小马加编信息学教案(九)

C++循环结构(三)

一. 课程内容

1. 循环嵌套
2. 循环控制语句- `break` 与 `continue`
3. 循环综合理解

二. 知识讲解

1. 循环嵌套

1.1 循环嵌套的含义

> 循环结构与分支结构的嵌套类似，也可以**在一个循环语句的循环体里出现另一个循环语句**
> 不管是 `while` 语句、 `do-while` 语句还是 `for` 语句都可以进行嵌套使用
> **这样的循环结构称为“循环嵌套”**

1.2 `for` 循环嵌套分析

```
#include <bits/stdc++.h>

int main() {
    for (int i = 1; i <= 4; i++) {  //外层for循环
        for (j = 1; j <= 4; j++) {  //内层for循环
            printf("i=%d, j=%d\n", i, j);
        }
        cout << endl;
    }
    return 0;
}
```

> 运行结果

运行结果：

```
i=1, j=1
i=1, j=2
i=1, j=3
i=1, j=4

i=2, j=1
i=2, j=2
i=2, j=3
i=2, j=4

i=3, j=1
i=3, j=2
i=3, j=3
i=3, j=4

i=4, j=1
i=4, j=2
i=4, j=3
i=4, j=4
```

- 代码分析

> 本例是一个简单的 `for` 循环嵌套，**外层循环和内层循环交叉执行**，外层 `for` 每执行一次，内层 `for` 就要执行四次。

> 在 C++ 语言中，**代码是顺序、同步执行的**，当前代码必须执行完毕后才能执行后面的代码。这就意味着，**外层 `for` 每次循环时，都必须等待内层 `for` 循环完毕（也就是循环4次）才能进行下次循环**。虽然 i 是变量，但是对于内层 `for` 来说，每次循环时它的值都是固定的。

> 再看如下输出九九乘法口诀表的 for 循环结构的 C++ 代码

```
#include <bits/stdc++.h>

int main() {
    for (int i = 1; i <= 9; i++) {  //外层for循环
        for(int j = 1; j <= i; j++) {  //内层for循环
            printf("%d*%d=%-2d  ", i, j, i * j);
        }
        cout << endl;
    }
    return 0;
}
```

> 运行结果

```
1*1=1
2*1=2   2*2=4
3*1=3   3*2=6   3*3=9
4*1=4   4*2=8   4*3=12  4*4=16
5*1=5   5*2=10  5*3=15  5*4=20  5*5=25
6*1=6   6*2=12  6*3=18  6*4=24  6*5=30  6*6=36
7*1=7   7*2=14  7*3=21  7*4=28  7*5=35  7*6=42  7*7=49
8*1=8   8*2=16  8*3=24  8*4=32  8*5=40  8*6=48  8*7=56  8*8=64
9*1=9   9*2=18  9*3=27  9*4=36  9*5=45  9*6=54  9*7=63  9*8=72  9*9=81
```

- 代码分析

  > 内层 for 每循环一次输出一条数据，外层 for 每循环一次输出一行数据。
  > 需要注意的是，内层 for 的结束条件是 $j \leq i$。外层 for 每循环一次，i 的值就会变化，
  > **所以每次开始内层 for 循环时，结束条件是不一样的。**

> 具体如下：
>
> - 当 $i = 1$ 时，内层 for 的结束条件为 $j \leq 1$，只能循环一次，输出第一行。
> - 当 $i = 2$ 时，内层 for 的结束条件是 $j \leq 2$，循环两次，输出第二行。
> - 当 $i = 3$ 时，内层 for 的结束条件是 $j \leq 3$，循环三次，输出第三行。
> - 当 $i = 4$、5、6... 时，以此类推。

1.3 循环嵌套注意事项

> - 在使用循环嵌套的时候，一定要注意分清楚内外层循环的变量分别是谁，应该改变的是谁。
>   如下是初学者常见错误

```
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= m; i++) {

        }
```

> 内层循环的循环值改变语句错写成了 `i++` ，导致 j 的值永远无法改变，而在内层 `for` 循环发生死循环。正确代码为:

```
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n; j++) {

        }
```

2. 循环控制语句

2.1 循环控制语句的种类

> 在循环结构中，**有时需要提前跳出循环体，或者忽略本次循环的后续语句而去执行下一次循环**。为此， `C++` 提供了 `break` 语句和 `continue` 语句。

2.2 `break` 语句概念

> **在循环体中遇到 `break` 语句，就会立刻跳出循环体，执行循环结构后面的语句。**
> **`break` 关键字通常和 `if` 语句一起使用，即满足条件时便跳出循环。**

2.3 `break` 语句的应用与注意

```
#include <bits/stdc++.h>
int main() {
    int i = 1, sum = 0;
    while(1) {  //循环条件为死循环
        sum += i;
        i++;
        if (i > 100)
            break;
    }
    cout << sum << endl;
    return 0;
}
```

> `while` 循环条件为 1，是一个死循环。
> 当执行到第100次循环的时候，计算完 `i++` 后 i 的值为 101，此时 `if` 语句的条件 $i > 100$ 成立，执行$break$语句，结束循环。

- 注意

> 在多层循环中，一个 $break$ 语句只向外跳一层。例如，输出一个 $4 * 4$ 的整数矩阵:

```
#include <bits/stdc++.h>
int main(){
    int i = 1, j;
    while (1) {  // 外层循环
        j = 1;
        while (1) {  // 内层循环
            printf("%-4d", i*j);
            j++;
            if(j>4)
                break;  //跳出内层循环
        }
        cout << endl;
        i++;
        if (i>4)
            break;  // 跳出外层循环
    }
    return 0;
}
```

> 运行结果

```
1   2   3   4
2   4   6   8
3   6   9   12
4   8   12  16
```

- 代码分析

> 当 $j > 4$ 成立时，执行$break;$，跳出内层循环；
> 外层循环依然执行，直到 $i > 4$ 成立，跳出外层循环。
> 内层循环共执行了 4 次，外层循环共执行了 1 次。

2.4 continue语句概念

> `continue` 语句的作用是跳过循环体中剩余的语句而强制进入下一次循环。
> `continue` 语句只用在 `while` 、 `for` 循环中，常与 `if` 条件语句一起使用，判断条件是否成立。

2.5 continue语句的应用

```
#include <bits/stdc++.h>
int main() {
    char c = 0;
    while (c != '\n') {  //回车键结束循环
        c = getchar();
        if (c == '4' || c == '5'){  //按下的是数字键4或5
            continue;  //跳过当次循环，进入下次循环
        }
        putchar(c);
    }
    return 0;
}
```

- 代码分析

> 程序遇到 `while` 时，变量c的值为 `'\0'` ，循环条件 `c != '\n'` 成立，开始第一次循环。 `getchar()` 使程序暂停执行，等待用户输入，直到用户按下回车键才开始读取字符。

> 本例假设我们输入的是 `0123456789` ，当读取到 `4` 或 `5` 时，if 的条件 `c == '4' || c == '5'` 成立，就执行 `continue` 语句，结束当前循环，直接进入下一次循环，也就是说 `putchar(c);` 不会被执行到。而读取到其他数字时， `if` 的条件不成立， `continue` 语句不会被执行到， `putchar(c);` 就会输出读取到的字符。

2.6 `break` 与 `continue` 语句的对比

> **`break` 用来结束所有循环，循环语句不再有执行的机会**
> **`continue` 用来结束本次循环，直接跳到下一次循环，如果循环条件成立，还会继续循环。**

三. 经典例题

1. 输出矩形

> 输入 n 和 m，输出一个 n 行 m 列的“*”矩形图案。

> 输入格式
> 一行两个正整数 n 和 m ，中间用一个空格隔开

> 输出格式
> 输出一个 n 行 m 列的 `*` 矩形图案。

样例输入	样例输出
3 4	**** **** ****

> 数据范围
> $1 \leq n$、$m \leq 100$

2. 数字三角形

> 输入一个正整数 n，输出 `n` 行的数字三角形。其中，第 1 行为数字 1，第 2 行为数字 23，第 3 行为数字 456，第 4 行为数字 7890，第 5 行为数字 12345 ...

> 输入格式
> 一行一个正整数 n。

> 输出格式
> n 行的数字三角形。

样例输入	样例输出
4	1 23 456 7890

> 数据范围
> $1 \leq n \leq 100$

3. 素数判定

> 输入一个正整数，判断其是否为素数。如果是，则输出 `prime` ；否则，输出 `not prime` 。(请使用 `break` 完成)

> 输入格式
> 一行一个正整数 n

> 输出格式
> 一行一个字符串,如果是素数则输出 `prime` ,否则输出 `not prime`

样例输入	样例输出
8	not prime
7	prime

> 数据范围
> $2 \leq n \leq 10^7$

四. 提高巩固

1. 金币 ($NOIP2015$普及组第一题)

> 国王将金币作为工资，发放给忠诚的骑士。
> 第一天骑士收到一枚金币；之后两天（第二天和第三天），每天收到两枚金币；之后三天（第四、五、六天），每天收到三枚金币；之后四天，每天收到四枚金币，以此类推；
> 这种工资发放模式会一直延续下去，当连续N天收到N枚金币后，骑士会在之后的$N+1$天，每天收到$N+1$枚金币。
> 请计算前K天里，骑士一共获得了多少金币。

> 输入格式
> 输入包含一个正整数K，表示发放金币的天数。

> 输出格式
> 输出一个正整数，即骑士收到的金币数。

样例输入	样例输出
6	14
1000	29820

> 数据范围
> 对于全部数据，
> $1 \leq K \leq 100001$

> 样例提示
> 对于样例一，骑士一共收到$1+2+2+3+3+3=14$枚金币。

2. 素数的统计

> 输入两个正整数 m 和 n，判断 m 和 n 之间（含 m 和 n）一共有多少个素数。

> 输入格式
> 一行两个正整数 m 和 n

> 输出格式
> 一行一个整数，表示素数的个数。

样例输入	样例输出
5 10	2

> 数据范围
> $2 \leq m \leq n \leq 10^4$

3. 数字统计

> 请统计某个给定范围$[L, R]$的所有整数中，数字 2 出现的次数。
> 比如给定范围$[2, 22]$,数字 2 在数 2 中出现了 1 次，在数 12 中出现 1 次，在数 20 中出现 1 次，在数 21 中出现 1 次，在数 22 中出现 2 次，所以数字 2 在该范围内一共出现了 6 次。

> 输入格式
> 2 个正整数 L 和 R ，之间用一个空格隔开。

> 输出格式
> 数字 2 出现的次数。

样例输入	样例输出
2 22	6
2 100	20

数据范围
$1 \leq L, R \leq 100000$

同型附加题

1. 计数问题($NOIP2013$普及组第一题)

> 试计算在区间 1到 n的所有整数中，数字$x(0 \leq x \leq 9)$共出现了多少次?
> 例如，在1到11中，即在 $1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11$ 中，数字 1 出现了 4 次。

> 输入格式
> 一行两个整数n, x，之间用一个空格隔开

> 输出格式
> 一个整数，表示数字x的个数

样例输入	样例输出
11 1	4

> 数据范围
> 对于100%的数据，$1 \leq n \leq 1,000,000, 0 \leq x \leq 9$