

# 一起++ 第6弹

# 循环(上)、作用域

作者: @孤言 原审校: @Alex Cui

循环结构,执行一遍又一遍~

### 知识点 1 变量的作用域

#### 1.作用域

在开始学习循环结构之前呢,我们先来了解一下变量的作用域。

作用域是变量、函数等可访问的一个区域。也就是说,对于某些变量(或函数等),在某些地方可以使用, 在另一些地方就不能用了。

不同的变量声明方式可能导致变量的作用域不同。之前的几弹中,孤言一直让大家把变量的声明写在开头, 正是因为写在其它地方,这个变量的作用域就可能不同了。

#### 2.全局变量

在所有函数外部声明的变量,称为**全局变量。**之前我们紧跟着头文件、<mark>using namespace <u>std</u>;</mark>等的后面, 在 main 函数外写的那些变量声明,其实都是全局变量的声明。

scratch 创建变量的时候就可以选择全局变量。聪明的你一定知道,Scratch 里全局变量一旦创建,在所有角色里都是可以使用的。C++里也类似,全局变量的作用域是全局的。全局变量的值可以被该程序的所有对象或函数等使用。

#### 3.局部变量

在函数或一个代码块内部声明的变量,称为**局部变量**。它们只能被函数内部或者代码块内部的语句使用。

通俗但**不是很严谨**的说法就是,C++中,你在某一个大括号里声明了一个变量,那么它只能在这个大括号括起来的部分起作用。在这个大括号外的地方不能使用它。如图 1,2 所示,我们在 if 语句的"地盘"里声明了变量b,那么这个变量 b 就仅能在它所在的大括号内使用,在 if 外使用局部变量 b 时会报错。

图 2 局部变量错误使用(在 Dev C++中)

图 1 局部变量错误使用(在 VScode 中)

一般地,局部变量在声明时的初始值为原来内存中的值,你无法事先确定。而全局变量在声明时会初始化, 比如,int 型变量初始值为 0.

#### 4.关于 "同名"

局部变量和全局变量的作用域不同,那么可不可以声明 2 个名字相同的变量呢?完全可以哒!但是要注意,声明的这两个变量是完全不相干的两个变量,占据各自不同的内存空间。形象地说,这两个变量就是两个名字相同的人,长成什么样和名字没关系。变量名虽然相同,但存放的值是不互通的。"同名"的变量在各自的作用域发挥自己的作用。

但这里就有矛盾了,如果某个域同时属于两个同名变量的作用域呢?我们来看下面这个例子。

```
#include<iostream>
1
2
     using namespace std;
     int a=1;//声明全局变量 a
6
     int main()
         cout<<a<<endl;</pre>
9
         if(true)
10
         {
11
              int a=2;//声明局部变量 a,这两个变量 a 不同
12
13
              cout<<a<<end1;</pre>
14
15
         cout<<a<<end1;</pre>
16
17
         return 0;
     }
18
```

输出:

2

1

可见,if 语句中的局部变量 a "覆盖"了全局变量的 a. 为什么会这样呢?我们知道,Ln 11-Ln 14 既属于全局变量 a 的作用域,也属于局部变量 a 的作用域。这时**作用域范围越小的变量越"优先"**,所以 L13 中的 a 是局部变量 a. <del>(幼吾幼以及人之幼嘛~)</del>

这一规则既可以用于全局变量和局部变量,也可以用于局部变量和局部变量。

## 知识点 2 while 语句

#### 1.从重复执行直到谈起(仅用作过渡,可以跳过不读)

在 Scratch 里,我们经常会用到重复执行、重复执行直到<>积木。类似于这样的程序设计结构称为循环结构,这一点我们在上一弹已经谈到了。现在,我们重点来康康图 3 的这个重复执行直到积木。积木告诉计算机,



图 3 Scratch 积木

这里再举几个例子。

满足条件 a>b 时就停下来。

而 C++中的 while 语句不是这样的。while 是 "在……时"的意思,表示当满足条件时不要停下来,也就是**不满足条件时才停下来**。

所以说,while 语句的条件和重复执行直到的条件表达式是完全相反的。为了帮助大家顺利完成过度,孤言总结以下规则:

**等于**改成**不等于,和**改成**或,大于**改成**小于,小于**改成**大于。** 

当然,这里的规则并不很全面,但相信你已经 get 到了精髓。

```
重复执行直到 a = 50
```

```
重复执行直到 a > b 与 b = 2333 不成立
```

```
while(a!=50) //1
while(a<b) //2
while((a<=b)||(b==2333)) //3</pre>
```

- 1.Scratch 中是 a==50,取其"反面"为 a!=50
- 2.Scratch 中是 a>b 或 a==b 也就是 a>=b,那么"反面"为 a<b

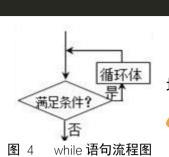
3.Scratch 中是(a>b)&&(b!=2333),那么根据规则,"反面"为(a<=b)||(b=2333)

当然,以上的转换只是一个为过渡阶段的同学提供一条道路,即使看不懂以上的转换,也没有关系,只要明确一点就可以了——while 语句只要当条件成立时,才继续循环。

#### 2.while 语句

while(条件表达式)

while 语句的语法格式为:



它是怎么运作的呢?可以用流程图 4 来表示。

while 语句先计算条件表达式,如果满足条件,就执行循环体(花括号里的语句块)一遍,然后继续判断条件,直到条件不满足时,跳出循环,执行后续语句。



while 语句先判断条件,再执行循环体,循环次数可能为 0 次。

例如,对于下面的代码段:

```
int a=5;
while(a<=4)
{
     a-=1;</pre>
```

因为一开始就没有满足 a<=4 这个条件,所以循环体根本没有执行,a 的值还是 5.

#### 算法表述的常见方法

1.自然语言:用"讲话"的方式直接描述算法,成为**自然语言。** 

2.流程图:以特定的图形符号加上说明,表示算法的图,称为流程图。

3. 伪代码: **伪代码**是一种类似于英语结构的,用于描述算法结构的方式(如图 5)。

# 将移动n个盘从A,通过B,移到C { 如果n等于1 直接将盘子从A移到C; 否则 { 将上面的n-1个盘从A,通过C,移到 B; 将最底下的一个盘从A移到C; 将B上的n-1个盘从B,通过A,移到C; } }

#### 下面,我们隆重请出 6 佬来表演跳舞!

//被本教程坑害的大佬还少吗

-6 总共要跳满 4 个小时的舞蹈,但中间需要休息。他第一次跳 2 个小时,

第二次为第一次的 3/4,也就是 90 分钟,以此类推,那么至少要跳几次呢?以下程序展示了求解方式。(也可以 使用等比数列求和公式来求和,这里不作引申)。

```
#include <iostream>
    #include <cstdlib>
    using namespace std;
    int main()
        double t=120,s=0; //时间增量(分钟),总用时(分钟)
        int n=0; //次数
9
10
        while(s<5*60) //在 s 没有达到 5h 时
11
12
        {
13
            s+=t;
            t*=2.0/3; //注意这里的 3.0, 用于隐式类型转换
14
15
            n+=1;
        }
16
17
18
        cout<<n<<endl;</pre>
19
        system("pause");
20
        return 0;
21
22
    }
23
```

与 for 语句类似, 当循环体中仅有一条语句时, 大括号可以省略不写。

# 知识点 3 do-while 语句

#### 1.来做个猜数游戏!

while 语句是先判断、再执行,但有的时候,我们还会遇到先执行一遍、再判断的情况。例如,你想编写一个 让别人猜随机数的程序。

#### 随机数函数

C++ 中有一个名为 rand() 的函数,每次调用该函数时都将返回一个非负整数(比如 1804289383)。要使 用 rand() 函数,必须在程序中包含 cstdlib 头文件(<del>当然你要用万能头我也实在拦不住</del>):

#### #include <cstdlib>

但要注意,这个函数产生的随机数是伪随机数,也就是用一些确定的计算公示计算出来的随机数,rand 函数 通常在启动计算机以后,生成的随机数就固定不变了。当然也有办法改变这种情况,这里就不深入了。

这样一个随机数显然不能满足你的欲望。我们想要的应该是像图 6 这样的特定范围内的随机数。



方法也很容易想到,我们以较简单的生成 1~10 的随机整数为例。因为 rand()对 10 取模的结果是在 0~9 间的整数,因此对其+1 也就得到了 1~10 的随机整数。即:

#### rand()%10+1

下面我们就来编写这个猜数游戏。

```
</>>
```

#include <iostream>

```
#include <cstdlib>
     using namespace std;
     int main()
         int n = rand() % 10 + 1;
         int ans;
10
11
12
         {
              cout << "Guess the number: ";</pre>
13
              cin >> ans;
14
15
              if (ans > n)
16
                   cout << "Too large!" << endl;</pre>
17
18
              else if (ans < n)</pre>
                   cout << "Too small!" << endl;</pre>
19
20
         } while (ans != n);
21
         cout << "U GOT IT!!" << endl;</pre>
22
23
         system("pause");
24
25
         return 0;
26
     }
```

#### 示例 I/O:

```
Guess the number: 3
Too large!
Guess the number: 1
Too small!
Guess the number: 2
U GOT IT!!
```

#### 2.do-while 语句

do-while 语句基本格式为:

# 注意 while 后面有分号。

它与 while 的不同之处是,会先执行一次循环体。然后再判断条件,如果条件成立,就继续循环,否则退出循环。因此循环体至少执行一次。do-while 语句的流程图如图 7 所示。因此即使 Ln 9 改为:

#### int ans=n;

也并不影响程序运行,因为第一次的回答已经将会赋给 ans.

再如,对于图 8的两个代码段,代码段 1输出为 0,代码段 2输出为 1.原因很容易想到。



图 7 do-while 语句流程图

```
//代码段1
                                                            #include <iostream>
    #include <iostream>
                                                           using namespace std;
    using namespace std;
                                                            int main()
    int main()
                                                                int i = 0;
         int i = 0;
         while ((i \ge 5) \&\& (i \le 10))
                                                       10
10
                                                       11
                                                                {
                                                       12
12
                                                                } while ((i >= 5) \&\& (i <= 10));
                                                       13
        cout << i;</pre>
13
                                                       14
14
                                                       15
                                                                cout << i;
15
        return 0;
16
   }
                                                       16
                                                       17
                                                                return 0;
17
18
```

图 8 while 和 do-while 的比较

根据运算符优先级,&&两侧的子条件表达式的括号原则上可以不写,但加上以后可以提高代码可读性。

和分支语句一样,循环语句也可以嵌套使用。

此外,当循环体只有一句语句时,大括号可以省略不写。例如上面代码段 1 的 Ln 11. 代码段 2 中 do-while 语句的大括号也可以省略。当然有的时候,大括号不省略,层次更加清晰一些。

循环还没有讲完,所以本弹不设代码练习。更加丰富的练习将在下一弹呈现。



©2019-2021 孤言, 版权所有。

未经作者许可,不得以任何形式和方式使用本文的任何内容(包括但不限于文字、程序等)。

第一版日期: 2019.12.5 第二版日期: 2021.2.4