Algorithm and Ambiguity

Upsorn Praphammontripong

CS 1111
Introduction to Programming
Spring 2018
Computing is Everywhere
Computing

Art of computer science (problem solving)
how to come up with solution
how to know if solution will work

Programming skill
how to automate solution
Software Development Life Cycle

- Requirements
- Specification
- Design
- Algorithm
- Implementation
- Testing
- Deployment
- Maintenance

Formal and informal
Notation
Step-by-steps
Coding
Syntax error
Semantic error
Logical error

Functional and Non-functional

When to stop?
Is the program good enough?
Deliverable
Types of Errors

Syntax error
- Does not conform to the rules of the programming language (e.g., incorrect grammar, typo)

Semantic error
- Yields nothing meaningful

Logical error
- Causes the program to operate incorrectly, not crash
- The syntax is correct, but executes without performing the intended action, may produce incorrect output or unintended behavior

Runtime error
- Happens when running the program, generates an exception that terminates the program with an error message
Programming Languages

High-level Language

```plaintext
z = 0;
x = 3;
while (x != 0)
{
    z = z + y;
x = x - 1;
}
y = z;
```

Assembly Language

- `ADD R3 R2 R3`
- `SUB R0 R0 R1 BZERO 4`
- `BRANCH 0`
- `MOVE R2 R3`
- `HALT`

Machine Language

```
10100001000000110
10100010000000110
00000010000000100
0000001000000000
1001000100001011
1111111111111111
```

Compiler / Interpreter  Assembler
Algorithms

Algorithm
• A step by step, list of instructions that if followed exactly will solve the problem under consideration.
• Can be described in many ways. Two commonly used methods:
  • Pseudocode
  • Flowchart

Pseudocode
• An informal description of an algorithm (in English language) which can easily translate into a high-level language (like Python).

Always think about a general solution then write it in a programming language so the computer can do it.
 Algorithms

Algorithms must be:

- **Unambiguous**
  - There are precise instructions for what to do at each step and where to go next.

- **Executable**
  - Each step can be carried out in practice.

- **Terminating**
  - It will eventually come to an end.

---

Don’t think about implementation yet. Try to focus on **how you want to solve the problem**
Example

Problem:
- A company is planning to have a storewide sale of 20%
- Sales tax is 5%
- You enter the price of an item based on the price tag
- Calculate the final sale price of an item after applying the discount and the sales tax
- Display the final sale price

Pseudocode
- Get item price
- Apply 20% discount
- Add 5% sales tax
- Display final sale price

Flowcharts
Pseudocode

- Pseudocode is one of the methods that can be used to represent / describe an algorithm (usually in English)
- Not use specific programming language syntax
- Can be easily translated into a high-level programming language
- Usually include terms specifying a sequence of actions the a program will take
Example: Pseudocode

Test the pseudocode with some simple inputs

Let item price = 100
Does the pseudocode work as expected?
final sale price = 84

If yes, let’s test with more inputs
item price = 0
item price = -100
Does it still work?

Always test the pseudocode and rewrite until it works properly

1. Get item price
2. Apply 20% discount
3. Add 5% sales tax
4. Display final sale price

rewrite

1. Get item price
2. Check if price is <= 0, then repeat step 1
3. Apply 20% discount
4. Add 5% sales tax
5. Display final sale price
Control Structures

**Sequence**
- A series of statements that execute one after another

**Condition (if)**
- To decide which of the two or more different statements to execute depending on a certain condition

**Repetition (loop)**
- To repeat statements while certain conditions are true

**Subprogram**
- A small part of another program solving a certain problem
- A collection of subprograms solves the original problem
Control Structures

Sequence

- A series of statements that execute **one after another**

walk, walk, walk, walk, walk, walk, walk, right-turn-180-degree, sit
**Control Structures**

**Condition (if)**

- **To decide** which of the two or more different statements to execute depending on a certain condition

\[
\text{If (condition):} \\
\text{statement1} \\
\text{else:} \\
\text{statement2}
\]

- **Diagram:**
  - Condition (if) branch
  - If true, execute statement 1
  - If false, execute statement 2
  - Flowchart with decision point and branches
Repetition (loop)

- To repeat statements while certain conditions are true

```
while (condition):
    statement1
    statement2
    statement3
...
```

Repeat until you are in front of the chair, turn, sit

? steps
Control Structures

Subprogram

- A small part of another program solving a certain problem
- A collection of subprograms solves the original problem

A meaningful collection of sequence, conditions, repetitions, and subprograms
Activity

Refer to cs1110.cs.virginia.edu/syllabus.html

For this activity, form a team of 2-3. You will write a pseudocode to compute a final letter grade

• You will enter a percentage of each task
• Compute and display a final letter grade
• Be prepare to share your pseudocode

Ambiguity?

Does your pseudocode solve the problem?