Algorithms
Pseudocode
Flowcharts

Instructor Özgür ZEYDAN (PhD)

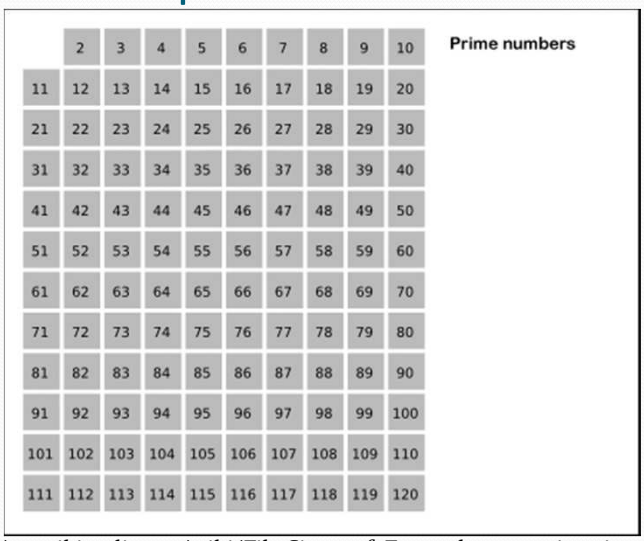
CIV 112 – Computer Programming
http://cevre.beun.edu.tr/zeydan/

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

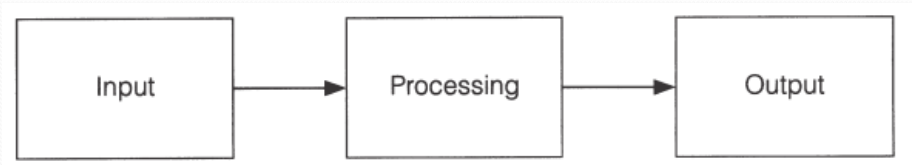


http://en.wikipedia.org/wiki/File:Sieve_of_Eratosthenes_animation.gif

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

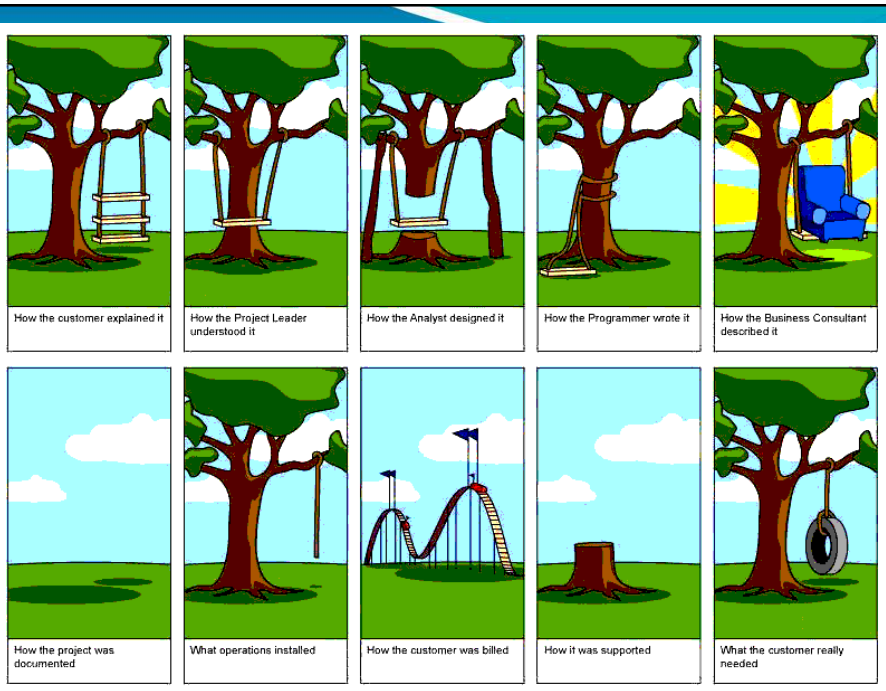
3. Choose the interface: Select the objects.
   - Determine how the input will be obtained and how the output will be displayed. Then create appropriate commands to allow the user to control the program.

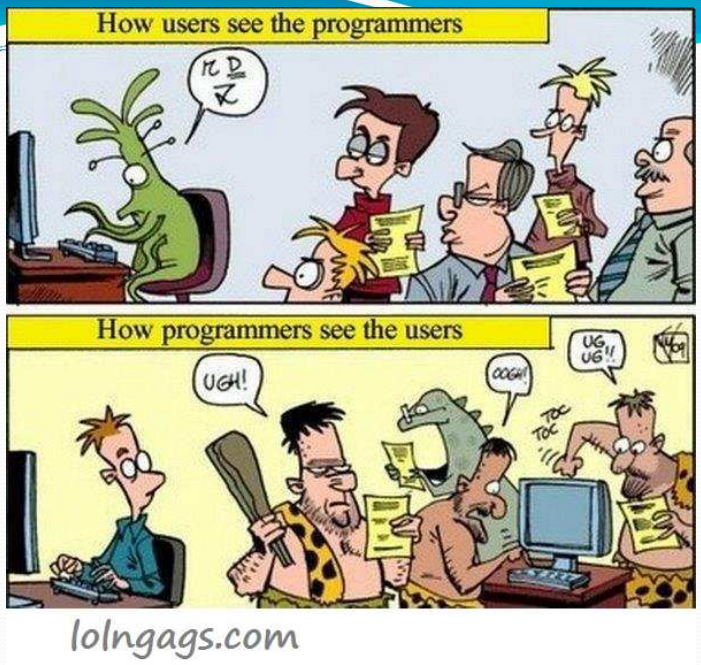
4. Code: Translate the algorithm into a programming language.
   - Coding is the technical word for writing the program.

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


How users see the programmers
How programmers see the users
UGH!
OOGH!
UG, UG!!
TOC TOC
lolngags.com

Programming Tools

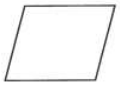
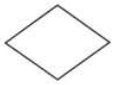
- Tools used to convert algorithms into computer programs:
  - Pseudocode: An informal high-level description of the operating principle of a computer program. It uses the structural conventions of a programming language, but is intended for human reading rather than machine reading.
  - Flowcharts: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

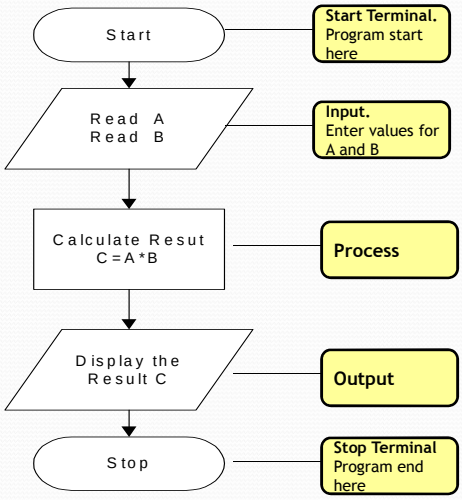
Flowchart Symbols

Symbol,	Name,	Meaning
	Flowline	Used to connect symbols and indicate the flow of logic.
	*Terminal*	Used to represent the beginning (Start) or the end (End) of a task.
	*Input/Output*	Used for input and output operations, such as reading and printing. The data to be read or printed are described inside.
	*Processing*	Used for arithmetic and data-manipulation operations. The instructions are listed inside the symbol.
	*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	Start - Terminal
Read A, B	Read A, B – Input
Calculate C = A*B	Calculate C = A*B - Action
Display C	Display C - Output
Stop	Stop - Terminal

Example Flowchart



Instructor Özgür ZEYDAN

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

Instructor Özgür ZEYDAN

Question ???

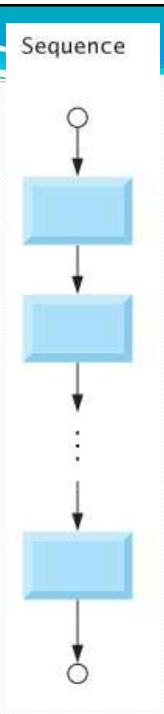
- Write an algorithm to calculate Fahrenhayt value of temperature if celcius valeu is given.
- $\frac{(F-32)}{180} = \frac{(C)}{100}$
- Write a pseudocode.
- Draw a flowchart.

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.

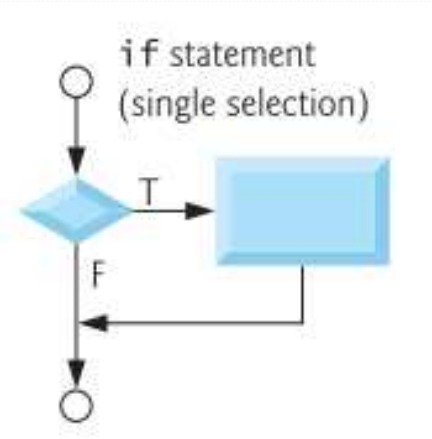
Sequences



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

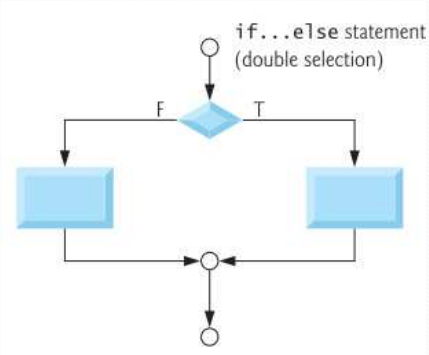
Decisions (selections)

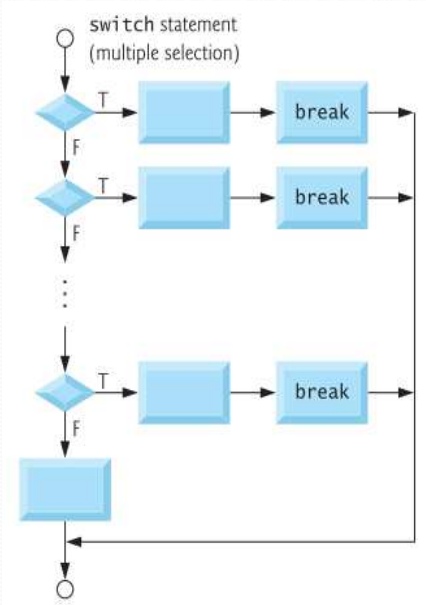
- Three selection structure in C programming:
  - If
  - If – else
  - Switch



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

Decisions (selections)

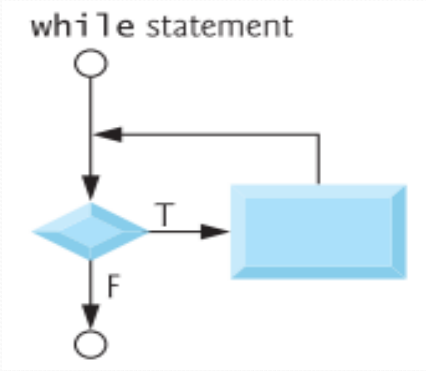




Ref: Deitel P J (Ed.) (2010) C How to Program, 6th Edition, Prentice Hall

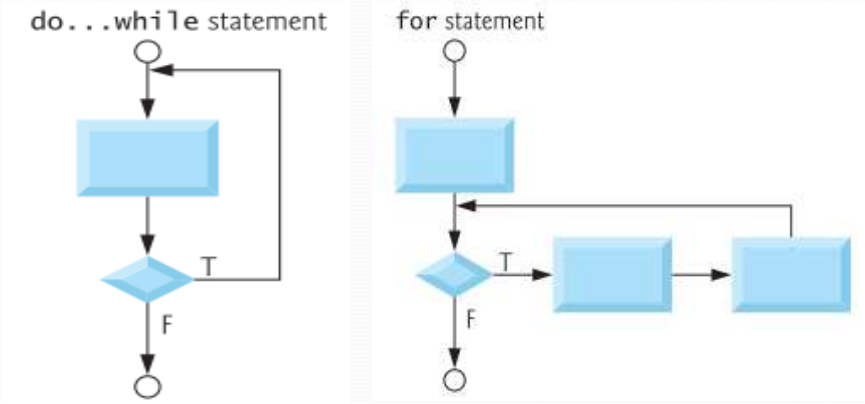
Loops (repetition)

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



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

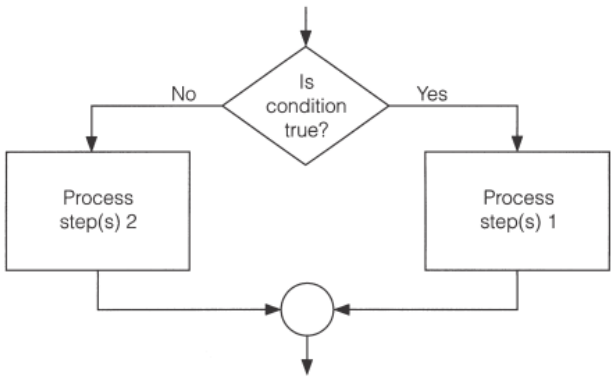
Loops (repetition)



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"

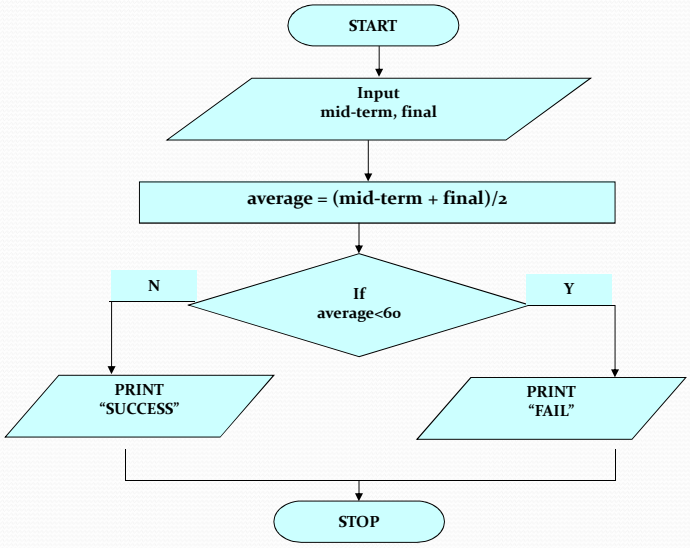
  else

  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



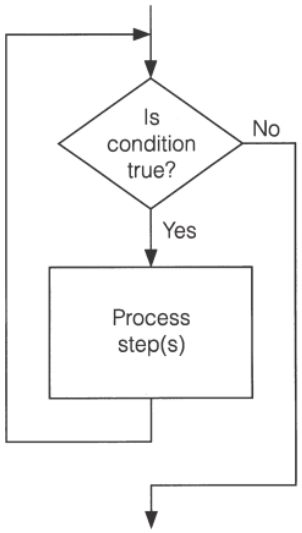
Nested If

- 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
```



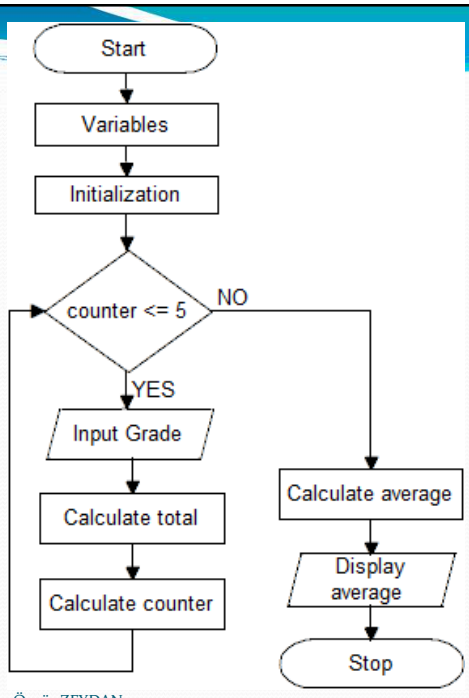
Example - 4

- 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

Flowchart



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.

Fatal Error – Memorizing

- If (memorize) then Display "sure that you will fail this course!!!"