Tin học cơ sở 4

Introduction

Course Introduction

- Số tín chỉ: 3
- Lý thuyết: 19-20 tiết.
- Bài tập: 5 tiết.
- Thực hành: 18 tiết.
- Lecturer:
  - Phạm Bảo Sơn, Khoa CNTT

Objectives

- Procedural Programming in C
- Data types and Algorithms
- Levels of abstraction in describing software systems
- Background for later subjects
- Goals after completing this course:
  - Design, construct, document and test a C program complying given specs.
  - Understand and use common data structures to solve problems.

Textbooks

- Kernighan & Ritchie, *The C Programming Language 2nd ed.*, Prentice Hall, 1988.
- Bùi Thế Duy, Hồ Sĩ Đàm (chủ biên), *Lập trình cơ bản với C*, NXB ĐHQGHN 2012.

Course Website

- www.coltech.vnu.edu.vn/~sonpb/THCS4
- www.bbc.vnu.edu.vn
- Nội dung:
  - Bài giảng, bài tập, thông báo, và mọi thông tin đều được đăng tại đây, ***sinh viên có trách nhiệm thường xuyên theo dõi***
  - Mỗi sinh viên sẽ có một tài khoản riêng để nộp bài tập và tham gia diễn dàn, sinh viên có trách nhiệm bảo vệ tài khoản của mình
- Diễn đàn: trao đổi các nội dung liên quan đến môn học
  - Những bài có nội dung không liên quan hoặc lời lẽ thiếu lịch sự sẽ bị xóa bỏ
  - Sinh viên nào cố tình gửi các bài thuộc loại trên sẽ bị cắt tài khoản website

Programming Language

- C/C++
- Compiler:
  - C: GNU gcc in Linux/Cygwin
  - C++: GNU g++
- IDE: Dev-C++:
  - bin\g++
  - bin\gcc
  - Download:
    - Windows 8: http://sourceforge.net/projects/orwelldevcpp/
    - Alternatives for Windows 7/XP http://www.bloodshed.net/devcpp.html (version 5)
- Labs and assignments will be marked using Dev-C++.

Assessment

- Class mark (40%):
  - Attendance
  - Weekly labs
  - Assignments
- Final exam (60%):
  - Programming tasks
  - Oral exam
- Conditions for taking final exam:
  - 80% attendance for both lectures and labs
  - Have all mark components

Policy

- Encourage discussion but labs and assignments must be your individual work
- Codes copied from books or other libraries but be explicitly acknowledged
- Sharing or copying codes is NOT allowed.

Top 10 ways to piss off your lecturer

1. Don't think, just take notes in lectures.
2. Don't ask when you don’t understand.
3. Skip lectures or labs and then ask for help.
4. Ask "Is this on the exam?".
5. Don't explore.
6. Don't do the lab exercises
7. Copy someone else lab exercises.
8. Talk in lectures.
9. Disrupt lectures with mobile phones.
10. Cheat on assignments.

Main Topics

- Procedural Programming
- Program Structure
- Variables and Data types
- Control structure
- Functions
- Array and String
- Pointer and Dynamic memory allocation
- Simple Algorithms
- Program Design

Why C/C++

- Good example of an imperative language
- Widely used in industry (and science)
- Many libraries and resources
- Fast compilers
- Provide low level access to machine

Brief History of C

- C and Unix share a complex history
- C was originally designed for and implemented on the Unix operating system on a PDP-11 computer
- Dennis Ritchie was the author of C (1971).
- In 1973 Unix was rewritten in C
- BCPL (mid-60s) strongly influenced the C language
- B (1970, cut-down of BCPL) was the predecessor to C, but there was no A
- BCPL and B are typeless languages, C is a *typed* language (but not strongly).

Brief History of C (cont.)

- In 1978 - The C Programming Language by Kernighan & Ritchie (1st edition) published
  sets first standard for C - "K&R C"
- In 1983, American National Standards Institute (ANSI) established a committee to clean up and standardise the language.
- In 1988, the ANSI C standard was published.
- In 1990, ISO adopts ANSI C with no changes – now an international standard
- This greatly improved source code portability
- See http://cm.bell-labs.com/cm/cs/who/dmr/chist.html for more details

C Offspring

- Concurrent C (1989)
- Objective C (1986)
- C* (1990)
- C++ (1986)
- Java (1993)
- C# (2001)

C++

- Backward compatible with C
- Better support for data abstraction
- Object oriented
- Support generic programming
- ***We only teach C in this course. C++ will be taught at later courses.***

C Program

hello.c:

```
#include <stdio.h>

int main() {
    printf("Hello, world");
    return 0;
}
```

Breakdown of C Program

hello.c:

```
//include standard library defs and functions
#include <stdio.h>

int main() //function and entry point
{
      //library call to print constant
      printf(”Hello, world”);
      return 0; // tell OS that no error occured
}
```

Program and States

- Procedural programs work by state-change.
- The state of a program:
  - Variables (mutable typed containers)
  - Position (place in code where executing)
  - External files (input/output file positions)
- The program comprises:
  - Declarations (description of variables: names and types)
  - Statements (list of operations to be performed).
- State is dynamic; program is static.

Key Points

- Functions can be called anything except *main* which is special.
- *main* is where the program starts executing.
- There must be one and only one *main*.
- Functions can have parameters or no parameters
- Curly brackets “{ }” are used to enclose statements in a function.
- A function can be called by naming it in a statement.
- Each statement is terminated by a semicolon “;”.
- “//” comments do not extend beyond the end of line.
- Comments between “/*” and “*/” can span multiple lines.

Statement example

```
int a, b, c;
a = b + c;
if (a > b) a = b;
printf(”Hello, world”);
printf(“%d\n”, a);
```

Header files

- Contains interface declarations of the library used.
  - #include
- stdio.h: input/output (C)
- stdlib.h: standard library (C)
- string.h (C)
- math.h: (C)

Reference Material

- K&R Chapter 1.