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 7-8)
  - 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)

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Chapter 2. Types, operators and expressions

Chaochun Wei

Shanghai Jiao Tong University

Spring 2019

- **Variable （变量）**
  - **int i, a = 0;**
- **Assignment （赋值）**
  - **i = 1; i= a;**
  - **i == 1; /* this is not an assignment */**
- **Expression （表达式）**
  - **Contains variables and operators**
  - **Example:**
    - i =(i + 2)* 5;

2.1 Variable names（变量名）

- The first character must be a letter
  - “_” is a letter
  - A library function often starts with “_”
- Upper case and lower case are distinct
  - Lower case for variable names
  - Upper case for symbolic constant
- Keywords（关键词）can NOT be used for variable names
  - if, else, int, float, …
- Make the variable names meaningful

2.1 Variable names

```
1.  /* Here is a program converting Fahrenheit temperatures to their Centigrade or
    Celsius equivalents. */
2.  //================================================
3.  #include <stdio.h>
4.  /* print Fahrenheit - Celsius table  for fahr = 0, 20 , ..., 300; */
5.  main() {
6.    float fahr, celsius;
7.    int lower, upper, step;
8.    lower = 0;
9.    upper = 200;
10.   step = 20;
11.   fahr = lower;
12.   while(fahr <= upper) {
13.     celsius = (5.0/9.0)* (fahr - 32.0);
14.     printf("%3.0f %6.1f\n", fahr, celsius);
15.     fahr = fahr + step;
16.   }
17. }
```

See example code
chpt2.1_variable_name.c

2.2 Data types and sizes

- **Basic data types**
  - char one character, a single byte
  - int integer
  - float single-precision floating point
  - double double-precision floating point
- **Qualifiers of basic types:**
  - short
  - long
  - signed
  - unsigned
  - const

2.2 Data types and sizes

chpt2.2_type_size.c

Type	Size （bytes)
int	4
char	1
float	4
double	8
short	2
long	8
long double	16

2.3 Constant （常数）

- **Integer constant**
  - **1234 , 123456789L, 123456789UL**

Chpt2.3_const_int.c

- **Character constant**
  - **‘x’, ‘A’, ‘Z’**

Chpt2.3_const_char.c

- **Constant expression**
  - **31+28+31**
- **String constant**
  - **“Hello world”**
- **Enumeration constant**
  - **enum months {JAN=1, FEB, MAR, APR, MAY, JUN}**

2.4 Declarations （声明）

- **All variables must be declared before use**
- **A declaration**
  - **specify a type**
  - **contains a list of variables of that type**
    - int lower=0, upper, step;
    - char c, line[1000];

2.5 Arithmetic operators

- +, -, *, / and %

  x % y

- Precedence (优先级）

  - ++, -- > *, /, % > +, -

2.6 Relational and logical operators

chpt2.6_precedence.c

- **Relational operators**
  - **> >= < <=**
  - **== !=**
  - **Precedence**
    - =<==, != < >, >=, <, <=<+, -, *, /, %
- **Logical operators**
  - **&& ||**
    - The order of evaluation

2.7 type conversion

chpt2.7_type_convert.c

- **Without information loss: “narrower” to “wider”**

```
1. float a = 1.2, f;
2. int i = 2;
3. f = a + i;   /* this is ok */
4. i = a;       /* this will lose information */
```

2.9 Bitwise operators

- **Bitwise operators**
  - & bitwise AND
  - | bitwise inclusive OR
  - ^ bitwise exclusive OR
  - << left shift
  - >> right shift
  - ~ one's complement (unary)
- **Examples**
  - N=~0 ➔ N= ffffffff 37777777777

2.9 Bitwise operators

```
1. /* getbits: get n bits from position p of x*/
2. unsigned getbits(unsigned x, int p, int n)
3. {
4.         return (x >> (p+1-n)) & ~(~0 <<n);
5. }
```

getbits(511, 7, 3) ➔ 111

f	e	d	c	b	a	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	511
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	511>>(7+1-3)
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	(511>>(7+1-3)) & ~(~0<<3)
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	~(~0<<3)
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	~0

2.10 Assignment operator (op=)

- op=, where op is one of

  + - * / % << >> & ^ |

Example: bitcount

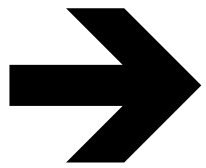
```
1. /* bit count: count 1 bits in x */
2. int bitcount (unsigned x) {
3.       int b;
4.       for (b = 0; x!=0; x>>= 1) {
5.             if (x & 01) b++;
6.       }
7.       return b;
8. }
```

2.11 Conditional Expressions

Chpt2.11_condexp.c

/* z = max (a, b) */

1. If (a > b)
2. z = a
3. else
4. z = b



z = (a>b) ? a:b

Basic Types and Operators

- **Basic data types**
  - Types: ***char, int, float and double***
  - Qualifiers: ***short, long, unsigned, signed, const***
- **Constant: 0x1234, 12, "Some string"**
- **Enumeration:**
  - Names in different enumerations must be distinct
  - ```
    enum WeekDay_t {Mon, Tue, Wed, Thur, Fri};
    enum WeekendDay_t {Sat = 0, Sun = 4};
    ```
- **Arithmetic: +, -, *, /, %**
  - prefix ++i or --i ; increment/decrement before value is used
  - postfix i++, i--; increment/decrement after value is used
- **Relational and logical: <, >, <=, >=, ==, !=, &&, ||**
- **Bitwise: &, |, ^ (xor), <<, >>, ~(ones complement)**
    ```

2.12 Precedence and associativity of operators

Operators	Associativity
( ) [ ] -> .	Left to right
! ~ ++ -- + - * & (type) sizeof	Right to left
* / %	Left to right
+ -	Left to right
<< >>	Left to right
< <= > >=	Left to right
== !=	Left to right
&	Left to right
^	Left to right
|	Left to right
&&	Left to right
||	Left to right
?:	Right to left
= += -= *= /= %= &= ^= |= <<= >>=	Right to left
,	Left to right