Course organization

- **Course introduction (Week 1)**
  - Code editor: Emacs （Week 2）
- **Part I: Introduction to C programming language (Week 3 - 12)**
  - Chapter 1: Overall Introduction (Week 3-4)
  - Chapter 2: Types, operators and expressions (Week 5)
  - Chapter 3: Control flow (Week 6)
  - **Chapter 4: Functions and program structure (Week 7)**
  - Chapter 5: Pointers and arrays (Week 9)
  - Chapter 6: Structures (Week 10)
  - Chapter 7: Input and Output (Week 11)
- **Part II: Skills others than programming languages (Week 12- 13)**
  - Debugging tools（Week 12）
  - Keeping projects documented and manageable （Week 13）
  - Source code managing （Week 13）
- **Part III: Reports from the battle field (student forum) (Week 14– 16)**
  - Presentation (week 14-15)
  - Demo (week 16)

Chapter 4. Function and Program Structure

Chaochun Wei

Shanghai Jiao Tong University

Spring 2019

Functions

- Break large program into smaller ones
- Enable people to build on existing codes
- Hide details of operation
  - Clarify the whole program
  - Make it easier to modify a program

4.1 Basics of functions

Function definition

```
return-type function-name(argument declarations)
{
    declarations and statements
}
```

E.g.: A minimal function

```
dummy () {}
/* this is a do-nothing and return-nothing function.
   return type int is assumed */
```

4.1 Basics of functions

Eg: chpapt4_1_functions.c

/* print lines containing a certain text pattern */

```
#include <stdio.h>
#define MAXLINE 1000    /* maximum input line length */

int getline_ ( char line[], int max);
int strindex (char  source[], char searchfor[]);
char pattern[] = "ould"; /* pattern to search for
        */

/* find all lines matching pattern */
main () {
  char line[MAXLINE];
  int found = 0;
  while (getline_ (line, MAXLINE) > 0)
    if (strindex(line, pattern) >= 0 ) {
      printf("%s", line);
      found ++; }
  return found;
}
```

```
/* getline: get line into s, return length */
int getline_ (char s[], int lim) {
  int c, i;
  i = 0;
  while (--lim > 0 && (c=getchar()) != EOF && c != '\n' && c
        != ';')
    s[i++] = c;
  if (c == '\n' )
    s[i++] = c;
  else if (c == ';') return i;

  s[i] = '\0';
  return i;
}
/* strindex: return index of t in s, -1 if one */
int strindex (char s[], char t[]) {
  int i, j, k;
  for (i = 0; s[i] != '\0'; i ++ ) {
    for (j = i , k= 0; t[k] != '\0' && s[j] == t[k]; j ++, k++)
      ;
    if (k > 0 && t[k] == '\0')
      return i;
  }
  return -1;
}
```

4.1 Basics of functions

- **Compile and load a C program from multiple source files**
  - **gcc -c main.c getline.c strindex.c**

  or

  - **gcc main.c getline.o strindex.o**

  will create an executable called a.out

Please divide the chpat4_1_functions.c into three files, compile, and run as above.

4.2 Functions returning non-integers

- **Functions other than void and integer type**

Example:

```
/* atof: convert string s to double */
double atof(char s[ ])
{
  double val, power; int I,k, sign;
  for (i = 0; isspace(s[i]); i ++) /* skip white spaces */
    ;
  sign = (s[i] == '-') ? -1: 1;
  if (s[i] == '+' || s[i] === '-') i ++;
  for ( val = 0.0; isdigit(s[i]); i ++)
    val = 10.0 * val + (s[i] - '0');
  if ( s[i] == '.')
    i ++;
  for (power = 1.0; isdigit(s[i]); i ++) {
    val = 10.0 * val + ( s[i] - '0');
    power *= 10.0;  }
  return sign * val/power;
}
```

Eg:chapt4_2_atof_main.c

4.3 External variables （外部变量）

- **Are globally accessible and they can be referenced by**
  - **Many functions**
  - **With the same name**
- **Provide a communication between functions**
  - **Can have a bad effect leading to too many data connections between functions**

Eg: chapt4_3_polish.c

4.4 Scope rules

- **Source codes can be in different files**
  - **Variable declaration organization**
  - **Variable initialization**

- **Declaration and definition of an external variable**

extern int sp;

extern double val [];

/* this is a declaration */

Eg:polish_cal_main.c

Initialization goes with the definition

4.5 Header files （头文件）

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include “calc.h”

#define MAXOP 100

Main ( ) {
…
}
```

calc.h

```
#define NUMBER ‘0’
void push (double);
double pop (void);
int getop (char [ ]);
int getch ( void );
void ungetch (int);
```

getop.c

```
#include <stdio.h>
#include <ctype.h>
#include “calc.h”
getop() { …}
```

getch.c

```
#include <stdio.h>
#define BUFSIZE 100
char buf[BUFSIZE]
int bufp = 0;
int getch (void) {
…. }
void ungetch (int) {
…}
```

stack.c

```
#include <stdio.h>
#include “calc.h”
#define MAXVAL 100

int sp = 0;
double val[MAXVAL];
void push (double) {
…
}
double pop(void) {
…
}
```

Eg: header_file

4.6 Static variables（静态变量）

- *Stored in the data segment*
- *Exist across the whole run of the program*
- *No routines in other files will be able to access static variables*

4.7 Register variables （寄存器变量）

- If variables will be heavily used, define them as register variables

4.8 Block structure（模块结构）

- **Variables can be defined after a left brace**

```
if ( n > 0) {
    int i; /* declare a new I */

    for ( i = 0; i < n; i++)
        ….
}
```

The variable *i* is initialized each time the block is entered

4.9 Initialization (初始化）

- **Without explicit initialization**
  - **External variables**
  - **Static variables**

  are initialized to 0
  - **Other variables have undefined initial values**
- **Explicit initialization**
  - **Character array can be initialized with a string**

    char pattern[] = “ould”

4.10 Recursion （递归）

A function may call itself

Eg：recursion

```
1.  /* qsort sort v[left]...v[right] into increasing order */
2.  void qsort (int v[], int left, int right) {
3.      int i, last;
4.      void swap ( int v[], int i, int j);
5.      if (left >= right) /* do nothing if array contains */
6.          return; /* fewer than two elements */
7.      swap(v, left, (left + right)/2); /* move partition element to v[0] */
8.      last = left;
9.      for ( i = left + 1; i <= right; i++) /* partition */
10.         if (v[i] < v[left]) swap(v, ++last, i);
11.     swap(v, left, last); /* restore partition elem */
12.     qsort ( v, left, last - 1);
13.     qsort (v, last +1, right);
14. }
```

4.11 The C preprocessor

- **The first step in compilation**
  - **Most frequently used features**
  - **#include**
    - ***#include <file>***

    or
    - *#include "file"*
  - **#define**
    - ***#define name replacement_text***
- **•Conditional compilation**
- **Macros with arguments**
  - **#define max(A, B) ((A)>(B) ? (A): (B))**

4.11 The C preprocessor

- **Conditional compilation**

```
#ifndef HDR
#define HDR

/* contents of hdr.h go here */

#endif
```

4.11 The C preprocessor

- **Conditional compilation**

```
#if SYSTEM == SYSV
        #define HDR "sysv.h"
#elif SYSTEM == BSD
        #define HDR "bsd.h"
#elif SYSTEM == MSDOS
        #define HDR "msdos.h"
#else
        #define HDR "default.h"
#endif
#include HDR
```