


UEH UNIVERSITY

College of
Technology
and Design



Fundamentals of Programming

By: Võ Văn Hải
Email: vovanhai@ueh.edu.vn

1

Course Introduction

- ▶ Syllabus
- ▶ Course objectives
 - The course provides basic knowledge of programming techniques as one of the fundamental knowledge in the field of Management Information Systems. In addition, the course presents basic concepts of algorithms and how to develop algorithms. The course also provides knowledge of the C# programming language, such as data types; declaration of variables, constants, and expressions; assignment commands; data input and output; branching and loop control structures; 1- and 2-dimensional arrays; functions/methods; string types; files,... so that students can apply and install algorithms from basic to advanced levels.
- ▶ Learning Outcomes
 - Knowledge
 - CLO1.1: Understand the basic concepts of structured programming.
 - CLO1.2: Understand how to build and implement algorithms.
 - CLO1.3: Master knowledge of structured programming languages and their components.
 - Skill
 - CLO2.1: Analyze problems and build algorithms.
 - CLO2.2: Proficient in using compilers.
 - CLO2.3: Proficient in using structures and data types.
 - CLO2.4: Proficient in using functions and files.
 - CLO2.5: Build computer programs.
 - Autonomy and Responsibility
 - CLO3.1: Can analyze and build programs to solve problems.
 - CLO3.2: Be responsible and proactive in doing group exercises and homework.

2

Course Structure

- ▶ Textbook: *Mark Michaelis, Essential C# 12.0. 8 Ed, Addison-Wesley Publisher, 2024.*
- ▶ Reference: *Joseph Albahari, C# 12 in a Nutshell: The Definitive Reference 1st Edition, O'Reilly Media Publisher, 2024*
- ▶ Content:
 - Introduction to Computer Programming
 - Structure Programming Language Fundamentals (SPL)
 - Function in SPL
 - Arrays and Strings
 - File Handling
 - Advance topics

3

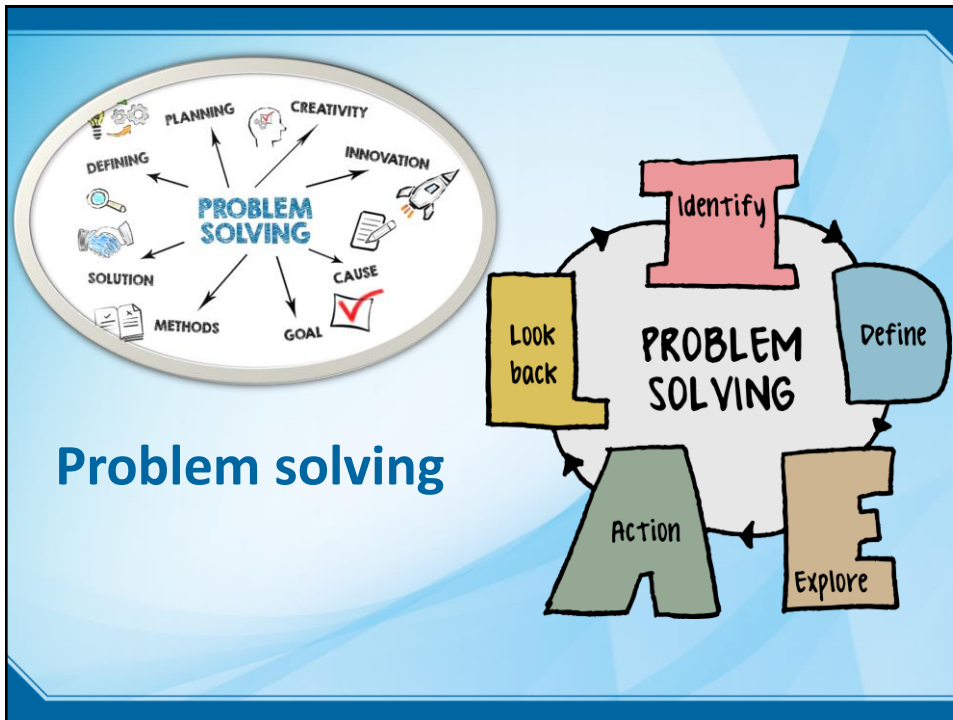
3

Assessment

Criteria	Percentage
Attendance:	10%
Discuss:	0%
Harvest:	0%
Give a talk:	10%
Report:	0%
Mid-term exams:	30%
Exam at the end of the module:	50%

4

4



5

What is a computer problem?

- ▶ A matter can be defined as a problem in computer science if it has two of the following features:
 - a) The problem is general enough to appear repeatedly in slightly different forms or instances; and
 - b) You can tell whether a potential answer is right or wrong.
- ▶ Before trying to find a process to solve a computer problem,
 - Make sure you understand exactly the answers that will be accepted.
 - Not every problem has these two features.
 - Problems that lack one or the other are generally outside the scope of computer science.



6

Computer problem definition

- ▶ A problem is generally considered to be a task, a situation, or person which is difficult to deal with or control due to complexity and intransparency. [1]
 - In everyday language, a problem is a question proposed for solution, a matter stated for examination or proof.
 - In each case, a problem is considered to be a matter which is difficult to solve or settle, a doubtful case, or a complex task involving doubt and uncertainty.
- ▶ A distinction can be made between “task” and “problem.”
 - Generally, a task is a well-defined piece of work that is usually imposed by another person and may be burdensome.

[1] Seel, N.M. (2012). Problems: Definition, Types, and Evidence. In: Seel, N.M. (eds) Encyclopedia of the Sciences of Learning. Springer, Boston, MA, pp 2690–2693.
https://doi.org/10.1007/978-1-4419-1428-6_914

7

7

Problem Solving

- ▶ Workaround
 - Initiated by Rene Descartes (1596-1650).
 - Observation: the problem solver must go about things in the right way and must use the **right method** to arrive at a solution.
 - Right method \approx algorithm
 - The term of **algorithm** was introduced by Abu Jafar Muhammad bin Musa al-Khwarizmi in AD 820
- ▶ Definition 1:
 - Problem solving is a process by which a situation is analyzed, and solutions are formed to solve a problem with identified steps to resolve, eliminate or mitigate the problem.
- ▶ Definition 2:
 - Problem solving is the process of constructing and applying mental representations of problems to finding solutions to those problems that are encountered in nearly every context. [1]

[1] Jonassen, D.H., Hung, W. (2012). Problem Solving. In: Seel, N.M. (eds) Encyclopedia of the Sciences of Learning. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-1428-6_208, pp 2680–2683

8

8

Problem Solving

Process



Fig 1. Problem solving process from the initial situation to the final goal

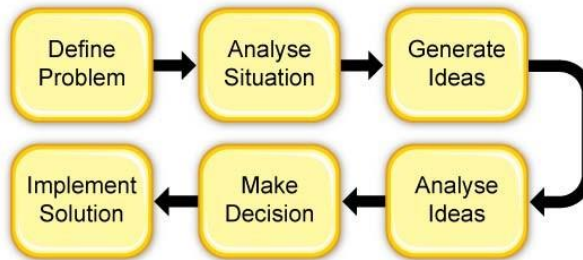


Fig 2. The general problem-solving process

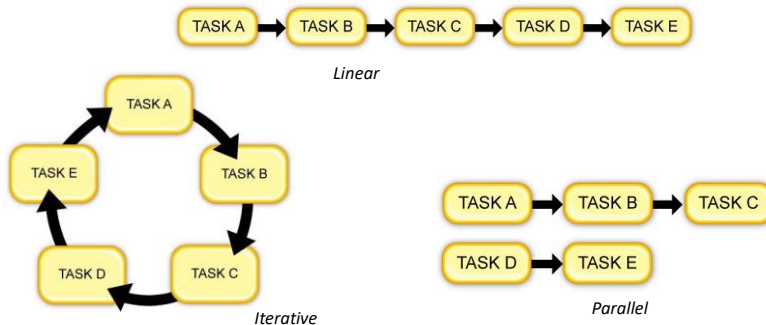
9

Problem Solving

Strategies

- There are four basic modelling structures used in solving problems:

- Linear
- Iterative
- Parallel
- Dynamic (based on interactions and interrelations between tasks)



10

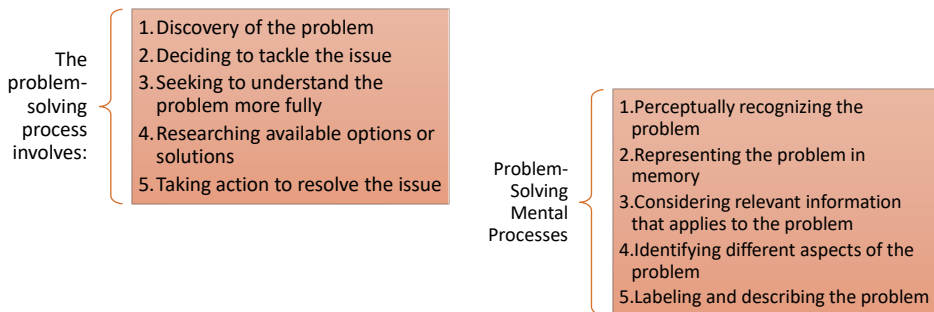
Problem Solving

Real World Problem-Solving (bonus)

► Definition 4: (Real World Problem-Solving)

- In cognitive psychology, the term 'problem-solving' refers to the mental process that people go through to discover, analyze, and solve problems. [1]

[1] Sarathy V. Real world problem-solving. *Front Hum Neurosci.* 2018;12:261. doi:10.3389/fnhum.2018.00261 (<https://www.frontiersin.org/articles/10.3389/fnhum.2018.00261/full>)



11

Problem Solving

Real World Problem-Solving (bonus) - Strategies

► Algorithms

- An algorithm is a step-by-step procedure that, by following certain "rules" produces a solution.

► Heuristics

- Heuristics are shortcut strategies that people can use to solve a problem at hand. These "rule of thumb" approaches allow you to simplify complex problems, reducing the total number of possible solutions to a more manageable set.

► Trial and Error


- A trial-and-error approach to problem-solving involves trying a number of potential solutions to a particular issue, then ruling out those that do not work.

► Insight

- In some cases, the solution to a problem can appear as a sudden insight. Insight can occur when the problem in front of you is similar to an issue that you've dealt with in the past.

12

Funny




Problem Solving Flowchart

```
graph TD
    Q1{DOES IT WORK?} -- YES --> Q2{DOES ANYONE KNOW?}
    Q1 -- NO --> Q3{DID YOU MESS WITH IT?}
    Q2 -- YES --> Q4{YOU POOR IDIOT}
    Q2 -- NO --> H1[NO PROBLEM]
    Q3 -- YES --> Q4
    Q3 -- NO --> Q5{WILL YOU GET INTO TROUBLE?}
    Q4 --> Q6{CAN YOU BLAME SOMEONE ELSE?}
    Q5 -- YES --> Q4
    Q5 -- NO --> T1[TRASH IT]
    Q6 -- YES --> H1
    Q6 -- NO --> Q7{MESS WITH IT?}
    Q7 -- YES --> Q8{YOU IDIOT!!!}
    Q7 -- NO --> H1
    Q8 --> Q2
```

13

13



```
while (again) {
    iN = -1;
    again = false;
    getline(cin, sInput);
    system("cls");
    stringstream(sInput) >> dblTemp;
    ilength = sInput.length();
    if (ilength < 4) {
        again = true;
        continue;
    } else if (sInput[0] == '0') {
        again = true;
        continue;
    }
    while (++iN < ilength) {
        if (isdigit(sInput[iN])) {
            continue;
        }
        if (iN == ilength - 1) {
            break;
        }
    }
}
```

Algorithms

Design

Experiment

Implement

Analyze

Problem Definition

Problem Analysis

Algorithm Design

Pseudocode/Flowchart

Implementation

Testing

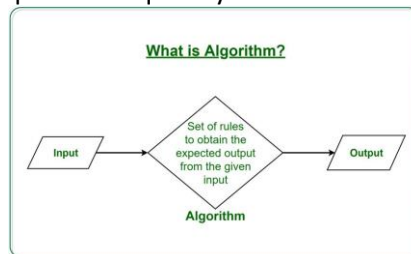
14

14

Algorithm

Definition

- ▶ **Definition 1:** An algorithm can be defined as "an *ordered* set of unambiguous, executable steps that defines a terminating process" (Brookshear, 2006).
- ▶ **Definition 2:** "A set of finite rules or instructions to be followed in calculations or other problem-solving operations"
- ▶ **Definition 3:** "A procedure for solving a mathematical problem in a finite number of steps that frequently involves recursive operations".



15

15

Algorithm

Usage

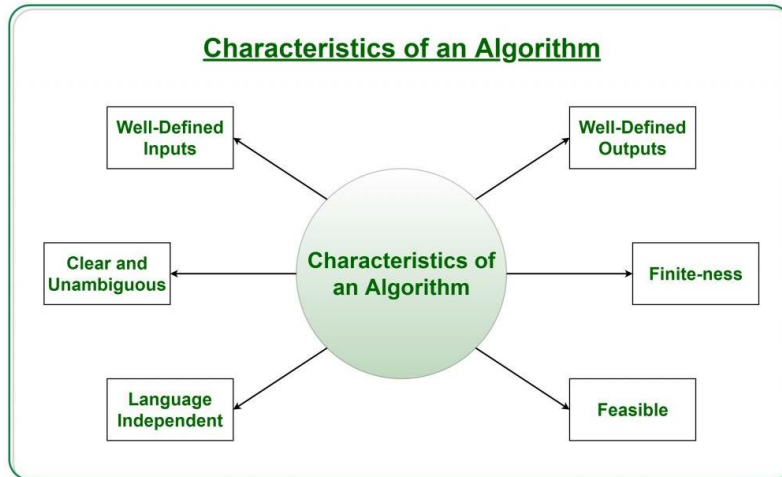
- ▶ Algorithms are necessary for solving complex problems efficiently and effectively.
- ▶ They help to automate processes and make them more reliable, faster, and easier to perform.
- ▶ Algorithms also enable computers to perform tasks that would be difficult or impossible for humans to do manually.
- ▶ They are used in various fields such as mathematics, computer science, engineering, finance, and many others to optimize processes, analyze data, make predictions, and provide solutions to problems.

16

16

Algorithm

Characteristics



17

17

Algorithm

Characteristics

- ▶ **Clear and Unambiguous:** The algorithm should be clear and unambiguous. Each of its steps should be clear in all aspects and must lead to only one meaning.
- ▶ **Well-Defined Inputs:** If an algorithm says to take inputs, it should be well-defined inputs. It may or may not take input.
- ▶ **Well-Defined Outputs:** The algorithm must clearly define what output will be yielded and it should be well-defined as well. It should produce at least 1 output.
- ▶ **Feasible:** The algorithm must be simple, generic, and practical, such that it can be executed with the available resources. It must not contain some future technology or anything.
- ▶ **Language Independent:** The Algorithm designed must be language-independent, i.e. it must be just plain instructions that can be implemented in any language, and yet the output will be the same, as expected.
- ▶ **Finiteness:** An algorithm must terminate after a finite number of steps in all test cases. Every instruction which contains a fundamental operator must be terminated within a finite amount of time. Infinite loops or recursive functions without base conditions do not possess finiteness.

18

18

Algorithm

Other characteristics

- ▶ **Input:** An algorithm has zero or more inputs. Each that contains a fundamental operator must accept zero or more inputs.
- ▶ **Output:** An algorithm produces at least one output. Every instruction that contains a fundamental operator must accept zero or more inputs.
- ▶ **Definiteness:** All instructions in an algorithm must be unambiguous, precise, and easy to interpret. By referring to any of the instructions in an algorithm one can clearly understand what is to be done. Every fundamental operator in instruction must be defined without any ambiguity.
- ▶ **Effectiveness:** An algorithm must be developed by using very basic, simple, and feasible operations so that one can trace it out by using just paper and pencil.

19

19

Algorithm

Types of Algorithms

- ▶ Brute Force Algorithm
- ▶ Recursive Algorithm
- ▶ Backtracking Algorithm
- ▶ Searching Algorithm
- ▶ Sorting Algorithm
- ▶ Hashing Algorithm
- ▶ Divide and Conquer Algorithm
- ▶ Greedy Algorithm
- ▶ Dynamic Programming Algorithm
- ▶ Randomized Algorithm
- ▶ ...

20

20

Algorithm

Pros/cons

► Advantages of Algorithms:

- It is easy to understand.
- An algorithm is a step-wise representation of a solution to a given problem.
- In Algorithm the problem is broken down into smaller pieces or steps hence, it is easier for the programmer to convert it into an actual program.

► Disadvantages of Algorithms:

- Writing an algorithm takes a long time so it is time-consuming.
- Understanding complex logic through algorithms can be very difficult.
- Branching and Looping statements are difficult to show in Algorithms.



21

21

Algorithm

Strategy

1. Pre-requisite:

1. The **problem** that is to be solved by this algorithm i.e. clear problem definition.
2. The **constraints** of the problem must be considered while solving the problem.
3. The **input** to be taken to solve the problem.
4. The **output** to be expected when the problem is solved.
5. The **solution** to this problem, is within the given constraints.

2. Designing the algorithm

- Pseudo code (/steps) or,
- Flow chart

3. Testing the algorithm

- Implementing then testing
- Analyzing/check
 - Priori analysis
 - Posterior

22

22

Algorithm

Design

Example: Solve the equation $ax + b = 0$

► Pre-requisite

- Constraints $a, b \in R$
- Input: the values of a, b (e.g., $\{a=2.0, b=4.0\}$)
- Output: the value of x

► Pseudo-code

- If $a = 0$ the
 - If $b = 0$ then x is any values
 - If $b \neq 0$ the equation has no solution
- If $a \neq 0$ then
 - The equation has only one solution $x = -b/a$

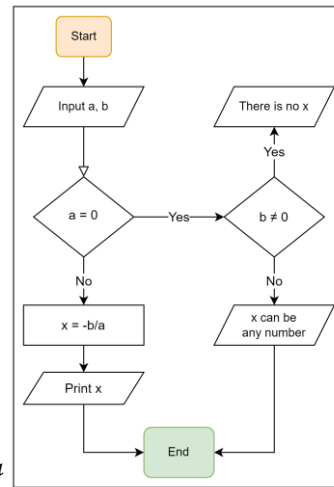


Fig. 1: Flowchart

23

23

Algorithm

Implement

```

#include <stdio.h>

void solve_equation(float a, float b) {
    if (a == 0)
        if (b == 0)
            printf("%s", "There is no x.");
        else
            printf("%s", "X can be any number");
    else {
        float x = -b / a;
        printf("x = %.3f", x);
    }
}

```

```

int main() {
    float a,b;
    printf("Enter a number a=");
    scanf("%f",&a);
    printf("Enter a number:");
    scanf("%f",&b);
    solve_equation(a, b);
    return 0;
}

```

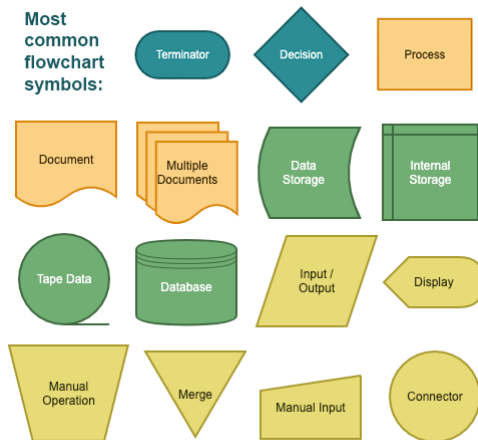
24

24

Flowchart

Introduction

Most common flowchart symbols:



<https://www.gliffy.com/blog/guide-to-flowchart-symbols>

► Flowchart is a diagram of the sequence of movements or actions of people or things involved in a complex system or activity.

► OR: A flowchart is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams.

25

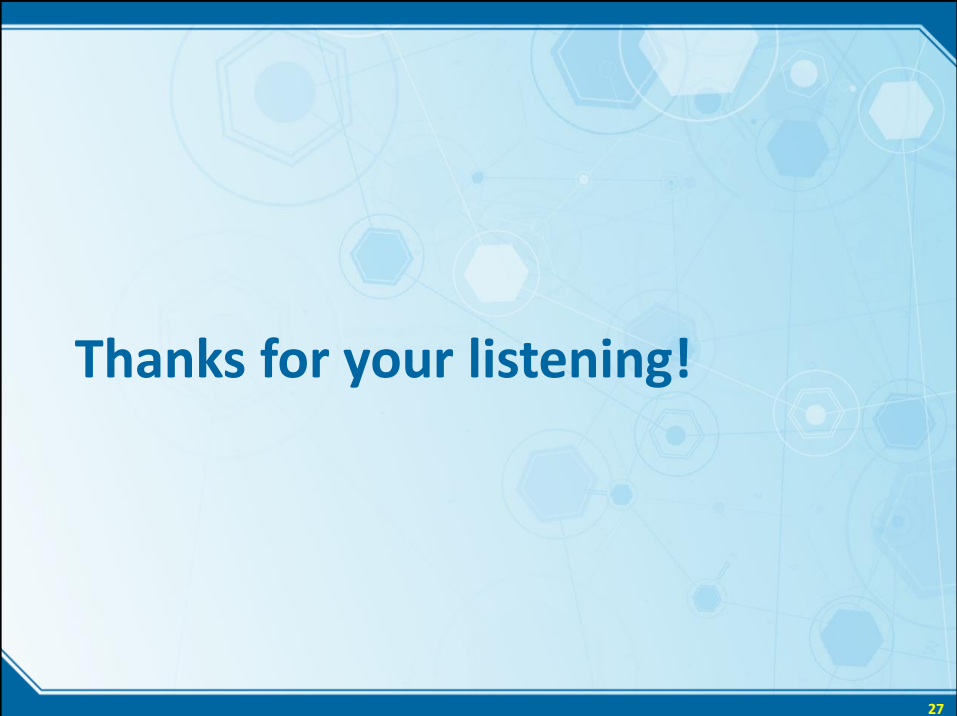
25

Developing Environment

- The Compiler
- The Integrated Development Environments (IDE) :
 - Visual Studio 2022
 - Visual Code
 - Sharp Developer
- Version control: Git ([GitHub](https://github.com)/[GitLab](https://gitlab.com)/[Bitbucket](https://bitbucket.org))
- Unit test

26

26



Thanks for your listening!

27