

语句

COMP250205： 计算机程序设计

李昊

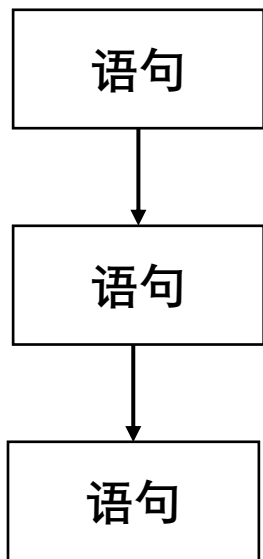
hao.li@xjtu.edu.cn

西安交通大学计算机学院

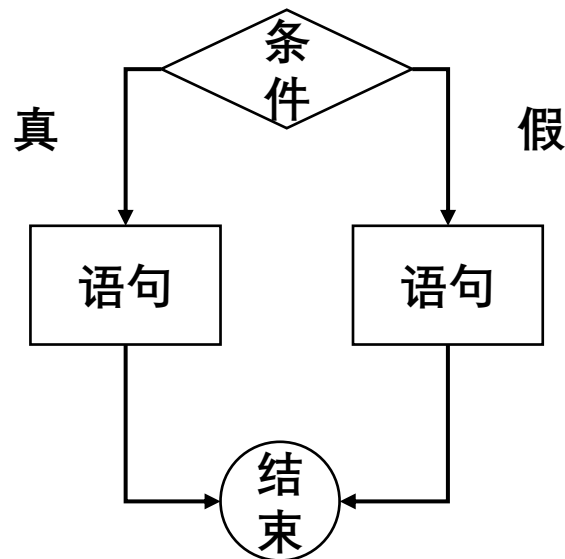
语句 (statement)

代表着一个由处理器执行的动作

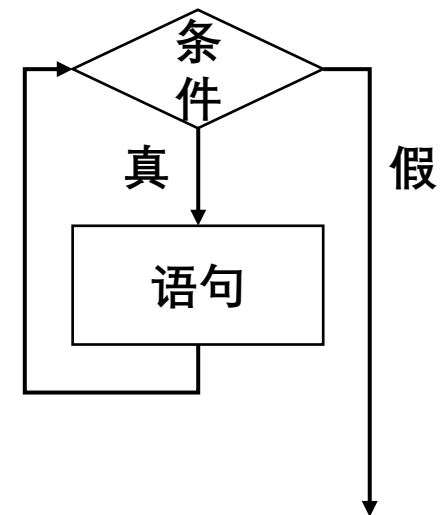
语句按照控制流 (control flow) 执行



顺序



分支



循环

简单语句

表达式语句：表达式后面直接加分号

```
c+1;
```

```
c++;
```

空语句：只包含一个单独的分号

```
;
```

```
c++;;
```

语句块：由花括号括起来的语句，结尾没有分号

```
while (cin << n && n != 42)
    cout << n;
```

while循环的循环体语法上只接受1条语句

```
while (cin >> n && n != 42) {
    cout << n;
    cout << n * 100 << endl;
}
```

使用语句块，将多条语句复合为1条语句

简单语句 - QUIZ

分析程序的控制流

```
while (n != 42) ; {  
    cin >> n;  
    cout << n;  
}
```

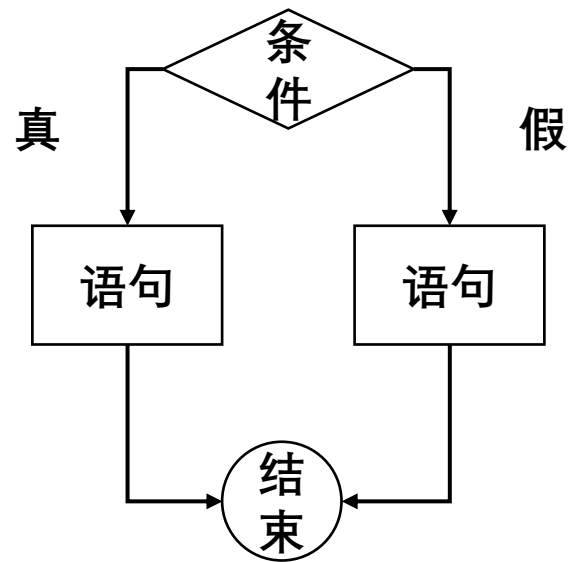
分号代表一个空语句
占据了while循环体

独立的语句块

条件语句

根据条件，进行分支控制流的执行

C++中有两种条件语句：if语句和switch语句



分支

条件语句 - if语句

简单的if语句格式

```
if (condition)  
    statement
```

If语句的分支语法上只支持1条语句，可以用语句块扩展

带有else分支的if语句格式

```
if (condition)  
    statement1  
else  
    statement2
```

else分支同样在语法上只支持一条语句

语句块的陷阱

区分“大于3且小于7的数”，以及“小于等于3的数”

```
if (n > 3)
    if (n < 7)
        cout << "n is larger than 3 and smaller than 7."
else
    cout << "n is not larger than 3."
```



会对大于3且不小于7的n输出!

建议使用语句块来明确结构，即使只有单条语句

```
if (n > 3) {
    if (n < 7) {
        cout << "n is larger than 3 and smaller than 7."
    }
}
else { cout << " n is not larger than 3. " }
```

条件语句 - switch语句

通过整型表达式的值来在多个分支间做选择

```
switch (expression) {  
    case var1:  
        statement1;  
        break;  
    case var2:  
        statement2;  
        break;  
    ...  
    default:  
        statement3;  
        break;  
}
```

一个整型表达式

一个整型值

使用switch语句 - 1

逐一接收键盘输入的字符，统计5个元音字母出现的次数，每次更新，直到输入0为止，输出结果

1. 使用while循环搭配cin并判断输入是否为0
2. 判断是否属于5个元音字母，在不同变量上累加
3. 结束循环后输出所有变量的值

5个变量分别计数

```

unsigned int a = 0, e = 0, i = 0, o = 0, u = 0;
char ch;
while (cin >> ch && ch != '0') {...}
cout << a << ' ' << e << ' ' << i << ' ' << o << ' ' << u;
  
```


使用循环输入ch，并判断ch不为'0'

如何判断ch是否为5个元音字母？

使用switch语句 - 2


逐一接收键盘输入的字符，统计5个元音字母出现的次数，每次更新，直到输入0为止，输出结果

```
while (cin >> ch && ch != '0') {  
    if (ch == 'a') {}  
    else if (ch == 'e') {}  
    else if (ch == 'i') {}  
    else if (ch == 'o') {}  
    else if (ch == 'u') {}  
}
```



嵌套层数过多，不易维护，且运算量大

```
while (cin >> ch && ch != '0') {  
    switch (ch) {  
        case 'a': a++; break;  
        case 'e': e++; break;  
        case 'i': i++; break;  
        case 'o': o++; break;  
        case 'u': u++; break;  
    }  
}
```



无嵌套，易维护，且运算量低

深入switch语句的控制流

switch语句的多个case条件是**同时判断**的

一旦进入一个case块，则会**顺序执行**，直到break

default语句匹配其他所有case

```
while (cin >> ch && ch != '0') {  
    switch (ch) {  
        case 'a': a++;  
        case 'e': e++; break;  
        case 'i': i++; break;  
        case 'o': o++; break;  
        case 'u': u++; break;  
        default: cout << "no vowel";  
    }  
}
```

对于'a'，会执行两个case的语句

如果不匹配任何一个case，则执行default块中的语句

使用switch语句的推荐原则

switch语句利于代码结构维护，且速度快

除非特殊情况，对每一个case都加上break

除非特殊情况，应加上default分支，可以留空

switch语句 - QUIZ

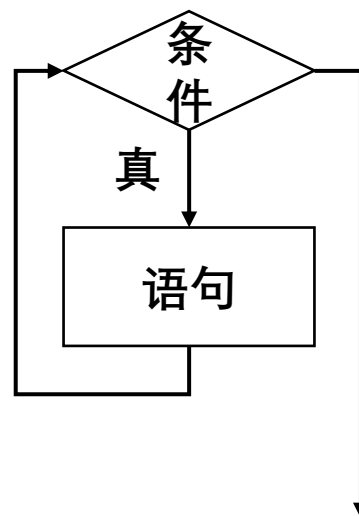
之前的代码中，只能匹配小写字母，请修改程序，使其可以同时匹配大写字母

```
while (cin >> ch && ch != '\0') {  
    switch (ch) {  
        case 'a': a++; break;  
        case 'e': e++; break;  
        case 'i': i++; break;  
        case 'o': o++; break;  
        case 'u': u++; break;  
    }  
}
```

```
while (cin >> ch && ch != '\0') {  
    switch (ch) {  
        case 'A':  
        case 'a': a++; break;  
        case 'E':  
        case 'e': e++; break;  
        ...  
    }  
}
```

迭代语句（循环）

重复执行操作，直到（不）满足某条件才停止



迭代语句 – while和do-while

只要条件为真，while语句就重复循环体

如果条件一开始就为假，则while循环体不执行

```
while (condition)  
    statement
```

do-while语句，首先执行循环体，再检查条件
要以分号结尾

```
do  
    statement  
while (condition) ;
```

迭代语句 – 主动跳出循环

与switch语句一样，break可以跳出当前控制流

```
int n = 0;
while (cin >> n) {
    if (n == 42) break;
    cout << n;
}
```

没有给出明确的停止条件

使用break主动跳出循环

迭代语句 – for语句

```
for (init-statement; condition; expression)  
    statement
```

首次循环前执行init-statement

每次循环前检查condition

每次循环执行statement

每次循环后执行expression

使用for语句

for语句条件中的每一个表达式都可以省略

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

省略初始化语句，要确保循环
条件中的变量可见

```
for (int i = 0; ++i) {  
    if (i >= 10) break;  
}
```

省略条件表达式，则需要主动
跳出循环

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

省略循环后表达式，则需要主
动在循环体内更新条件

for语句 – QUIZ - 1

说明下列循环的意义并改正错误

```
int sz = 10;
int n = 0;
for (int ix = 0; ix != sz; ++ix) {
    cin >> n;
    if (n == ix) break;
    cout << n;
}
if (ix != sz) {
    cout << ix;
}
```

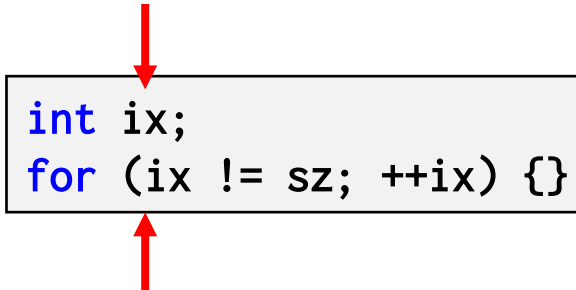
循环外看不到ix变量



for语句 – QUIZ - 2

说明下列循环的意义并改正错误

ix没有初始化，值不确定



```
int ix;  
for (ix != sz; ++ix) {}
```

表达式可以省略，但需要一个空语句，分号不能省


跳转语句

跳出当前的控制流，结束或继续迭代语句

break: 跳出控制流并结束迭代语句


continue: 跳出控制流并继续迭代语句

```
char ch;
while (cin >> ch) {
    if (ch == ' ') break;
    cout << ch;
}
```



跳出并结束循环

```
char ch;
while (cin >> ch) {
    if (ch == ' ') continue;
    cout << ch;
}
```




跳出并进入下一次循环

跳转语句的异类：goto语句

无条件的跳转到同一函数的另一位置

```
int main()
{
    begin:
        int x = 0;
        int y = 10;
        cin >> x;
        if (x > 10) goto end;
        else x = x + 10;
    end:
        cout << x + y << endl;
}
```

```
int main()
{
    begin:
        int x = 0;
        cin >> x;
        if (x > 10) goto end;
        else x = x + 10;
        int y = 10;
    end:
        cout << x + y << endl;
}
```




除非必要，**不要使用goto语句**

跳过了y的初始化，错误

真实问题中的循环：文本输入 - 1

输入字符并计数，直到遇到‘#’为止

```
#include <iostream>
int main()
{
    using namespace std;
    char ch;
    int count = 0;
    cout << "Enter characters; enter # to quit:\n";
    while (cin >> ch && ch != '#') {
        ++count;
    }
    cout << endl << count;
    return 0;
}
```



cin会忽略特殊字符，包括空格、回车

真实问题中的循环：文本输入 - 2

使用cin.get()来获取每一个字符的输入

```
#include <iostream>
int main()
{
    using namespace std;
    char ch;
    int count = 0;
    cout << "Enter characters; enter # to quit:\n";
    while (cin.get(ch) && ch != '#') {
        ++count;
    }
    cout << endl << count;
    return 0;
}
```

cin.get()会忠实的接受每一个字符，包括空格、回车

真实问题中的循环：文本输入 - 3

如果想要接收多行文本输入？

```
#include <iostream>
int main()
{
    using namespace std;
    char ch;
    int count = 0;
    cout << "Enter characters; enter # to quit:\n";
    while (cin.get(ch) && ch != '\n') {
        ++count;
    }
    cout << endl << count;
    return 0;
}
```

仍然只能处理单行

真实问题中的循环： 文本输入 - 4

使用cin.fail()检查文件结尾 (End-Of-File, EOF)

```
#include <iostream>
int main()
{
    using namespace std;
    char ch;
    int count = 0;
    cout << "Enter characters; enter # to quit:\n";
    while (cin.get(ch) && !cin.fail()) {
        ++count;
    }
    cout << endl << count;
    return 0;
}
```

直到手动触发EOF事件

内容总结

简单语句：

以分号结尾；顺序执行；不能随便写分号

条件语句：

if语句：一个表达式，选择两个分支

switch语句：一个整型表达式，选择多个整数值分支

迭代循环语句：

while和do-while语句：检查条件并执行/执行并检查条件

for语句：（初始化表达式；条件表达式；迭代表达式）

跳转语句：break、continue、goto