

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级：软件工程2304班**

**学 号：U202317310**

**姓 名：柯俊翔**

**指导教师：唐赫**

**报告日期：2023-10-20**

**软件学院**

# 实验2 流程控制实验

## 2.1 实验目的

（1）掌握复合语句、if语句、switch语句的使用，熟练掌握for、while、do-while三种基本的循环控制语句的使用，掌握重复循环技术，了解转移语句与标号语句。

（2）练习循环结构for、while、do-while语句的使用。

（3）练习转移语句和标号语句的使用。

（4）使用集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

## 2.2 实验内容及要求

### 1．程序改错

下面的实验2-1程序是合数判断器（合数指自然数中除了能被1和本身整除外，还能被其它数整除的数），在该源程序中存在若干语法和逻辑错误。要求对该程序进行调试修改，使之能够正确完成指定任务。

/\* 实验2-1改错题程序：合数判断器\*/

1. #include <stdio.h>
2. int main( )
3. {
4. int i, x, k, flag = 0;
5. printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");
6. while (scanf("%d", &x) !=EOF) {
7. for(i=2,k=x>>1;i<=k;i++)
8. if (!x%i) {
9. flag = 1;
10. break;
11. }
12. if(flag=1) printf("%d是合数", x);
13. else printf("%d不是合数", x);
14. }
15. return 0;
16. }

**解答：**

（1）错误修改：

1) 第4行flag定义为0后在循环开始时没有归0，正确形式为：在第六行后加语句flag = 0;

2) 第8行运算优先级错误，正确形式为：if (!(x % i))

3) 第12行运算符错误，不应为赋值符，正确形式为：if (flag == 1)

（2）错误修改后运行结果：

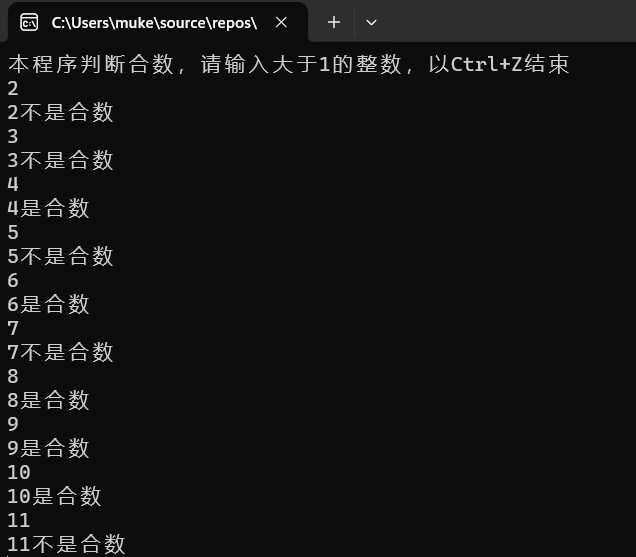


图 2-1 错误修改后的程序运行结果示意图

### 2．程序修改替换

（1）修改实验2-1程序，将内层两出口的for循环结构改用单出口结构，即不允许使用break、goto等非结构化语句。

**解答：**

程序等价为当flag=1后退出循环，可以在for循环中添加一个判断条件，因此，题目（1）替换后的程序如下所示：

1. #include <stdio.h>
2. int main()
3. {
4. int i, x, k, flag = 0;
5. printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");
6. while (scanf("%d", &x) != EOF)
7. {
8. flag = 0;
9. for (i = 2, k = x >> 1; i <= k&&!flag; i++)
10. if (!(x % i)) {
11. flag = 1;
12. }
13. if (flag == 1) printf("%d是合数\n", x);
14. else printf("%d不是合数\n", x);
15. }
16. return 0;
17. }

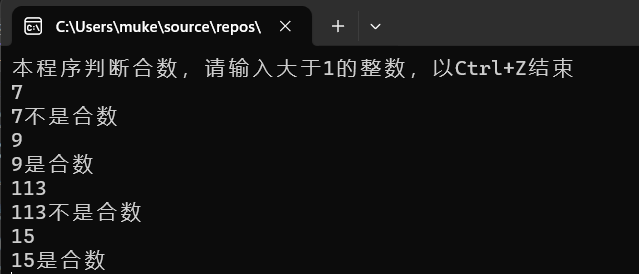


图 2-2 程序修改（1）后的程序运行结果示意图

1. 修改实验2-1程序，将for循环改用do-while循环。

**解答：**

由于do-while循环先进行do操作再进行判断，循环开始前需先获取一次x的值，因此，题目（2）替换后的程序如下所示：

1. #include <stdio.h>
2. int main()
3. {
4. int i, x, k, flag = 0;
5. printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");
6. scanf("%d", &x);
7. do
8. {
9. flag = 0;
10. for (i = 2, k = x >> 1; i <= k; i++)
11. if (!(x % i)) {
12. flag = 1;
13. break;
14. }
15. if (flag == 1) printf("%d是合数\n", x);
16. else printf("%d不是合数\n", x);
17. } while (scanf("%d", &x) != EOF);
18. return 0;
19. }

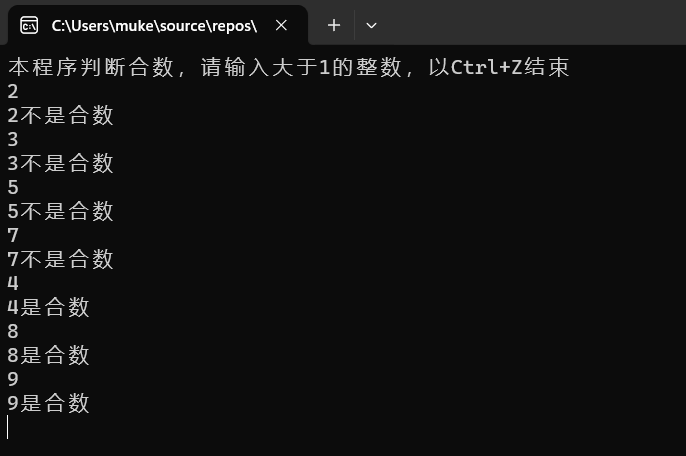


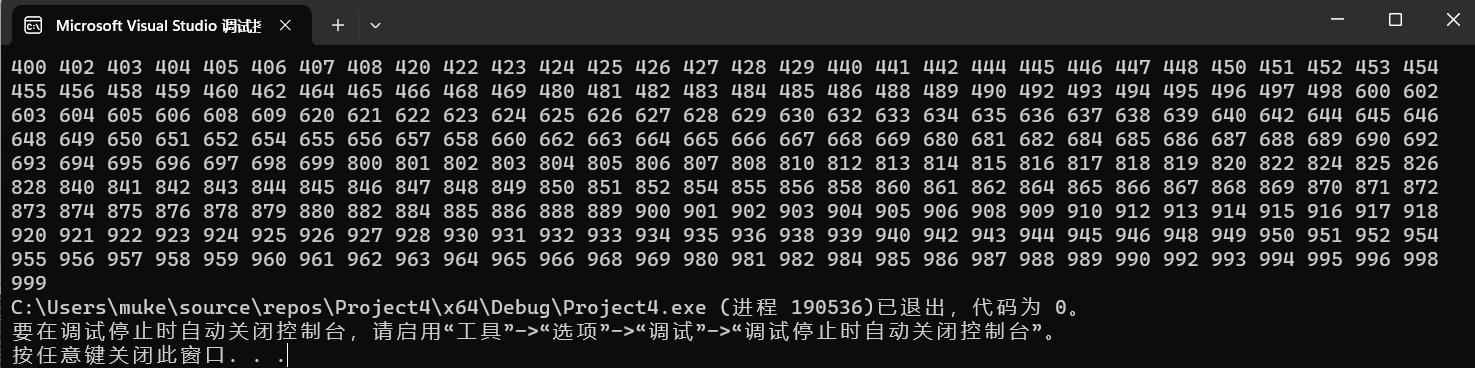
图 2-3 程序修改（2）后的程序运行结果示意图

（3）修改实验2-1程序，将其改为纯粹合数求解器，求出所有的3位纯粹合数。一个合数去掉最低位，剩下的数仍是合数；再去掉剩下的数的最低位，余留下来的数还是合数，这样反复，一直到最后剩下一位数仍是合数，这样的数称为纯粹合数。

**解答：**

根据题意，写出两个辅助函数判断是否为纯粹合数，再遍历三位数，得到答案，因此，题目（3）替换后的程序如下所示：

1. #include <stdio.h>
2. int isHeshu(int x){*//判断是否是合数*
3. int i, k, flag; flag = 0;
4. for (i = 2, k = x >> 1; i <= k; i++)
5. if (!(x % i)) {
6. flag = 1;break; }
7. if (flag == 1) return 1;
8. else return 0;
9. }
10. int PureHeshu(int n){*//判断是否是纯粹合数*
11. int ret=0;
12. if (isHeshu(n))
13. { n /= 10;
14. if (isHeshu(n)) {
15. n /= 10;
16. if (isHeshu(n)) {
17. ret = 1;  } } }
18. return ret;}
19. int main(){
20. for (int m = 100; m < 1000; m++)
21. {  if (PureHeshu(m))
22. printf("%d ", m); }
23. return 0;}

图 2-4 程序修改（3）后的程序运行结果示意图

### 3．程序设计

1. 假设工资税金按以下方法计算：x ＜ 1000元，不收取税金；1000 ≤ x ＜ 2000，收取5%的税金；2000 ≤ x ＜ 3000，收取10%的税金；3000 ≤ x ＜ 4000，收取15%的税金；4000 ≤ x ＜ 5000，收取20%的税金；5000 ≤ x，收取25%的税金。（注意税金的计算按照阶梯计税法，比如，工资为4500，那么税金=1000\*5% + 1000\*10% + 1000\*15% + 501\*20%）。编写一个程序,输入工资金额，输出应收取税金额度，要求分别用if语句和switch语句来实现。

**解答：**

1) 解题思路

1.输入x，根据实际情况，x声明为双精度浮点数

2.根据x的大小进行选择判断，计算税金

3. 打印结果；

4. 结束

1. 源程序清单

/\* if实现版本\*/

1. *#include <stdio.h>*
2. int main()
3. { double x, tax;
4. printf("请输入工资:");
5. scanf("%lf", &x);
6. if (x < 1000) tax = 0;
7. if (x >= 1000 && x < 2000) tax = (x+1 - 1000) \* 0.05;
8. if (x >= 2000 && x < 3000) tax = 1000\*0.05+(x+1-2000) \* 0.1;
9. if (x >= 3000 && x < 4000) tax = 1000 \* 0.05 + 1000\*0.1+(x+1 - 3000) \* 0.15;
10. if (x >= 4000 && x < 5000) tax = 1000 \* 0.05 + 1000 \* 0.1 + 1000\*0.15+(x+1 - 4000) \* 0.2;
11. if (x >= 5000) tax = 1000 \* 0.05 + 1000 \* 0.1 + 1000 \* 0.15 +1000\*0.2+(x+1 - 5000) \* 0.25;
12. printf("\n应收税金%lf元\n", tax);
13. }

/\* switch实现版本\*/

1. *#include <stdio.h>*
2. int main()
3. {
4. double x, tax;
5. printf("请输入工资:");
6. scanf("%lf", &x);
7. int a =(int) x / 1000;
8. switch (a) {
9. case 0:tax = 0;
10. break;
11. case 1:tax = (x + 1 - 1000) \* 0.05;
12. break;
13. case 2:tax = 1000 \* 0.05 + (x + 1 - 2000) \* 0.1;
14. break;
15. case 3:tax = 1000 \* 0.05 + 1000 \* 0.1 + (x + 1 - 3000) \* 0.15;
16. break;
17. case 4:tax = 1000 \* 0.05 + 1000 \* 0.1 + 1000 \* 0.15 + (x + 1 - 4000) \* 0.2;
18. break;
19. default:tax = 1000 \* 0.05 + 1000 \* 0.1 + 1000 \* 0.15 + 1000 \* 0.2 + (x + 1 - 5000) \* 0.25;
20. break;
21. }
22. printf("\n应收税金%lf元\n", tax);
23. return 0;
24. }

3）测试

（a） 测试数据：

表2-1 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例1 | 1500 | 25.05 |
| 用例2 | 2500 | 100.1 |
| 用例3 | 4500 | 400.2 |

（b） 对应测试数据的运行结果截图

图2-5 编程题1的测试用例1的运行结果



图2-6 编程题1的测试用例2的运行结果



图2-7 编程题1的测试用例3的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

1. 输入一段以!结尾的短文(最多5行,每行不超过50个字符)，要求将它复制到输出，复制过程中将每行一个以上的空格字符用一个空格代替。

**解答：**

1） 算法流程如图2.8所示。



图2-8 编程题2的程序流程图

2）源程序清单

1. #include <stdio.h>
2. int main() {
3. char input[5][51];   char c;
4. int line = 0;   int charCount = 0; int spaceCount = 0;
5. while ((c = getchar()) != '!' && line < 5){
6. if (c == ' ') {
7. spaceCount++;              *//空格计数*
8. if (spaceCount <= 1) {      *//保证只有一个空格*
9. input[line][charCount] = c;
10. charCount++;
11. }
12. }
13. else {
14. spaceCount = 0;      *//归零空格计数*
15. input[line][charCount] = c;   *//将c存入数组*
16. charCount++;
17. }
18. if (c == '\n') {
19. input[line][charCount] = '\0';    *//添加终止符并换行*
20. line++;
21. charCount = 0;
22. spaceCount = 0;
23. }
24. }
25. input[line][charCount] = '!';    *//添加结尾*
26. input[line][charCount + 1] = '\0';
27. for (int i = 0; i <= line; i++) {
28. printf("%s", input[i]);   }     *//打印短文*
29. return 0;
30. }

3）测试

（a） 测试数据：

表2-2 编程题2的测试数据

|  |  |
| --- | --- |
| 测试输入 | 理论结果 |
| Mary had a little lamb .  It is a good pet for Mary  The lamb did not live in the water The lamb did not fly away  The lamb liked to look at books! | Mary had a little lamb .  It is a good pet for Mary  The lamb did not live in the water  The lamb did not fly away  The lamb liked to look at books! |

 （b） 对应测试数据的运行结果截图

图2-9 编程题2的测试用例的运行结果

1. 打印如下的杨辉三角形。

1 /\*第0行 \*/

1 1 /\*第1行 \*/

1 2 1 /\*第2行 \*/

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

1 9 36 84 126 126 84 36 9 1

第i行第j列位置的数据值可以由组合表示，而的计算如下：

|  |  |
| --- | --- |
|  | (i=0,1,2,…) |
|  | (j=0,1,2,3,…,i) |

根据以上公式，采用顺推法编程，输入最后一行的编号N（0<=N<=6），要求输出金字塔效果的杨辉三角形。

特别要注意空格的数目，一位数之间是3个空格，两位数之间有2个空格，3位数之间只有一个空格。第N行行首是N个空格。每行末尾是3个空格和换行。

**解答：**

1） 解题思路

1. 输入整数N
2. 利用for循环打印N个空格，并用第二个for循环控制每行空格的数量比上一行少两个
3. 利用递推关系求出coef
4. 利用双层循环打印出杨辉三角
5. 结束

2）源程序清单

1. #include <stdio.h>
2. int main() {
3. int N;
4. scanf("%d", &N);
5. for (int i = 0; i <= N; i++)  {
6. for (int a = 1; a <= N; a++)   {
7. printf(" ");
8. }
9. for (int space = 1; space <= N - i; space++)    {
10. printf("  ");
11. }
12. int coef = 1;
13. for (int j = 0; j <= i; j++) {
14. printf("%d", coef);
15. if (coef < 10)
16. printf("   ");
17. if (coef >= 10)
18. printf("  ");
19. coef = coef \* (i - j) / (j+1);
20. }
21. printf("\n");
22. }
23. return 0;
24. }

3）测试

（a） 测试数据：

表2-3 编程题2的测试数据

|  |  |
| --- | --- |
| 测试输入 | 理论结果 |
| 5 | 1  1 1  1 2 1  1 3 3 1  1 4 6 4 1  1 5 10 10 5 1 |

（b） 对应测试数据的运行结果截图



图2-10 编程题3的测试用例的运行结果

（4）625这个数很特别，625的平方等于390625，其末3位也是625。请编程输出所有这样的3位数：它的平方的末3位是这个数本身。要求这些数字从小到大排列，每个数字单独占一行。

**解答：**

1） 算法流程如图2.11所示。



图2-11 编程题4的程序流程图

2）源程序清单

1. #include <stdio.h>
2. #include <math.h>
3. int main()
4. {
5. for (int i = 100; i < 1000; i++)
6. {
7. int a = pow(i, 2);
8. if (a % 1000 == i)
9. {
10. printf("%d\n", i);
11. }
12. return 0;
13. }

3）测试

运行结果如图所示



图2-12 编程题4的运行结果

## 2.3 实验小结

在本实验中，我探索了复合语句、if语句、switch语句，以及for、while、do-while等循环控制语句的使用，还了解了转移语句和标号语句的应用。此外，我还学习了如何使用集成开发环境（IDE）的调试功能，包括单步执行、设置断点以及观察变量值。在完成这次实验后，我获得了以下几点体会和经验：

1. **善于纠错**：在编写程序时，仍然经常会遇到语法错误，例如拼写错误、缺少分号、括号不匹配等。这些错误可以导致程序无法编译或运行。通过检查代码，查看IDE提供的错误消息来识别和纠正语法错误。
2. **重视逻辑，学会调试**：在程序中，尤其是使用循环可能会出现逻辑错误，导致程序的输出不符合预期，或陷入死循环。我们可以使用调试工具，例如设置断点、单步执行，观察程序的执行过程，和中间变量的值，判断错误原因。
3. **善于选择合适结构**：通过实验，我了解每种循环结构和选择结构的特点和用途，知道了while和do-while的区别，if和switch的区别，学会如何选择适合特定情况的循环结构。
4. **更多样结构的流程图画法：**通过实验，我熟悉visio软件的基本使用，知道了如何用流程图表示循环结构和选择结构整理思路。