上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

Course organization

- Course introduction (Week 1)
  - Code editor: Emacs
- Part I: Introduction to C programming language (Week 1 - 12)
  - Chapter 1: Overall Introduction (Week 1-4)
    - C
    - Unix/Linux
  - Chapter 2: Types, operators and expressions (Week 4)
  - **Chapter 3: Control flow (Week 5)**
  - 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- 14)
  - Debugging tools（Week 12-13）
  - Keeping projects documented and manageable （Week 14）
  - Source code managing （Week 14）
- Part III: Reports from the battle field (student forum) (Week 15 – 16)

Chapter 3. Control Flow

Chaochun Wei
Shanghai Jiao Tong University
Spring 2014

3.1 Statement and Blocks

- Statement
  - Expression ;
- Block

```
{
 }
/* no ; after } */
```

3.2 if-else

- Syntax:

If (expression)

statement_1

else

statement_2

```
If( n > 0)
    If  (a > b)
        z = a;
    else
        z = b;
```

```
If( n > 0) {
    If  (a > b)
        z = a;
}
else
        z = b;
```

Note:

1. The else part is optional
2. The else is associated with the closest else-less if.

3.3 else-if

1. *If (expression)*
2. *statement*
3. *else if (expression)*
4. *statement*
5. *else if (expression)*
6. *statement*
7. *else*
8. *statement*

```
/* binsearch: find if x is included in an
array v[].
    return the index of x in v[] if yes   */
int binsearch (int x, int v[], int n) {
  int low, high, mid;
  low = 0;
  high = n - 1;
  while (low <= high) {
    mid = (low + high) / 2;
    if (x < v[mid])
      high = mid - 1;
    else if ( x > v[mid])
      low = mid +1;
    else /* found match */
      {
     return mid; }
  }
   return -1; /* no match */
}
```

3.4 switch

- Syntax

```
Switch (expression) {
    case const-expr: statements
    case const-expr: statements
    default: statements
}
```

3.4 switch

```
1.  #include <stdio.h>

2.  main() { /*count digits, white space, and others */
3.     int c, i, nwhite, nother, ndigit[10];
4.     nwhite = nother = 0;
5.     for (i = 0; i < 10; i ++) {
6.          ndigit[i] = 0;
7.      }
8.      while (( c = getchar ()) != EOF) {
9.          switch(c) {
10.             case '0': case '1': case '2': case '3': case '4':
11.             case '5': case '6': case '7': case '8': case '9':
12.                     ndigit[c-'0'] ++;
13.                     break;
14.              case ' ': case '\n': case '\t':
15.                     nwhite ++;
16.                     break ;
17.              default:
18.                     nother ++;
19.                     break;
20.         }
21.     }
22.      printf("digits = ");
23.      for (i = 0; i < 10; i ++) {
24.           printf (" %d", ndigit[i]);
25.      }
26.      printf(" , white space  = %d, other = %d\n", nwhite, nother);
27.   return 0;
28. }
```

Break: force an immediate exit from ***switch***, ***while***, ***for*** and ***do*** loops.

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

Course organization

- Course introduction (Week 1)
  - Code editor: Emacs
- Part I: Introduction to C programming language (Week 1 - 12)
  - Chapter 1: Overall Introduction (Week 1-4)
    - C
    - Unix/Linux
  - Chapter 2: Types, operators and expressions (Week 4)
  - **Chapter 3: Control flow (Week 5)**
  - 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- 14)
  - Debugging tools（Week 12-13）
  - Keeping projects documented and manageable （Week 14）
  - Source code managing （Week 14）
- Part III: Reports from the battle field (student forum) (Week 15 – 16)

3.5 loops: *while* and *for*

- Syntax:

while (expression)

statement

for(expr1; expr2; expr3)

statement

equivalent to

```
while(expr2) {
  statement
  expr3;
}
```

3.5 *loops: while* and *for*

```
while ((c = getchar()) == ‘ ‘ || c == ‘\n’ || c == ‘\t’)
        ;  /* skip white space characters */
```

```
for ( i = 0; i < n; i ++)
…
```

3.5 loop: while and for

```
1. int main(){
2.   int c, i = 0;
3.   char s[100];
4.   printf("Please input a number: \t");
5.   ;
6.   while ( c = getchar()  != '\n') {
7.     s[i] = c;
8.     i ++;
9.   }
10.  s[i]= '\0';
11.  printf (" \n The input number is %d\n",  atoi(s));
12.}
```

What's wrong with the while block?

3.5 loops: while and for

```
1. /* reverse: reverse string s in place */
2. void reverse(char s[]) {
3.       int c, i, j;
4.       for ( i = 0, j = strlen(s) - 1; i < j; i ++, j--)
5.           c = s[i], s[i]=s[j], s[j]=c;
6.  }
```

3.6 Loops: Do-while

- Syntax:

```
do
    statement
while (expression);
```

3.7 Break and Continue

- Break:
  - Exit the innermost enclosing loop
- Continue
  - Start the next iteration of a loop