Ho Chi Minh City University of Technology
Faculty of Computer Science and Engineering

Chapter 1: Introduction to Computers and Programming

Introduction to Computer Programming
(C language)

Nguyễn Tiến Thịnh, Ph.D.

Email: ntthinh@hcmut.edu.vn

2022 – 2023, Semester 1

Course Content

- **C.1. Introduction to Computers and Programming**
- C.2. C Program Structure and its Components
- C.3. Variables and Basic Data Types
- C.4. Selection Statements
- C.5. Repetition Statements
- C.6. Functions
- C.7. Arrays
- C.8. Pointers
- C.9. File Processing

References

- [1] "*C: How to Program*", 7th Ed. – Paul Deitel and Harvey Deitel, Prentice Hall, 2012.
- [2] "*The C Programming Language*", 2nd Ed. – Brian W. Kernighan and Dennis M. Ritchie, Prentice Hall, 1988
- and others, especially those on the Internet

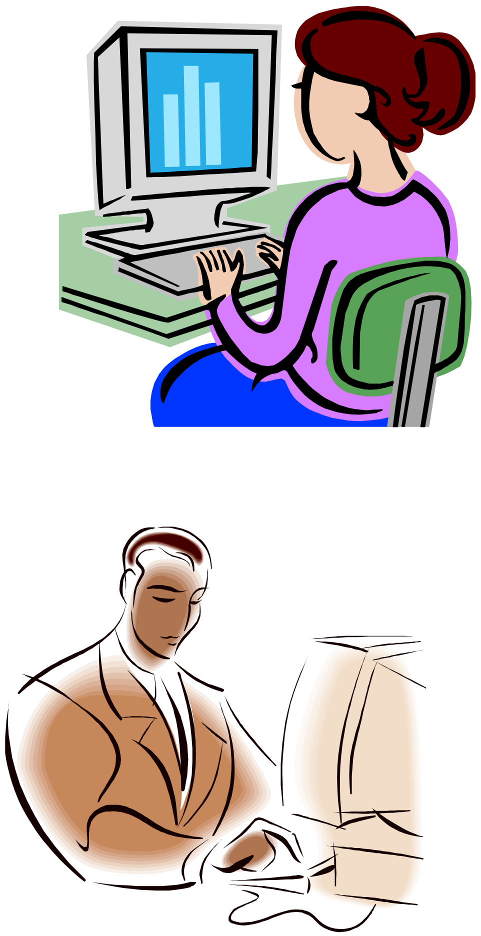
Content

Introduction

- Computer Programming
  - Computer
    - a device that can perform computations and make logical decisions billions of times faster than human beings can
  - Programming
    - The act of writing the programs executable on the computers to produce intended results
  - Program
    - A sequence of instructions written in a programming language to perform a specified task by the computer

Introduction

Computers

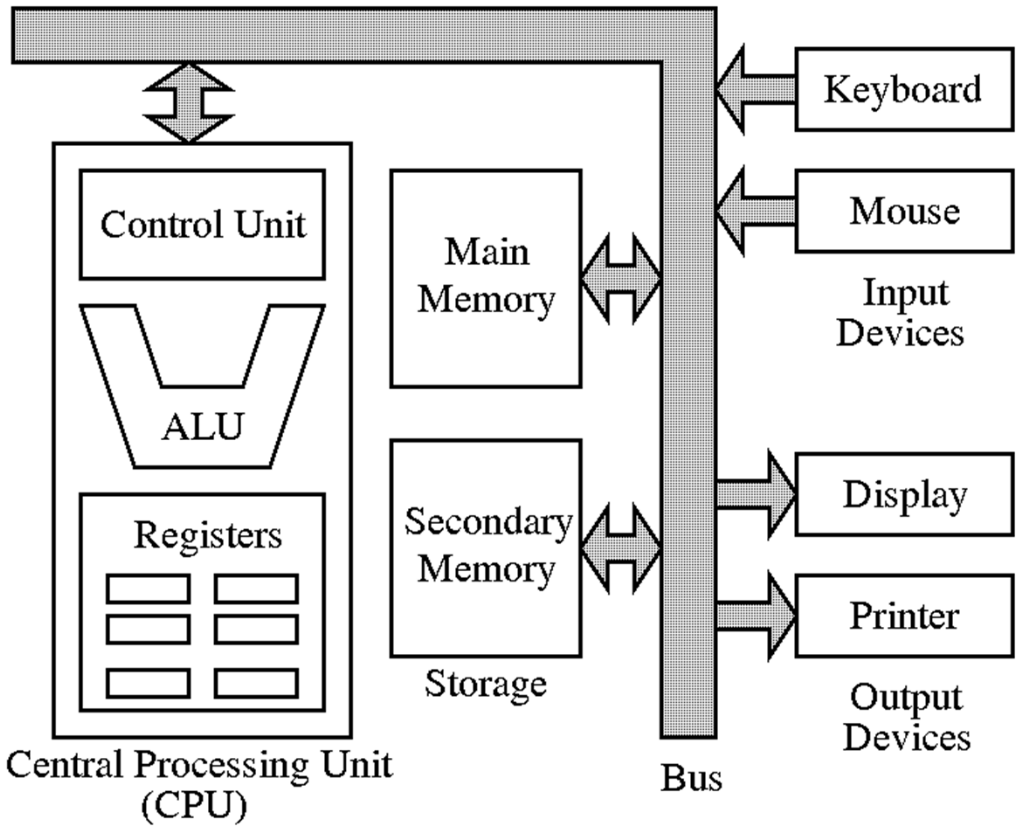

Programming

Programs and their Results

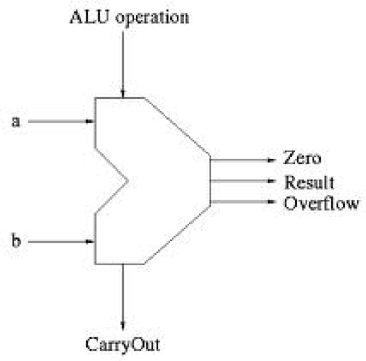
Computer Organization

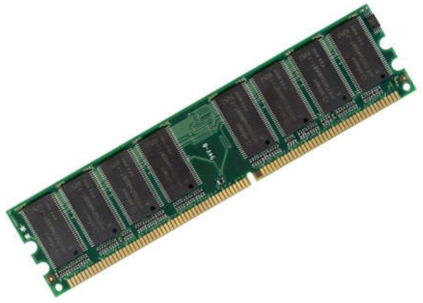
- Hardware: physical components of computer (including peripherals)
  - the keyboard, screen, mouse, hard disk, memory, DVDs and processing units, …
- Software: a set of machine-readable instructions that directs a computer's processor to perform specific operations [Wikipedia]
  - Application softwares
  - Operating system
  - System softwares

Computer Organization - Hardware



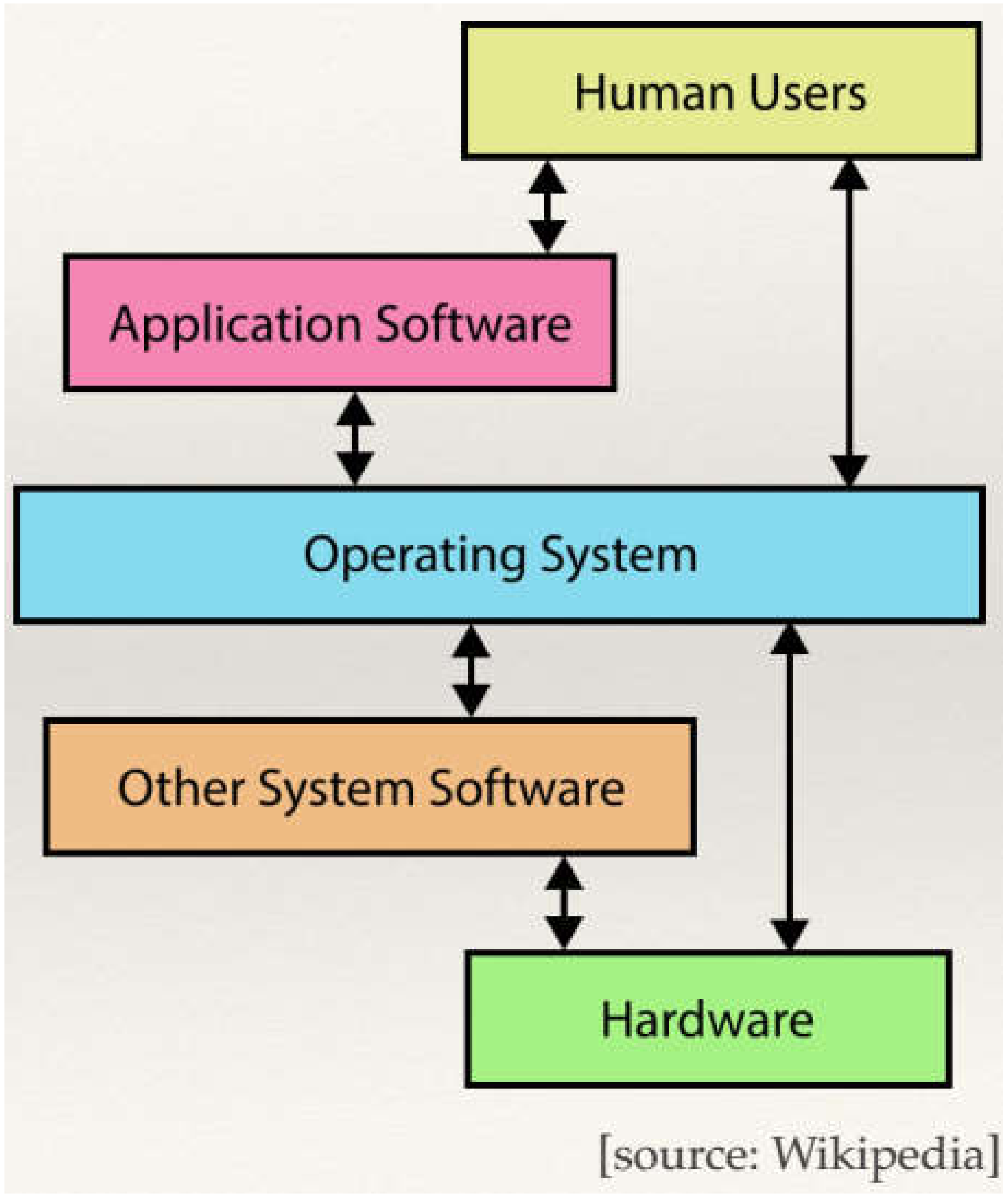
Computer Architecture





ALU = Arithmetic/logic gate unit: performing arithmetic and logic operations on data

Computer Organization – Software



Programming Languages

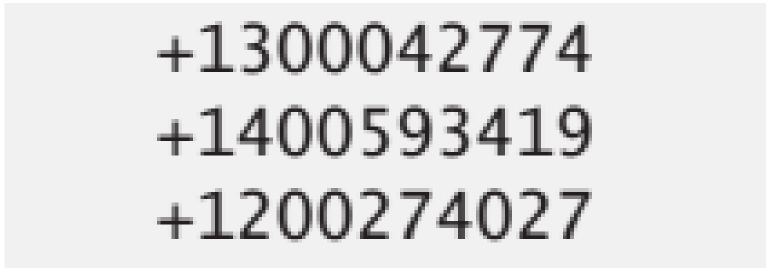
- Programming language: a formal language for writing a computer program as a sequence of instructions
  - C, C++, C#, Java, PHP, Python, …
- Three general types
  - Machine languages
  - Assembly languages
  - High-level languages

  → Providing a sequence of instructions that directly understandable by computers or requiring some intermediate translation steps

Programming Languages – Machine Languages

- First-generation language: strings of numbers (ultimately reduced to 1s and 0s) that instruct computers to perform their most elementary operations one at a time
  - Directly understandable by computers
  - Machine-dependent

For example, instructions for adding overtime pay to base pay and then storing the result in gross pay

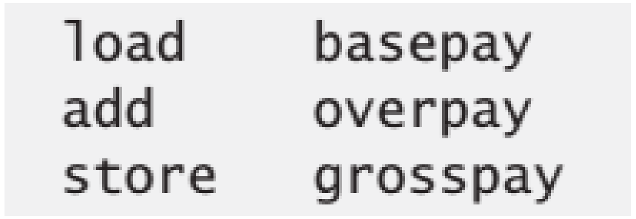


```
+1300042774
+1400593419
+1200274027
```

Programming Languages – Assembly Languages

- Second-generation language: a low-level language used to interface with computer hardware using English-like abbreviations to represent elementary operations
  - Less understandable by computers
  - Need for translation steps to convert an assembly language program to machine codes
    - Translator = Assembler

For example, instructions for adding overtime pay to base pay and then storing the result in gross pay



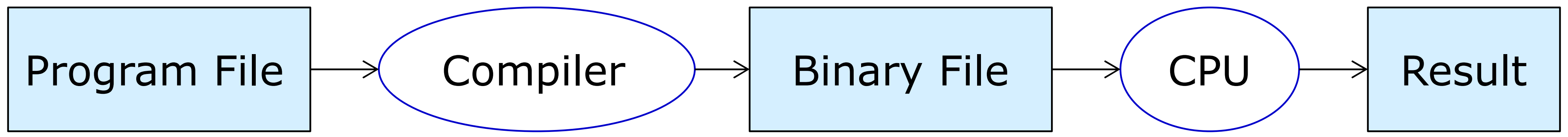
```
load      basepay
add       overpay
store     grosspay
```

Programming Languages – High-level Languages

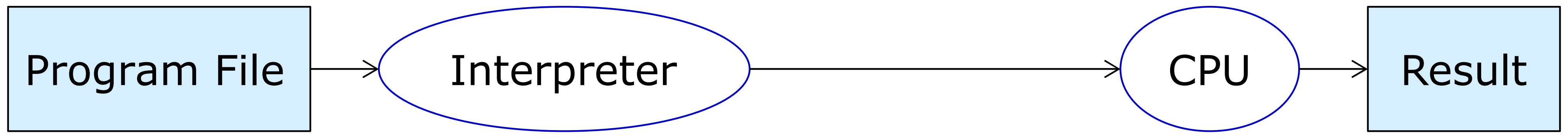
- Third-generation language: written instructions that look almost like everyday English and contain commonly used mathematical notations
  - Less understandable by computers
  - Translator program is called compiler.
  - The C language is a high-level language that needs a compiler.
  - Scripting languages such as PHP and Perl need an interpreter.

For example, instructions for adding overtime pay to base pay and then storing the result in gross pay: grosspay = basepay + overpay.

Programming Languages – High-level Languages



C, C++, Java, …



PHP, Perl, …

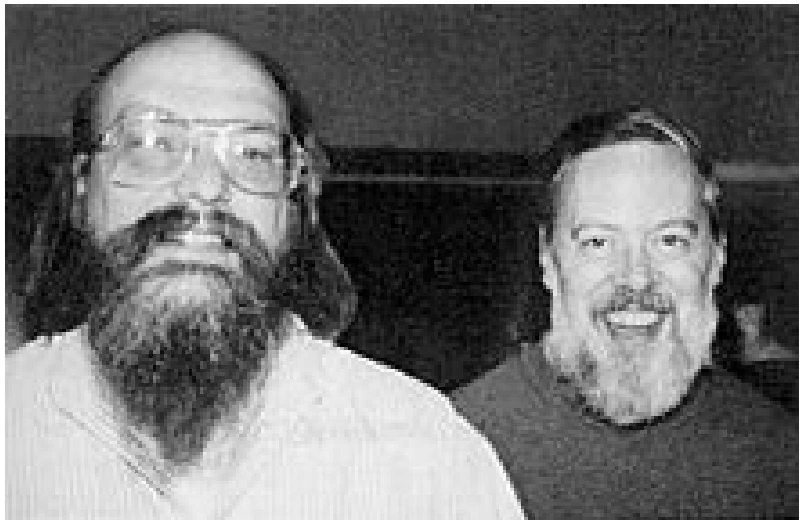
A history of computer programming languages – Wikipedia

Graph of programming language history – www.levenez

Programming Languages – The C language

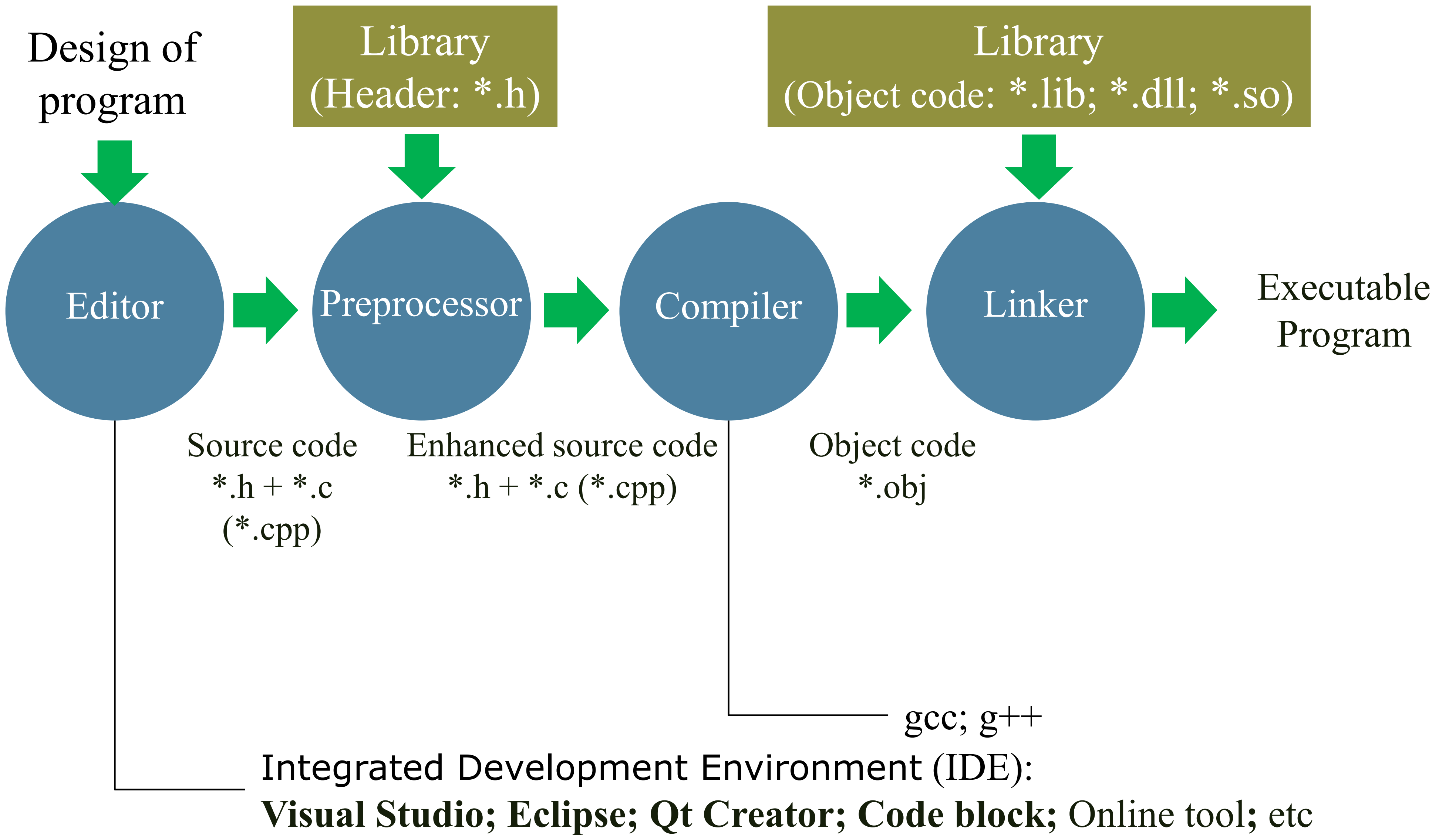
- Evolved from B by Dennis Ritchie at Bell Laboratories and originally implemented on a DEC PDP-11 computer in 1972
- Using many of the important concepts of BCPL and B while adding data typing and other powerful features
- Used for many important application trends
  - Developing new major operating systems: UNIX, Linux, Android, …
  - Developing programs in the embedded systems in cars, medical machines, …

Programming Languages – The C language


Ken Thompson (left) with Dennis Ritchie (right, the inventor of the C programming language) [Wikipedia]

- The development of the C language
  - Dennis M. Ritchie
- Full history of the C language
  - Wikipedia

Programming Tasks



Integrated Development Environment (IDE):
Visual Studio; Eclipse; Qt Creator; Code block; Online tool; etc

Programming Tasks

- Editor: supports text editing feature for writing source code
- Preprocessor: preprocesses the source code with replacing macro, inserting library files *.h, …
- Compiler: translates the source code into target machine language
- Linker: links the object code to other library files

Data and Algorithms – Concepts

- Program

= A Sequence of Instructions Written in a Programming Language to Perform a Specified Task by the Computer

$$= \underbrace{\text{Data and their Structures}}_{\text{Input/Output/…}} + \underbrace{\text{Algorithms}}_{\text{Process}}$$

Example 1: instructions for adding overtime pay to base pay and then storing the result in gross pay: grosspay = basepay + overpay.

Example 2: given n positive numbers, find the smallest one.

Data and Algorithms – Data

- Atomic data: int, double, char, ...
- Non-atomic data: array, struct, enum, …
- A strong relationship between the data structures and the operations on the data in the corresponding structures

Example 1: instructions for adding overtime pay to base pay and then storing the result in gross pay: grosspay = basepay + overpay.
- Input Data: basepay and overpay are positive real numbers (double).
- Output Data: grosspay is also a positive real number (double).

Example 2: given n positive numbers, find the smallest one.
- Input Data: n positive real numbers are treated individually OR as a collection (double)
- Output Data: minNumber is a positive real number (double).

Data and Algorithms – Algorithms

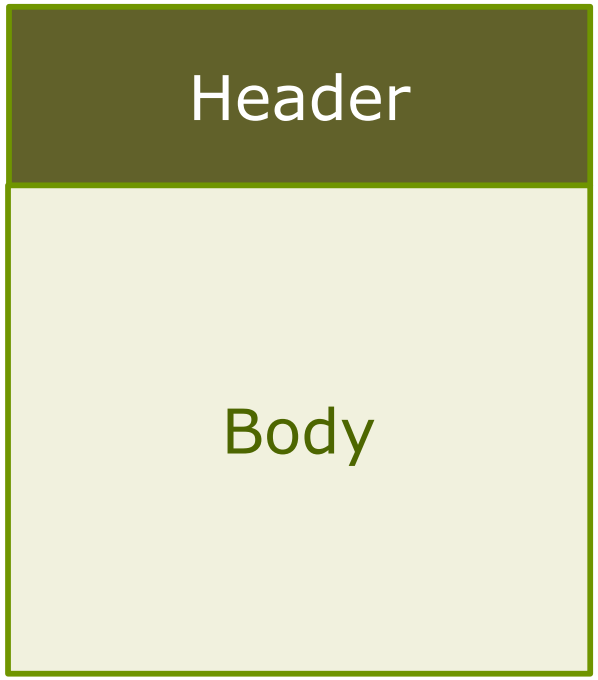
- Algorithm = a sequence of unambiguous instructions for solving a problem, i.e. for obtaining a required output for any legitimate input in a finite amount of time
  - Anany Levitin, *Introduction to the Design and Analysis of Algorithms*, 2nd Edition, Addison Wesley, 2007
- Algorithm representation
  - Pseudo code
  - Flowchart
  - Real code in a high-level programming language

Data and Algorithms – Algorithms

- Example 2: given n positive numbers, find the smallest one.
- Task solution:
  - 1. Suppose that the first number is the smallest one (current one).
  - 2. Check if the current smallest one is a real one as compared to the next number.
    - If yes then compared to the next number of the next one like step 2 till all numbers are checked.
    - Otherwise,
      - update the smallest one with the smaller one
      - And then move next to check with the next number of the next number like step 2 till all numbers are checked.

Data and Algorithms – Algorithms – Pseudo Code

- Header
  - Algorithm name
  - Input data and their data types
  - Task purpose
  - Pre-conditions
  - Post-conditions
  - Output data and their data types
- Body
  - (Numbered) (control) statements
  - Comments



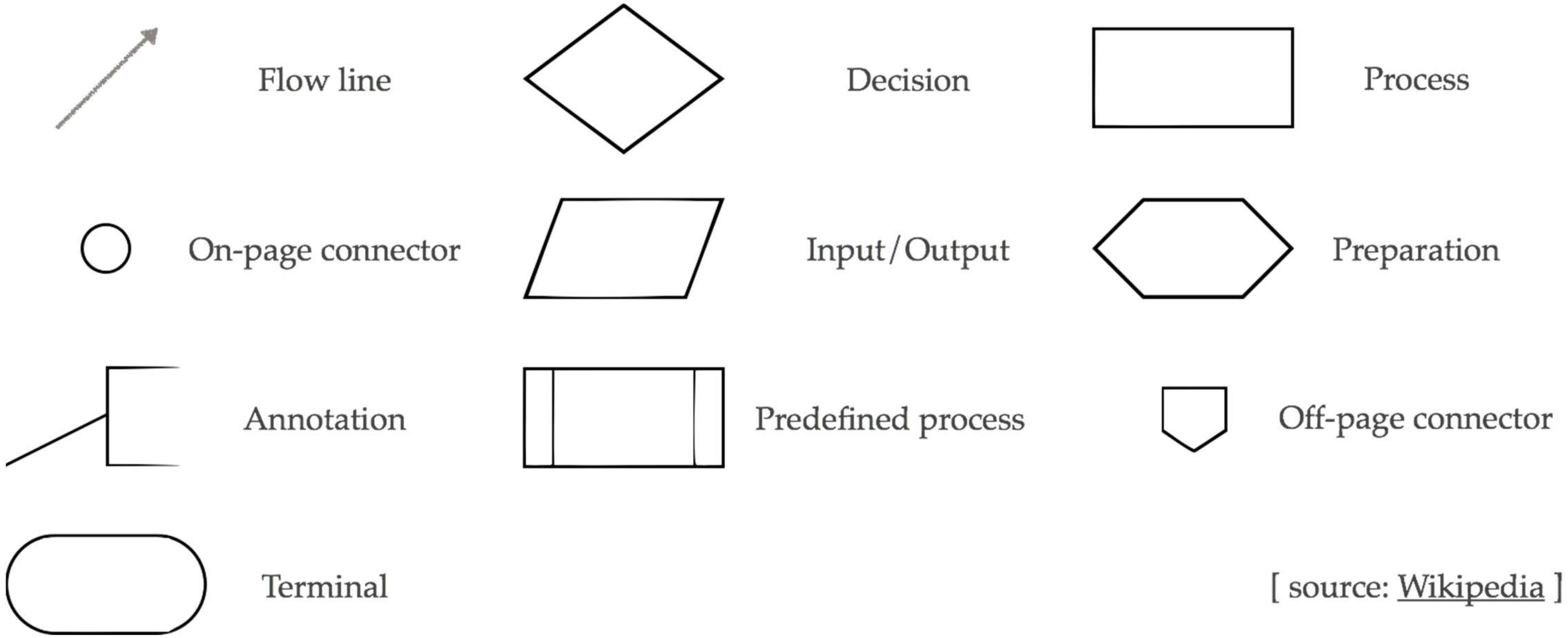
Data and Algorithms – Algorithms – Pseudo Code

- Example 2: given n positive numbers, find the smallest one.

```
Algorithm findMinNumber
-    Input: positiveNumber[n] which is an array of n positive double values
-    Output: minNumber which is the smallest one whose type is double
-    Purpose: find the smallest number in a collection
-    Precondition: n data inputs are positive.
Begin Algorithm
        Check positiveNumber[n] contains only positive values
        minNumber = positiveNumber[1]
        iteration = 2
        While (iteration <= n)
        Begin While
                If (minNumber <= positiveNumber[iteration]) Then
                        iteration = iteration + 1
                Else
                Begin
                        minNumber = positiveNumber[iteration]
                        iteration = iteration + 1
                End
        End While
End Algorithm
```

Data and Algorithms –
Algorithms – Flowchart

- Symbols used for drawing a flowchart



[source: Wikipedia]

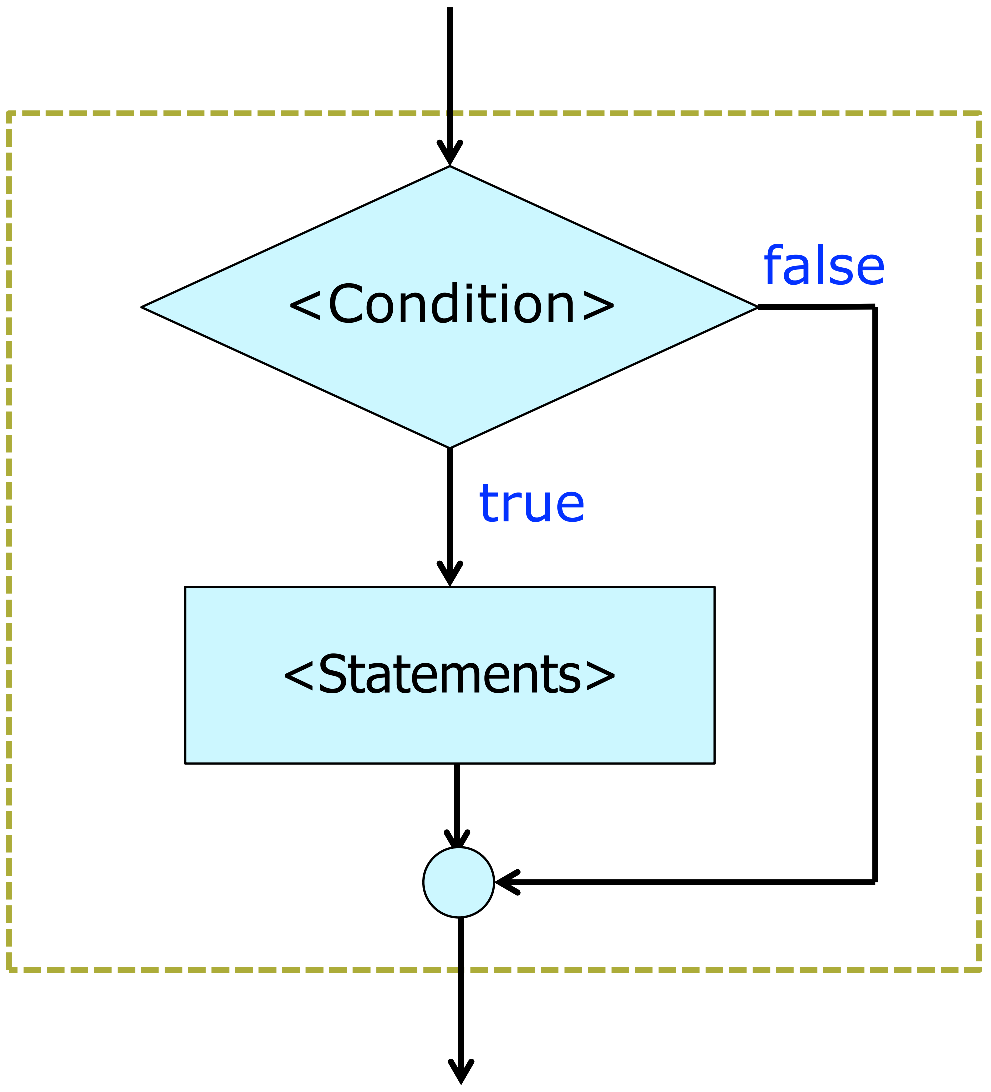
Data and Algorithms – Algorithms - Flowchart

- Terminal: starting point or end point
- Input/Output: input data/output data of the algorithm
- Flow line: shows a control flow of the algorithm. Execution follows this part.
- Decision: allows a condition (expressed as a boolean expression) to be checked
- Process: data processing block

Data and Algorithms – Algorithms - Flowchart

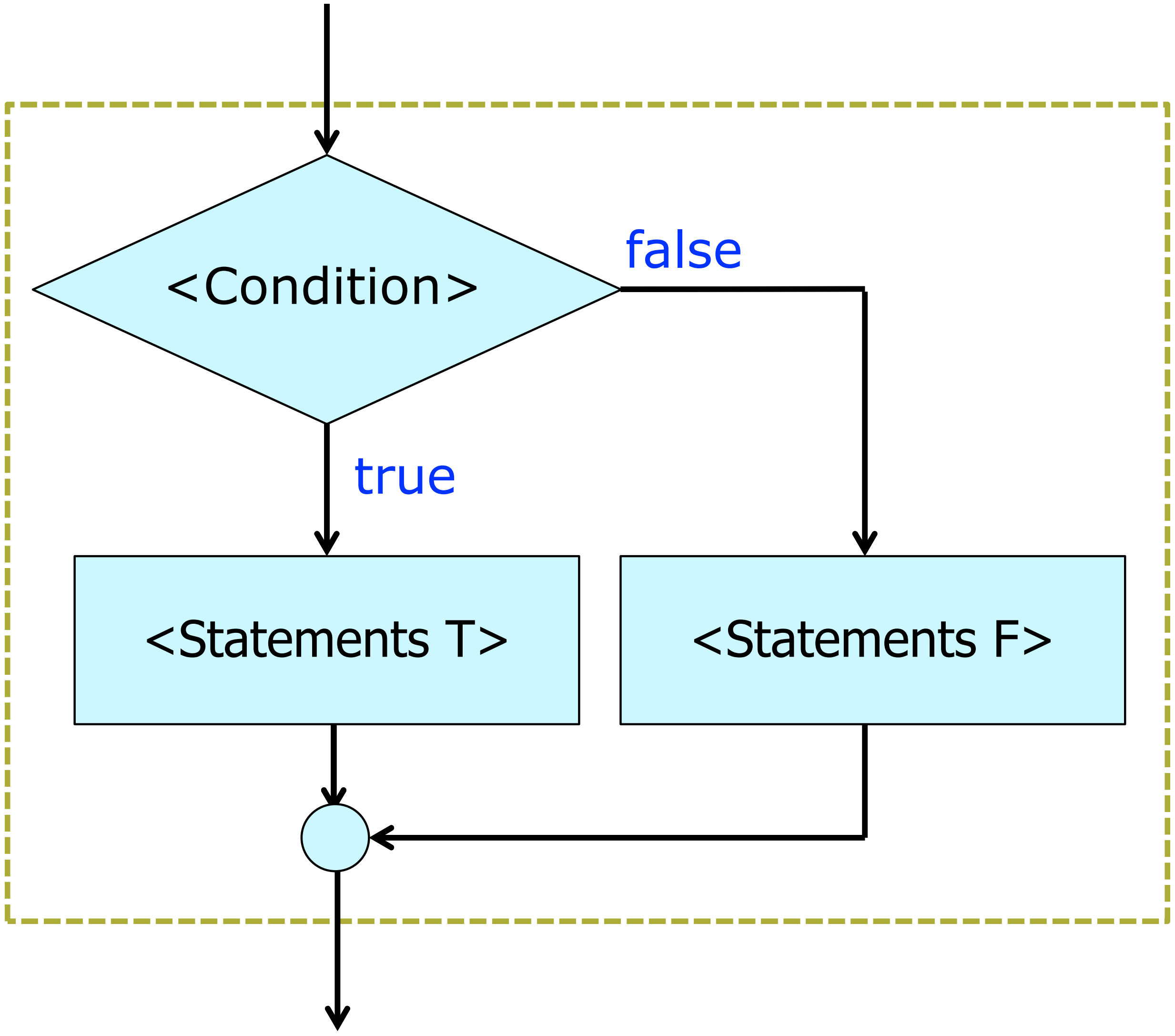
- Predefined process: an existing data processing block
- On-page connector: a gathering point of the flow lines in a flowchart
- Off-page connector: a gathering point of the flow lines from another page
- Preparation: preparation steps, setting for initial conditions
- Annotation: comments

Data and Algorithms –
Algorithms - Flowchart



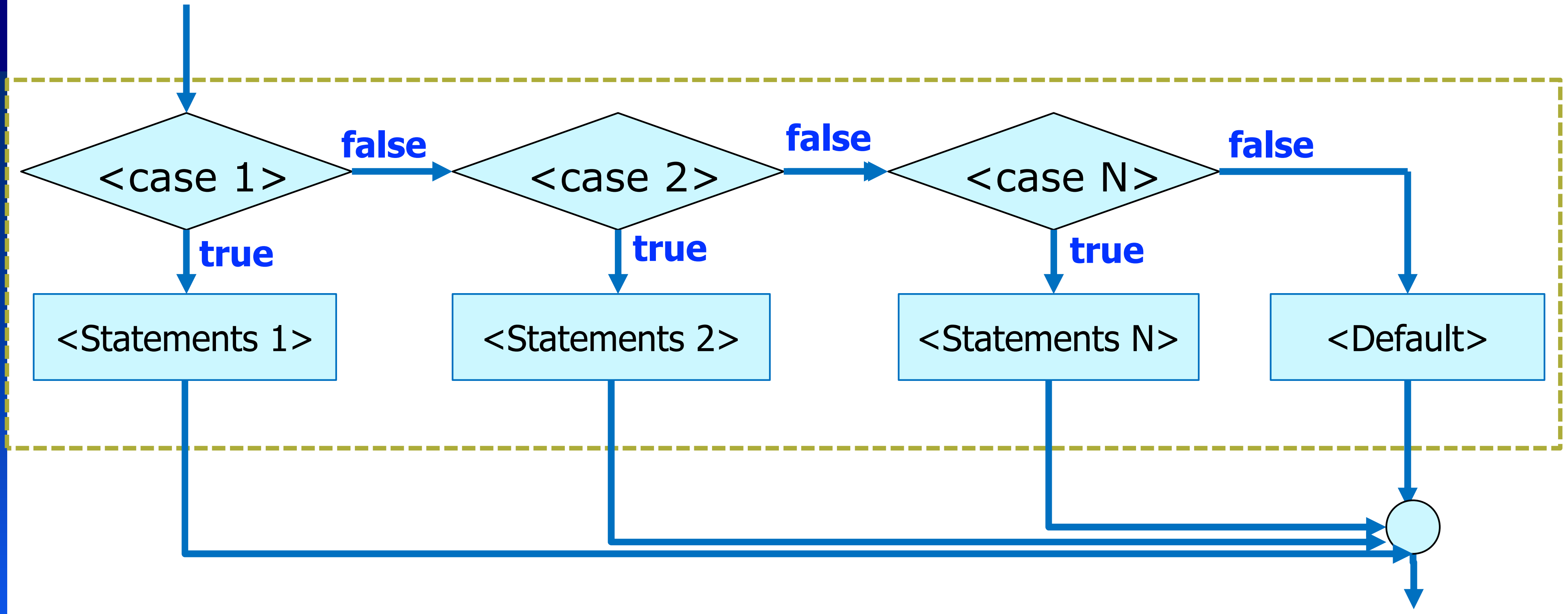
`if Statement`

Data and Algorithms – Algorithms - Flowchart



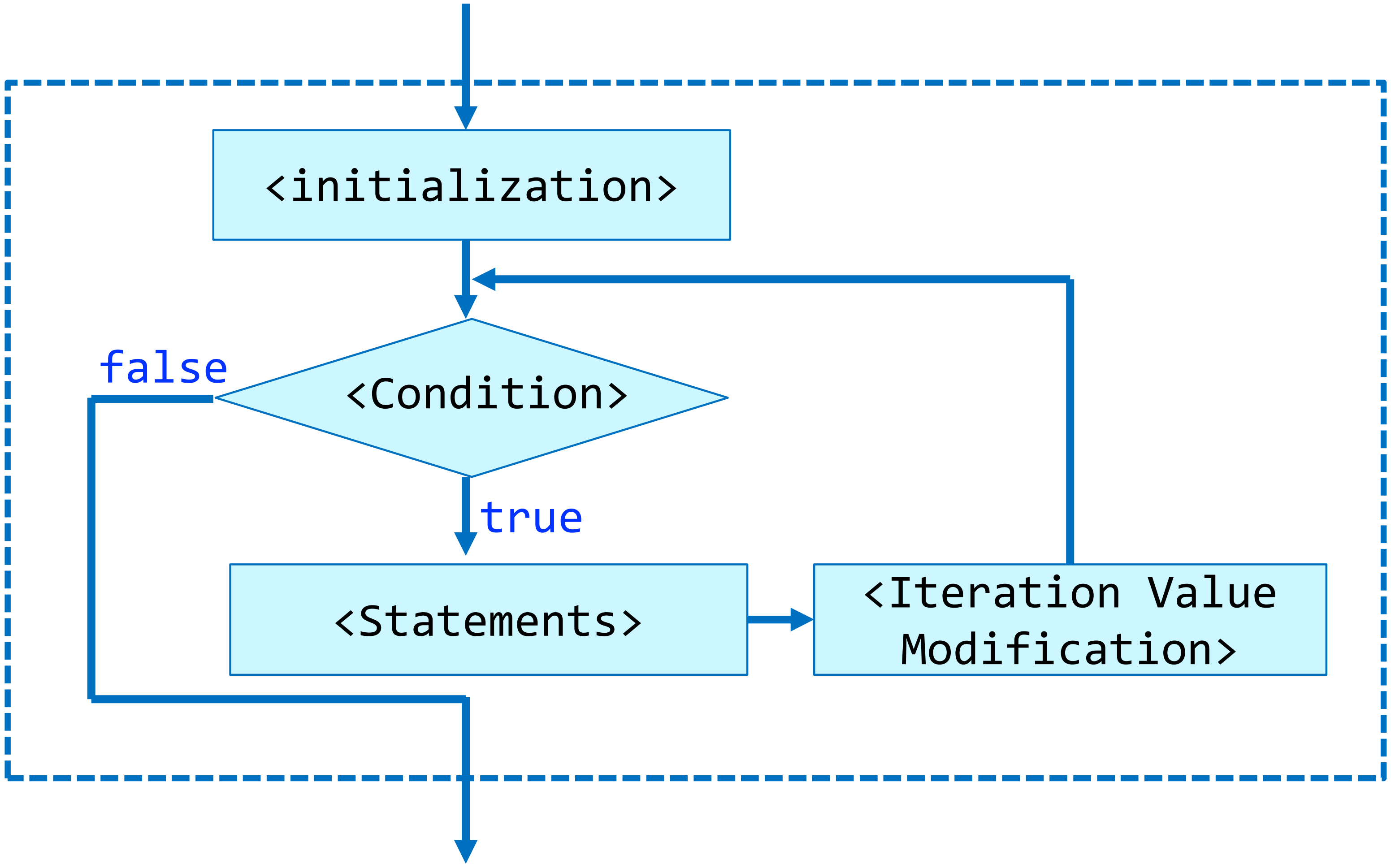
`if-else` Statement

Data and Algorithms –
Algorithms - Flowchart



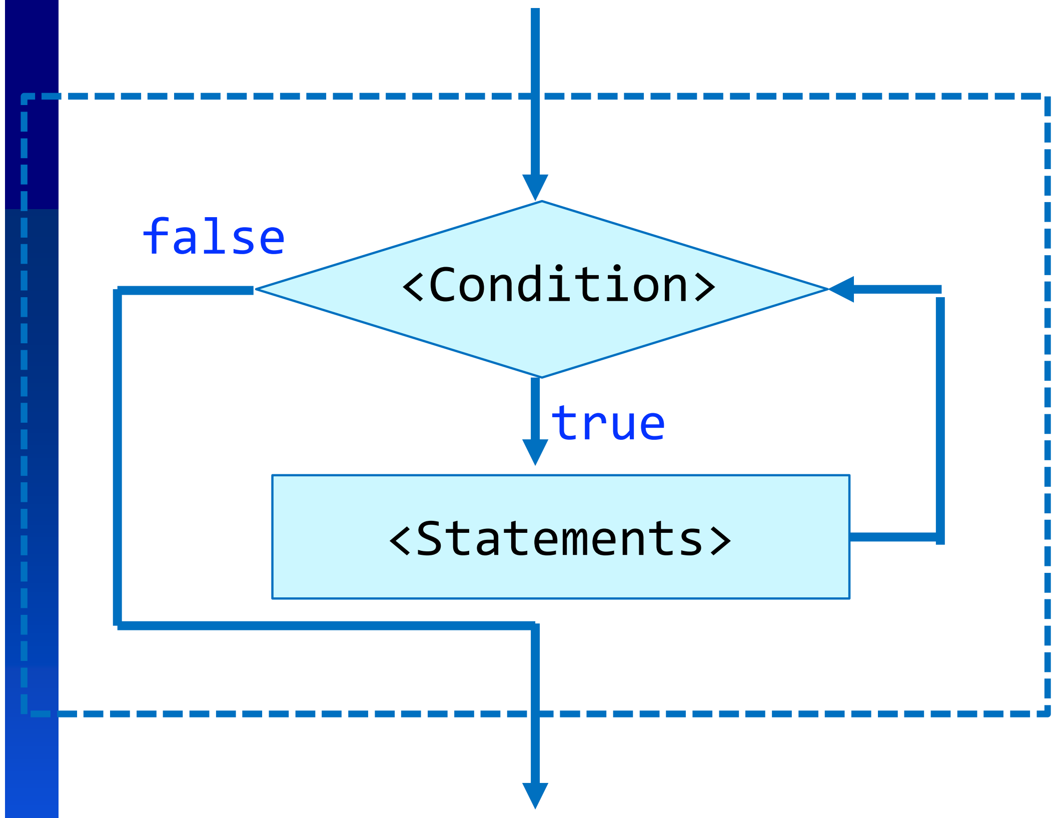
`switch-case` Statement

Data and Algorithms – Algorithms - Flowchart

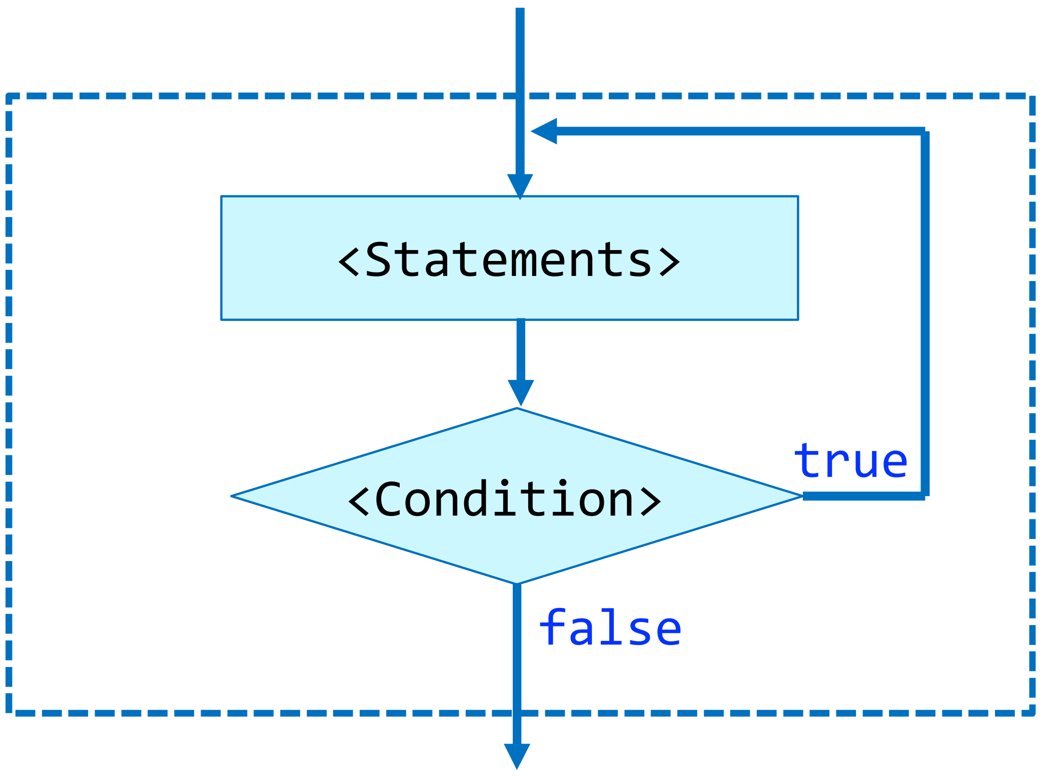


for Statement

Data and Algorithms – Algorithms - Flowchart



while Statement

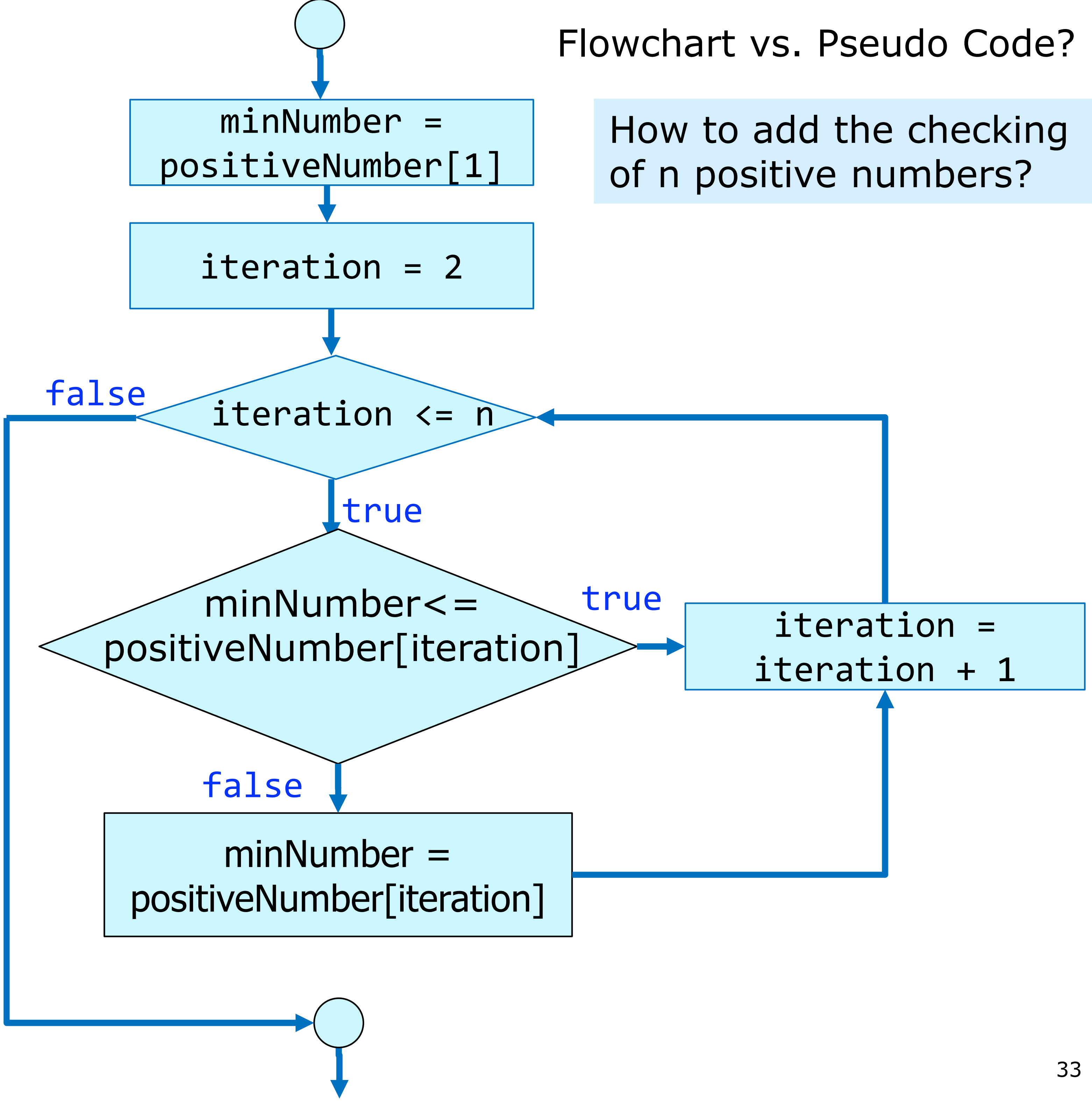


do while Statement

Flowchart vs. Pseudo Code?

- Example 2: given n positive numbers, find the smallest one.

How to add the checking of n positive numbers?


minNumber = positiveNumber[1]
iteration = 2
iteration <= n
false
true
minNumber<= positiveNumber[iteration]
true
iteration = iteration + 1
false
minNumber = positiveNumber[iteration]

Data and Algorithms – Algorithms – Real code in C

```
void main() {
    double positiveNumber[10] = {2, 1, 3, 10, 8, 3, 4, 5, 9, 12};
    int n = 10;
    double minNumber = positiveNumber[0];
    int iteration = 1;
    while (iteration < n) {
        if (minNumber <= positiveNumber[iteration]) iteration = iteration + 1;
        else {
            minNumber = positiveNumber[iteration];
            iteration = iteration + 1;
        }
    }
}
```

How to add the checking of n positive numbers?

Pseudo Code vs. Flowchart vs. Real Code in C?

Summary

- Concepts related to computer programming
- Short introduction to computers, programs, programming, and programming languages
- Short introduction to the C language
- Preparation for computer programming
  - Programming tasks
  - Data and basic data types
  - Algorithms and their representations

Chapter 1: Introduction to Computers and Programming