



南方科技大学  
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

# C/C++ Program Design

## CS205

**Prof. Shiqi Yu (于仕琪)**

yusq@sustech.edu.cn

<http://faculty.sustech.edu.cn/yusq/>



南方科技大学  
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

# `if` Statement



# if and if-else

- Statements are executed conditionally

```
int num = 10;  
if (num < 5)  
    cout << "The number is less than 5. " << endl;
```

```
if (num == 5 )  
{  
    cout << "The number is 5." << endl;  
}  
else  
    cout << "The number is not 5." << endl;
```



# if-else if-else

```
if (num < 5)
```

```
    cout << "The number is less than 5." << endl;
```

```
else if (num > 10)
```

```
    cout << "The number is greater than 10." << endl;
```

```
else
```

```
    cout << "The number is in range [5, 10]." << endl;
```



# A little more complex

When will "Where I'm?" be printed?

How to make the code easier to understand?

```
if(num < 10)
if(num < 5)
cout << "The number is less than 5" << endl;
else
cout << "Where I'm?" << endl;
```



# ? : operator

- When can we use the ternary conditional operator?

```
bool isPositive = true;  
int factor = 0;  
//some operations may change isPositive's value  
if(isPositive)  
    factor = 1;  
else  
    factor = -1;
```



```
factor = isPositive ? 1 : -1;
```



```
factor = (isPositive) * 2 - 1;
```



南方科技大学  
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

# Conditions



# Condition

- What should be a condition?

```
int num = 10;  
if (num < 5)  
    cout << "The number is less than 5. " << endl;
```

A large red arrow points from the top right towards the condition `num < 5` in the code snippet. The condition is also enclosed in a red rectangular box.

- The condition should be an expression which is convertible to bool
  - Its value can be `bool`, `char`, `int`, `float`





# Relational Expressions

- The condition can be a relational expression
- The 6 relational/comparison operators

Operator name	Example
equal to	a == b
not equal to	a != b
less than	a < b
greater than	a > b
less than or equal to	a <= b
greater than or equal to	a >= b

- Return **1** if the condition (such as a==b) is true,
- Return **0** if the condition is false.



# Logical Expressions

- If an operand is not `bool`, it will be converted to `bool` implicitly.

Operator name	Symbol-like operator	Keyword-like operator	Example
negation	<code>!</code>	<code>not</code>	<code>!a</code>
AND	<code>&amp;&amp;</code>	<code>and</code>	<code>a &amp;&amp; b</code>
Inclusive OR	<code>  </code>	<code>or</code>	<code>a    b</code>

- Precedence: `!` > `&&` > `||`
- What's the value of the follow expressions?

```
if(-2 && true)
    cout << "The condition is true." << endl;
```

```
if(not -2)
    cout << " (!-2) is true, really?" << endl;
```



# Non-Boolean Expressions

- They will be converted to `bool` implicitly if it is feasible.

```
float count = 0.2f;  
if (count) //not recommend to use a float-point number  
    cout << "There are some." << endl;
```

- Pointers are also frequently used as conditions

```
int * p = new int[1024];  
if (!p) // if(p == NULL)  
    cout << "Memory allocation failed." << endl;
```



南方科技大学  
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

while loop



# while loop

- Syntax :

```
while( expression )  
{  
    //...  
}
```

- If the condition is true, the statement (loop body) will be executed.

while.cpp

```
int num = 10;  
while(num > 0)  
{  
    cout << "num = " << num << endl;  
    num--;  
}
```



# do-while loop

- The test takes place **after** each iteration in a do-while loop.
- The test takes place **before** each iteration in a while loop.

while.cpp

```
int num = 10;  
do  
{  
    cout << "num = " << num << endl;  
    num--;  
}while (num > 0);
```



# break statement

- Terminate a loop

while.cpp

```
int num = 10;
while (num > 0)
{
    if (num == 5)
        break;
    cout << "num = " << num << endl;
    num--;
}
```



# continue statement

- Skip the remaining part of the loop body and continue the next iteration.

while.cpp

```
int num = 10;
while (num > 0)
{
    if (num == 5)
        continue;
    cout << "num = " << num << endl;
    num--;
}
```





# The Condition, Be Careful!


- Can you find any problem from the code?

```
size_t num = 10;  
while(num >= 0)  
{  
    cout << "num = " << num << endl;  
    num--;  
}
```



# The Condition, Be Careful!

```
bool flag = true;
int count = 0;
while(flag = true)
{
    cout << "count = " << count++ << endl;
    // and do sth
    if (count == 10) //meet a condition
        flag = false; //set flag to false to break the loop
}
```

A large red arrow points from the top right towards the condition 'flag = true' in the while loop, highlighting the assignment operator '=' instead of the equality operator '=='. This is a common mistake in programming that causes an infinite loop.



# Why?

- Expression `3+4` has a value;
  - Expression `a+b` has a value;
  - Expression `(a==b)` has value (true or false);
  - `a=b` is an assignment, also an expression and has a value
- 
- The follow code can be compiled successfully!

```
int b = 0;  
int m = (b = 8);  
cout << "m=" << m << endl;
```



南方科技大学  
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

for loop



# for loop

- Syntax

**for** (init-clause; cond-expression; iteration-expression)  
loop-statement

- Example

for.cpp

```
int sum = 0;
```

```
for(int i = 0; i < 10; i++)
```

```
{
```

```
    sum += i;
```

```
    cout << "Line " << i << endl;
```

```
}
```

```
cout << "sum = " << sum << endl;
```



# for loop VS while loop

```
int sum = 0;  
for(int i = 0; i < 10; i++)  
{  
    sum += i;  
    cout << "Line " << i << endl;  
}
```



```
int sum = 0;  
int i = 0;  
while (i < 10)  
{  
    sum += i;  
    cout << "Line " << i << endl;  
  
    i++;  
}
```





# for loop VS while loop

```
while(num > 0)
{
    cout << "num = " << num << endl;
    num--;
}
```



```
for(; num > 0; )
{
    cout << "num = " << num << endl;
    num--;
}
```



# Endless loop

- Sometimes we need it

```
for(;;)
{
    // some statements
    cout << "endless loop!" << endl;
}
```

```
while(true)
{
    // some statements
    cout << "endless loop!" << endl;
}
```





# break/continue statement

- break and continue statements behavior the same with while loops.



南方科技大学  
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

# goto and switch Statements



# goto Statement

- Jump to the desired location
- An unrecommended statement

goto.cpp

```
float mysquare(float value)
{
    float result = 0.0f;

    if(value >= 1.0f || value <= 0)
    {
        cerr << "The input is out of range." << endl;
        goto EXIT_ERROR;
    }
    result = value * value;
    return result;

EXIT_ERROR:
    //do sth such as closing files here
    return 0.0f;
}
```



# switch Statement

- Execute one of several statements, depending on the value of an expression.
- `break` prevents executing some following statements. **Don't forget break!**
- More similar to `goto`, not `if-else if-else`

switch.cpp

```
switch (input_char)
{
    case 'a':
    case 'A':
        cout << "Move left." << endl;
        break;
    case 'd':
    case 'D':
        cout << "Move right." << endl;
        break;
    default:
        cout << "Undefined key." << endl;
        break;
}
```