VE280

Programming and Elementary Data Structures

Introduction

Instructor

- Weikang Qian
- Email: qianwk@sjtu.edu.cn
- Phone: 3420-6765 (Ext. 4301)
- Office: Room 430, JI Building
- Office hour
  - Tuesday 7:00 – 8:00 pm
  - Thursday 7:00 – 8:00 pm
  - Or *by appointment*

Time & Physical Location

- Monday 12:10-1:50 pm (odd weeks), D100 (East Upper Hall 100)
- Tuesday 2:00-3:40 pm, A100 (Upper Hall 100)
- Thursday 2:00-3:40 pm, A100 (Upper Hall 100)

Notes on Attending the Lectures

- **Strongly encourage to attend them onsite**
- If you have to attend remotely (hopefully, in **very rare** cases), here are the links for the online lecture:
  - For Monday lectures: https://vc.feishu.cn/j/802486971
  - For Tuesday & Thursday lectures: https://vc.feishu.cn/j/254945235

Textbook for Reference (Not Required)

- "C++ Primer, 4th Edition," by Stanley Lippman, Josee Lajoie, and Barbara Moo, Addison Wesley Publishing, 2005.
- "Problem Solving with C++, 8th Edition," by Walter Savitch, Addison Wesley Publishing, 2011.
- "Data Structures and Algorithm Analysis," by Clifford Shaffer. Online available: http://people.cs.vt.edu/~shaffer/Book/C++3e20120605.pdf

Teaching Assistants

- Chen, Yunuo
  - Email: cyril-chenyn@sjtu.edu.cn

- Ma, Pingchuan
  - Email: hensonma@sjtu.edu.cn

Teaching Assistants

- Su, Zhenxuan
  - Email: sinesu@sjtu.edu.cn

- Sun, Jiajun
  - Email: sunjiajun2007@sjtu.edu.cn

Grading

- Composition
  - Class participation: 5%
  - (About) 5 programming projects: 50%
  - Midterm exam (written): 20%
  - Final exam (written): 25%
- We will assign grades on a curve, in keeping with past grades given in this course.
- Questions about the grading?
  - Must be mentioned to TAs or instructor within one week after receiving the item.

Class Participation

- I may ask a question from time to time and randomly pick a student to answer it
- If you're there every time when I ask you, you get all participation points (That's easy!)
- Otherwise, you'll lose some points.

Projects

- Projects require:
  - Read and understand a problem specification
  - Design a solution (in your mind)
  - Implement this solution (simply and elegantly)
  - Convince yourself that your solution is correct

Projects

- We will give you a few simple test cases to get started. You should design your own set of tests (very important!).
- You will have chance to pre-test your program before the deadline.
  - We will use an online judge.
  - Pre-test cases are a subset of final test cases.
- Grading projects will be done by a combination of testing (correctness) and reading (implementation requirement and simplicity/elegance).

Programming Environment

- We require you to develop your programs on **Linux operating systems** using compiler g++.

- C++11 standard is allowed.
  - Compile with the option –std=c++11

- We will grade your programs in the Linux environment.
  - They must compile and run correctly on this operating system.

Project Deadline

- Each project will be given a due date. Your work must be turned in by 11:59 pm on the due date to be accepted for full credit.
- However, we still allow you to submit your homework within 3 days after the due date, but there is a late penalty.

Hours Late	Scaling Factor
(0, 24]	80 %
(24, 48]	60 %
(48, 72]	40 %

- No work will be accepted if it is more than 3 days late!

Project Deadline

- In **very occasional** cases, we accept deadline extension request.
  - Deadline extension requests will only be considered if you contact the course instructor in person. Do not contact TAs!
  - **ONLY** granted for **documented** medical or personal emergencies that could not have been anticipated.
  - **NOT** granted for reasons such as accidental erasure/loss of files and outside conflicting commitments.

Some Suggestions

- Practice! Build demos yourself
  - You have the freedom. Even try something wrong on purpose
- Learn from your mistakes!
  - Take notes on the mistakes you make. Review frequently
- Start your project early!
  - Don't wait until the last minute. Numerous lessons before
  - **Hofstadter's Law:** It always takes longer than you expect, even when you take into account Hofstadter's Laws
- Make copies frequently in case your computer crashes.
  - Consequence: "computer crash" is NOT a reason for late submission!

Honor Code: Collaboration and Cheating

- You may discuss in oral with your classmates.
- **But** you must do all the assignments yourself.
- Some behaviors that are considered as cheating:
  - Reading another student's answer/code, including keeping a copy of another student's answer/code.
  - Copying another student's answer/code, in whole or in part.
  - Having someone else write part of your assignment.
  - Using test cases of another student.
  - Testing your code with another one's account.

"**Another student**" includes a student in the current semester or in the previous semester.

Honor Code: Collaboration and Cheating

- The previous lists of behaviors are **deliberate** cheating, but some **unintentional** actions could make you look like cheating. For example,
  - You use another's computer to upload your code (in some cases like network/computer problems), but upload another's copy.
- You should be extremely careful!
  - If due to network/computer problem, you need to use another's computer, double check the uploaded file.

Honor Code: Collaboration and Cheating

- In summary, you should be responsible for all answers/codes you submit. If you submit a copy of another student's work (or overwrite another student's work), it is considered cheating, **no matter of the reason**!

Honor Code: Online Attending

- Online Presence and Activities
  - The Joint Institute imposes a "real name" policy for all online activities
  - Students are required to use their actual name (for Chinese students, use Pinyin) as part of their online presence
  - No share of the meeting ID of a given course to any person who is not enrolled in that course
- Online Etiquette
  - Spam messages, verbal and other forms of abuse, and disturbance of the learning experience of other students are not permitted

Honor Code: Teaching and Learning Materials

- Teaching and learning materials, such as lecture slides, assignments, **your solutions**, quizzes, videos etc. are copyrighted and may not be passed on to others without the permission of the course instructor.
    - This applies to recordings of the lectures
    - In particular, it is not permissible to upload videos to sharing platforms (such as Youku or YouTube) or to post lecture slides, assignment questions, project descriptions etc. on public sites such as SlideShare
    - If you use Github to back up your code, make your repository **private**

Consequence of Honor Code Violation

- Any suspect of honor code violation will be reported to **the Honor Council at JI**.
- For programming assignments, we will run an automated test to check for unusually similar programs. Those that are highly similar - in whole or in part - will be reported to **the Honor Council at JI**.
- **Penalty** of honor code violation

1. Reduction of the grade for this assignment to 0, **plus**
2. Reduction of the final grade for the course by one grade point, e.g., B+ → C+, for **both students** involved

Canvas

- Log into Canvas: https://umjicanvas.com

- Check the class webpage on Canvas regularly for
  - Announcements
  - Slides
  - Grades

- Course slides will be uploaded onto Canvas before each lecture.

Getting Help

- If you have any questions, you can come to see TAs and instructor during the office hour
  - Better choice for questions that are not easy to solve!

- You can also post it on **piazza**
  - You can help answer your fellow students' questions

- For private question, you can also write emails to us

Aside: Fun Quizzes!

- What?
  - Multiple-choice questions on slides with ❓
  - **Non-graded** and **Anonymous**
  - Feel free to answer even if you're not sure!
- How?
  - Scan a QR code on your smartphone
  - Answer
  - Note: Some are single-choice; some are multiple-choice
- Why?
  - Have fun!
  - Allow you to check your understanding
  - Allow the instructor to adapt his teaching
- Let's try one!

Do You Like Programming?

Choose one answer:

- **A**. I like it very much!
- **B.** I more or less enjoy it.
- **C**. I’m OK with it.
- **D**. I hate it.

What I Assume You Know

- Some basics of C++
  - Variables
  - Built-in data types, e.g., int, double, etc.
  - Operators, e.g., +, -, *, etc.
  - Flow of controls, e.g., if/else, while, for, switch/case, etc.
  - Functions; function declaration versus definition.
  - Arrays
  - Pointers
  - References
  - Struct

What Does foo(1, 2, 0) Print and Return?

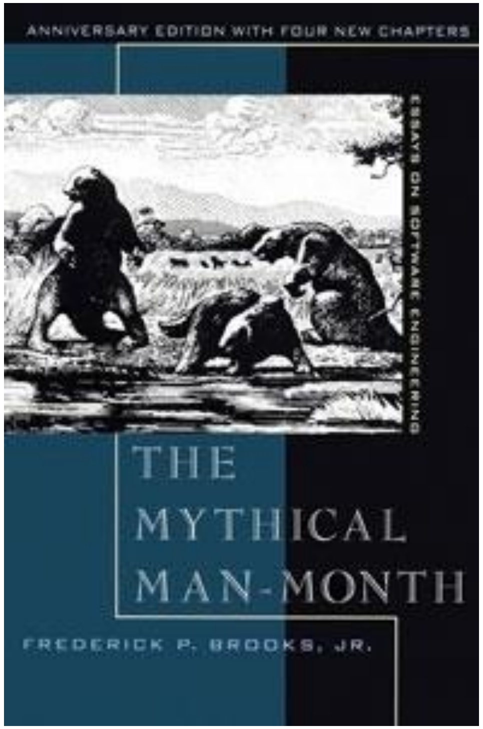
```
double foo(int a, double b, int c){
  while (c<=1) c++;
  cout << (a/b);
  return (a/c);
}
```

Choose the correct answer:

- **A**. It prints “0.5” and returns 0.5.
- **B**. It prints “0.5” and returns 0.
- **C**. It prints “0.5” and returns 1.
- **D**. It prints “0” and returns 0.

Why is Programming Fun?

- First is the sheer joy of making things.
- Second is the pleasure of making things that are useful to other people.
- Third is the fascination of fashioning complex puzzle-like objects of interlocking moving parts and watching them work in subtle cycles.
- Fourth is the joy of always learning, which springs from the nonrepeating nature of the task.
- Finally, there is the delight of working in such a tractable medium.


ANNIVERSARY EDITION WITH FOUR NEW CHAPTERS
ESSAYS ON SOFTWARE ENGINEERING
THE MYTHICAL MAN-MONTH
FREDERICK P. BROOKS, JR.

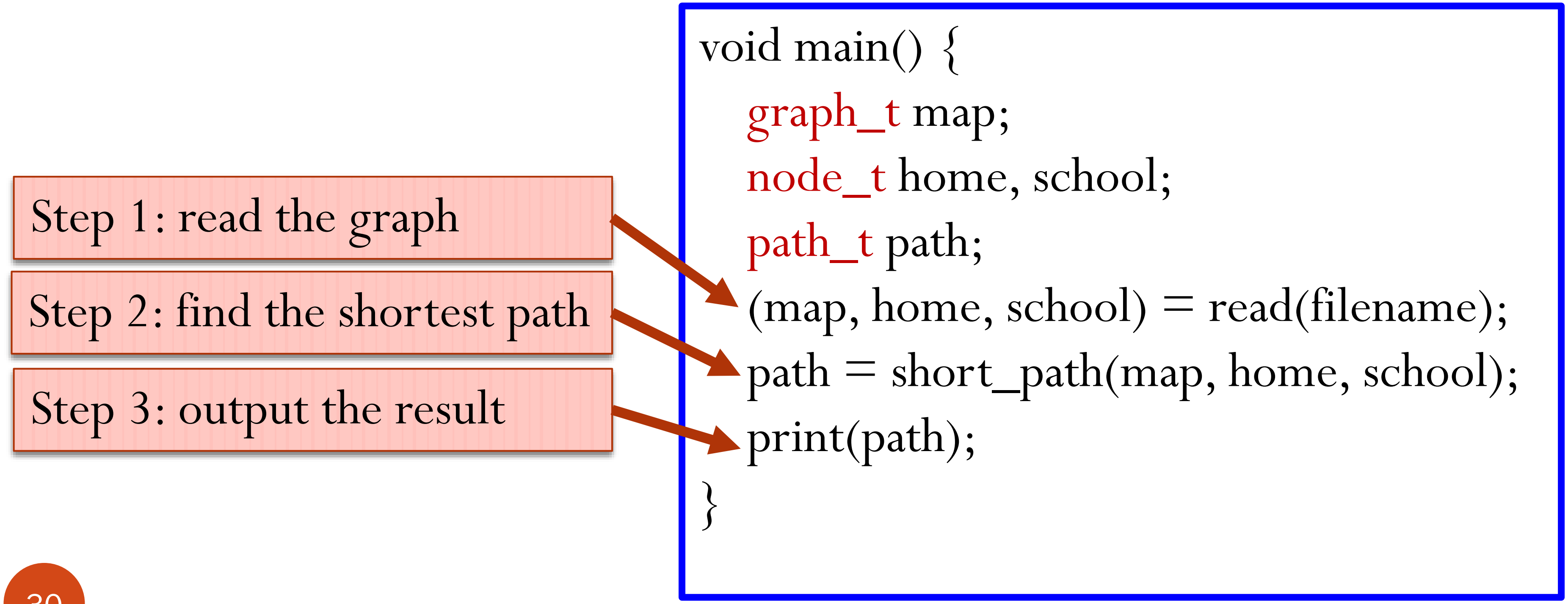
The Mythical Man-Month: Essays on Software Engineering

The Task of Programming

- Accept some specifications of the problem. (E.g., find the shortest way to go from my home to school.)
- Problem solving phase:
  - Design an algorithm (perhaps in pseudo-code/flow chart) that
    1) correctly satisfies the specification.
    2) is efficient in its usage of **space** and **time**.
- Implementation phase:
  - Implement the algorithm **correctly** and **efficiently**
    1) An implementation of an algorithm is correct if it behaves as the algorithm is intended for all inputs and in all situations. **Correctness is never negotiable!**
    2) **Efficient** can mean fast, simple, and/or elegant.

Problem Solving Phase

- Usually, hierarchical design: decompose into sub-tasks
- Example: find the shortest path from home to school



Step 1: read the graph

Step 2: find the shortest path

Step 3: output the result

Pseudocode

```
void main() {
    graph_t map;
    node_t home, school;
    path_t path;
    (map, home, school) = read(filename);
    path = short_path(map, home, school);
    print(path);
}
```

Key Points of Ve280

- The focus of Ve280 is on the **implementation** part. Some **key points** we will learn include
  - Abstraction and its realization mechanism
  - Techniques to increase code reuse
  - Techniques to efficiently use memory
  - Elementary data structures
  - Some other essential parts of C++ programming

Abstraction

- One important concept about programming
  - Provides only those details that matter
  - Eliminates unnecessary details and reduces complexity
  - You already know one realization of abstraction: function (e.g. exp(x)), which is procedural abstraction

Abstraction

- We will talk about
  - Basics about abstraction
  - Procedure abstraction (i.e., function), in more detail
  - Data abstraction (i.e., class)
    - Basics about class: constructor, destructor, etc.
    - Abstract base class

Elegant code!

```
void main() {
    graph_t map;
    node_t home, school;
    path_t path;
    (map, home, school) = read(filename);
    path = short_path(map, home, school);
    print(path);
}
```

Data abstraction

Procedure abstraction

Techniques to Increase Code Reuse

- Function and class, which are basic ways to increase code reuse
- Class inheritance and virtual function
- Template and polymorphism
  - Template: write one thing, used for many different types

Techniques to Efficiently Use Memory

- Sometimes, the amount of memory needed to solve a problem can vary a lot
- Of course, you can write your program considering the worst-case memory usage
  - For example, a large enough array to hold data
  - However, this may lead to some waste in memory use

- We will learn a solution: **dynamic memory management**
  - Dynamic memory allocation and de-allocation

Elementary Data Structures

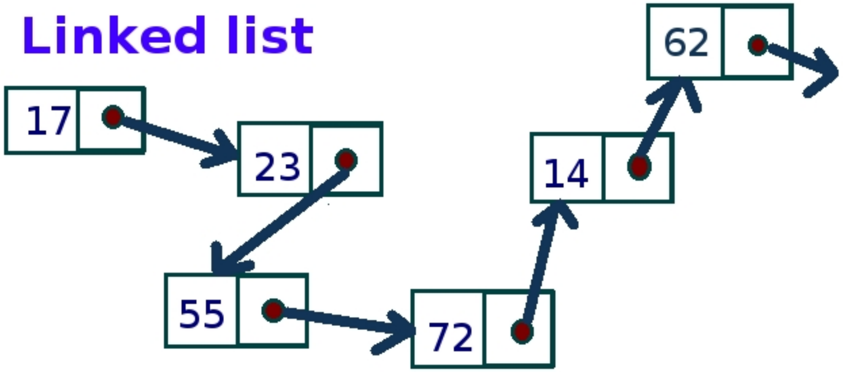
- Data structures are concerned with the **representation** and **manipulation** of data.
- All programs manipulate data.
- So, all programs represent data in some way.

Example: Store a list of numbers

Array

17	23	55	72	14	62

Linked list



Elementary Data Structures

- We will learn
  - Linked list
  - Linear list
  - Stack
  - Queue

- **Note**: This course only shows a few elementary data structures
  - More data structures will be taught in a following course, Ve281 Data Structures and Algorithms

Other Essential Parts

- Writing programs that take arguments
- I/O streams, including file I/O
- Error handling
- Testing
- Linux

What Are the Issues with this Code?

```
int f(int a, int *b, unsigned c)
{
 int s = 0;  int p = 1;
for(unsigned i = 0; i <= c; i++) {
 s = s + b[i] * p;
  p = p * a; }
   return s; }
```

Choose all correct answers:

- **A**. There is no comment.
- **B**. The naming of variables/function is not clear.
- **C**. The code is not indented.
- **D**. The style is not consistent.

Good Programming Style

```
// Evaluate the polynomial on x
int poly_eval(int x, int *coef, unsigned degree) {
      int result = 0;
      int x_power = 1;
      for(unsigned i = 0; i <= degree; i++)
            result += coef[i] * x_power;
            x_power *= x;
      }
      return result;
}
```

Comments

Meaningful Naming

Indentation

Consistency!

Relation with Other Courses

- Vg101 Introduction to Computers and Programming
  - Very basic programming skills.
  - Ve280 will go in depth. To connect, we will review some basics.

- Ve281 Data Structures and Algorithms
  - Focus on the efficiency of the algorithms.
  - Ve280 focuses on correctness. It will show you some very basic data structures.

Questions?