CP264 Data Structures II

Instructor: Dr. Hongbing Fan (HBF)

1. Introduction to the course
2. Introduction to C
    1. C background
    2. C features
    3. Applications of C
    4. Compiling & execution

What and why data structures?

- Data structures define how data are represented, organized, stored, and operated in programs and computers.

- Data structures are essential for data representations and operations in programming, and critical for time and space efficiency of algorithms and programs.

CP164 vs CP264

- What we learned in CP164 Data Structure I
  1. The basic concepts of data structures, abstract data type (ADT)
  2. Fundamental DS: array, list, stack, queue, tree, dictionary.
  3. Application programming of the data structures in **Python**.

- What we will do in CP264 Data Structures II
  1. Go deep into the principles of data structures and algorithms, to **learn data structure design, analysis and implementation at low memory level in C programming language.**
  2. Go deep into fundamental DS: arrays, linked lists, queues, stacks, trees, hash tables**, heaps, graphs**.
  3. **Design, analyze, and implement high performance application specific data structures in C.**

Why do we need to learn C?

- **C is a foundation of programming languages.**
  - Over 50 years old, the first standardized programming language, the mostly used programming language.
  - Many programming languages are based on C, e.g., C++, Java, C#.
  - Many interpreters are written in C, e.g., Python, PHP, JavaScript.
- **C has been used in many applications.**
  - C is used to write operating systems, e.g., Unix, Linux, Windows, Mac OS, iOS, Android.
  - C is used to write high performance application programs.
  - C is used to write programs for embedded systems.
- **C has powerful features**
  - C is a high level compiled program language.
  - C is capable to do memory address and bit level operations.
  - C is small, extensible, stable, and portable.

Why do we use C in DS II?

- Understand fundamentally how data structures work in programs at memory and CPU instruction level.

- Design and implement data structures down to memory level for efficient data representation, storage, and processing.

- Design and implement application specific data structures to improve the performance of algorithms and programs.

Why this is a required core course?

- DS II provides essential understanding of how data structures and programs work at memory and CPU level.
- DS II provides foundational knowledge and skills for the careers of computer science and software engineering.
- DS II provides the foundation for other CS courses including
  - CP312 Algorithm Design & Analysis I
  - CP367 Introduction to System Programming
  - CP386 Operating Systems
  - CP411 Computer Graphics, using C/C++

Teaching, learning, doing

1. 12 weekly lessons
   - Read weekly lessons
   - Do self-evaluated non-graded quizzes after each lesson section
2. 34 Lectures
   - Focus on core concepts, principles, and practices.
   - Delivered with notes/slide, demonstrations, assignment helps, Q&A.
3. 10 assignments (individual work, graded)
   - Design and implement DS to solve computing problems in C.
   - Submit through MyLS
4. Midterm and final exam
   - Scoop: subjects covered in lectures, lessons, and assignments
   - Style: in person, paper-based
   - Question types: selection, short answer, programming

5. Labs are provided as optional practices; lab tasks are not graded.

Evaluation

1.	10 assignments	30%
2.	Midterm	15%
3.	Final exam	55%

All tasks count, no dropping, no weight transfer.

You are required to achieve a minimum of grade of 50% on the final exam, and 50% on overall grade to pass the course.

Q&A

Check course syllabus and web site for more info.

2. Introduction to C

1. C background
2. C features
3. Applications of C
4. Compiling & execution

1. C background

- **Origin**
  - C is a general-purpose programming language initially created and developed by Dennis Ritchie between 1969-1973 at AT&T Bell Labs. Dennis Ritchie invented C to export Unix OS to different platforms.
  - UNIX OS was originally written in assembly by Ken Thompson.

- **Standardization**
  - American National Standards Institute (ANSI) approved first C standard in 1989 (known as ANSI C or C89).
  - International Organization for Standardization (ISO) approved a revised C standard in 1999 (known as C99). Followed by C11, C17, C23.

2. C features

1. **C is a high-level compiled programming language**.
   - C source code program is written in **easily understandable form**, converted (compiled) to **executable machine code program** stored in file**,** then loaded to memory to run.
   - **C compiler** is the program to convert source code programs to executable programs.
   - C compilers are available for most computer platforms.

2. **C is small, simple, but powerful**
   - Small: **32 keywords**, a few libraries
   - Simple: simple syntax and simple program structure
   - Powerful: supports memory address operations of variables, data structures, and functions; Access processor instruction operations, including bitwise operations.

3. **C is stable, portable, and extensible.**
   - Stable: not much changes due to its standardization.
   - Portable: source code programs can be compiled by cross platform compilers; compiled executable programs run different computers of the same platform.
   - Extensible: allow to add libraries for extended functionalities.

4. **C is the primary programming language for procedural programming paradigm**.
   - Structured programs featured with program controls, blocks, and subroutines (functions).
   - Extensive use of function (also known as procedure, routine, or subroutine) calls.

3. Applications of C

1. C is used to write almost all OS kernels, device drivers, system programs, utility programs, compilers.
   E.g. Unix, Linux, Windows, Mac OS, …, drivers for monitors, mice.
2. C is used to write almost all interpreters of other programming languages. E.g. Python, Java (JVM), JavaScript, …
3. C is used to write high-performance computing programs.

   Graphics programs, browsers, GUI, large scale parallel programs on multiple CPUs and GPUs.

- C influenced other programming languages.
  - C++ (created by Bjarne Stroustrup, 1979) is an extension to C for Objected-Oriented Programming (OOP).
  - Java (created by James Gosling, 1984), OOP like C++, run on JVM.

4. Compiling and execution

Quick start: the first program in C to print hello, world

```
// hello.c
#include <stdio.h>                    // include a header file
void main()                           // the main function
{
   printf("hello, world\n");   // function call
}
```

Refer to page 6 of reference book The C programming Language.

C compiling

C compiling has four steps:

1. **Pre-processing**: Resolve lines with pre-processor directives, i.e., lines start with #
2. **Compilation**: Convert C program into assembly program.
3. **Assembling**: Convert the assembly program to binary format, called object program.
4. **Linking** : link necessary object programs to create an executable binary machine code program.

An executable program is stored in a file, which can be loaded directly into memory to run.

- C compiler is the program to do C compiling.

We use MinGW (**Min**imalist **G**NU C compiler for **W**indows).
gcc is GNU C compiler name, GNU Compiler Collection

Mac OS comes with Clang, a C compiler by Apple.

- Compile by command line operation:

```
gcc hello.c                // use default C99 standard
```

  This creates executable program:
  a.exe on Windows
  a.out on Linux/Mac

- Execution

```
a.exe     on Windows
./a.out   on Linux/Mac
```

Compiler options

-std=c99 use C99 standard

```
gcc –std=c99  hello.c
```

-c suppress linking step of compilation, generate an object file with .o extension

```
gcc –c hello.c          => create hello.o
```

-g (gdb) embbed diagnostic information into the object/executable program file.

```
gcc –g hello.c          => a.exe
```

-o specify name of the executable program from linking instead of a.out

```
gcc hello.c  –o hello.exe => hello.exe
or
gcc hello.c  –o hello => hello.exe
```

-S suppress the assembling and linking steps, generate assembly program file .s

```
gcc –S hello.c          => hello.s
```

-On (optimize) 0 turn off, 1 default, 2, 3, 6 higher optimized

gcc –O3 –o hello hello.c => hello.exe (executable, optimized with level 3)

-Wall   give compiling warning

-w not give compiling warning

-I <dir>: (capital i), adds <dir> to the list of directories searched for header files.

-L <dir>: Adds <dir> to the list of directories searched for libraries.

-l <lib>: (little l), links with the specified library <lib>

File name extension convention

.c	C language source code file
.h	C language header file
.exe	executable (.out for Unix/Linux)
.o	Object file
.s	Assembly language source file
.lib	static library of object modules, (.a for Unix/Linux)
.dll	dynamic library file (.so for Unix/Linux)

4. C Program Execution

1. An executable program, consisting of machine code instructions, is loaded into the main memory.
2. The execution starts from the first instruction of the main function, one by one controlled by the program's flow control. The execution stops at the end of the main function.
3. When executing an instruction of a program, the instruction is read from memory to CPU register and then executed by CPU.
4. Basic categories of CPU instructions:
   – Data transfer
   – Arithmetic/logic operations
   – Control flow