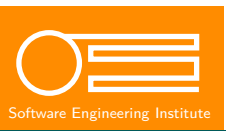
Control Flows

Lecture 04

Min Zhang

zhangmin@sei.ecnu.edu.cn

2020.10.19



10:00-11:40, Monday, Room 319

Software Engineering Institute, East China Normal University

Review

Last lecture:

1. Data representation in computer
2. Types
3. Constants
4. Variables
5. Operators
6. Type conversion
7. Assignment operator
8. ++,−−
9. Privilege of operators
10. Bit operators
11. Conditional expression

control flow

Expression and statement

Expression

An expression is one of the following form:

- a constant;
- a variable;
- an operator with its arguments.

Statement

A statement is an expression followed by `;`.

Quiz

Expression or statement?

- `x+y` — expression
- `x=y` — expression
- `x=y+z` — expression
- `scanf("%d",&y)` — expression
- `x=scanf("%d",&y)` — expression
- `x=scanf("%d",&y);` — statement
- `;` — statement
- `x+y;` — statement

Difference between expression and statement

Every expression has a value.

Statements do not have value.

BAD: `x=(y+z;)` ×

Block

Definition (Block)

A block is a sequence of statements in { and }.

```
{
  x=1;
  y=x+1;
}
```

```
{
}
```

conditional statement: `if`

Syntax of `if`

```
1 if( expression ) a statement or a block
```

Semantics of `if`

- if `expression`'s value is 0, skip the statement or the block
- if `expression`'s value is not 0, execute the statement or the block

Example

Calculate an integer a's absolute value.

```
1 if(a<0)
2   a*=-1;
```

Some quiz

Are they valid statements?

- `if(a<0)a=0;` Yes
- `if(a<0){a=0};` No
- `if(a<0;)a=0;` No
- `if(a==0)a+1;` Yes
- `if(a==0);` Yes
- `if(a=0)a=1;` Yes
- `if(a<0)a=1;a++;` Yes
- `if(a<0)a=1;{a++;}` Yes
- `if(a<0){a=1;a++;}` Yes

Warning

Example

```
int a=-1;
char s='+';
if(a<0)
  a*=-1;
  s='-';
```

The value of a and s is ? 1 and '-'

Example

```
int a=1;
char s='+';
if(a<0)
  a*=-1;
  s='-';
```

The value of a and s is ? 1 and '-'

The right way

Example

```
int a=-1;
char s='+';
if(a<0){
  a*=-1;
  s='-';
}
```

Always use { and } in your if statement.

if-else statement

Syntax of if-else statement

```
if( expression )
   a statement1 or a block1
else
   a statement2 or a block2
```

Semantics of if-else statement

- If expression is not 0, then execute a statement1 or a block1
- Otherwise, execute a statement2 or a block2

See which is valid

```
1 if(a<0)
2   a*=-1;
3   s='-';
4 else
5   s='+';
```

Invalid

```
1 if(a<0){
2   a*=-1;
3   s='-';
4 }else
5   s='+';
```

Valid

```
1 if(a>0)
2   s='+';
3 else
4   a*=-1;
5   s='-';
```

Valid but not correct

```
1 if(a>0)
2   s='+';
3 else{
4   a*=-1;
5   s='-';
6 }
```

Valid and correct

if-else-if-else-... statement

Remember that: if statement and if-else statement are just statements.

```
1  if(score>=90)
2    printf("Excellent\n");
3  else
4    if(score>=80)
5      printf("Good\n");
6    else
7      if(score>=70)
8        printf("Fine\n");
9      else
10       if(score>=60)
11         printf("Pass\n");
12       else
13         printf("FAIL!\n");
```

Valid but BAD style

```
1  if(score>=90){
2    printf("Excellent\n");
3  }else{
4    if(score>=80){
5      printf("Good\n");
6    }else{
7      if(score>=70){
8        printf("Fine\n");
9      }else{
10       if(score>=60){
11         printf("Pass\n");
12       }else{
13         printf("FAIL!\n");
14       }
15     }
16   }
17 }
```

Valid and good style

Switch statement

Syntax of switch

```
switch(expression){
  case constant-expression1 : statement(s)1
  case constant-expression2 : statement(s)2
  ...
  case constant-expressionn : statement(s)n
  default: statement(s)
}
```

Semantics of switch

From constant-expression1 to constant-expressionn, when the value of expression is equal to one of them, then execute the corresponding statement(s) and the following ones until it meets a break; statement.

计算 expression 的值，并从第一个 case 逐个判断是否与其常量表达式相等。若相等，则执行其对应的所有语句以及后续的语句直到遇到 break; 语句。

An example

Print the rank of students score.

```
if(score>=90){
  printf("Excellent\n");
}else{
  if(score>=80){
    printf("Good\n");
  }else{
    if(score>=70){
      printf("Fine\n");
    }else{
      if(score>=60){
        printf("Pass\n");
      }else{
        printf("FAIL!\n");
      }
    }
  }
}
```

```
switch(score/10){
  case 10: printf("Excellent\n"); break;
  case 9: printf("Excellent\n"); break;
  case 8: printf("Good\n"); break;
  case 7: printf("Fine\n"); break;
  case 6: printf("Pass\n"); break;
  default: printf("FAIL!\n");
}
```

OR

```
switch(score/10){
case 10: // no statements here!!!
case 9: printf("Excellent\n"); break;
case 8: printf("Good\n"); break;
case 7: printf("Fine\n"); break;
case 6: printf("Pass\n"); break;
default: printf("FAIL!\n");
}
```

Remember: Do

while statement

Syntax of while

```
1 while(expression)
2   a statement or a block
```

Exactly the same as if statement, very simple.

Semantics of while

1. evaluate expression
2. if it is not 0
   1. execute a statement or a block
   2. go to 1
3. otherwise, done.

Remark:

- If statement break; in the block is executed, exit the loop
- If statement continue; in the block is executed, go to 1

Calculate the sum from 1 to 100

```c
int i=0,sum=0;
while(i<=100){
  sum+=i;
  i++;
}
```

Code 1

```c
int i=0,sum=0;
while(i<=100){
  sum+=i++;
}
```

Code 2

```c
int i=0,sum=0;
while(i++<100){
  sum+=i;
}
```

Code 3

```c
int i=0,sum=0;
while(sum+=i++,i<=100);}
```

Code 4

```c
int i=0,sum=0;
while(sum+=i,i++<100);}
```

Code 5

Remark: ; in the `while` statement of Code 4 and 5 is a statement.

Code like 4 and 5 is not recommended. Do not use too complex expression as condition in `while` statement, The simpler, the better.

Summary

Control flow:

1. Expression & statement
2. Block
3. `if` statement
4. `switch` statement
5. `while` statement
6. `break` statement
7. `continue` statement