SUSTech Southern University of Science and Technology

CS111，C Programming

Lab / Intro & Basic

黄嘉炜

huangjw3@mail.sustech.edu.cn

深港微电子学院
SCHOOL OF MICROELECTRONICS

Outline

SUSTech Southern University of Science and Technology

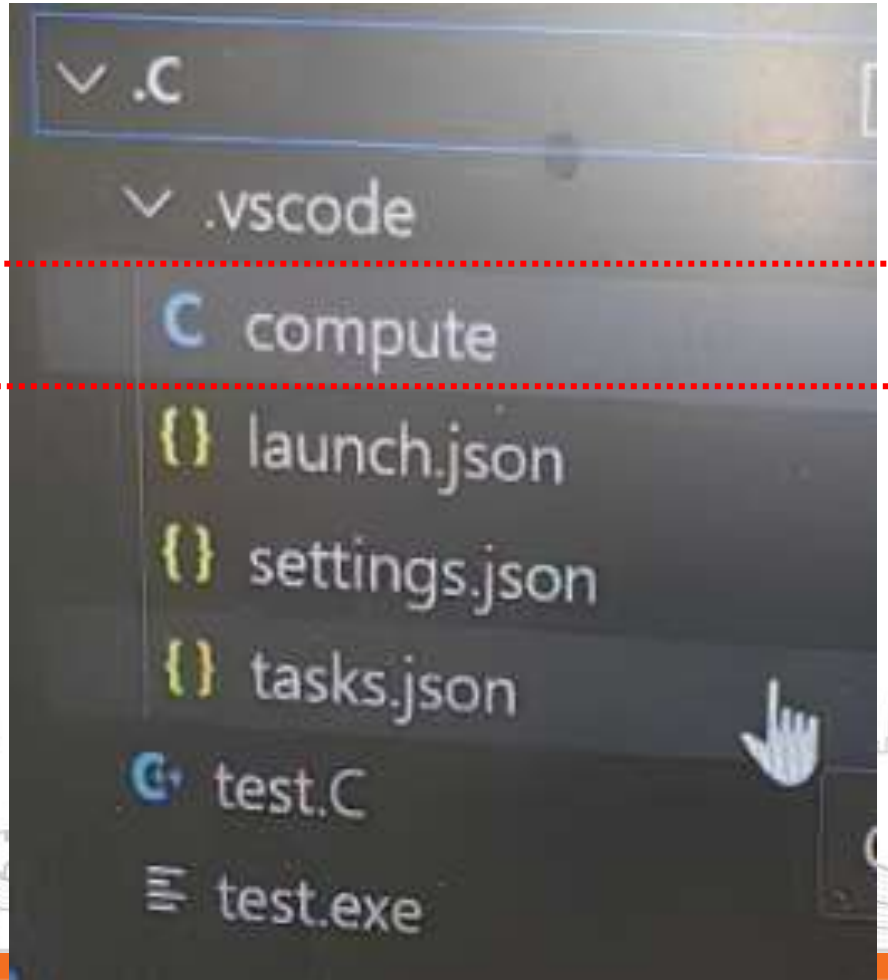
- Review
- Operators：More，Precedence(优先级)
- Conditional statement：Preliminary, if … else if … else …
- Assignment

Review： Problem

Code organization

SUSTech Southern University of Science and Technology

错误做法：子目录 .vscode 下添加代码



Review： Problem

Find syntax bug ?

SUSTech Southern University of Science and Technology

```c
dadaw.c > main()
1   #include <stdio.h>
2   #include <stdlib.h>
3   int main()
4   {
5       float a=100
6       float b=200
7       printf("%f",a/b)
8       system("pause");
9       return 0;
10  }
```

问题 2 输出 调试控制台 终端 端口

正在启动生成...

E:\mingw64\bin\gcc.exe -fdiagnostics-color=always -g "C:\User[redacted]\Desktop

C:\Users\[redacted]esktop\csc program\dadaw.c: In function 'main':

C:\Users\[redacted]ktop\csc program\dadaw.c:6:5: error: expected ',' or ';' b

float b=200

^~~~~

生成已完成，但出现错误。

Review： Problem

Find logical bug ?

SUSTech Southern University of Science and Technology

```
4  int main() {
5      int a;
6      int b;
7      int c = a + b;
8      scanf("%d", &a);
9      scanf("%d", &b);
10     printf("a+b=%d \n", c);
11     return 0;
12 }
```

Review：Variable initial value

SUSTech Southern University of Science and Technology

内部运行

- 申请一块内存空间：32 bits
- 给这块空间一个名称: a

可理解为 **32 个**盒子，
每个盒子只能放：0 或 1

int a; 声明语句，创建变量时用且仅用一次

a = 10; 赋值语句，改变变量时使用

c语言声明的变量初始值是多少

在C语言中，变量的初始值取决于变量的存储类别：

1. **全局变量** 和 **静态存储持续性变量**（包括文件作用域的静态变量和函数内部的静态局部变量）：
   - 默认初始值都是 `0`（对于数字类型为整型或浮点型的变量是0，对于字符型变量是 `\0`，对于指针则是NULL）。这些变量位于数据段，会在程序启动时自动初始化为零。
2. **局部变量**（包括函数内的自动变量）：
   - 若未初始化，它们不会被赋予任何特定的初始值，其值是不确定的，可能是任意的随机值，这是因为局部变量存储在栈上，这部分内存先前的内容并未清除，因此新声明的局部变量可能会保留上次使用这块内存时遗留的数据。

为了确保程序的正确性，强烈建议程序员在声明变量时显式地为其赋予初始值，尤其是在声明局部变量时。

Review: puts vs printf

puts:
- 简洁，
- 使用常见：简单输出一行文本且不需要任何格式转换

printf:
- 强大，但稍复杂
- 可混合输出不同类型的数据
- 可对输出样式有特定要求

c语言中，puts 和 printf 区别

C语言中的 `puts()` 和 `printf()` 函数都是用于向标准输出设备（通常是屏幕）输出信息，但它们之间存在明显的区别：

1. **参数处理**:
   - `puts()` 函数接收一个指向字符串的指针作为参数，并在显示字符串后自动添加一个换行符 `\n` 。
   - `printf()` 函数则更加灵活，它可以接收可变数量的参数，并且第一个参数是一个格式化字符串，其中可以包含格式说明符，如 `%d` 用于整数、 `%f` 用于浮点数等。后面的参数将根据格式说明符被正确地格式化并输出。
2. **格式化输出**:
   - `puts()` 不支持格式化输出，只能直接输出整个字符串。
   - `printf()` 支持格式化输出，可以根据格式串来控制输出数据的格式、宽度、精度等。

Outline

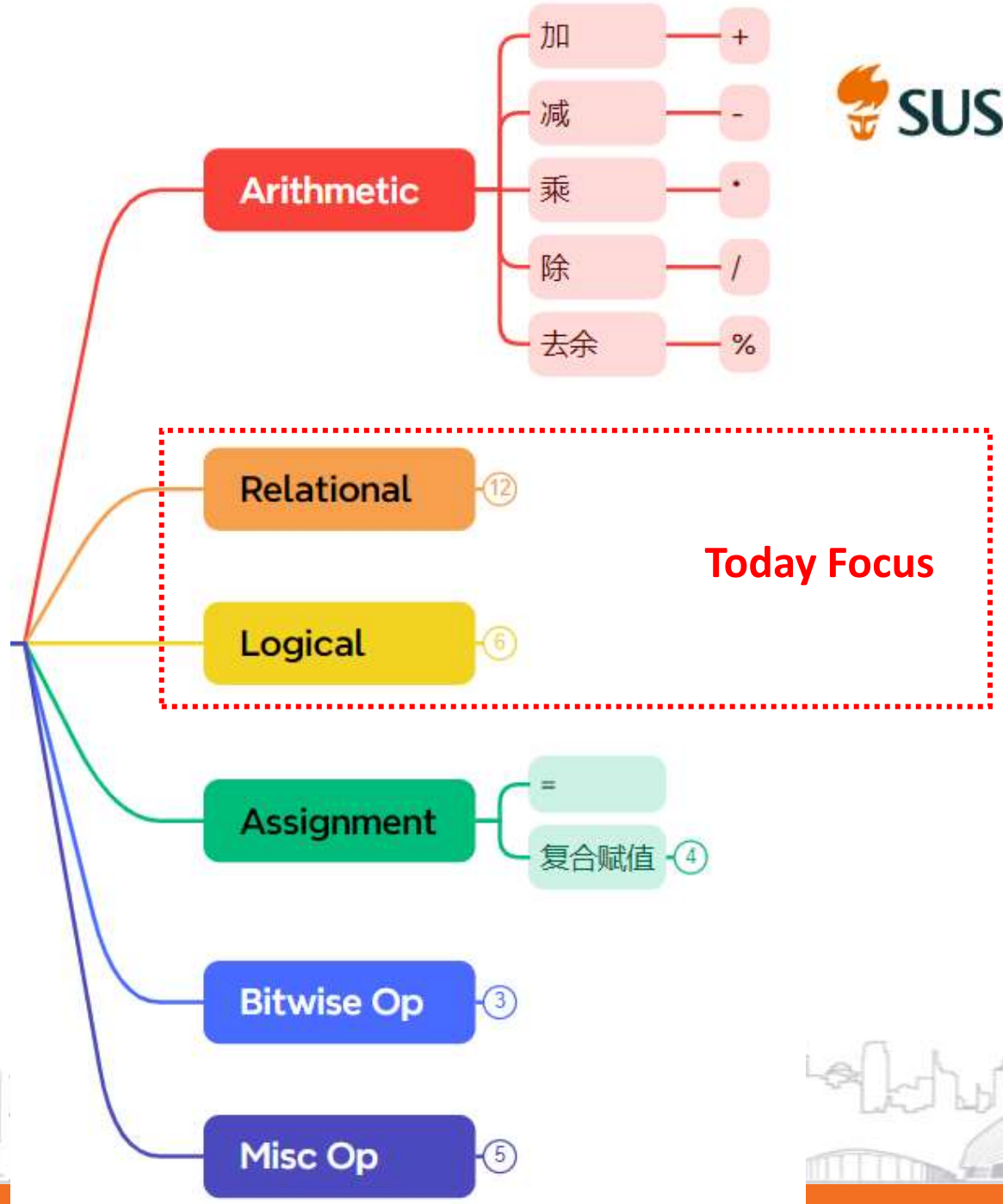
SUSTech Southern University of Science and Technology

- Review
- **Operators：More，Precedence(优先级)**
- Conditional statement：Preliminary, if … else if … else …
- Assignment

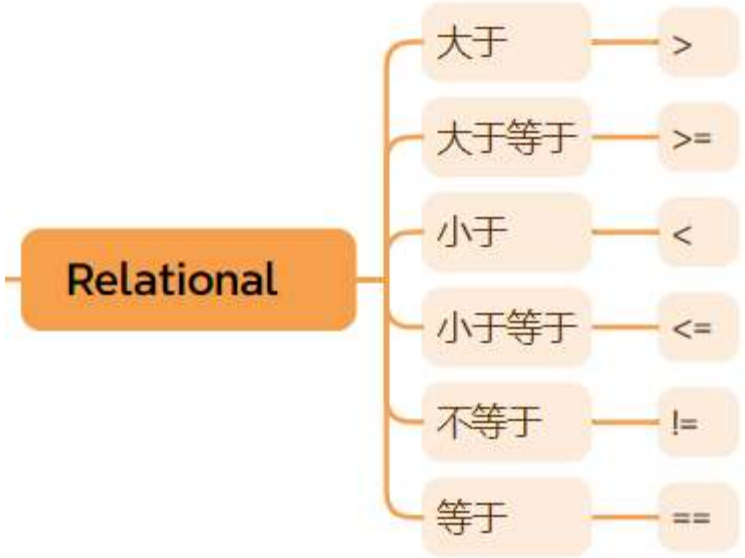
Operator： Review

SUSTech Southern University of Science and Technology

Whole Picture



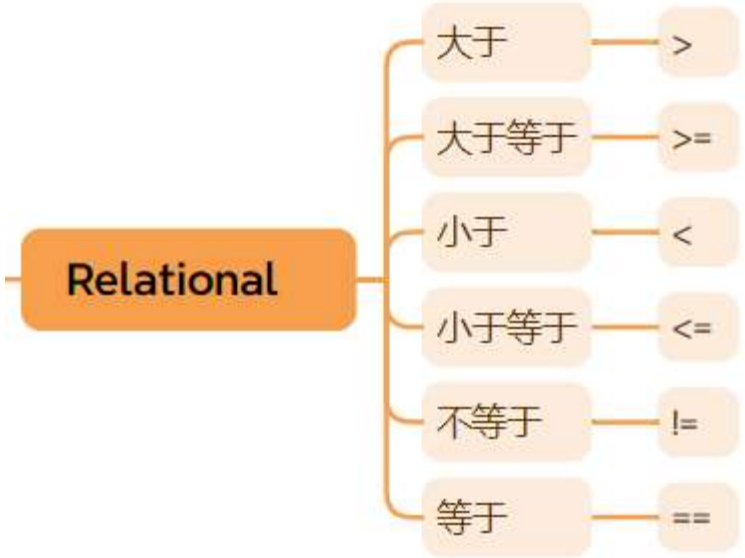
Operator：More




```c
int main()
{
    float a, b;
    scanf("%f %f", &a, &b);

    int ret = a > b;
    printf("a > b, %d\n", ret);

    ret = a >= b;
    printf("a >= b, %d\n", ret);

    ret = a < b;
    printf("a < b, %d\n", ret);

    ret = a <= b;
    printf("a <= b, %d\n", ret);

    ret = a == b;
    printf("a == b, %d\n", ret);

    ret = a != b;
    printf("a != b, %d\n", ret);

    return 0;
}
```

SUSTech Southern University of Science and Technology

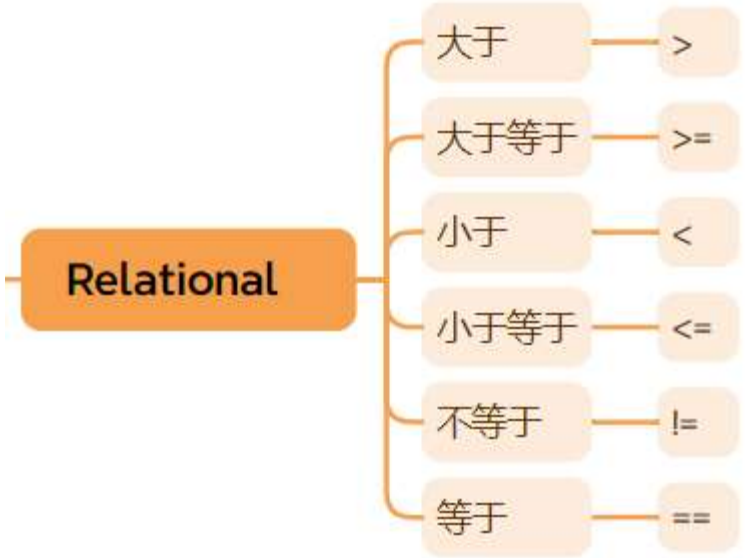
Operator：More




```c
int main()
{
    float a, b;
    scanf("%f %f", &a, &b);

    int ret = a > b;
    printf("a > b, %d\n", ret);

    ret = a >= b;
    printf("a >= b, %d\n", ret);

    ret = a < b;
    printf("a < b, %d\n", ret);

    ret = a <= b;
    printf("a <= b, %d\n", ret);

    ret = a == b;
    printf("a == b, %d\n", ret);

    ret = a != b;
    printf("a != b, %d\n", ret);

    return 0;
}
```

```
100 200
a > b, 0
a >= b, 0
a < b, 1
a <= b, 1
a == b, 0
a != b, 1
```

Operator：More




```c
int main()
{
    float a, b;
    scanf("%f %f", &a, &b);

    int ret = a > b;
    printf("a > b, %d\n", ret);

    ret = a >= b;
    printf("a >= b, %d\n", ret);

    ret = a < b;
    printf("a < b, %d\n", ret);

    ret = a <= b;
    printf("a <= b, %d\n", ret);

    ret = a == b;
    printf("a == b, %d\n", ret);

    ret = a != b;
    printf("a != b, %d\n", ret);

    return 0;
}
```

```
100 100
a > b, 0
a >= b, 1
a < b, 0
a <= b, 1
a == b, 1
a != b, 0
```

Operator：More


```c
int main()
{
    int a, b;
    scanf("%d %d", &a, &b);

    int ret = 0 || 1;
    printf("0 || 1, %d\n", ret);

    ret = (a >= b);
    ret = ret && (a >= 100);
    printf("(a >= b) && (a >= 100), %d\n", ret);

    ret = !( (a < b) || (a < 100) );
    printf("!( (a < b) || (a < 100) ), %d\n", ret);

    ret = a * b < a + b || a >= 100 && a > b;
    printf("a * b < a + b || a >= 100 && a > b, %d\n", ret);

    return 0;
}
```

Operator：More


```
100 50
0 || 1, 1
(a >= b) && (a >= 100), 1
!( (a < b) || (a < 100) ), 1
```

```c
int main()
{
    int a, b;
    scanf("%d %d", &a, &b);

    int ret = 0 || 1;
    printf("0 || 1, %d\n", ret);

    ret = (a >= b);
    ret = ret && (a >= 100);
    printf("(a >= b) && (a >= 100), %d\n", ret);

    ret = !( (a < b) || (a < 100) );
    printf("!( (a < b) || (a < 100) ), %d\n", ret);

    return 0;
}
```

Operator：More


```
100 50
0 || 1, 1
(a >= b) && (a >= 100), 1
!( (a < b) || (a < 100) ), 1
```

```
3   int main()
4   {
5       int a, b;
6       scanf("%d %d", &a, &b);
7
8       int ret = 0 || 1;
9       printf("0 || 1, %d\n", ret);
10
11      ret = (a >= b);
12      ret = ret && (a >= 100);
13      printf("(a >= b) && (a >= 100), %d\n", ret);
14
15      ret = !( (a < b) || (a < 100) );
16      printf("!( (a < b) || (a < 100) ), %d\n", ret);
17
18      ret = a * b < a + b || a >= 100 && a > b;
19      printf("a * b < a + b || a >= 100 && a > b, %d\n", ret);
20
21      return 0;
22  }
```

Operator: Precedence

Focus on Red Firstly

```
a * b < a + b || a >= 100 && a > b;
```

Precedence	Operator	Description	Associativity
1	++ --	Suffix/postfix increment and decrement	Left-to-right
	()	Function call	
	[]	Array subscripting	
	.	Structure and union member access	
	->	Structure and union member access through pointer	
	(*type*){*list*}	Compound literal(C99)	
2	++ --	Prefix increment and decrement[note 1]	Right-to-left
	+ -	Unary plus and minus	
	! ~	Logical NOT and bitwise NOT	
	(*type*)	Cast	
	*	Indirection (dereference)	
	&	Address-of	
	sizeof	Size-of[note 2]	
	_Alignof	Alignment requirement(C11)	
3	* / %	Multiplication, division, and remainder	Left-to-right
4	+ -	Addition and subtraction	
5	<< >>	Bitwise left shift and right shift	
6	< <=	For relational operators < and ≤ respectively	
	> >=	For relational operators > and ≥ respectively	
7	== !=	For relational = and ≠ respectively	
8	&	Bitwise AND	
9	^	Bitwise XOR (exclusive or)	
10	|	Bitwise OR (inclusive or)	
11	&&	Logical AND	
12	||	Logical OR	
13	?:	Ternary conditional[note 3]	Right-to-left
14[note 4]	=	Simple assignment	
	+= -=	Assignment by sum and difference	
	*= /= %=	Assignment by product, quotient, and remainder	
	<<= >>=	Assignment by bitwise left shift and right shift	
	&= ^= |=	Assignment by bitwise AND, XOR, and OR	
15	,	Comma	Left-to-right

Highest

Lowest

Ref,
https://en.cppreference.com/w/c/language/operator_precedence

Operator：Precedence

Tips: 不确定时，用 () 保证计算顺序

=> 额外好处：保证代码可读性

```
((a * b) < (a + b)) || (a >= 100) && (a > b);
```

Precedence	Operator	Description	Associativity
	++ --	Suffix/postfix increment and decrement	Left-to-right
	()	Function call	
1	[]	Array subscripting	
	.	Structure and union member access	
Highest	->	Structure and union member access through pointer	
	(*type*){*list*}	Compound literal(C99)	
2	++ --	Prefix increment and decrement[note 1]	Right-to-left
	+ -	Unary plus and minus	
	! ~	Logical NOT and bitwise NOT	
	(*type*)	Cast	
	*	Indirection (dereference)	
	&	Address-of	
	sizeof	Size-of[note 2]	
	_Alignof	Alignment requirement(C11)	
3	* / %	Multiplication, division, and remainder	Left-to-right
4	+ -	Addition and subtraction	
5	<< >>	Bitwise left shift and right shift	
6	< <=	For relational operators < and ≤ respectively	
	> >=	For relational operators > and ≥ respectively	
7	== !=	For relational = and ≠ respectively	
8	&	Bitwise AND	
9	^	Bitwise XOR (exclusive or)	
10	|	Bitwise OR (inclusive or)	
11	&&	Logical AND	
12	||	Logical OR	
13	?:	Ternary conditional[note 3]	Right-to-left
14[note 4]	=	Simple assignment	
	+= -=	Assignment by sum and difference	
	*= /= %=	Assignment by product, quotient, and remainder	
	<<= >>=	Assignment by bitwise left shift and right shift	
	&= ^= |=	Assignment by bitwise AND, XOR, and OR	
Lowest	15	,	Comma	Left-to-right

Ref,
https://en.cppreference.com/w/c/language/operator_precedence

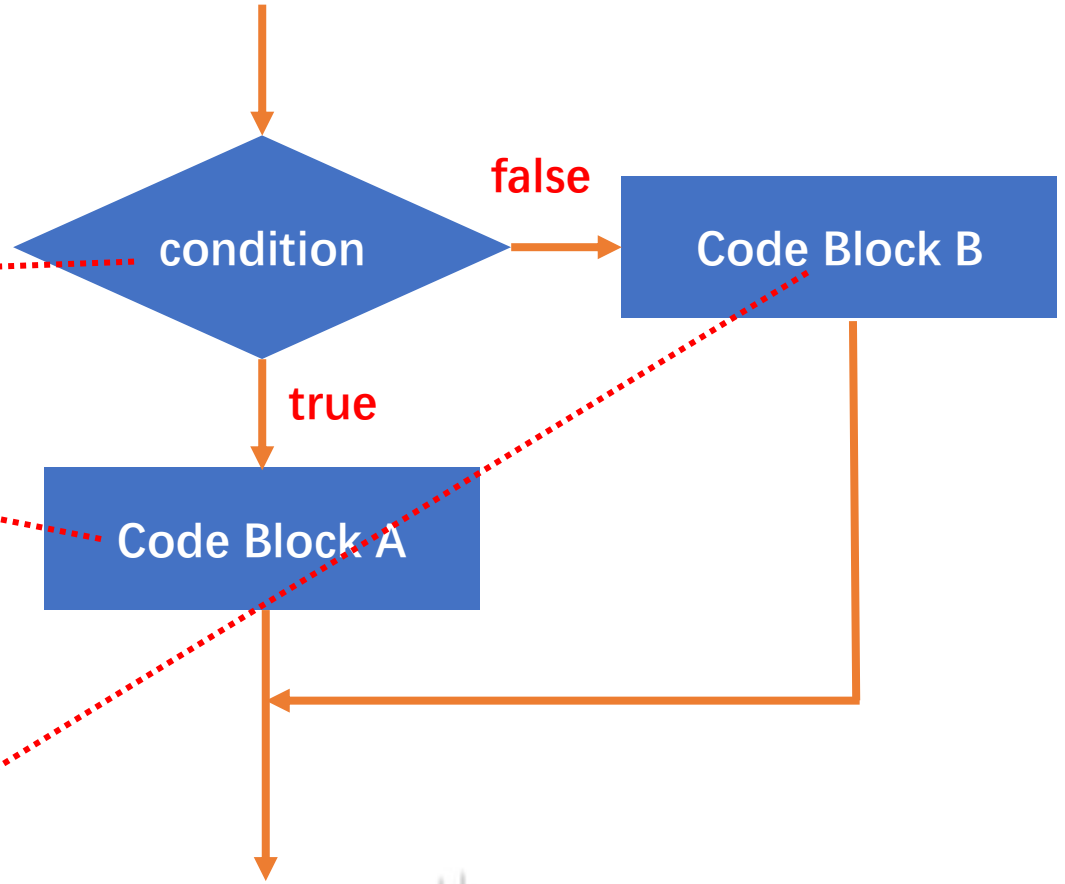
Outline

SUSTech Southern University of Science and Technology

- Review
- Operators：More，Precedence(优先级)
- **Conditional statement：Preliminary, if … else if … else …**
- Assignment

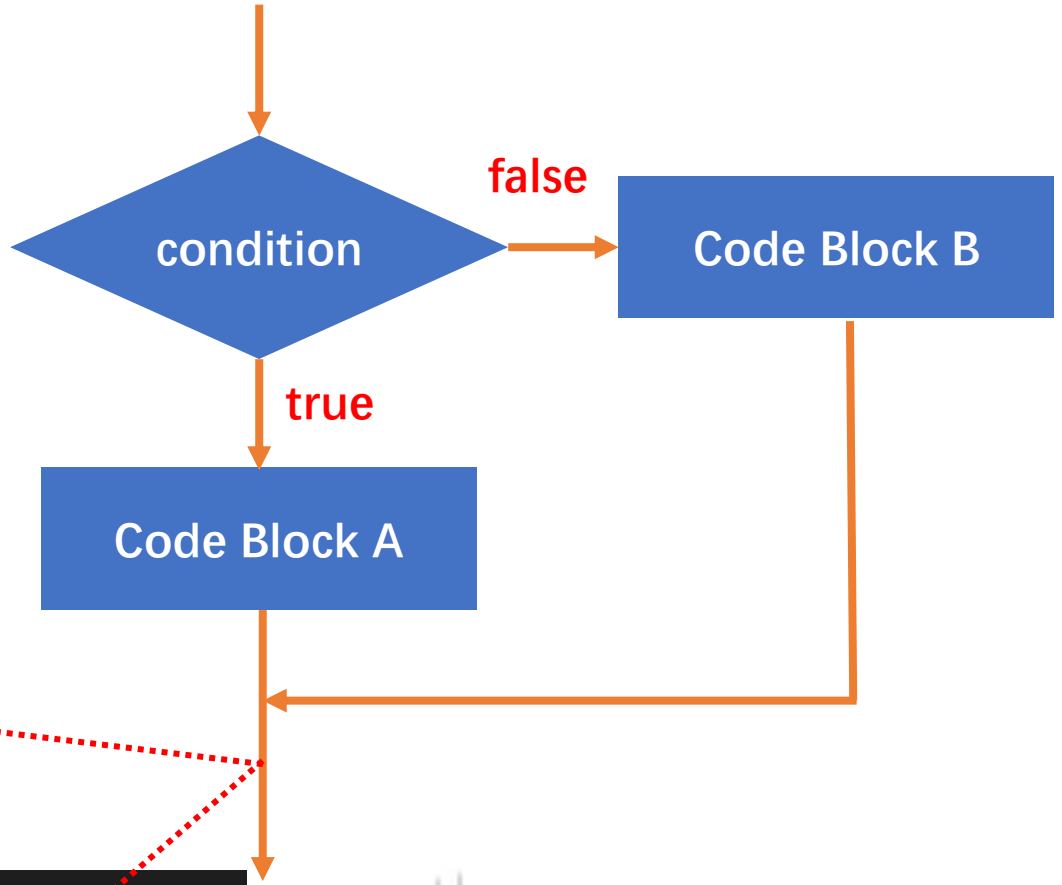
Conditional statement： if () … else …

```c
int main()
{
    int a = 0;
    scanf("%d", &a);

    if (a == 3)
    {
        printf("a == 3\n");
        puts("run A block");
    }
    else
    {
        printf("a != 3\n");
        puts("run B block");
    }
    puts("run after if..else.. block");
    return 0;
}
```



Conditional statement： if () … else …

```c
int main()
{
    int a = 0;
    scanf("%d", &a);

    if (a == 3)
    {
        printf("a == 3\n");
        puts("run A block");
    }
    else
    {
        printf("a != 3\n");
        puts("run B block");
    }
    puts("run after if..else.. block");
    return 0;
}
```




```
3
a == 3
run A block
run after if..else.. block
```

```
4
a != 3
run B block
run after if..else.. block
```

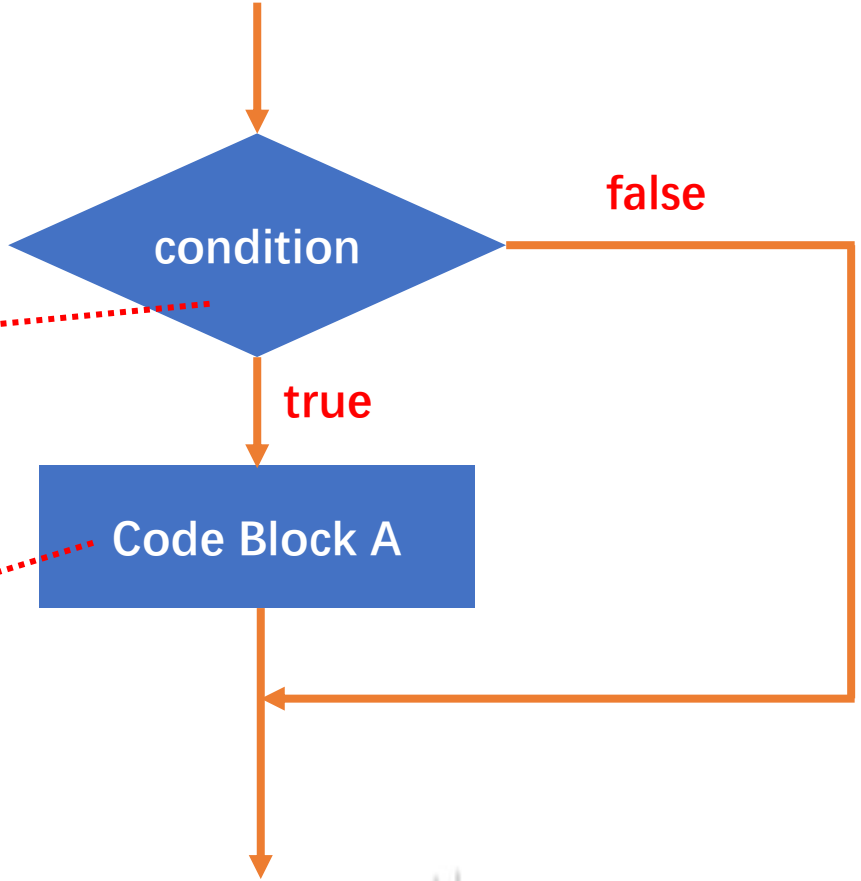
Conditional statement： if () …

```
3   int main()
4   {
5       int a = 0;
6       scanf("%d", &a);
7
8       if (a == 3)
9           printf("a == 3\n");
10          puts("run A block");
11
12      puts("run after if.. block");
13      return 0;
14  }
```

```
4
??
```

Conditional statement： if () ...

```
3   int main()
4   {
5       int a = 0;
6       scanf("%d", &a);
7
8       if (a == 3)
9       {
10          printf("a == 3\n");
11          puts("run A block");
12      }
13
14      puts("run after if.. block");
15      return 0;
16  }
```



Conditional statement： if () …

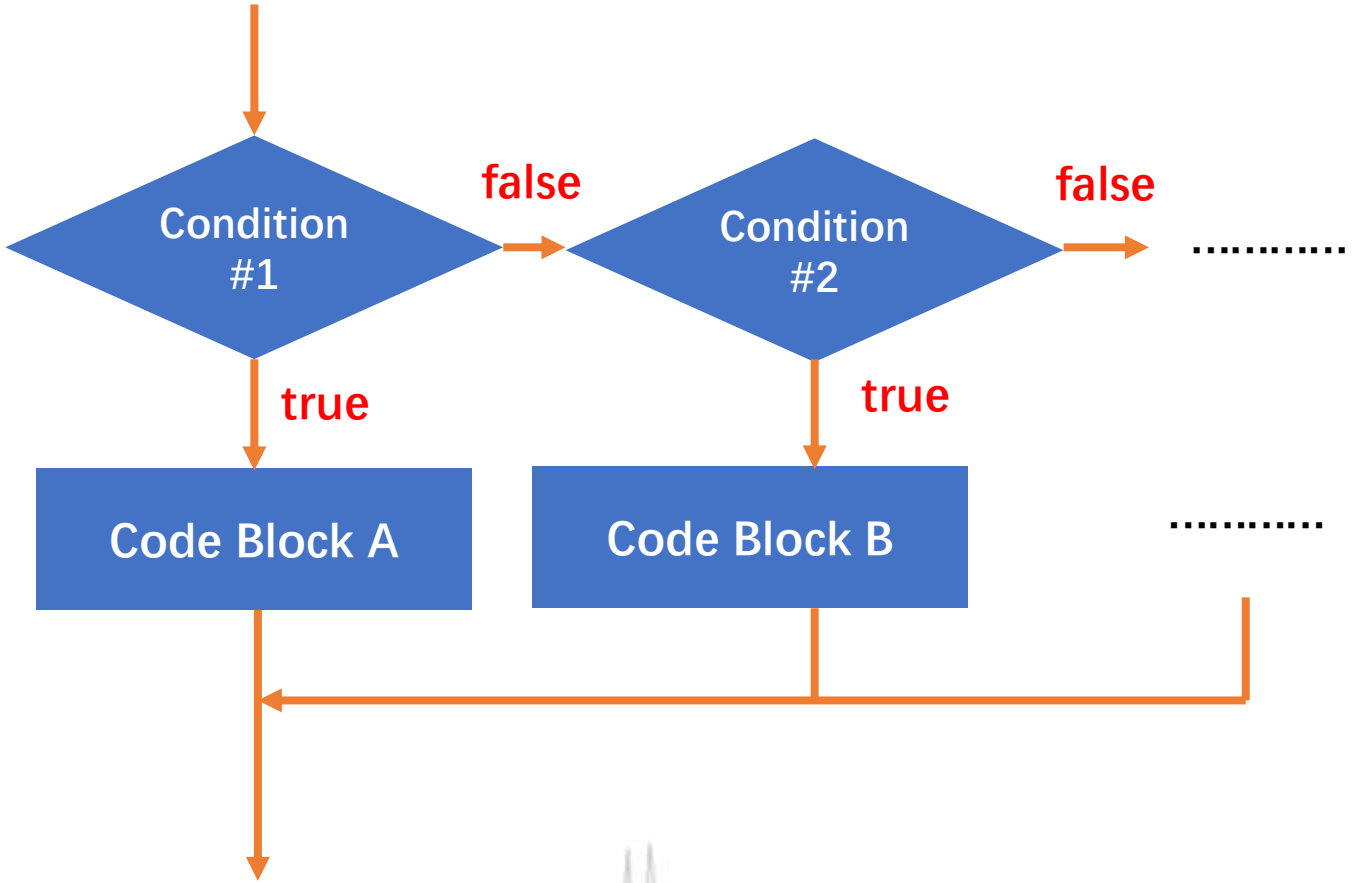
```
3   int main()
4   {
5       int a = 0;
6       scanf("%d", &a);
7
8       if (a == 3) ;
9           printf("a == 3\n");
10          puts("run A block");
11
12      puts("run after if.. block");
13      return 0;
14  }
```

```
4
```

??

Conditional statement: if () ... else if () ... else

```
8       if (a == 13) // Condition #1
9       {
10          puts("Card Number: K"); // Code Block A
11      }
12      else if (a == 12) // Condition #2
13      {
14          puts("Card Number: Q"); // Code Block B
15      }
16      else if (a == 11)
17      {
18          puts("Card Number: J");
19      }
20      else if (a == 1)
21      {
22          puts("Card Number: A");
23      }
24      else if (a > 1 && a <= 10)
25      {
26          printf("Card Number: %d\n", a);
27      }
28      else
29      {
30          puts("Error Card Number!");
31      }
```




```
13
Card Number: K
run after if...else... block
```

```
1
Card Number: A
run after if...else... block
```

Showcase

SUSTech Southern University of Science and Technology

Write a program to determine whether a given year is a leap year (闰年) or not.

Print “YES" if it is a leap year, otherwise print "NO".

Hint: A leap year is divisible by 4 but not by 100 unless it is also divisible by 400.

Outline

SUSTech Southern University of Science and Technology

- Review
- Operators：More，Precedence(优先级)
- Conditional statement：Preliminary, if … else if … else …
- **Assignment**

Assignment 1): Average

SUSTech Southern University of Science and Technology

Write a program

- **input 1 line which contains 3 numbers: 1st is integer (a), 2nd is float (x), 3rd is float (y), and space separation.**
- **print out the result (浮点数，保留2位小数) based on following calculation.**
  - **When a equal to 1, result as: (int)(x + y) / 2**
  - **When a equal to -1, result as: ((int)x + (int)y) / 2**
  - **Otherwise, result as: (x + y) / 2**

```
1 3.8 4.5
4.00
```

```
-1 3.8 4.5
3.00
```

```
0 3.8 4.5
4.15
```

Extended thinking: why there are different results for above 3 calculations?

Assignment 2): GPA Conversion

SUSTech Southern University of Science and Technology

Write a program that takes a score in 100-point scale as input, and outputs the equivalent GPA on a 4.0 scale and corresponding letter grade on a 13-level grading scale.

```
99
A+ 4.00
```

```
101
error
```

```
95
A 3.95
```

```
-1
error
```

Note:
When input score invalid, print "error"

Ref, 《南方科技大学考试工作及成绩管理条例》

第二十五条 平均学分绩点（Grade Point Average，GPA）是衡量学生学习质量的重要指标。

（一）成绩与绩点的换算关系

1. 百分制成绩

绩点= $4-3\times(100-Z)^2\div1600$

Z 是百分制的成绩，Z 不得大于 100 或小于 60。Z 若小于 60，绩点为 0。绩点取四舍五入后保留小数点后两位数字。

2. 十三等级制成绩

等级	A+	A	A-	B+	B	B-	C+	C	C-	D+	D	D-	F
绩点	4.00	3.94	3.85	3.73	3.55	3.32	3.09	2.78	2.42	2.08	1.63	1.15	0
百分参考	97~100	93~96	90~92	87~89	83~86	80~82	77~79	73~76	70~72	67~69	63~66	60~62	<60

Appendix, 一行输入多个数字

SUSTech Southern University of Science and Technology

```
scanf("%d %f %f", &a, &x, &y);
```

```
1 3.8 4.5
4.00
```

Appendix, 浮点数打印保留2位小数

```
printf("A+ %.2f\n", gpa);
```

```
99
A+ 4.00
```

More Ref, https://www.geeksforgeeks.org/printf-in-c/

Appendix, OJ

校内网：http://10.16.27.156/

用户名：学号 （如：12345678）
初始密码：helloclang （首次登陆后尽快修改）

SUSTech Southern University of Science and Technology

THANK YOU