控制流Flow of Control

动机/ Motivation

- 常规的，程序从第一个到最后一个表达式语句一条一条执行。

```
{ statement1; statement2; statement3; }
```

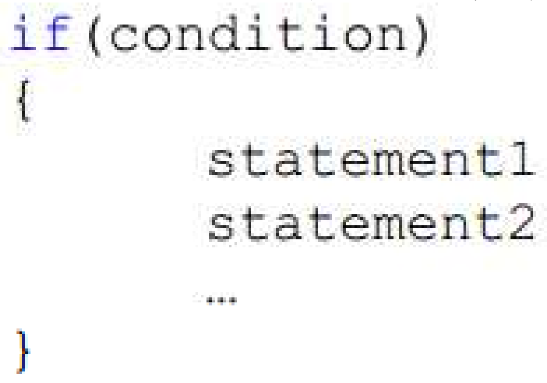
- 但有用的程序通常会根据情况决定需要执行那些语句。比如统计一篇文章中某个特定单词出现的次数，在进行这种计数时需要跳过其他的单词。再如一个游戏中的角色需要根据用户的操作指令移动到相应位置。

```
string word;  int count;  …
if (word==“teacher”)   count++;
```

- 程序的控制结构(Control Structures): 依赖环境的程序代码，最典型的两种：条件(conditionals) 和循环(loops)

条件(conditionals)

- C++主要有if 和 switch-case 两种条件语句
- if 关键字用于当且仅当条件满足时执行相应的程序块。



```
if(condition)
{
        statement1
        statement2
        ...
}
```

  这里的condition是一个逻辑表达式，如果值是true 则执行statement， 如果值是 false就不执行.

- 如

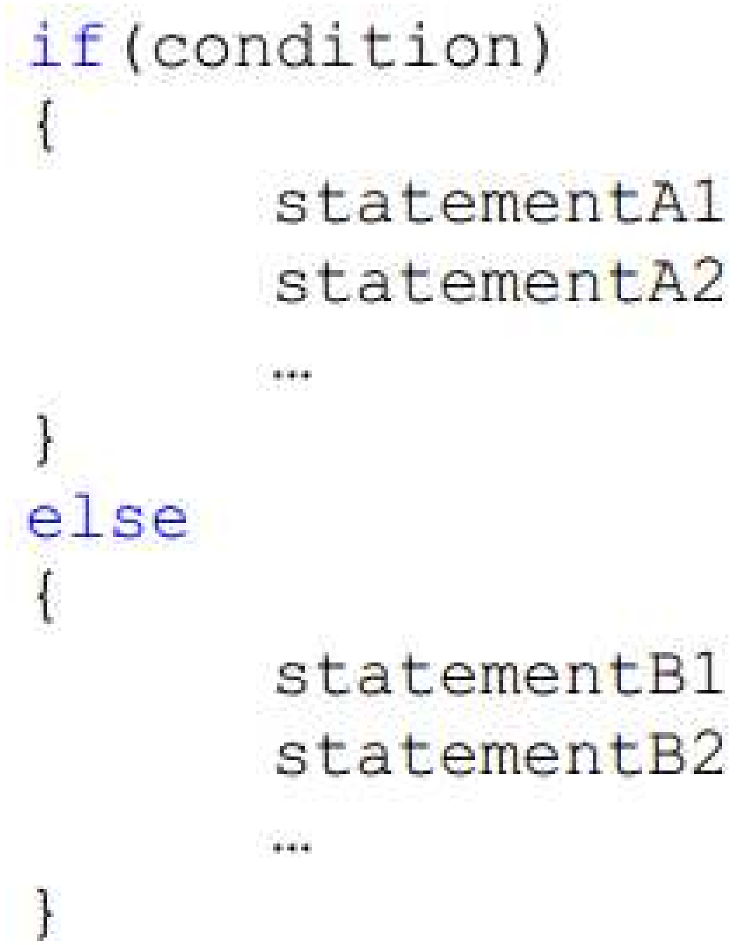
```
if (x == 100)
  cout << "x is 100";
```

- 如果if块内只有一个语句，可省略{ }

```
  if (condition) statement
```

条件(conditionals)

- 另一种 if 语句表达式形式：



```
if(condition)
{
        statementA1
        statementA2
        ...
}
else
{
        statementB1
        statementB2
        ...
}
```

- 如

```
if  (x == 100)
   cout << "x is 100";
else
   cout << "x is not 100";
```

- 如果if-else块内只有一个语句，可省略{ }
  if (condition) statement1 else statement2

条件(conditionals)

- 如果有多个判断条件，可用 else if.

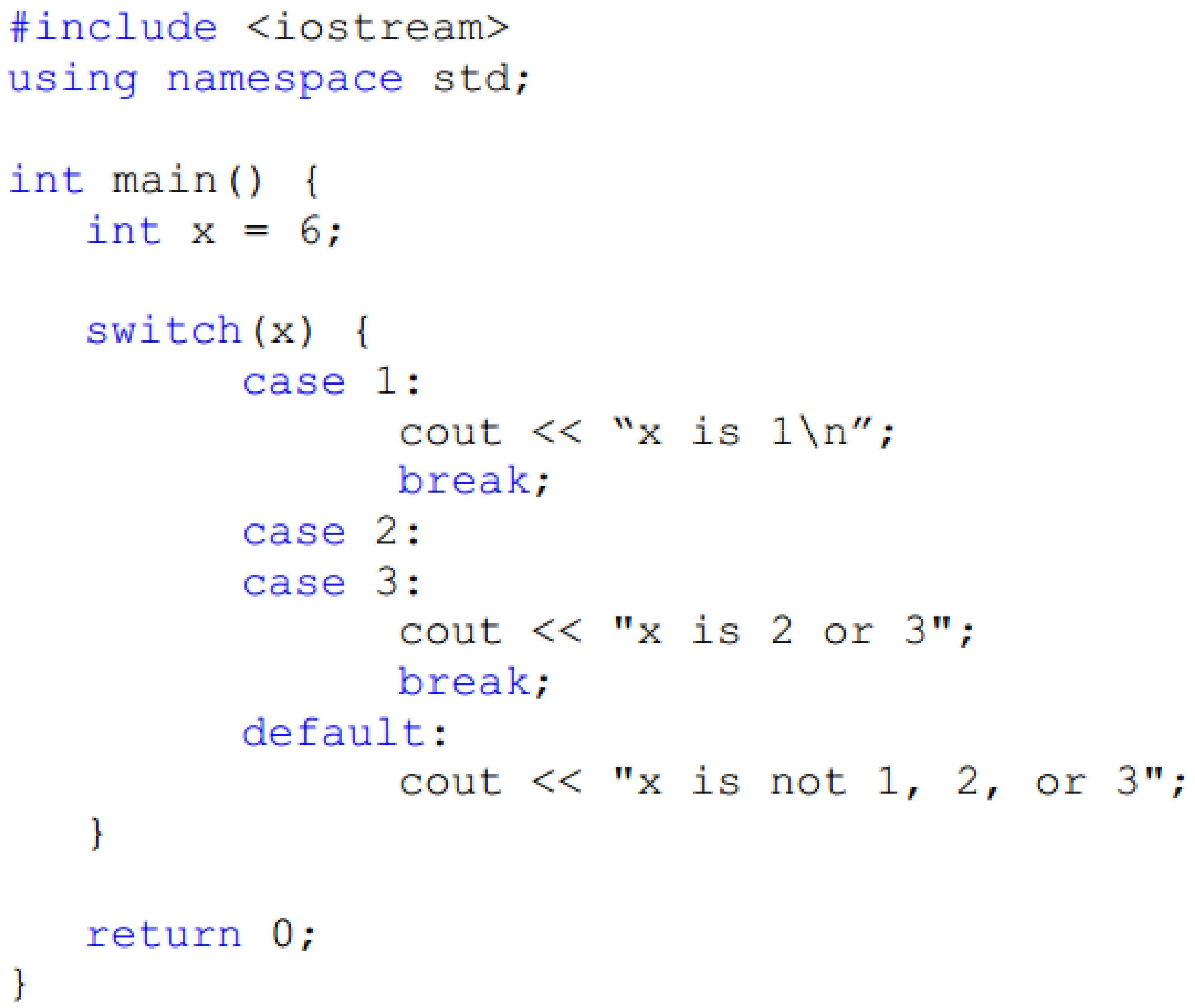
```
if(condition1)
{
        statementA1
        statementA2
        …
}
else if(condition2)
{
        statementB1
        statementB2
        …
}
```

```
#include <iostream>
using namespace std;

int main() {
   int x = 6;
   int y = 2;

   if(x > y)
          cout << "x is greater than y\n";
   else if(y > x)
          cout << "y is greater than x\n";
   else
          cout << "x and y are equal\n";

   return 0;
}
```

条件(conditionals)-switch-case

```
switch(expression)
{
      case constant1:
            statementA1
            statementA2
            ...
            break;
      case constant2:
            statementB1
            statementB2
            ...
            break;
      ...
      default:
            statementZ1
            statementZ2
            ...
}
```

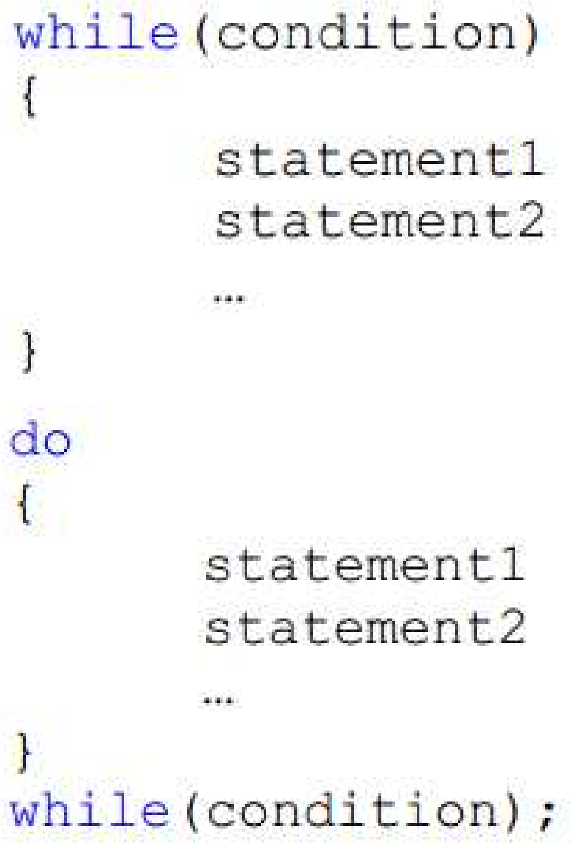
条件(conditionals)-switch-case



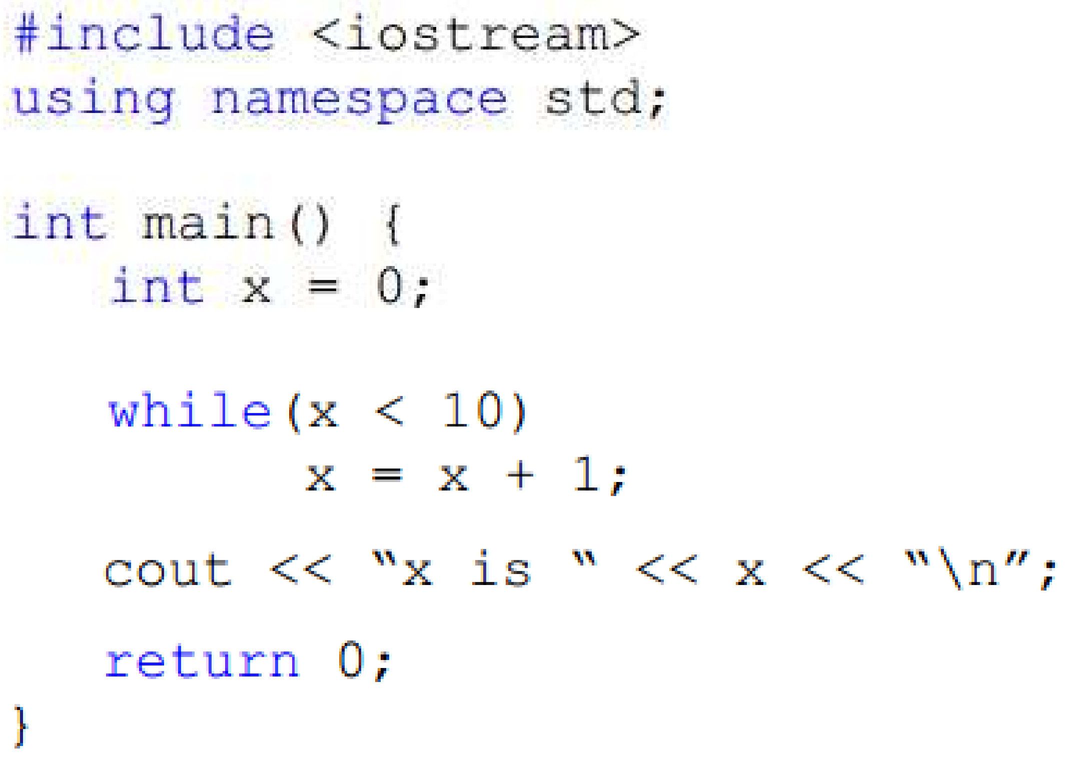
```
#include <iostream>
using namespace std;

int main() {
    int x = 6;

    switch(x) {
        case 1:
            cout << “x is 1\n”;
            break;
        case 2:
        case 3:
            cout << "x is 2 or 3";
            break;
        default:
            cout << "x is not 1, 2, or 3";
    }

    return 0;
}
```

循环(Loops)

- 满足条件，就重复执行循环体。三个关键字 while,do,for.
- while和do-while

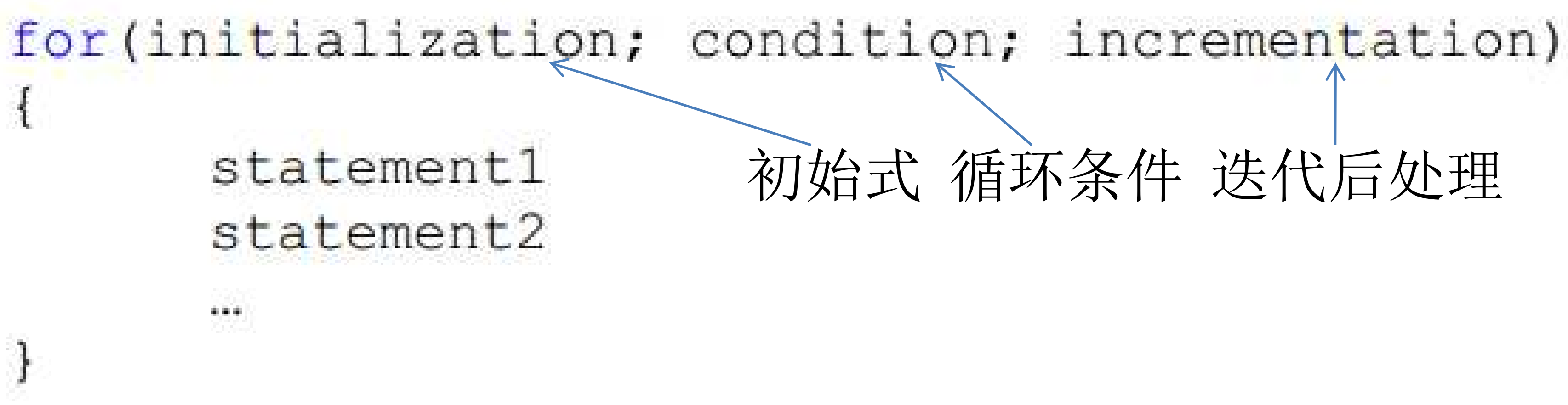


```
while(condition)
{
        statement1
        statement2
        …
}

do
{
        statement1
        statement2
        …
}
while(condition);
```



```
#include <iostream>
using namespace std;

int main() {
    int x = 0;

    while(x < 10)
          x = x + 1;

    cout << "x is " << x << "\n";

    return 0;
}
```

循环(Loops)



```
for(initialization; condition; incrementation)
{
        statement1
        statement2
        ...
}
```

```
#include <iostream>
using namespace std;

int main() {

   for(int x = 0; x < 10; x = x + 1)
         cout << x << "\n";

   return 0;
}
```

循环(Loops)

```
for(initialization; condition; incrementation)
{
        statement1
        statement2
        …
}
```

初始式 循环条件 迭代后处理

```
#include <iostream>
using namespace std;

int main() {

    int x = 0;
    for(; x < 10; x = x + 1)
          cout << x << "\n";

    return 0;
}
```

循环(Loops)

- for 和while循环是等价的.

```
for(initialization; condition; incrementation)
{
        statement1
        statement2
        …
}
```

等价于

```
initialization
while(condition)
{
        statement1
        statement2
        …
        incrementation
}
```

嵌套控制结构
(Nested Control Structures)

- 可以将if语句放在另一个if语句块内或者Loops语句放在另一个Loops语句块内,从而构成复杂的程序。

```
#include <iostream>
using namespace std;

int main() {
    int x = 6;
    int y = 0;

    if(x > y) {
            cout << “x is greater than y\n”;
            if(x == 6)
                    cout << “x is equal to 6\n”;
            else
                    cout << “x is not equalt to 6\n”;
    } else
            cout << “x is not greater than y\n”;

    return 0;
}
```

嵌套控制结构
(Nested Control Structures)

- 可以将if语句放在另一个if语句块内或者Loops语句放在另一个Loops语句块内,从而构成复杂的程序。

```
#include <iostream>
using namespace std;

int main() {
   for(int x = 0; x < 4; x = x + 1) {
         for(int y = 0; y < 4; y = y + 1)
               cout << y;
         cout << "\n";
   }

   return 0;
}
```