Introduction to Computer Programming

Assoc. Prof. Özgür ZEYDAN

https://www.ozgurzeydan.com/

Computer Programming

- A computer is a programmable machine. This means it can execute a programmed list of instructions and respond to new instructions that it is given.

- Computer Programming is the process of developing and implementing various sets of instructions to enable a computer to do a certain task.

- Programs are written to solve problems or perform tasks on a computer.

Computer Programming

➤ Programmers translate the solutions or tasks into a language the computer can understand.

➤ As we write programs, we must keep in mind that the computer will only do what we instruct it to do.

➤ Because of this, we must be very careful and thorough with our instructions.

CAN'T YOU DO ANYTHING RIGHT?

Copyright 2003 Randy Glasbergen.   www.glasbergen.com

First Computer Programmer: Ada Lovelace

Ada Lovelace is the first person to develop an algorithm for a machine.

In 1842-1843, Lovelace translated an article about Charles Babbage's proposed Analytic Engine. In her notes, she describes an algorithm that is cited as the first computer program, making her the first computer programmer.

She also theorized that the computer could, one day, play music and chess.

Ada, a U.S. Department of Defense computer language, is named in her honor.

Algorithm

- An **algorithm** is a list of instructions, procedures, or formulas used to solve a problem.
- The word derives from the name of the mathematician, Mohammed ibn-Musa al-Khwarizmi (**El-Harezmî**), (780 – 850).

Pseudocode

➤ Pseudocode is a computer programming language that resembles plain English that cannot be compiled or executed, but explains a resolution to a problem.

Source Code

➢ The source code consists of the programming statements that are created by a programmer with a text editor or a visual programming tool and then saved in a file.

➢ For example, a programmer using the C language types in a desired sequence of C language statements using a text editor and then saves them as a named file.

➢ This file is said to contain the source code.

Flowchart

- A flowchart is a formalized graphic representation of a logic sequence, work or manufacturing process, organization chart, or similar formalized structure.

- The purpose of a flow chart is to provide people with a common language or reference point when dealing with a project or process.

- Flowcharts use simple geometric symbols and arrows to define relationships.

Programming Languages

- Computer programming is almost always done by means of Programming Language.

- There exists 8945 programming languages in the world (https://hopl.info).

- Some of them are known by only their developers!

- For further information: http://en.wikipedia.org/wiki/List_of_programming_languages

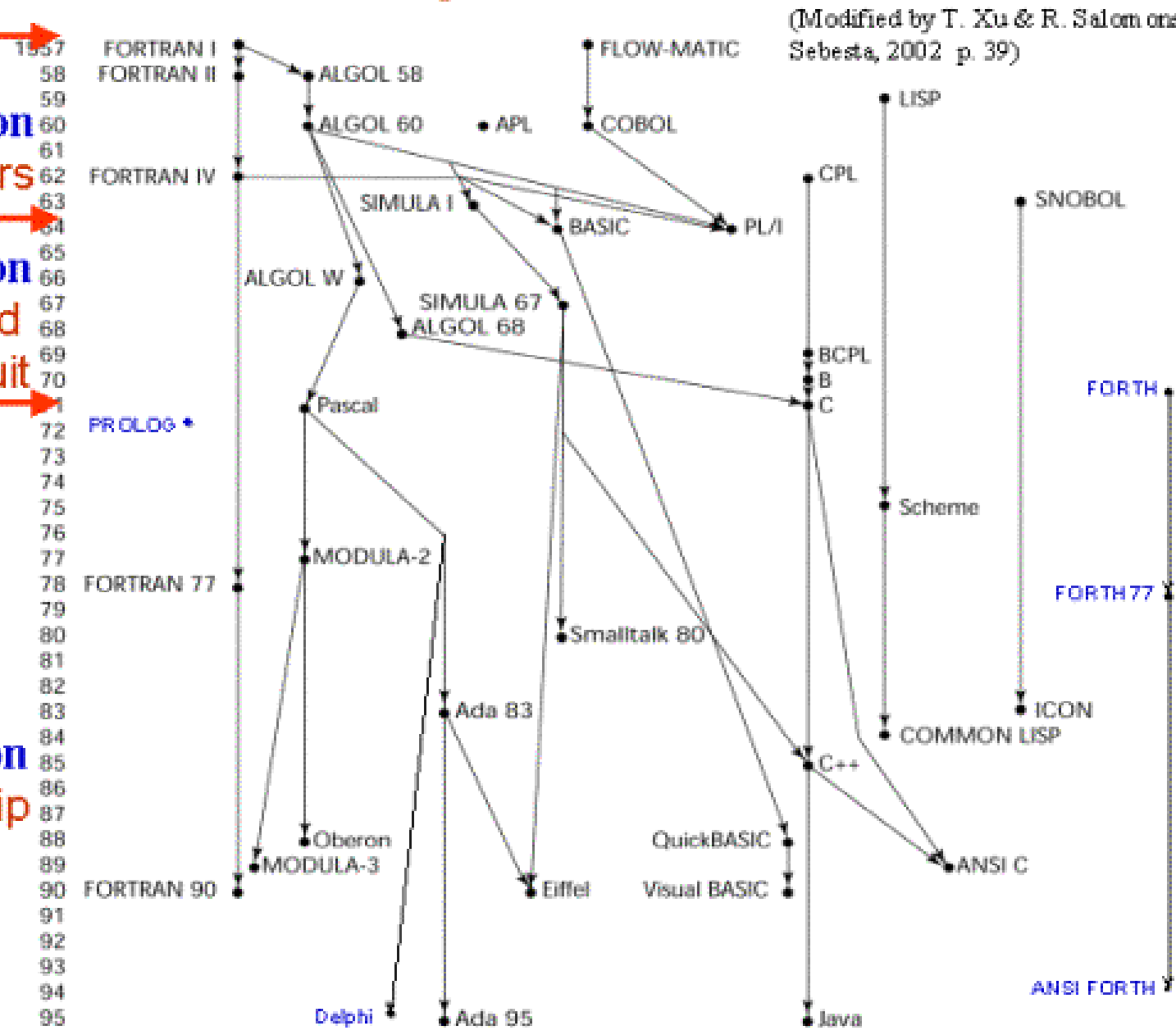# Programming Languages

History of Programming Languages
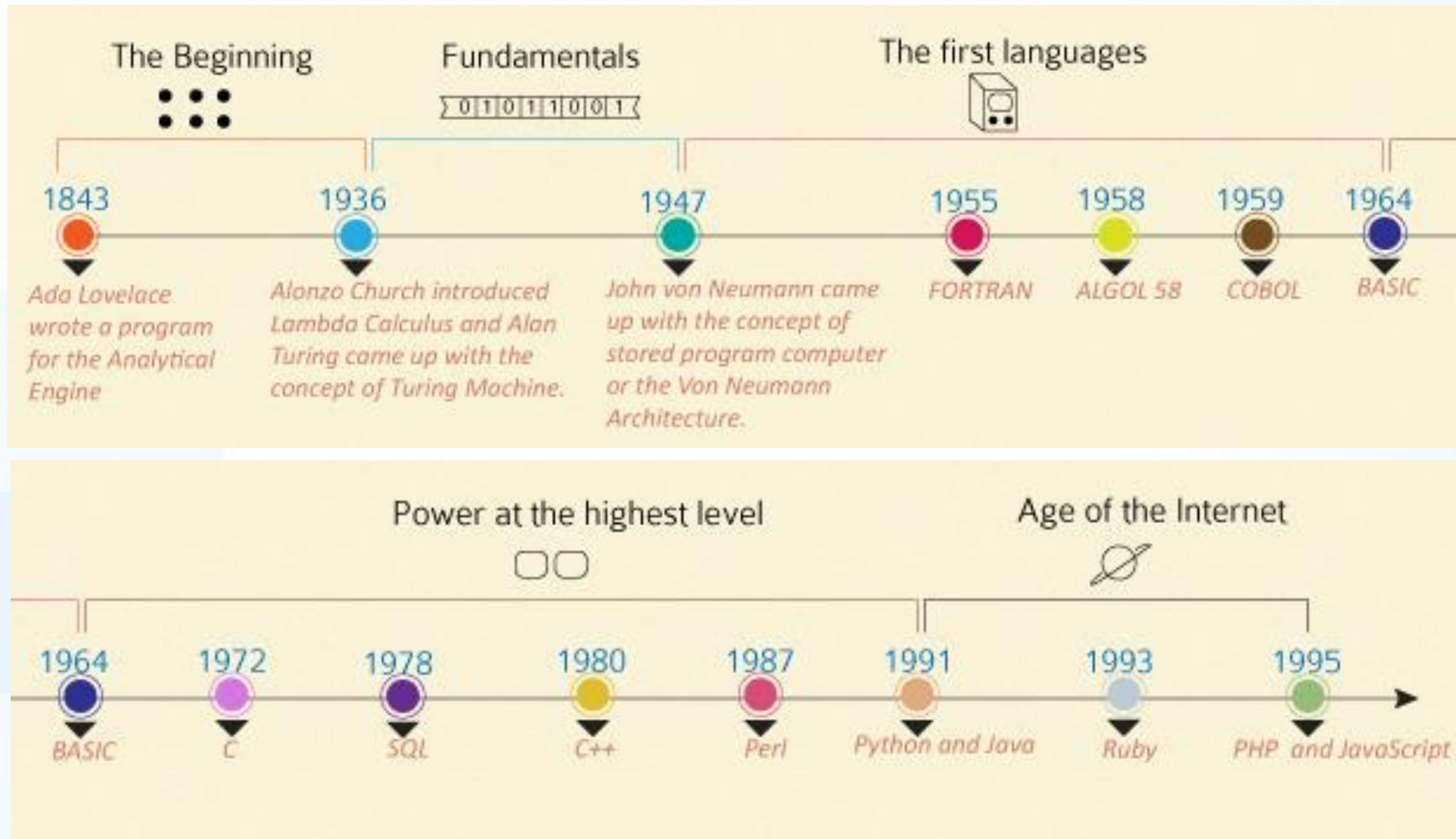
1st Generation Vacuum Tubes, Magnetic drums

2nd Generation
Transistors

3rd Generation
Integrated
Circuit

4th Generation
Chip

(Modified by T. Xu & R. Salomons from Sebesta, 2002 p. 39)

12

Programming Language Generations

➤ 1GL or first-generation language was (and still is) machine language or the level of instructions and data that the processor is actually given to work on.

➤ 2GL or second-generation language is assembler (sometimes called "assembly") language.

Programming Language Generations

➢ 3GL or third-generation language is a "high-level" programming language, such as PL/I, C, or Java. A compiler converts the statements of a specific high-level programming language into machine language. A 3GL language requires a considerable amount of programming knowledge.

➢ 4GL or fourth-generation language is designed to be closer to natural language than a 3GL language. Languages for accessing databases are often described as 4GLs.

Programming Language Generations

➢ 5GL or fifth-generation language is programming that uses a visual or graphical development interface to create source language that is usually compiled with a 3GL or 4GL language compiler. Microsoft, Borland, IBM, and other companies make 5GL visual programming products for developing applications in Java, for example. Visual programming allows you to easily envision object-oriented programming class hierarchies and drag icons to assemble program components.

Machine Code (machine language)

- Machine code, also known as machine language, is the elemental language of computers, comprising a long sequence of binary digital zeros and ones (bits).

- Sometimes referred to as machine code or object code, machine language is a collection of binary digits or bits that the computer reads and interprets. Machine language is the only language a computer is capable of understanding.

Machine Code
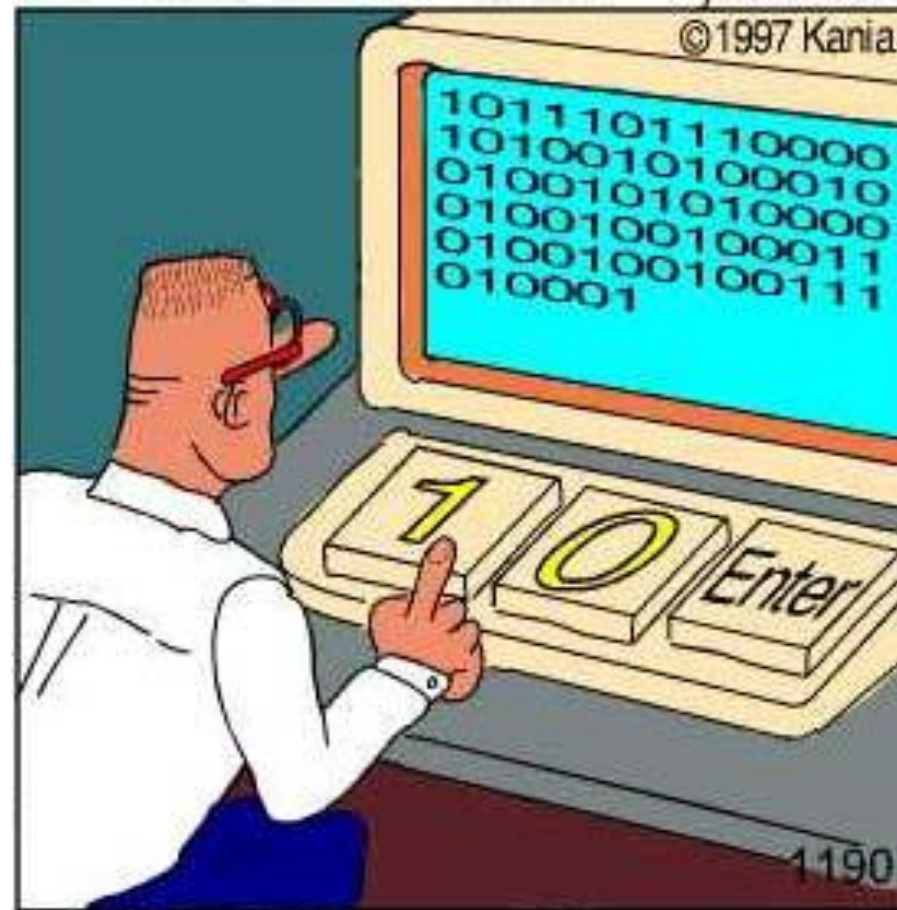
Real programmers code in binary.

Assembly Language

➢ Sometimes referred to as assembly or ASL, assembly language is a low-level programming language used to interface with computer hardware.

➢ Assembly language uses structured commands as substitutions for numbers allowing humans to read the code easier than looking at binary. Although easier to read than binary, assembly language is a difficult language and is usually substituted for a higher language such as C.

Low-level Languages
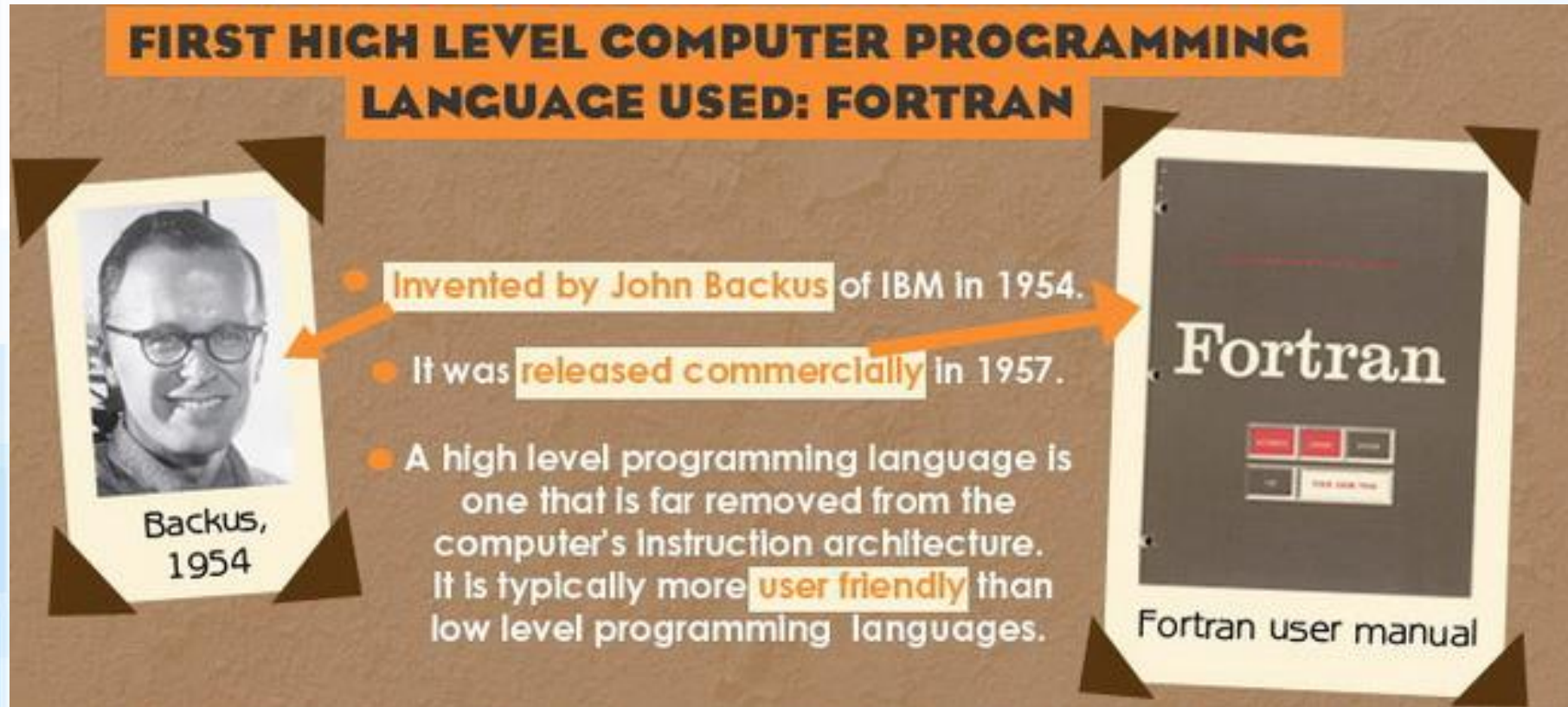
➢ Low-level languages have the advantage that they can be written to take advantage of any peculiarities in the architecture of the central processing unit (CPU).

➢ Thus, a program written in a low-level language can be extremely efficient, making optimum use of both computer memory and processing time.

➢ However, to write a low-level program takes a substantial amount of time, as well as a clear understanding of the inner workings of the processor itself. Therefore, low-level programming is typically used only for very small programs, or for segments of code that are highly critical and must run as efficiently as possible.

High-level Languages

➢ High-level languages permit faster development of large programs. The final program as executed by the computer is not as efficient, but the savings in programmer time generally far outweigh the inefficiencies of the finished product.

➢ This is because the cost of writing a program is nearly constant for each line of code, regardless of the language.

➢ Thus, a high-level language where each line of code translates to 1-0 machine instructions costs only one tenth as much in program development as a low-level language where each line of code represents only a single machine instruction.

First High-Level Language

FIRST HIGH LEVEL COMPUTER PROGRAMMING LANGUAGE USED: FORTRAN

Invented by John Backus of IBM in 1954.

It was released commercially in 1957.

A high level programming language is one that is far removed from the computer's instruction architecture. It is typically more user friendly than low level programming languages.

Backus, 1954

Fortran

Fortran user manual

Design and Illustration
By Ellie Koning

Programming Language Popularities

Aug 2023	Aug 2022	Change		Programming Language	Ratings	Change
1	1			Python	13.33%	-2.30%
2	2			C	11.41%	-3.35%
3	4	⌃		C++	10.63%	+0.49%
4	3	⌄		Java	10.33%	-2.14%
5	5			C#	7.04%	+1.64%
6	8	⌃		JavaScript	3.29%	+0.89%
7	6	⌄		Visual Basic	2.63%	-2.26%
8	9	⌃		SQL	1.53%	-0.14%
9	7	⌄		Assembly language	1.34%	-1.41%
10	10			PHP	1.27%	-0.09%
11	21	⌃⌃		Scratch	1.22%	+0.63%
12	15	⌃		Go	1.16%	+0.20%
13	17	⌃⌃		MATLAB	1.05%	+0.17%

https://www.tiobe.com/tiobe-index/

Compiler

➢ A compiler is a special program that processes statements written in a particular programming language and turns them into machine language or "code" that a computer's processor uses.

➢ After you write a program, your source language statements are compiled into machine code that is stored as an executable file.

➢ Scripting languages like Perl and PHP do not need to be compiled.

Compiler
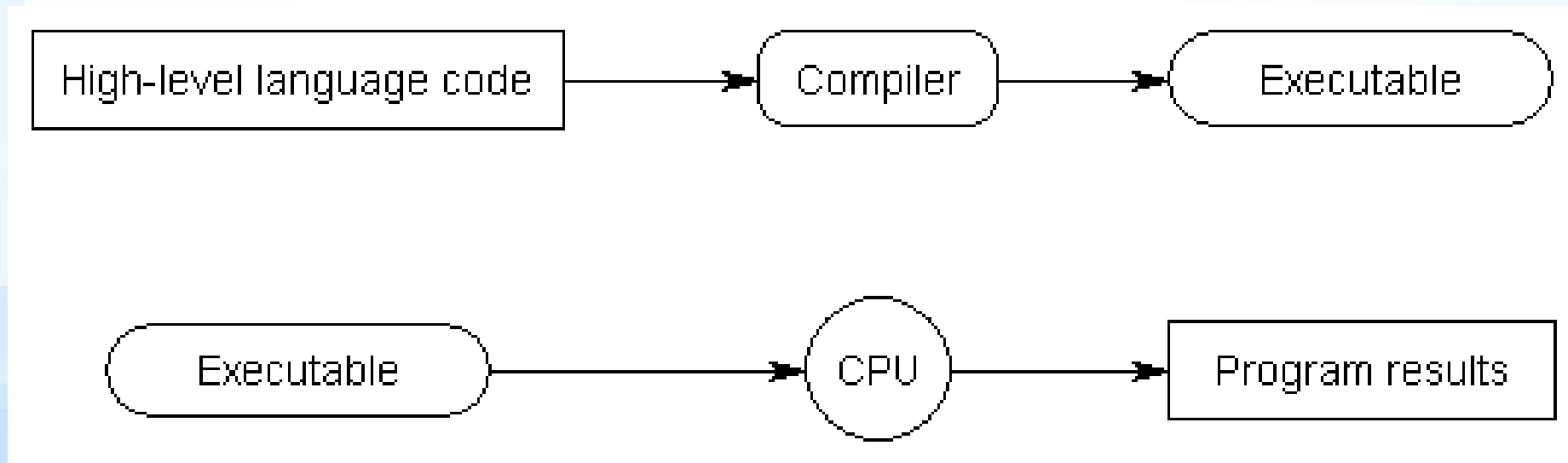
Structured Programming (modular programming)

- Structured programming (sometimes known as modular programming) is a subset of procedural programming that enforces a logical structure on the program being written to make it more efficient and easier to understand and modify. Certain languages such as Ada, Pascal, and dBASE are designed with features that encourage or enforce a logical program structure.

- Structured programming frequently employs a top-down design model, in which developers map out the overall program structure into separate subsections.

- Program flow follows a simple hierarchical model that employs looping constructs such as "for", "repeat", and "while" Use of the "Go To" statement is discouraged.

- Structured programming was first suggested by Corrado Bohm and Guiseppe Jacopini. The two mathematicians demonstrated that any computer program can be written with just three structures: decisions, sequences, and loops.

Object-Oriented Programming (OOP)

> Object-oriented programming (OOP) is a programming language model organized around "objects" rather than "actions" and data rather than logic.

> Historically, a program has been viewed as a logical procedure that takes input data, processes it, and produces output data.

Classification of Programming Languages

- **Procedure-oriented programming**
- COBOL, FORTRAN, Pascal and C

- **Object oriented programming**
- Objective C, C++, Java, and PHP

Integrated Development Environment (IDE)

An IDE or Integrated Development Environment is a software program that is designed to help programmers and developers build software.

Most IDEs include:

➢a source code editor
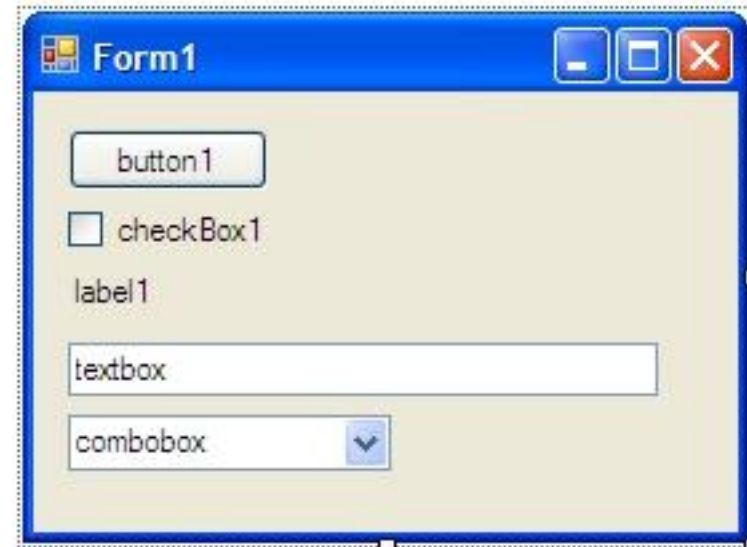
➢a compiler

➢build automation tools

➢a debugger

Debugger

➢ A special program used to find errors (bugs) in other programs. A debugger allows a programmer to stop a program at any point and examine and change the values of variables.
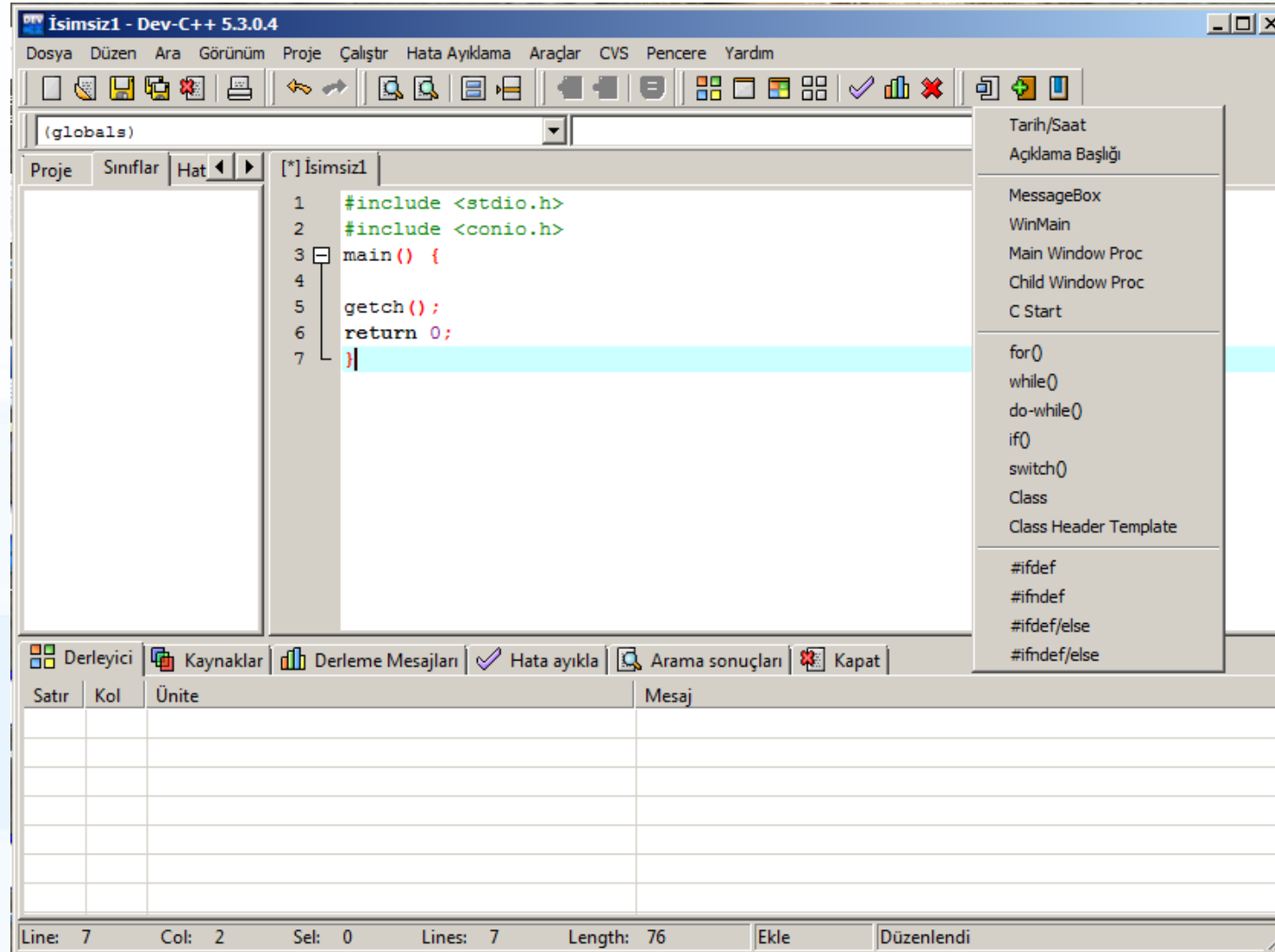
➢ https://www.webopedia.com/definitions/debugger/

Graphical User Interface (GUI)

➢ A GUI is a graphical (rather than purely textual) user interface to a computer.

➢ Elements of a GUI include textboxes, buttons, pulldown menues, list and combo boxes

DEV C++ IDE

DEV C++ IDE download page

http://dev-cpp.com

Visual Studio Community

Visual Studio Community

Windows kullanan .NET ve C++ geliştiricileri için en iyi kapsamlı IDE. Yazılım geliştirmenin her aşamasını yükseltip geliştiren bir dizi güzel araçlar ve özelliklerle tam olarak paketlenmiştir.

Daha fazla bilgi edinin →

Ücretsiz indirin

https://visualstudio.microsoft.com/tr/free-developer-offers/

Software Development Languages

- C
- C++ (C-plus-plus)
- C# (C-Sharp)
- Python

Web Languages

- HTML (Hyper Text Markup Language)
- XML (Extensible Markup Language)
- Javascript
- VBScript
- PHP (Hypertext Preprocessor)
- Java
- ASP (Active Server Pages)

Important People in Computer Programming

Charles Babbage
{He first came up with the idea of difference engine & analytical engine and is regarded as father of computer}

John Backus
{He is well known for the development of FORTRAN and ALGOL. He is also the inventor of Backus-Naur form and has also helped to popularize functional level programming}

Alan Turing
{He is well known for the Halting problem, Turing machines, crypto-analysis of Enigma & Turing test. Turing award is given annually for exceptional work in the field of computing}

Dennis Ritchie
{He is the creator of C programming language and was also amongst the key developers of UNIX operating system. He received the Turing award in 1983}

John von Neumann
{He came up with the concept ofstored program computer that uses a CPU and a separate storage to hold both instructions and data. This is also known as von Neumann architecture}

Ken Thompson
{He is well known as the principal creator of the UNIX operating system and is also the co-creator of the Go programming language}

36

Important People in Computer Programming

Bjarne Stroustrup

{He is well known for the creation and development of C++programming language and currently holds the college of engineering chair in computer science at Texas A&M.}
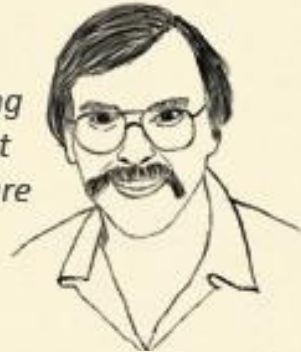
Larry Wall

{He is well known for the creation of Perl programming language and is also the first recepient of the Free Software Foundation Award for the Advancement of Free Software}

Guido van Rossum

{He is well known as the author of Python programming language and is currently employed by Google}

Richard Stallman

{He is the creator of Emacs editor and the lead architect and organizer of the GNU project. He has been actively involved in the free software movement}

Linus Torvalds

{He is best known for having initiated the development of Linux Kernel and the Git revision control system. He is also a strong supporter of Open Source software}

G² GeekGraphic
part of what - a - geek.com

Designed By *Adit Gupta*

37

Algorithms
Pseudocode
Flowcharts

Assoc. Prof. Özgür ZEYDAN

https://www.ozgurzeydan.com/

Why do we have to learn computer programming?

➢ Computers can make calculations at a blazing speed without any error as compared to the humans.

➢ Example: calculate the prime numbers until 121.

➢ How long does it take?

➢ Do you do any error?

➢ Prime numbers: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 109, 113

➢ Such a problem is so easy for a computer...

Calculate the prime numbers until 121

http://en.wikipedia.org/wiki/File:Sieve_of_Eratosthenes_animation.gif

Performing a Task on The Computer

➢ The first step in writing instructions to carry out a task is to determine what the output should be—that is, exactly what the task should produce.

➢ The second step is to identify the data, or input, necessary to obtain the output.

➢ The last step is to determine how to process the input to obtain the desired output, that is, to determine what formulas or ways of doing things can be used to obtain the output.

A pictorial representation of problem-solving
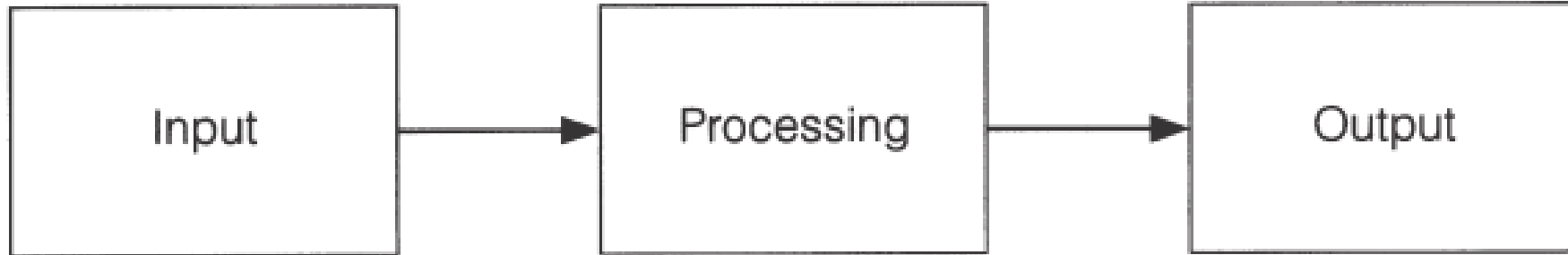
Program Development Cycle

1. Analyze: Define the problem.

- Be sure you understand what the program should do, that is, what the output should be. Have a clear idea of what data (or input) are given and the relationship between the input and the desired output.

2. Design: Plan the solution to the problem.

- Find a logical sequence of precise steps that solve the problem. Such a sequence of steps is called an algorithm. Every detail, including obvious steps, should appear in the algorithm.

Program Development Cycle

3. **Choose the interface**: Select the objects.
  - Determine how the input will be obtained and how the output will be displayed. Then create appropriate commands to allow the user to control the program.

4. **Code**: Translate the algorithm into a programming language.
  - Coding is the technical word for writing the program.

5. **Test and debug**: Locate and remove any errors in the program.
  - Testing is the process of finding errors in a program, and debugging is the process of correcting errors that are found. (An error in a program is called a bug.)

Program Development Cycle

6. Complete the documentation: Organize all the material that describes the program.

- Documentation is intended to allow another person, or the programmer at a later date, to understand the program. Internal documentation consists of statements in the program that are not executed, but point out the purposes of various parts of the program.

- Documentation might also consist of a detailed description of what the program does and how to use the program (for instance, what type of input is expected).

- For commercial programs, documentation includes an instruction manual.

- Other types of documentation are the flowchart and pseudocode.

How the customer explained it
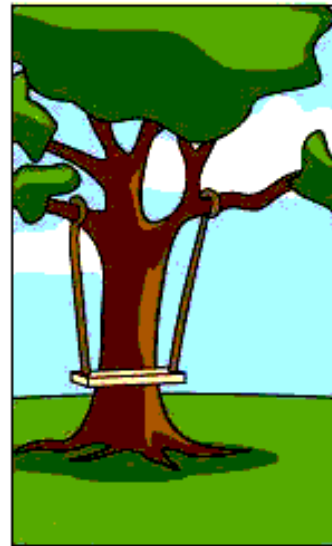
How the Project Leader understood it
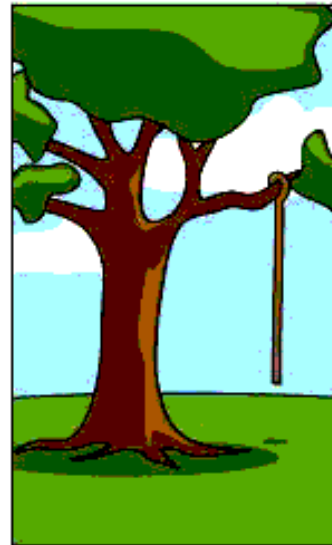
How the Analyst designed it

How the Programmer wrote it
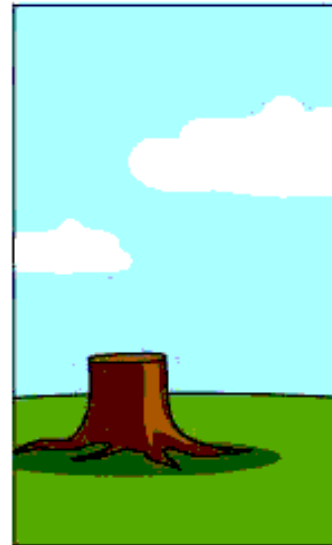
How the Business Consultant described it

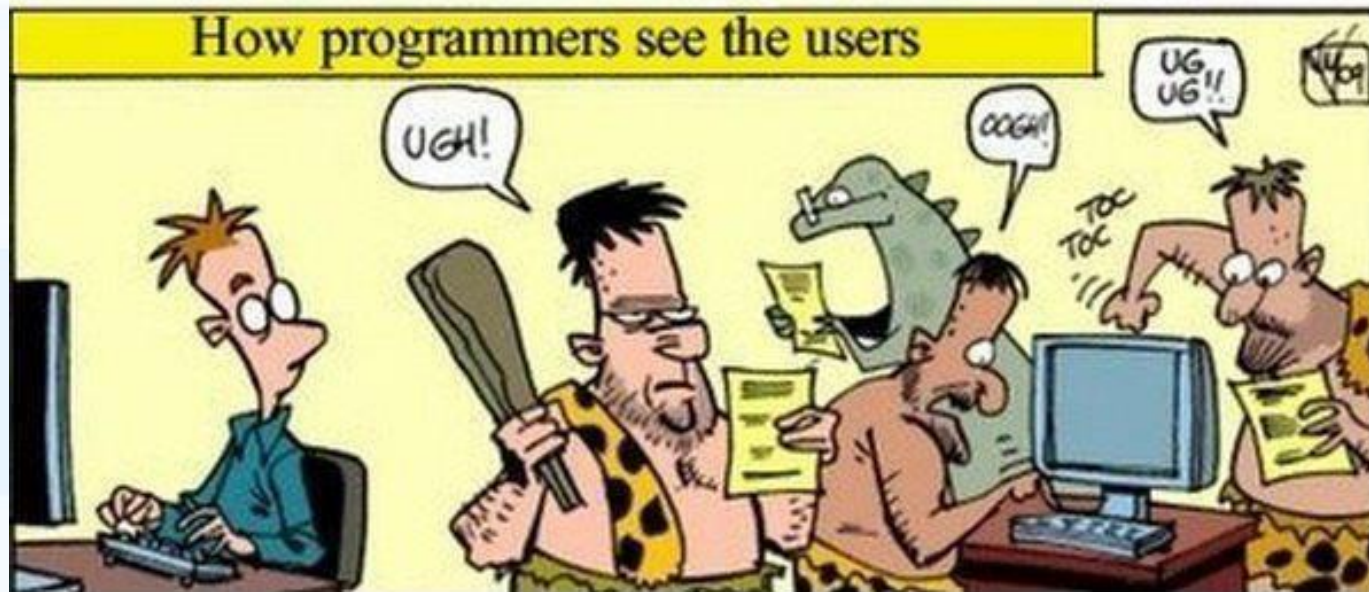How the project was documented

What operations installed
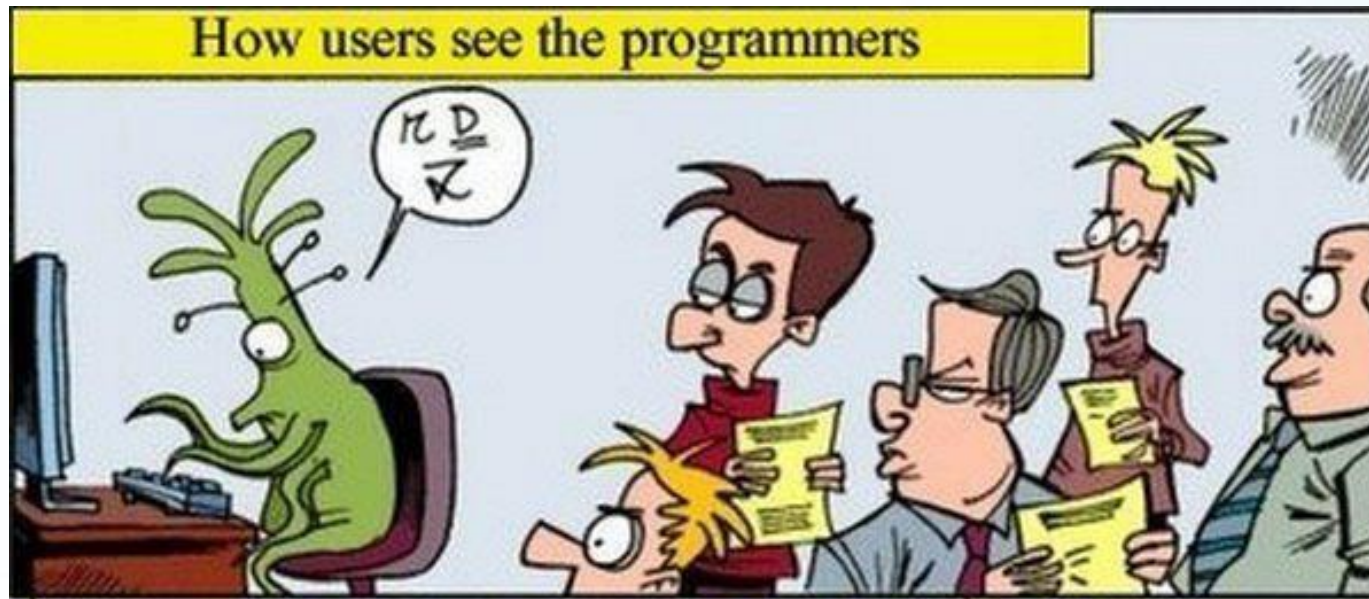
How the customer was billed

How it was supported

What the customer really needed

46

How users see the programmers

How programmers see the users

lolngags.com

47

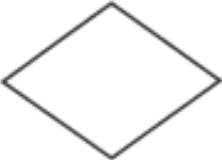# Programming Tools

➢ Tools used to convert algorithms into computer programs:

- Pseudocode: An informal high-level description of the operating principle of a computer program. It uses the structural conventions of a programming language, but is intended for human reading rather than machine reading.

- Flowcharts: Graphically depict the logical steps to carry out a task and show how the steps relate to each other.

Pseudocode vs Flowcharts

- Artificial and Informal language
- Helps programmers to plan an algorithm
- Similar to everyday English
- Not an actual programming language

- A graphical way of writing pseudocode
- Rounded rectangle – terminal
- Parallelogram – input / output
- Rectangle – actions
- Diamonds – decision / conditional
- Circles – connector

Flowchart Symbols

Symbol,	Name,	Meaning
→	Flowline	Used to connect symbols and indicate the flow of logic.
(rounded rectangle)	Terminal	Used to represent the beginning (Start) or the end (End) of a task.
(parallelogram)	Input/Output	Used for input and output operations, such as reading and printing. The data to be read or printed are described inside.
(rectangle)	Processing	Used for arithmetic and data-manipulation operations. The instructions are listed inside the symbol.
(diamond)	Decision	Used for any logic or comparison operations. Unlike the input/output and processing symbols, which have one entry and one exit flowline, the decision symbol has one entry and two exit paths. The path chosen depends on whether the answer to a question is "yes" or "no."

Example Pseudocode

Start

Read A, B

Calculate C = A*B
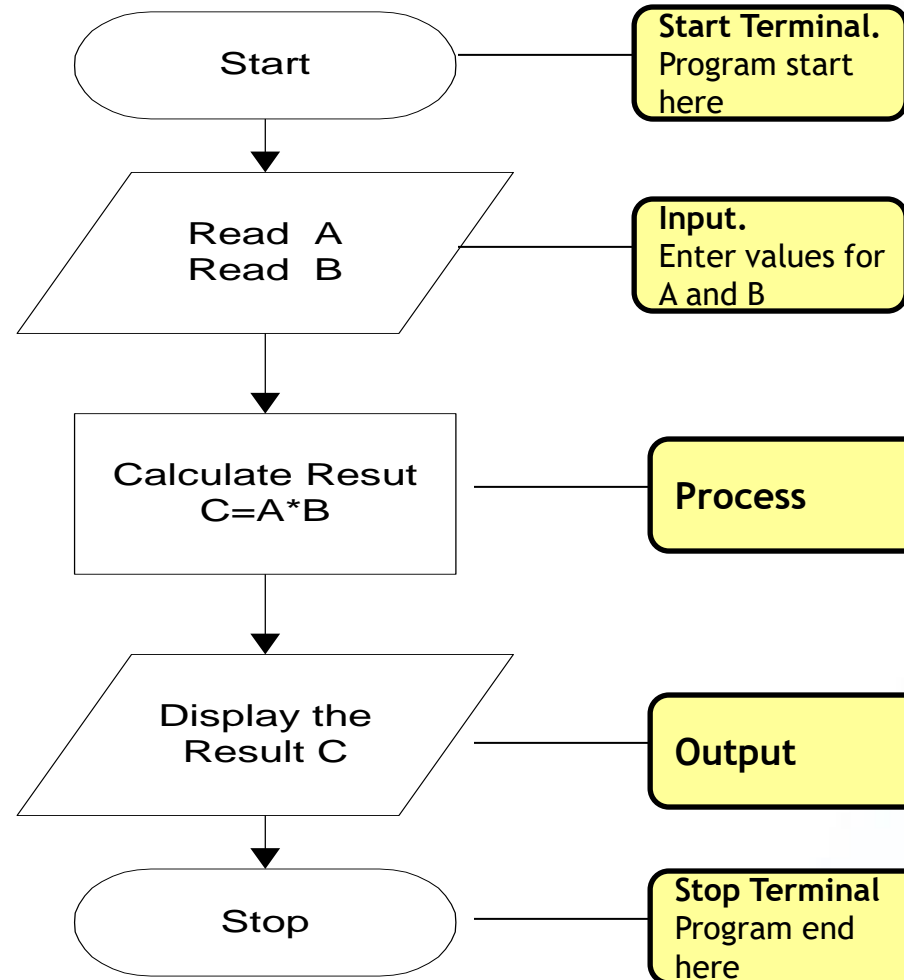
Display C

Stop

Start - Terminal

Read A, B – Input

Calculate C = A*B - Action

Display C - Output

Stop - Terminal

Example Flowchart

Start

Start Terminal.
Program start here

Read A
Read B

Input.
Enter values for A and B

Calculate Resut
C=A*B

Process

Display the
Result C

Output

Stop

Stop Terminal
Program end here

User Friendly Pseudocode

Start

Use variables A,B and C

Display "write two numbers"

Read A, B

Calculate C = A*B

Display "multiplication of numbers" , C

Stop

Question ???

➤ Write an algorithm to calculate Fahrenhayt value of temperature if celcius valeu is given.

- $\dfrac{(F-32)}{180} = \dfrac{(C)}{100}$

- Write a pseudocode.
- Draw a flowchart.

Structured Programming

➢ Structured programming was first suggested by Corrado Bohm and Guiseppe Jacopini. The two mathematicians demonstrated that any computer program can be written with just three structures: sequences, decisions, and loops.

➢ Sequences: one command is executed after previous one.

➢ Decisions (selections): statement(s) is (are) executed if certain condition gives TREU or FALSE value.

➢ Loops (repetition): statement(s) is (are) executed repeatedly until certain condition gives TREU or FALSE value.

➢ Corrado; B. and Jacopini, G. (May 1966). "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules". Communications of the ACM 9 (5): 366–371.
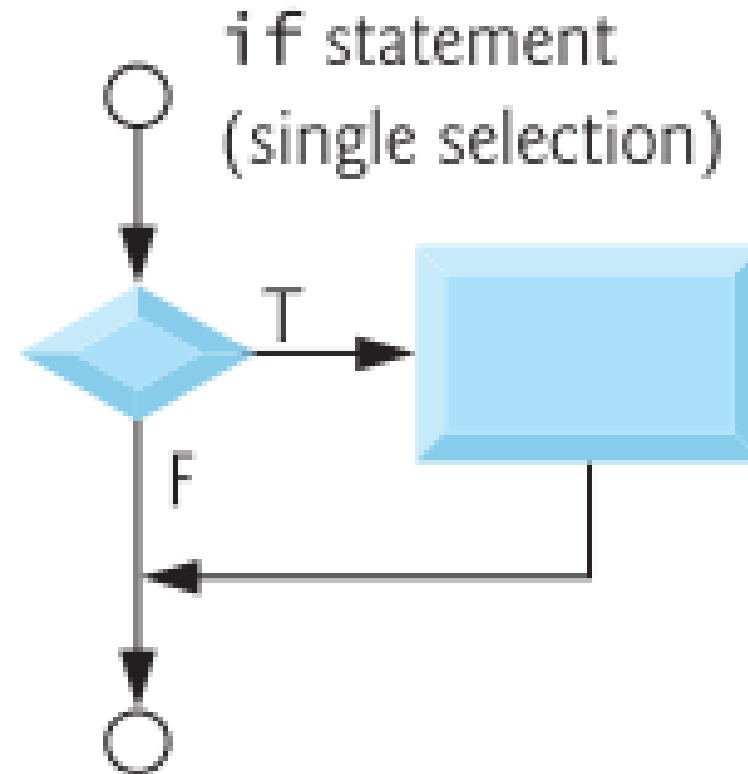
Sequences
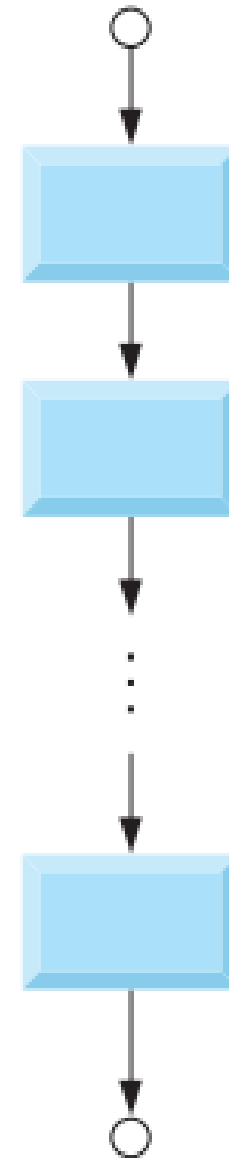
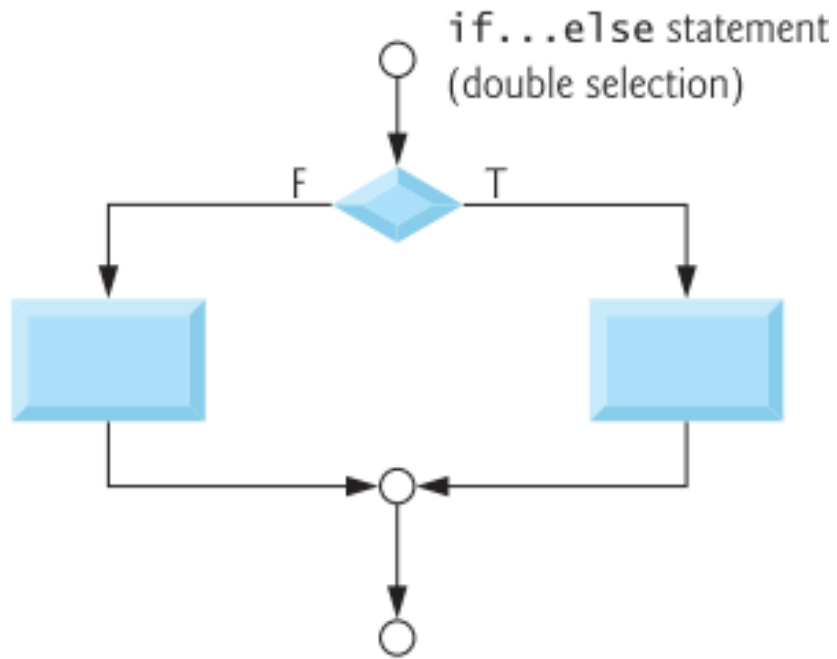Ref: Deitel P J (Ed.) (2010) C How to Program, 6th Edition, Prentice Hall

Decisions (selections)

➢ Three selection structure in C programming:
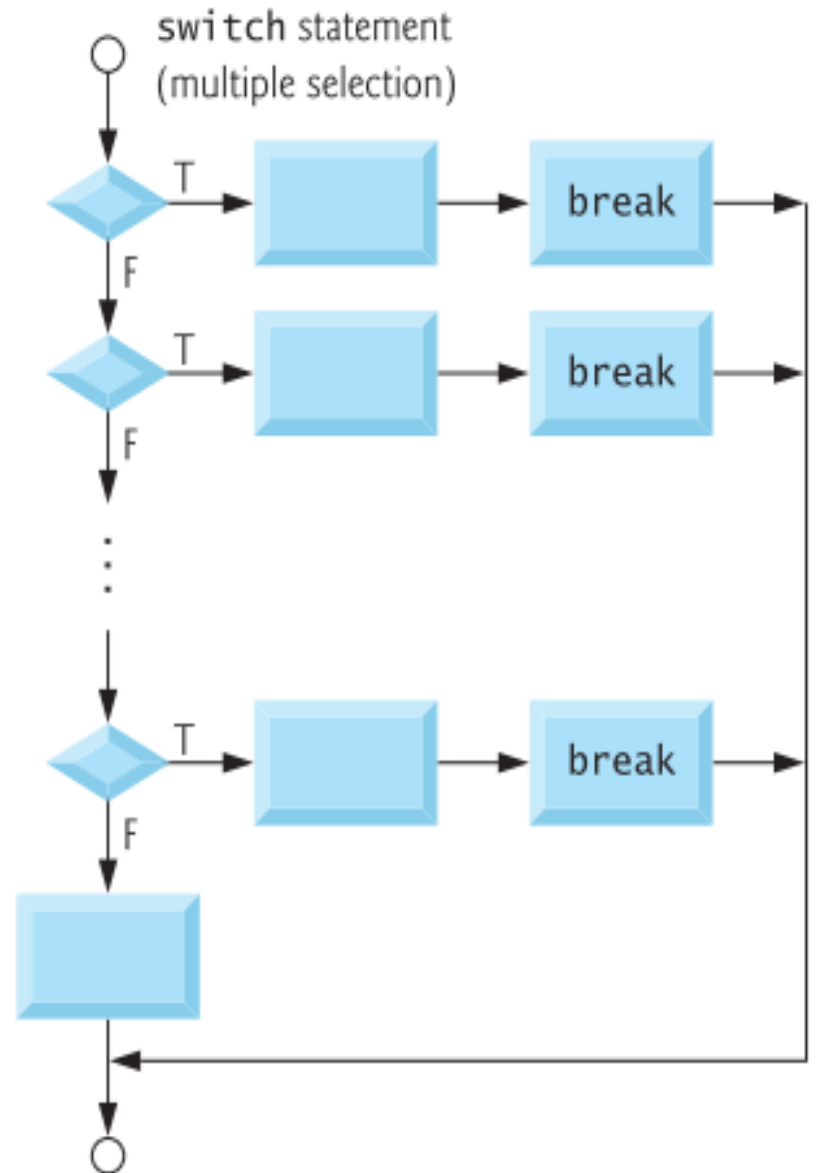
- If
- If – else
- Switch

Ref: Deitel P J (Ed.) (2010) C How to Program, 6th Edition, Prentice Hall

if statement
(single selection)

Decisions (selections)

if...else statement
(double selection)

switch statement
(multiple selection)

break

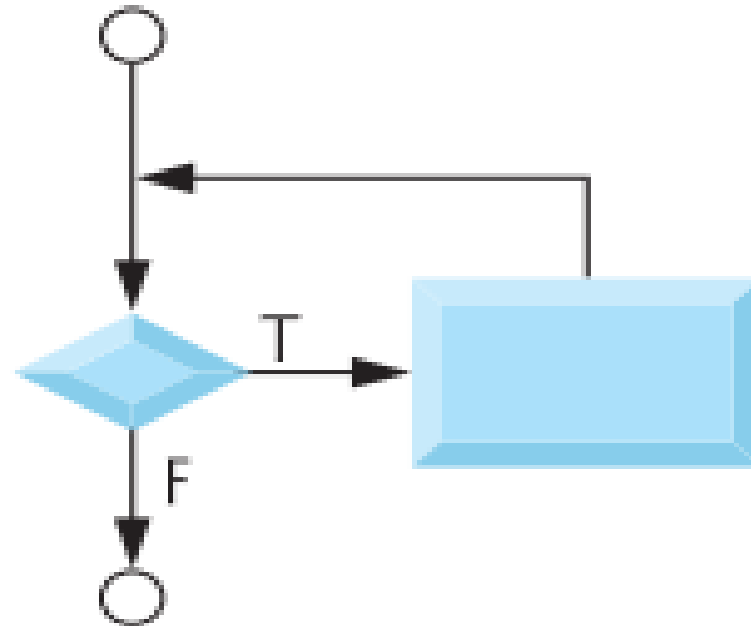Ref: Deitel P J (Ed.) (2010) C How to Program, 6th Edition, Prentice Hall

Loops (repetition)

➢ **Three repetition structure in C programming:**
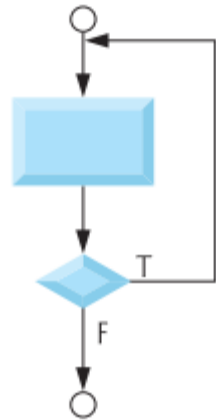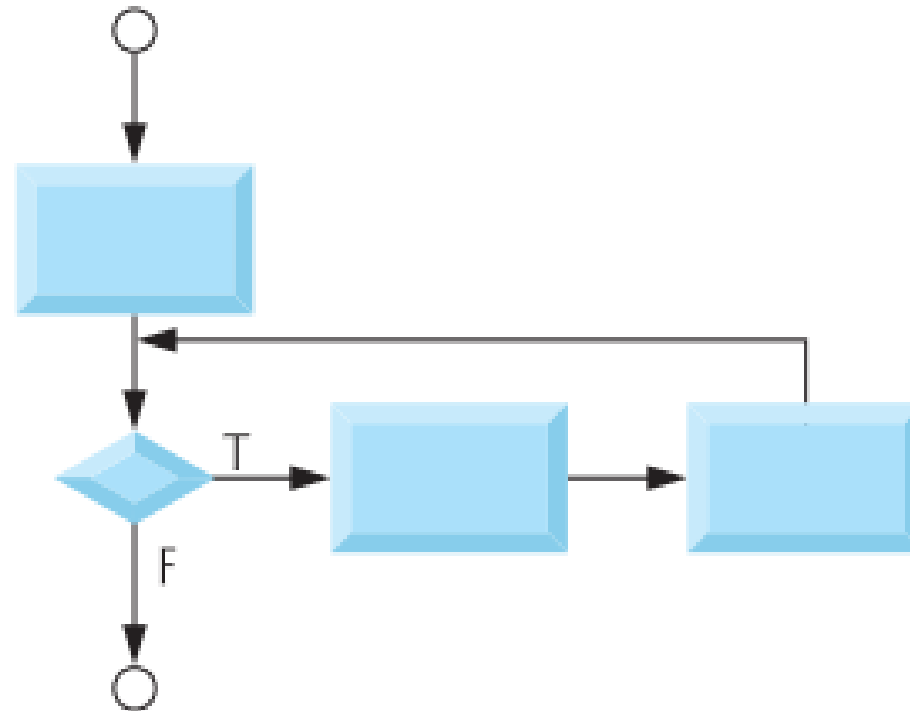
- ▪ While
- ▪ Do – while
- ▪ For

Ref: Deitel P J (Ed.) (2010) C How to Program, 6th Edition, Prentice Hall

while statement

Loops (repetition)
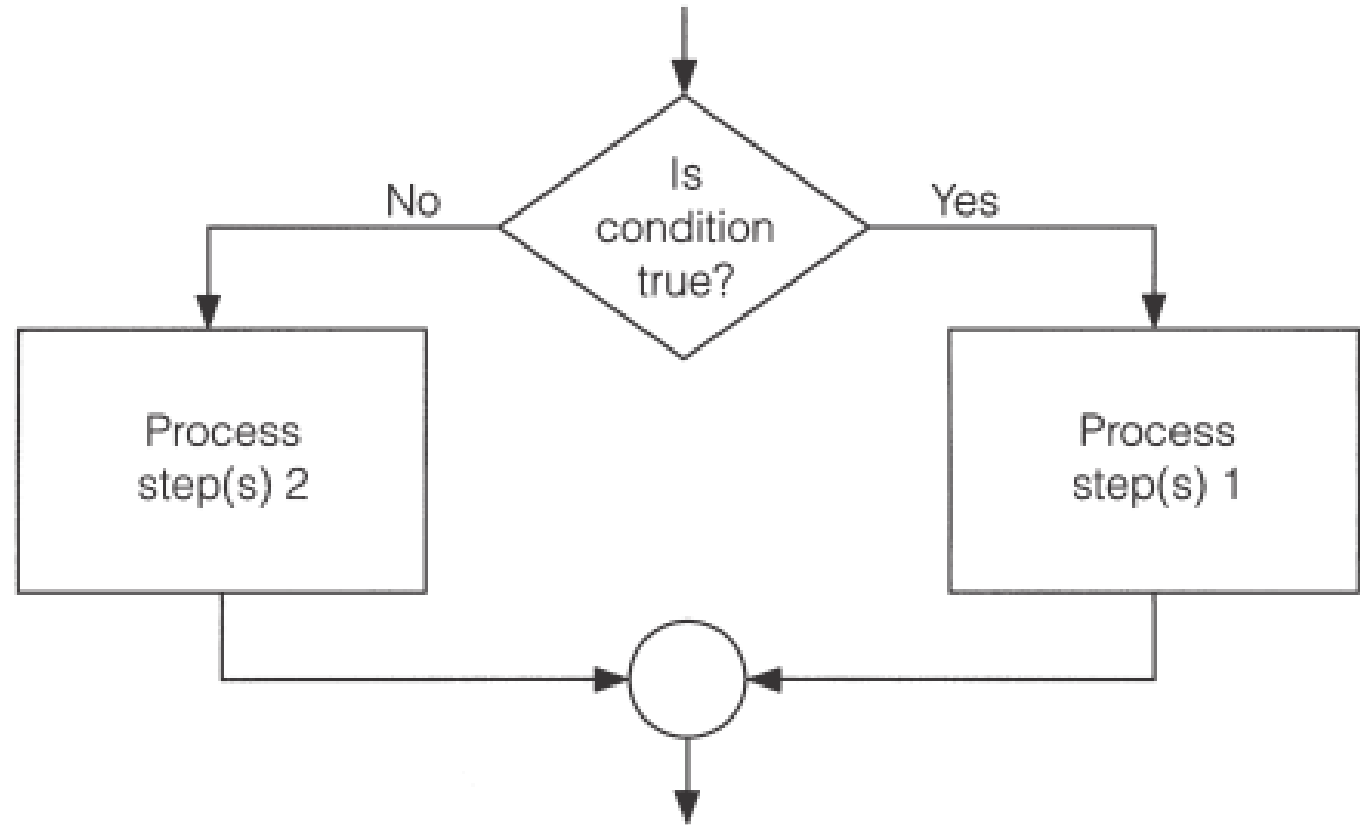

Ref: Deitel P J (Ed.) (2010) C How to Program, 6th Edition, Prentice Hall

Pseudocode and Flowchart for a Decision Structure

If condition is true Then
        Process step(s) 1
    Else
        Process step(s) 2
End If

Example - 2

- Write an algorithm to determine a student's average grade and indicate whether he is successful or not.

- The average grade is calculated as the average of mid-term and final marks.

- Student will be successful if his average grade is grater or equals to 60.

Pseudocode

- Start
- Use variables mid term , final and average
- Input mid term and final
- Calculate the average by summing mid term and final and dividing by 2
- if average is below 60

    Print "FAIL"

else

    Print "SUCCESS"

- Stop

Detailed Algorithm
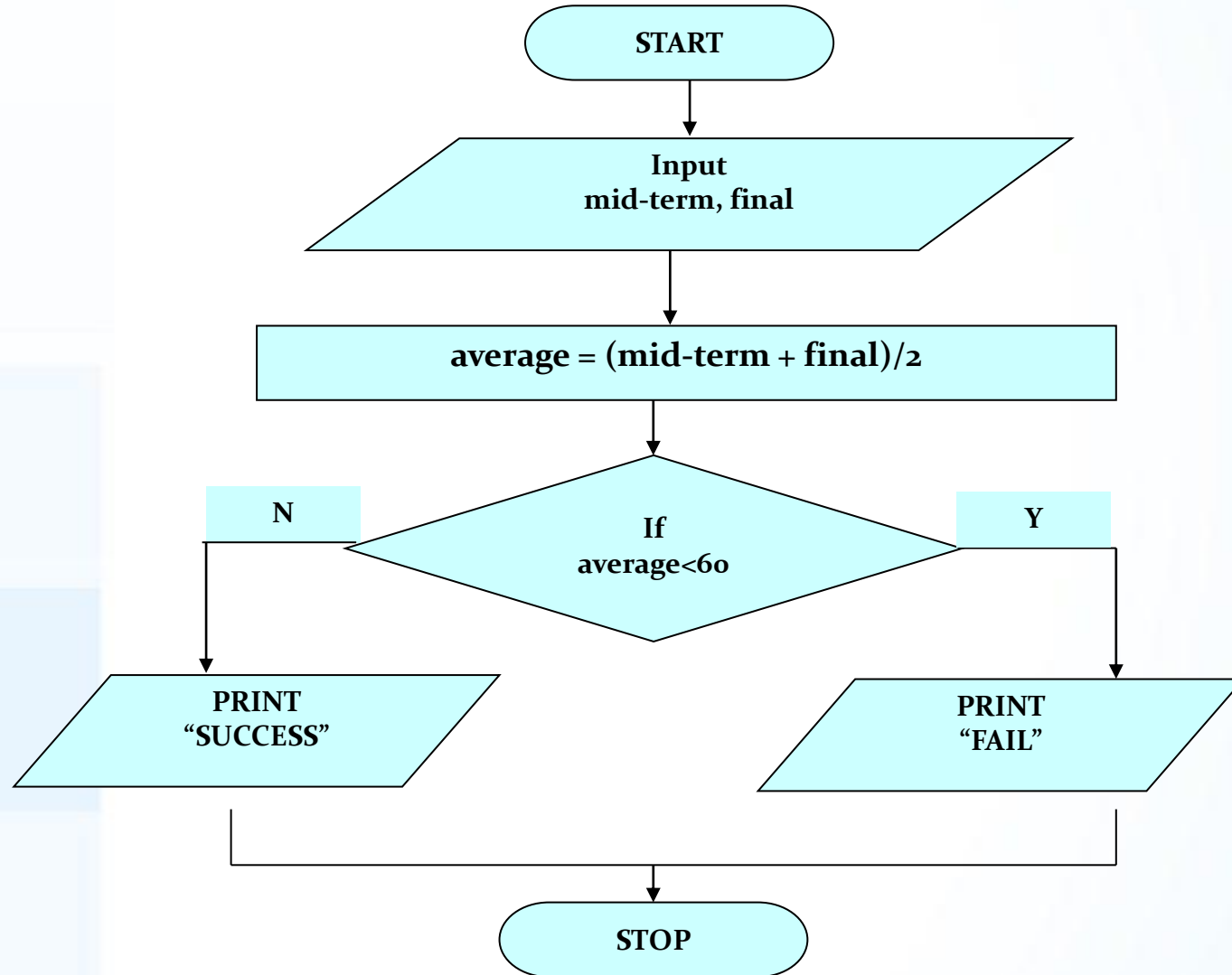
- 1. Step: Input mid-term and final
- 2. Step: average = (mid-term + final)/2
- 3. Step: if (average < 60) then
             Print "FAIL"

         else

             Print "SUCCESS"

     endif

Flowchart

Nested If

- Simply, if structure in if structure
- Example - 3: Both final and average grades must be grater than or equals to 35 for curve calculation in BEU.
- if (final >= 35) then

      {   if (average >= 35) then

         execute curve calculation commands

       endif   }

else

   Print "FF grade"

endif

Pseudocode and Flowchart for a Loop
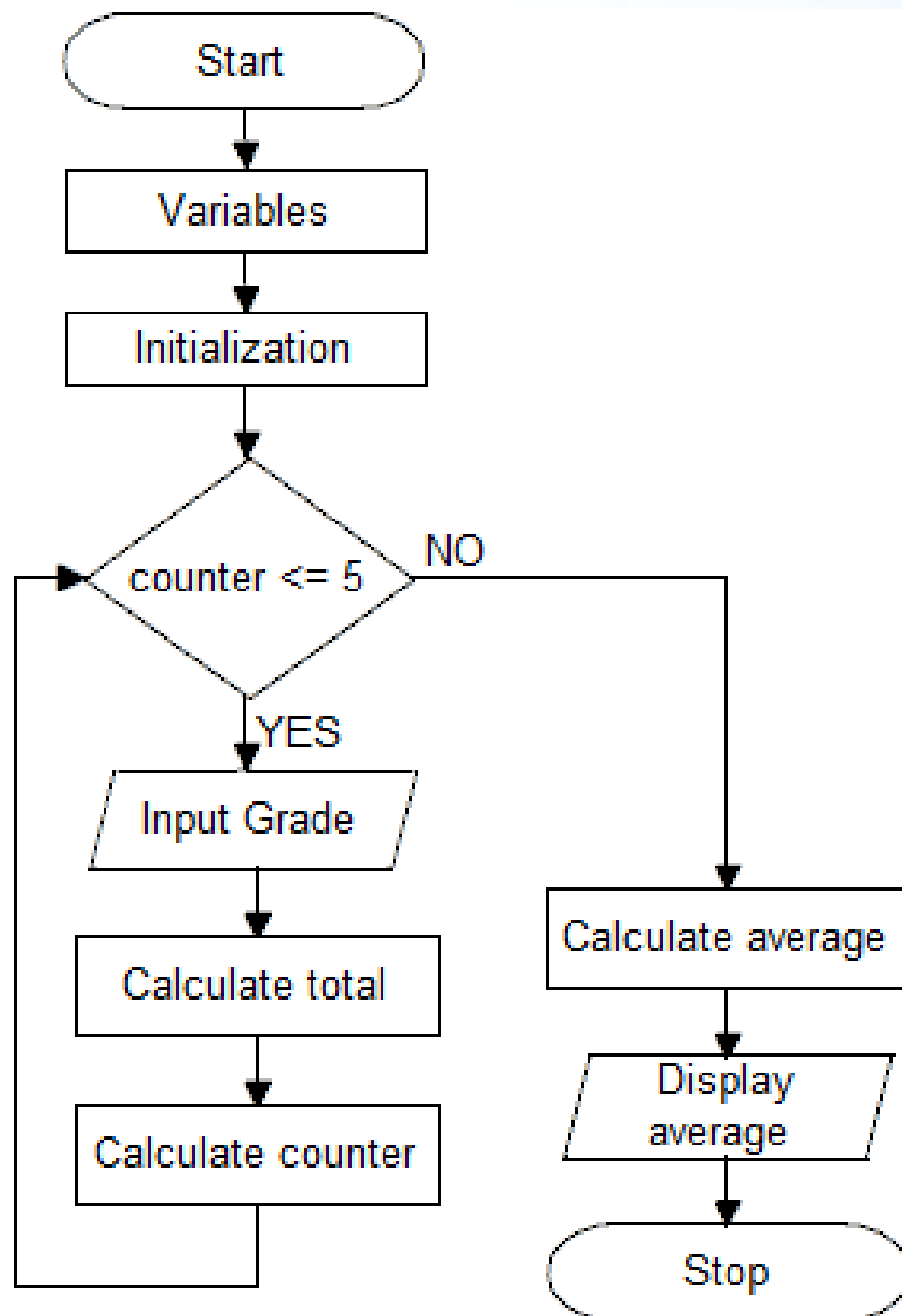
Do While condition is true
    Process step(s)
Loop
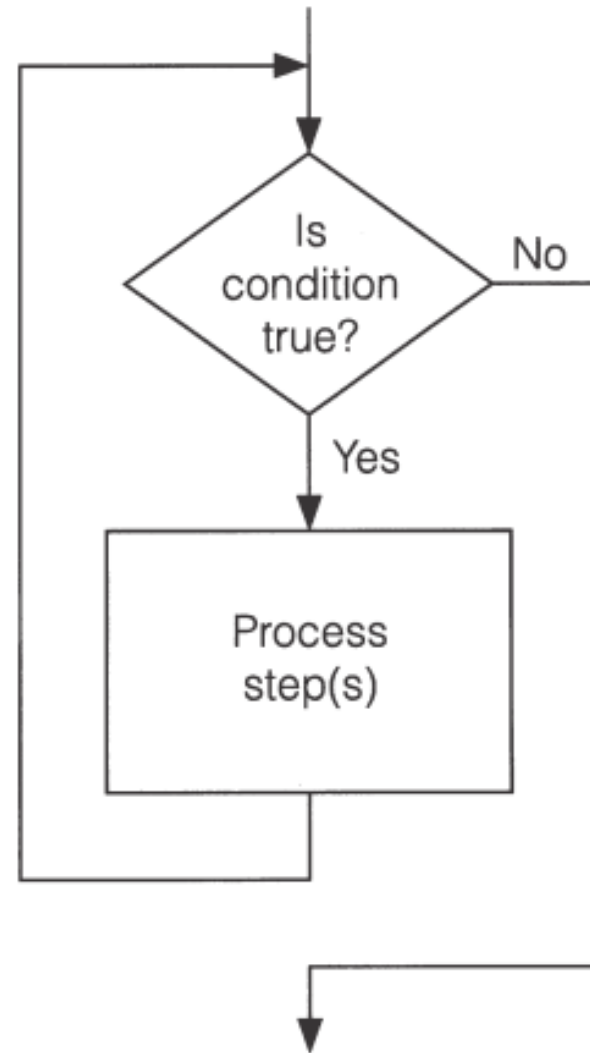
Example - 4

- Write an algorithm which calculates the average exam grade for a class of 5 students.
- What are the program inputs?
  - the exam grades
- Processing:
  - Find the sum of the grades;
  - count the number of students; (counter controlled)
  - calculate average grade = sum of grades / number of students.
- What is the program output?
  - the average exam grade

Pseudocode

- Start
- Use variables total, counter, grade, average
- Initialize total = 0
- Initialize counter = 1
- While (counter <= 5)
  - Input grade
  - Calculate total = total + grade
  - Calculate counter = counter + 1
- End-while
- Calculate average = total / 5
- Display average
- Stop

Example - 5

➢ Write an algorithm which calculates the average exam grade for a class of unknown number of students.
➢ What are the program inputs?
  ▪ the exam grades
➢ Processing:
  ▪ Find the sum of the grades till sentinel value is given; for example -99 to break loop (sentinel controlled)
  ▪ calculate average grade = sum of grades / number of students.
➢ What is the program output?
  ▪ the average exam grade

Pseudocode

- Start
- Use variables total, counter, grade, average
- Initialize total = 0
- Initialize counter = 0
- While (grade != -99)
  - Input grade
  - Calculate total = total + grade
  - Calculate counter = counter + 1
- End-while
- Calculate average = total / counter
- Display average
- Stop

Example - 6

- Write an algorithm which calculates the average exam grade for a class of unknown number of students.

- This time, the number of students have been asked at the beginning of the program.

- Use counter controlled structure.

Pseudocode

- Start
- Use variables total, counter, grade, average, number_of_students
- Initialize total = 0 , number_of_students = 0 , counter = 1
- Display "write number of students"
- Input number_of_students
- While (counter <= number_of_students)
  - Input grade
  - Calculate total = total + grade
  - Calculate counter = counter + 1
- End-while
- Calculate average = total / number_of_students
- Display average
- Stop

Question

➢ Draw a flowchart for example – 6.

Fatal Error – Memorizing

- Do not memorize any of the codes in programming.
- Read and try to understand what is given and what is asked in the question, then write your own codes.