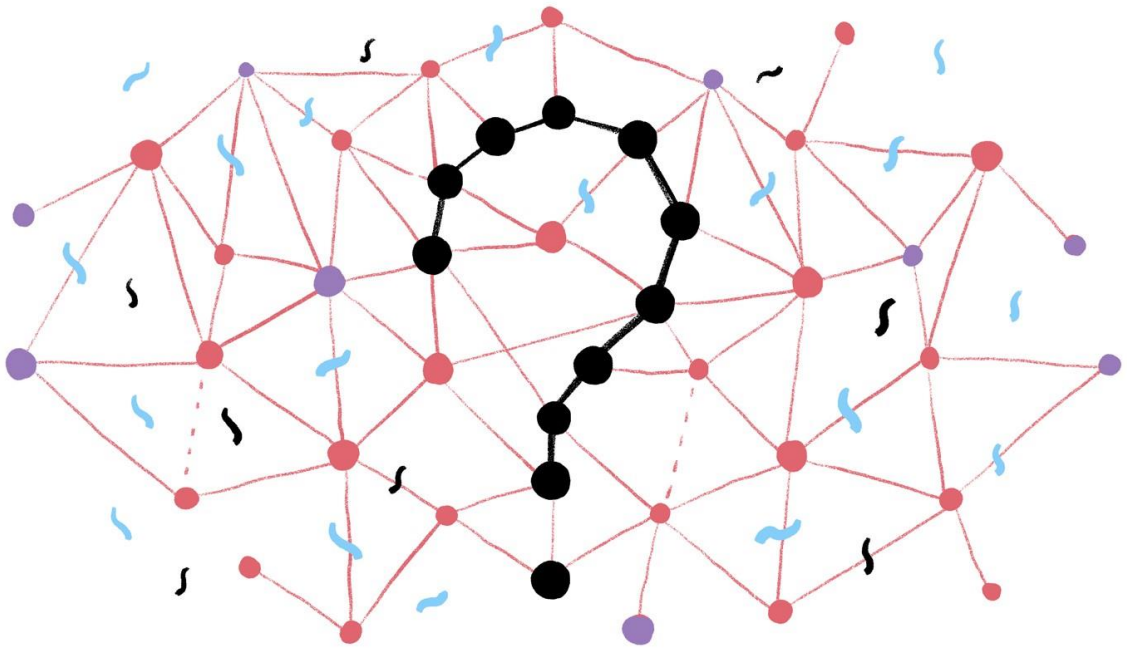
Programming

- Transfer of a very well-defined solution to a problem with the steps of the solution to a computer language with a programming language is called **Programming**.

Algorithm

- All problem solutions must have an algorithm.

- What is the algorithm?



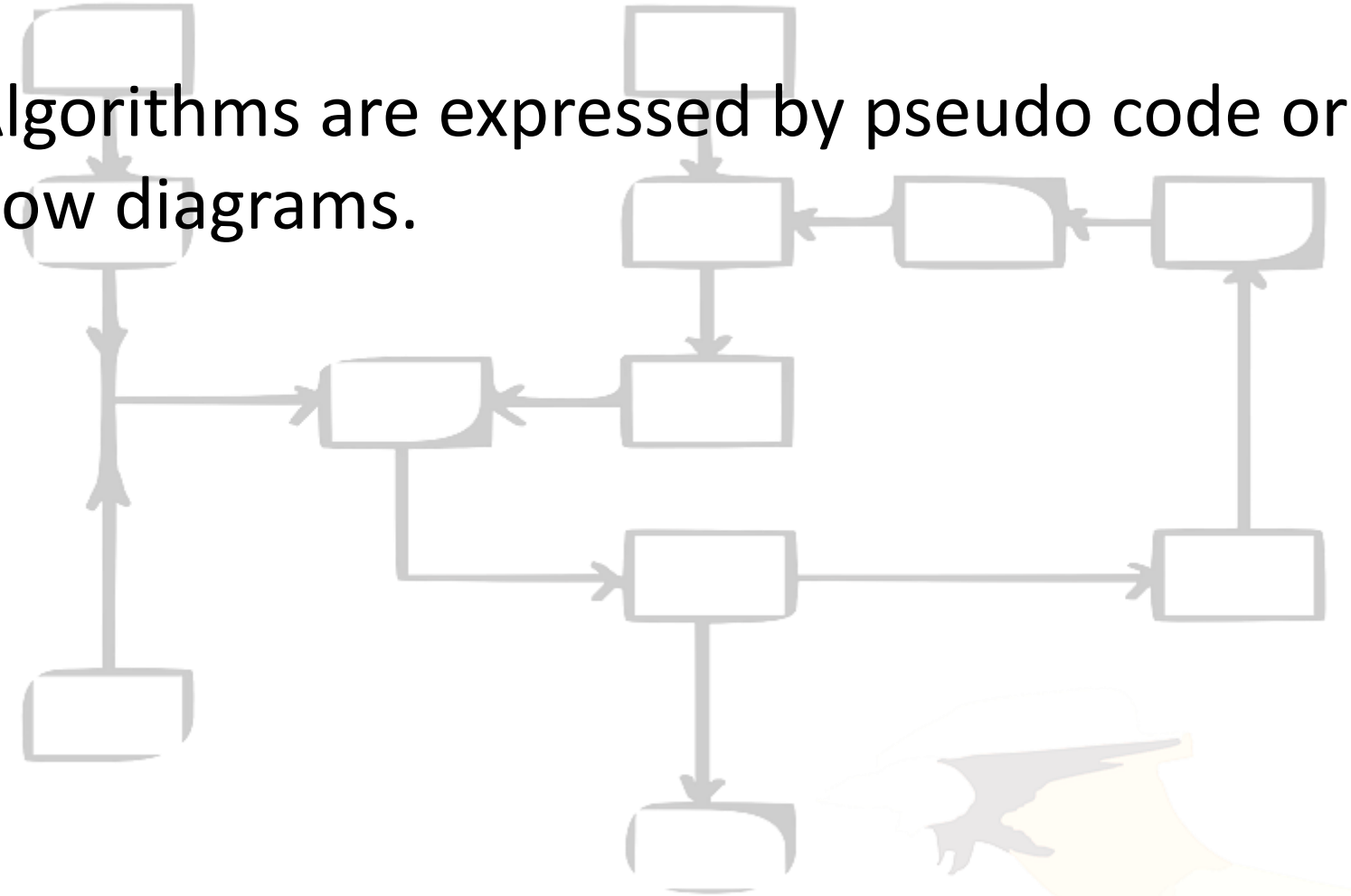
Algorithm

- The sequential logical steps necessary to solve a problem are called **algorithm**.

- An algorithm must verify the following steps.
  - Each step must be highly decisive. Nothing should depend on luck.
  - At the end of a certain number of steps, the algorithm must terminate.
  - The algorithms should be generic enough to handle all the possibilities that may be encountered.

Algorithm

- Algorithms are expressed by pseudo code or flow diagrams.



Pseudo Code

- Pseudo code is an algorithm that we use to establish algorithms, close to the language of the day, and does not have the features of a particular programming language.

- It is used to describe the structure of the algorithm in a flow with all the details and simplicity of everyday language.

Pseudo Code

- The person who reads the pseudo-code should be able to understand the algorithm completely and comprehend the logic of the work.
- There is no syntax to be considered when writing pseudo code.
- The important thing is that they are understandable and programmable.

Pseudo Code

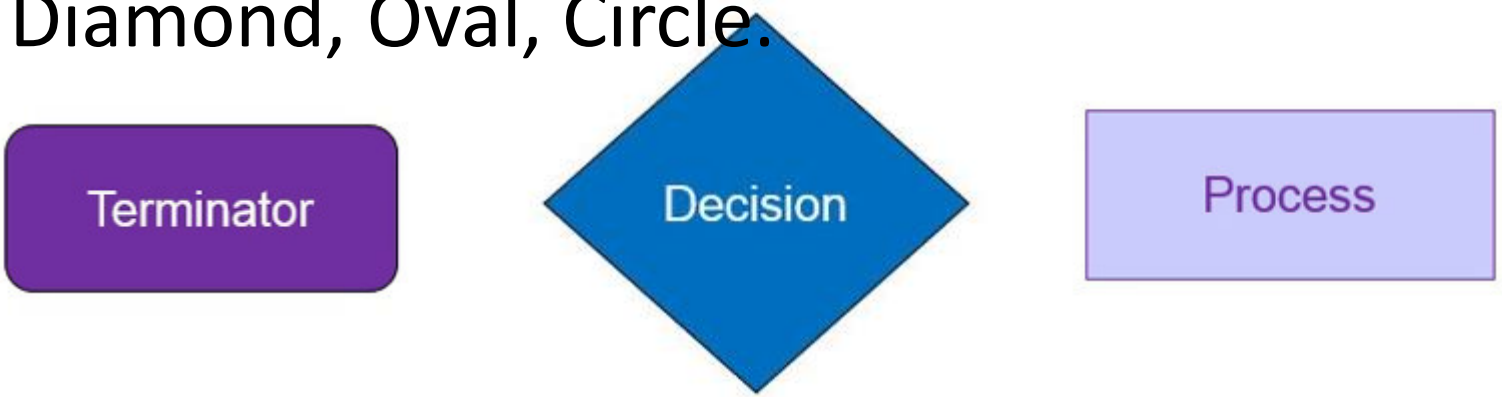
- Some basic Pseudo code commands are:

- **Start :** Indicates that the program has started.
- **End :** Indicates that the program has finished.
- **Read :** Used for user input.
- **Write :** Used to display information or results to the user.
- **If … :** It is used to change the flow according to the conditions.
- **If Not… :** It is used to change the flow according to the conditions.

Flowchart

- To express the steps of the algorithm visually with shapes that express various meanings and connect each other with arrows.

- Flow charts are created by drawing some special purpose symbols such as Rectangle, Diamond, Oval, Circle.

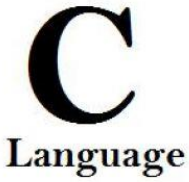


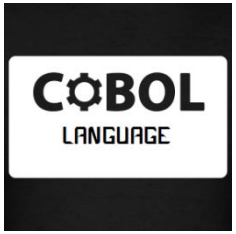
Programming Language

- Since the computer does not have a functional brain, people should write commands.

- An array of rules that have extremely strict rules is called **programming language**, which allows the algorithmic solution of a problem to be explained to the Computer.

Programming Language





[1]http://rigaux.org/language-study/diagram.html

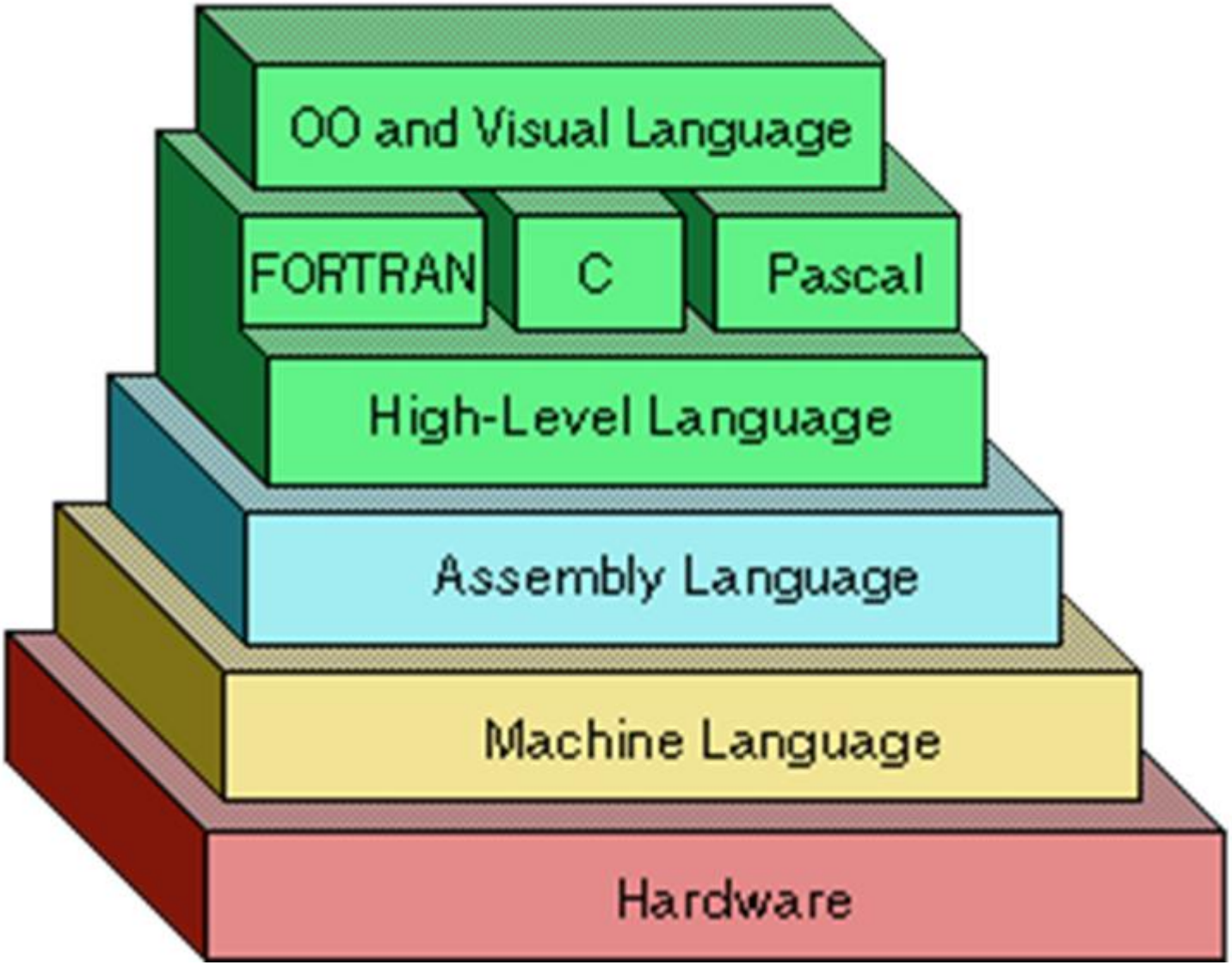
Classifying Programming Languages

- Programming languages are often classified according to their level.
- The level is a measure of the proximity of a programming language to human perception.
- Higher-level languages are closer to human perception, and low-level languages are closer to the computer's natural functioning.

Classifying Programming Languages

- The higher the level of the language, the easier it will be for the programmer.
- In high-level languages there are commands for what to do, not about how to do.
- Increasing the level makes everything easier for the programmer while reducing productivity and flexibility in general.

Classifying Programming Languages



Classifying Programming Languages

close to human

close to computer

- very high-level programming languages and visual languages
(Visual C# .NET, Visual Basic.NET, Java ...)
- high-level programming languages
(Pascal, Cobol, Fortran, Basic)
- medium level programming languages
(C)
- low-level programming languages
(Assembly)
- Machine Language (1 and 0)

Machine Languages

- Imagine them as the **“native tongue” of the computer**, the language closest to the hardware itself.
- Each unique computer has a unique machine language.
- A machine language program is **made up of a series of binary patterns** (e.g., 01011100) which represent simple operations that can be accomplished by the computer

Machine Languages

- Machine language programs are *executable,* meaning that they can be run directly.

- Programming in machine language requires memorization of the binary codes and can be difficult for the human programmer.

Assembly Languages

- Machine language programming is very slow and very convenient to make mistakes. Assembly languages represent an effort to make programming easier for the human.

- Instead of dealing with the number sequences that computers can directly understand, programmers have begun to use English abbreviations to represent basic operations.

- **The machine language instructions are replaced with simple pneumonic abbreviations (e.g., ADD, MOV).**

Assembly Languages

- These abbreviations form the basis of **assembly language**.

- Prior to execution, an assembly language program requires translation to machine language. This translation is accomplished by a computer program known as an **Assembler**.

Medium-level Programming Languages

- **Medium-level languages** contain structures close to both the user and the computer.
- Medium-level languages use the facility of high-level languages and the flexibility and naturalness of low-level languages.
- C is the most common mid-level language.
- Intermediate languages are used especially in writing system programs.

High-level Programming Languages

- **High-level programming languages** are more algorithmic languages.

- High level languages are written in a form that is close to our human language, enabling to programmer to just focus on the problem being solved.

High-level Programming Languages

- In these languages, algorithms for how to do things first are designed. Then these algorithms are converted to program code.

- These programmer friendly languages are called ‘high level’ as they are far removed from the machine code instructions understood by the computer.

- Examples include Basic, C++.

Very High-level Programming Languages

- **Very high-level languages** are languages in which the program code is produced partially or entirely by various means visually.

- C # and Visual Basic under the umbrella of .NET are used extensively in Windows systems.

- Very high-level languages are also called declarative languages.

Compiler

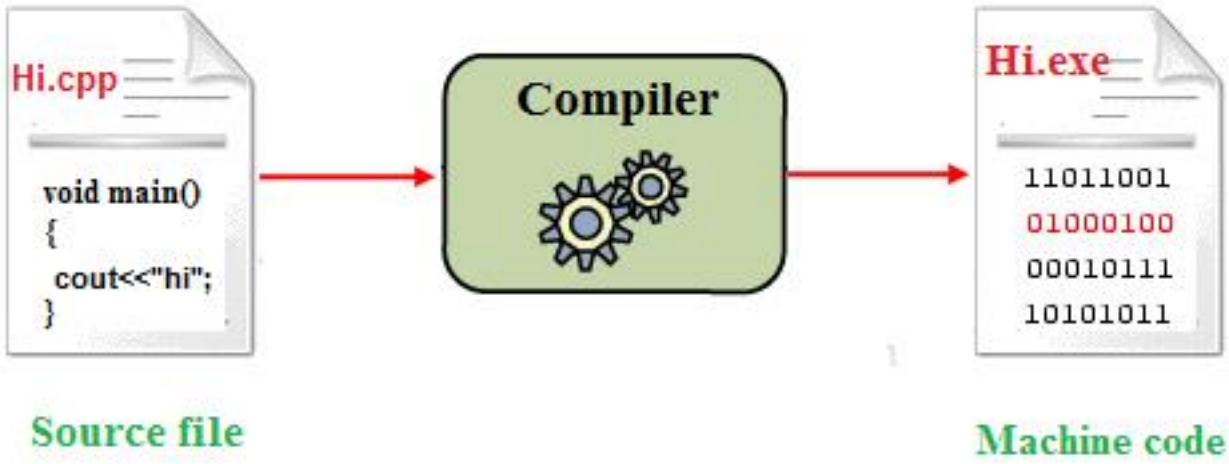
- A compiler is a software program that transforms high-level source code that is written by a developer in a high-level programming language into a low level object code (binary code) in machine language, which can be understood by the processor.

- The process of converting high-level programming into machine language is known as compilation.

Compiler

- The compiler will first check whether the program code written is correct, and if there are errors, inform the programmer.

- If the code is correct, it compiles the machine code consisting of 0 and 1s, which are suitable for the compiled system (EXE file).



Interpreter

- An interpreter is a computer program that is used to directly execute program instructions written using one of the many high-level programming languages.

- The interpreter transforms the high-level program into an intermediate language that it then executes, or it could parse the high-level source code and then performs the commands directly, which is done line by line or statement by statement.

Compiler vs Interpreter

Interpreter	Compiler
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
It takes less amount of time to analyze the source code but the overall execution time is slower.	It takes large amount of time to analyze the source code but the overall execution time is comparatively faster.
No intermediate object code is generated, hence are memory efficient.	Generates intermediate object code which further requires linking, hence requires more memory.
Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.	It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
Programming language like Python, Ruby use interpreters.	Programming language like C, C++ use compilers.

References

- Doç. Dr. Fahri Vatansever, “Algoritma Geliştirme ve Programlamaya Giriş”, Seçkin Yayıncılık, 12. Baskı, 2015.
- J. G. Brookshear, “Computer Science: An Overview 10th Ed.”, Addison Wisley, 2009.
- Kaan Aslan, “A’dan Z’ye C Klavuzu 8. Basım”, Pusula Yayıncılık, 2002.
- Paul J. Deitel, “C How to Program”, Harvey Deitel.
- T. Karaçay ve A. Karaçay, *Hiç Bilmeyenler İçin C# ile Programlamaya Giriş*, 2. Baskı. Ankara: Seçkin Yayıncılık, 2016.