Fundamentals of C++ Programming

Lecturer: Dustin Nguyen, PhD.
Introduction

❖ Audience: students who have no background in computer programming

❖ Aims: provide basic knowledge and skill on programming with two important programming paradigms: structure programming and object-oriented programming.

❖ Demonstration language: C++

❖ Prerequisite: basic math knowledge

❖ Requirement:
  ❖ Class attendance
  ❖ Self-study
  ❖ Work hard
Introduction

❖ What you will get from the course

• Be able to describe the algorithm for your problem
• Understand and be able to use structure programming techniques
• Be able to implement a given algorithm using C++
• Understand basic concepts of Object-Oriented Programming (OOP)
• Improve your coding style
• The process of solving problem
Syllabus

❖ Course meeting time:
  ❖ Lecture: 3 hours/week for 15 weeks
  ❖ Laboratory: 30 hours

❖ Course mechanics:
  ❖ Textbook: C++ How to program
  ❖ Lecture notes
Syllabus

❖ Assessment

❖ Report on problem sets (collaboration is encouraged but the report must be made by your own hand)

❖ Final lab exam: 60’

❖ Final exam: 90’

❖ Ratio: lab (20%), assignment (30%), final exam (50%)

❖ Coding environment:

❖ Visual studio (recommended), and other IDEs are welcome.
Today’s outline

❖ Hardware vs. Software
❖ Programming language
❖ Problem solving and software development
❖ Algorithm
❖ Discussion
Hardware vs. Software

- Hardware: input devices, CPU, mainboard, output devices, etc.
  - Physical components of computer (including peripherals)
- Software: Windows, OSX, Linux, drivers, MS Office, Adobe Photoshop, etc.
von Neumann architecture
Computer Architecture

- von Neumann architecture
  - Memory unit: holds both data and instructions.
  - Arithmetic/logic gate unit (ALU): is capable of performing arithmetic and logic operations on data.
  - Input unit: moves data from the outside world into the computer.
  - Output unit: moves results from inside the computer to the outside world.
  - Control unit: acts as the stage unit to ensure that all the other components act in concert.
Software

❖ Definition: A set of machine-readable instructions that directs a computer's processor to perform specific operations. [Wikipedia]

❖ Software vs. Program

❖ At the lowest level, executable code consists of machine language instructions specific to an individual processor.

❖ At the high level, programs are built from source code using development tools (also software)

❖ There are programmers who wrote tools to help other programmers and the other who used to solve problems.
Software

- **Software**: system software and application software
- **System software**: manages a computer system at a more fundamental level, provides the tools and an environment in which application software can be created and run.
- **OS**: manages computer resources, such as memory, and input/output devices, and provides an interface through which a human can interact with the computer.
- **Application software**: is written to address specific needs of real-world.

[source: Wikipedia]
Programming language

Let's see how to determine how much a software engineer or developer would be paid.

[source: lifehacker]
Programming language
Programming language

❖ Which language should you choose?
❖ What is your need? What is your programming style?
❖ Why C/C++?
    ❖ The language that places a foundation for many others
    ❖ Gain high-level programming skills
    ❖ Freedom, flexibility, advanced techniques, …
Programming language

❖ The high-level language code is called **source code**.

❖ The compiled machine language code is called **target code**.

❖ **Compiler** translates the **source code** into **target machine language**.

❖ Tools for development
  ❖ Editor: support **text editing** feature for writing source code
  ❖ Preprocessor, compiler, linker: tools for translating source code
  ❖ Debugger: traces program’s execution in order to locate **bugs**!
  ❖ Profiler: collects statistics about a program’s execution
Programming language

- The process of building a computer program

```plaintext
Editor → Preprocessor → Compiler → Linker → Executable program

Problem solution, design of program
Library declaration (source code)
Enhanced source code
Object code
Libraries (object code)
```
Problem solving

- Think about how you solved your problems before
  - Math, Physic, Chemistry, etc.
- How did you solved them?
  - Systematic
  - Random idea
  - Memorising solutions
  - Other approaches
Problem solving

Define problem
Gather information
Explore
Plan
Act
Generalize
Disseminate

criteria
constraints
strategies
known
unknown

look at the problem from different viewpoint
brainstorm
evaluate against criteria
troubleshooting
make sense

solution
method
apply to new situations

Don Woods & Philip Wankat

KuuDuongThanCong.com

https://fb.com/tailieudientucntt
Problem solving

❖ How will you solve a problem on computer?

❖ Computer deals best with performing easy tasks over and over again.

❖ Principle: break the big problem into smaller pieces

❖ We utilize the computer's ability by implementing repetitive techniques to incrementally solve our complex problems.

❖ Think about a problem that you can solve using this method.
Problem solving

❖ Two basic approaches

❖ **Iterative**: solve a part of the problem, repeat the process on the remaining problem until the original problem is solved.

❖ **Recursive**: define how to solve simplest problems. Then we break the problem into simpler and simpler pieces until they reach the level that computer know how to solve (by our definition).
Software development

- The software development process follows the principles of problem solving.
- Various software development models existed
- What do you need to develop a software at the basic level?
  - Compiler, Debugger, Editor
  - Integrated Development Environment (IDE)
  - Visual Studio, Eclipse, Xcode, etc.
Software development

❖ Address your problem, collect requirements
❖ Analyse feasible approaches, find solution
❖ Design algorithm
❖ Implement: write code
❖ Compile, debug
❖ Evaluate the program
❖ Deploy software
Steps in development of algorithm

❖ Problem definition
❖ Development of a model
❖ Specification of Algorithm
❖ Designing an Algorithm
❖ Checking the correctness of Algorithm
❖ Analysis of Algorithm
❖ Implementation of Algorithm
❖ Program testing
❖ Documentation Preparation
Algorithm

- Algorithm is a self-contained list of operations to be performed.
- An algorithm is an effective method that can be expressed within a finite amount of space and time and in a well-defined formal language for calculating a function.

Describe algorithm

- Text: details of data flow and processing steps
- Pseudo code
- Flowchart
Algorithm

❖ Pseudo code

❖ Independent from programming language.
❖ An informal high-level description of the operating principle of a computer program or algorithm.
❖ No standard for pseudo code syntax
❖ Algorithm is often designed with pseudo code description

```
function randomSound()
  Loop: from i = 1 to 100
    print_number = True
    If i is even Then
      If print_number Then
        Print (i + random())
      Else
        PlaySound( rand() )
      End If
    Else
      Print “Odd…”
    End If
  End (loop)
End (function)
```
Algorithm

❖ Pseudo code - guideline

❖ Mimic good code and natural language
❖ Ignore unnecessary details
❖ Don’t belabour the obvious
❖ Take advantage of programming shorthands
❖ Consider context
❖ Don’t lose sign of the underlying model
❖ Check for balance
Algorithm

❖ Pseudo code - example

INPUT A, B
Loop while B > 0
  If A > B then
    A = A - B
  Else
    B = B - A
  End
End loop
Print A
Algorithm

❖ Flow chart

❖ A type of diagram that represents the algorithm (in our context). The flow chart illustrates a solution model to a given problem.

❖ A flow chart is constructed from basic shapes that have specific meanings.
Algorithm

❖ Flow chart - Building blocks

- Flow line
- Decision
- Process
- On-page connector
- Input/Output
- Preparation
- Annotation
- Predefined process
- Off-page connector
- Terminal

[source: Wikipedia]
Algorithm

- **Flowchart**
  - Flow line: represents control pass from one symbol to another.
  - On-page connector: has more than one arrow coming into it but only one going out. It is useful to represent iterative processes.
  - Annotation: represents comments or remarks about the flowchart.
  - Terminal: usually has word/phrase to indicate the start/end of a process.
  - Decision: where the decision must be made (usually Y/N).
Algorithm

❖ Flowchart

❖ Input/Output: involves receiving data and displaying processed data.
❖ Predefined process: represents complex processing steps which maybe detailed in a separate flowchart.
❖ Process: shows that something is performed.
❖ Preparation: prepares a value for a subsequent conditional or decision step (replace decision symbol in case of conditional loop).
❖ Off-page connector: connect to another page.
**Algorithm**

*Flowchart - example*

**INPUT A,B**

Loop while \( B > 0 \)

If \( A > B \) then

\( A = A - B \)

Else

\( B = B - A \)

End

End loop

Print \( A \)
Summarise

❖ Understand concepts of Hardware/Software
❖ Understand problem solving process
❖ Definition of algorithm
❖ Role of algorithm in problem solving process
❖ Knowing how to describe an algorithm
❖ Understand concepts of pseudocode and flowchart
Homework

❖ Describe two or three problems in practice that should be solved by software.
❖ Analyse your problem following the process in this lecture.
❖ Write pseudocode to represent your algorithm.
❖ Draw flowchart to represent your algorithm.