

第三讲

Control Structures

薛浩

2023年3月21日

www.stickmind.com

- 话题 1：编程基础** 初学编程的新手，一般应该熟练使用函数和库处理字符串相关的编程任务。
- 话题 2：抽象数据类型的使用** 在尝试实现抽象数据类型之前，应该先熟练使用这些工具解决问题。
- 话题 3：递归和算法分析** 递归是一种强有力的思想，一旦掌握就可以解决很多看起来非常难的问题。
- 话题 4：类和内存管理** 使用 C++ 实现数据抽象之前，应先学习 C++ 的内存机制。
- 话题 5：常见数据结构和算法** 在熟练使用抽象数据类型解决常见问题之后，学习如何实现它们是一件很自然的事情。

话题 1: 编程基础

初学编程的新手，一般应该熟练使用函数和库处理字符串相关的编程任务。

- C++ 基础
- 函数和库
- 字符串和流

American Soundex		Daitch-Mokotoff Soundex	Phonetic Matching
Waagenasz	Wegonge	Bassington	Bassington
Wachenhausen	Weismowsky	Bazunachden	Vasington
Wacknocty	Weuckunas	Bechington	Washincton
Waczinjac	Wiggins	Bussington	Washington
Wagenasue	Woigemast	Fissington	
Waikmishy	Wozniak	Washington	
Washington	Wugensmid	Vasington	4 names
Washincton	...	Washincton	
Wassingtom	+ 3,900 more names	Wassington	
...			
		9 names	

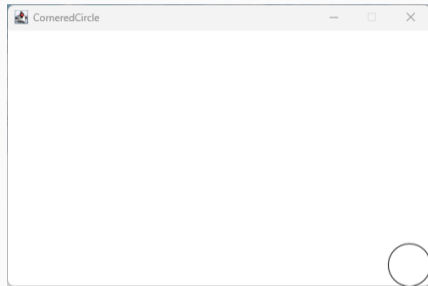
Figure 1: 语音算法

计算机如何作选择？

目录

1. 复习: Graphics 绘图
2. Boolean 布尔值
3. if 分支
4. for 循环
5. while 循环
6. switch 分支

复习: Graphics 绘图



通过**图形对象** (Graphics Objects) 可以更方便的创建图形应用。

创建一个图形对象分两个步骤：

- 声明一个指针类型的变量来记录对象的位置
- 使用关键字 `new` 来创建对象

指针变量常用于引用较大的数据结构以节省空间，声明指针变量需要做的就是类型后面加个星号 (*)。

例如，

```
#include "gobjects.h"  
G0val* circle = new G0val(x, y, 50, 50);
```

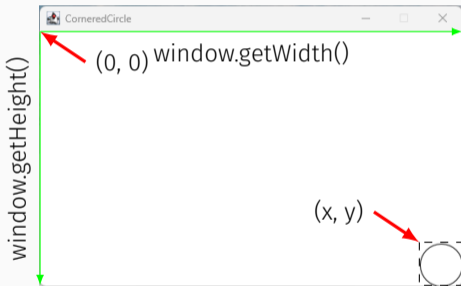

图形坐标

GWindow 对象通过定义 x 和 y 坐标来定位。

- x 从左向右增长; y 从上向下增长
- x 和 y 都以像素为单位

如何确定坐标值?

```
double x = ?;  
double y = ?;  
GOval* circle = new GOval(x, y, 50, 50);
```



魔法数字 (Magic Number) 表示无法通过代码直接推断其含义的数字。

```
double area = 3.14 * radius * radius;
```

在书写代码时，硬编码这些魔法数字是一个不好的习惯：

- 降低代码可读性
- 增加代码维护成本

常量 (constant) 表示初始化后无法修改值的变量。

常量一般定义在文件开头部分，使用大写字母表示，两个单词之间用下划线连接，比如 `PI`，`MAX_RADIUS`。

```
const double PI = 3.14;  
const double MAX_RADIUS = 300.0;
```

使用常量替换魔法数字可以极大地提高代码的可读性和可维护性。

Boolean 布尔值

布尔表达式 (Boolean Expression) 用于测试某个状态，只有 `true` 和 `false` 两个值。

比较运算符：用于比较两个值的大小。

`==` 等于 (区别于赋值运算符 `=`)

`!=` 不等于

`>` 大于

`<` 小于

`>=` 大于等于

`<=` 小于等于

逻辑运算符 (Logical Operators) 用于组合布尔值。

按优先级由高到低依次为：

逻辑非 ! !p

逻辑与 && p && q

逻辑或 || p || q

if 分支

if 分支

if 分支根据条件是否满足控制程序执行不同的部分。

```
if(condition) { statements }  
if(condition) { statements } else { statements}
```

例如，

```
if(n % 2) {  
    cout << "n is even" << endl;  
} else {  
    cout << "n is odd" << endl;  
}
```


更多分支:

```
if (score >= 90) {  
    println("Excellent!");  
} else if (score >= 80) {  
    println("Great Job!");  
} else if (score >= 70) {  
    println("Nice Work!");  
} else if (score >= 60) {  
    println("Would Be Better");  
} else {  
    println("Keep Practising");  
}
```

for 循环

好用的缩写

对变量修改后重新赋值的语句 `variable = variable op value;`
可以修改为 `variable op= value;`

例如,

```
x = x + 1;  y = y * 137;  
z = z / 4;  w = w - 1;
```

可以改写为,

```
x += 1;  y *= 137;  
z /= 4;  w -= 1;
```

对于加减 1 的操作还可以简写为,

```
x++;  
w--;
```

for 循环

for 循环适合用于重复执行固定次数的代码块。语法格式如下，

```
for (init; test; step) {  
    statements  
}
```

- `init` 在 for 循环开始前执行，用于设置循环计数器初值
- `test` 在代码块执行前检测，表达式为 `true` 时继续下一次循环
- `step` 在代码块执行后执行，用于更新计数器的值

for 循环

初始化语句 终止条件 步长

```
for(int i = 0; i < 3; i++) {  
    cout << i << endl;  
}
```

陷阱：初始化

以下代码的输出是什么？

```
int i = 0;
for(i = 5; i < 3; i++) {
    ...
}
cout << i << endl;
```

以下代码的输出是什么？

```
for(int i = 0; i < 3; i++) {  
    ...  
}  
cout << i << endl;
```

陷阱：终止条件

以下代码的输出是什么？

```
for(int i = 0; i < -3; i++) {  
    cout << i << endl;  
}
```

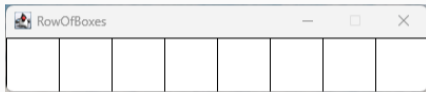

对比

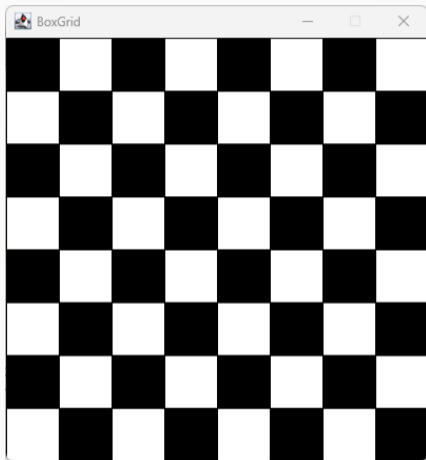
哪种写法更好? [1]

```
for(int i = 0; i < 3; i++) {  
    cout << i << endl;  
}
```

C89 风格

```
int i;  
for(i = 0; i < 3; i++) {  
    cout << i << endl;  
}
```





while 循环

while 循环

for 循环可以转换成 while 循环

```
for (init; test; step) {  
    statements  
}
```

while 循环等价形式

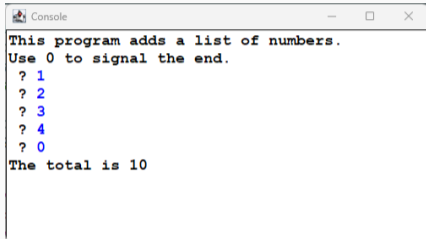
```
init;  
while (test) {  
    statements  
    step;  
}
```

Loop-and-a-half Idiom

根据某个**信号量** (sentinel) 确定是否要结束循环, 可以使用 `break` 语句。

```
while (true) {
    /* ... get a value from the user ... */
    /* ... decide whether to continue ... */
    if (condition == sentinel) {
        break;
    }
    /* ... process the value ... */
}
```

这种循环编程模式称为**读取直到信号量模式** (read-until-sentinel-pattern)。



```
Console
This program adds a list of numbers.
Use 0 to signal the end.
? 1
? 2
? 3
? 4
? 0
The total is 10
```

AddIntegerList.cpp

```
#include <iostream>
#include "console.h"
#include "simpio.h"
using namespace std;

/* Constant: SENTINEL */
const int SENTINEL = 0;

int main() {
    cout << "This program adds a list of
    ↪ numbers." << endl;
    cout << "Use " << SENTINEL << " to
    ↪ signal the end." << endl;
```

```
int total = 0;
while (true) {
    int value = getInteger(" ? ");
    if (value == SENTINEL)
        break;
    total += value;
}
cout << "The total is " << total <<
↪ endl;
return 0;
}
```


switch 分支

switch 分支

switch 分支的一般形式如下：

```
switch (expression) {  
  case c1:  
    statements  
    break;  
  case c2:  
    statements  
    break;  
  /* ... more cases ... */  
  default:  
    statements  
    break;  
}
```

expression 为控制表达式

与匹配控制表达式时执行对应的 case

c1, c2 必须为标量类型

控制表达式没有匹配项时，执行 default

陷阱：break 缺失

switch 分支的一般形式如下：

```
switch (expression) {  
  case c1:  
    statements  
  case c2:  
    statements  
    break;  
  /* ... more cases ... */  
  default:  
    statements  
    break;  
}
```

c1 没有 break 执行完后将继续 c2 分支

特例：空 case 分支

switch 分支的一般形式如下：

```
switch (expression) {  
  case c1:  
  case c2:  
    statements  
    break;  
  /* ... more cases ... */  
  default:  
    statements  
    break;  
}
```

c1 为空且没有 break, 将与 c2 共享 case

计算机如何作选择？

loop break
and if
switch for case a half
enum

问题?

- [1] 开源中国. *Linux 内核升级 C 语言标准*. URL:
<https://www.oschina.net/news/184153/linux-kernel-to-modern-c>.