信息学奥赛笔记03

循环结构——定量循环 | 循环结构——非定量循环

循环结构

如果我们希望让计算机输出`1000`遍$HelloWorld$，应该怎么办呢?

写一千遍`cout << "Hello World" << endl;`吗?

在用代码进行模拟和做数学计算的很多时候，我们都是在做重复的工作。

把大象放进冰箱需要三步：

- 打开冰箱门
- 把大象放进去
- 把冰箱门关上

再比如说，从`1`加到`1000000`，也只有两步：

- 加上一个数`i`
- `i`自身`+1`

第二个问题与第一个问题的区别就在于，把大象放进冰箱只有一次，这个行为主体只需要完成一次就行了。

而第二个问题需要把这个情况重复`1000000`遍，如果用人力进行计算，耗时耗力，这就是值得交给计算机做的事情了，让计算机来反复的执行一段相同的代码。这就是循环结构。

定量循环——For循环

代码语法

```
for (控制变量初始化表达式; 条件表达式; 增量表达式) {
    语句1;
    语句2;
}
```

如果语句1和语句2需要让计算机反复执行，则我们需要写一个for循环来让计算机这么做。

想让计算机反复循环执行代码，需要有三个要素：

- 从几开始循环
- 到什么时候结束
- 每次循环的时候控制循环次数的量变化多少

这就对应了我们for循环语法里的三个块。

第一块控制循环的初始值，我们一般习惯使用变量`i`来控制当前循环的标号。

当条件满足时，执行循环，直到条件不满足时结束，所以通过条件控制循环的执行次数。

既然用`i`来当作循环执行的标尺，所以我们需要每执行完代码后，来控制`i`的变化，来决定多少次后，达到条件不满足的情况，停止执行循环。

这样我们就能控制整个循环，从几开始，到几结束。

例题分析

[U424109] 求和 https://www.luogu.com.cn/problem/U424109

题目描述

输入一个整数n。计算输出 $1+2+3+...+n$ 的和。

输入格式

一个正整数，表示n.

输出格式

一个整数表示答案。

样例 #1

样例输入 #1

```
10
```

样例输出 #1

```
55
```

提示

对于100%的数据，$1 \leq n \leq 100$。

思路分析

> 本题中我们在写代码前需要想清楚两件事，累加器和循环变量。累加器的作用是记录当前求和的答案，既然我们想要去求$1+2+...+n$，起码我们要从小数开始加起，并且每一个数都加一遍，所以理清楚需要循环的代码部分。用`s`来做累加器进行求和，初始应该给`s`赋`0`，循环变量`i`从`1-n`遍历一遍，每个数都看一遍，每次`s += i`， 然后为了下一次能加下一个数，所以需要对`i++`，一共执行`n`遍后，就能获得求和的答案了。

标准代码

```cpp
#include <bits/stdc++.h>
using namespace std;
int n, s; // 变量定义在全局区自动赋0
int main() {
    cin >> n;
    //为了保证循环一定能执行n遍一般情况下在循环中一定不能改变结束的条件n
    for (int i = 1; i <= n; i++) { // 注意控制循环结束的n
        s += i; // 累加器每次加循环变量i
    }
    cout << s << endl; // 循环结束后，s存储答案，将答案输出
    return 0;
}
```

循环跳过语句——continue

在有的时候，我们并不需要将循环进行到底，我们来看一个问题：如果要找到班级里所有的男生去搬书，那么就得首先需要先看一遍班级所有的学生，因为我们只需要找到男生，女生虽然被找到了，但她并不是我们的选取范围，所以需要跳过，当男生这个条件符合的时候，选取，不符合的时候，不选取，不执行循环内的代码，就需要跳过，这就是`continue`的作用。

例题分析

[U424110]偶数输出 https://www.luogu.com.cn/problem/U424110

题目描述

输入一个整数n，按照由小到大的顺序，输出$1 \sim n$之间的所有偶数。

输入格式

一个正整数n。

输出格式

若干个整数，偶数之间用空格隔开。

样例 #1

样例输入 #1

```
10
```

样例输出 #1

```
2 4 6 8 10
```

提示

对于100%的数据，$1 \leq n \leq 100$。

思路分析

> 既然我们要找到`1-n`的所有偶数，所以咱们先把这些数都“看”一遍，也就是遍历这些数，如果是个奇数，就跳过它，如果是个偶数，就输出它。

标准代码（使用continue）

```cpp
#include <iostream>
using namespace std;
int n, s;
int main() {
    cin >> n;
    for (int i = 1; i <= n; i++) {
        if (i % 2 == 1) continue;
        cout << i << " ";
    }
    return 0;
}
```

标准代码（不使用continue）

```cpp
#include <iostream>
using namespace std;
int n, s;
int main() {
    cin >> n;
    for (int i = 1; i <= n; i++) {
        if (i % 2 == 0) {
            cout << i << " ";
        }
    }
    return 0;
}
```

标准代码（控制循环）

其实我们根本不用在循环里判断条件，只需要从`2-n`做循环，每次让`i`加`2`即可，直接在循环上动手脚。

```cpp
#include <iostream>
using namespace std;
int n, s;
int main() {
    cin >> n;
    for (int i = 2; i <= n; i += 2) {
        cout << i << " ";
    }
    return 0;
}
```

循环中止语句——break

和`continue`语句有些差异，直接将循环中止，不再进行任何操作。就比如刚刚的代码

```cpp
#include <iostream>
using namespace std;
int n, s;
int main() {
    cin >> n;
    for (int i = 1; i <= n; i ++) {
        if (i == 5) break;
        cout << i << " ";
    }
    return 0;
}
```

这个程序会输出`[1 2 3 4]`。

因为当`i == 5`的时候，程序执行了`break`，整个循环就被立刻中止掉了。

同学们一定要记住一句话：

continue只能在循环中使用，break不是只能在循环中使用

非定量循环——while循环

只要不达目的，我就不会停止，这句话就是一个循环的例子，我们把这句话拆分成两句，第一句：**只要不达目的**。这是一个条件，要么达到目的，要么不达到目的，不达到目的，就执行第二句话：**不停止**。达到目的就停止。那么，这个不停止的过程，我们就可以理解为while循环。

和`for`循环不同的是，`while`循环是一个专注于循环结束条件的语句，突出表现了**条件**的作用。

代码语法

```cpp
while (条件成立时) {
    语句1;
    语句2;
}
```

`while`循环看起来十分简单，但是想要操作起来却很难，如果想要用`while`循环执行定量的循环，需要在循环外面加上循环变量，并且赋初始值`0`，条件部分加上循环结束的条件，在循环内控制循环变量的增量，所以`for`循环将这三个步骤集合到了一个语句，而`while`循环把他们拆开了。

例题分析

上述for循环的第一道例题，我们想写成while循环，可以按照如下代码改写

例题1while循环

```cpp
#include <bits/stdc++.h>
using namespace std;
int n, s;
int main() {
    cin >> n;
    int i = 1;
    while (i <= n) {
        s += i;
        i++
    }
    cout << s << endl;
    return 0;
}
```

例题2while循环

```cpp
#include <iostream>
using namespace std;
int n, s;
int main() {
    cin >> n;
    int i = 1;
    while (i <= n) {
        if (i % 2 == 1) continue;
        cout << i << " ";
        i++;
    }
    return 0;
}
```

课后作业

在洛谷的团队——作业——周日校区-循环结构 中拿到`700`分