

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级：**

**学 号：**

**姓 名：**

**指导教师：**

**报告日期：**

**软件学院**

**目 录**

[实验2 流程控制实验 2](#_Toc180178704)

[2.1 实验目的 2](#_Toc180178705)

[2.2 实验内容及要求 2](#_Toc180178706)

[2.3 实验小结 14](#_Toc180178707)

# 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。

1 /\* 实验2-1改错题程序：合数判断器\*/

2 #include <stdio.h>

3 int main( )

4 {

5 int i, x, k, flag = 0;

6 printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");

7 while (scanf("%d", &x) !=EOF) {

8 for(i=2,k=x>>1;i<=k;i++)

9 if (!x%i) {

10 flag = 1;

11 break;

12 }

13 if(flag=1) printf("%d是合数", x);

14 else printf("%d不是合数", x);

15 }

16 return 0;

17 }

**解答：**

1. 错误修改
   1. 在每次输入新的数字后，未将flag重置为0，导致在输入一个合数后，所有的数均被判断为合数，应在第7、8行间添加：  
      flag = 0;
   2. 第9行if语句的条件中，由于运算符优先级，!x先得到运算，导致条件恒为假，应将条件修改为：  
      !(x%i) 或 x%i == 0
   3. 第13行if语句的条件中错误的使用了赋值运算符，使条件恒为真，应将条件修改为：  
      flag == 1
2. 修改后完整代码

/\* 实验2-1改错题程序：合数判断器\*/

#include <stdio.h>

int main( )

{

int i, x, k, flag = 0;

printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");

while (scanf("%d", &x) !=EOF) {

flag = 0;

for(i=2,k=x>>1;i<=k;i++)

if (!(x%i)) {

flag = 1;

break;

}

if(flag==1) printf("%d是合数", x);

else printf("%d不是合数", x);

}

return 0;

}

1. 文本

   描述已自动生成修改后效果

图2-1 实验2-1的运行结果图

**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：**

1. 修改后完整代码

#include <stdio.h>

int main()

{

int i, x, k, flag = 0;

printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");

while (scanf("%d", &x) !=EOF) {

flag = 0;

for(i=2,k=x>>1;i<=k && !flag;i++) {

if (!(x%i)) {

flag = 1;

}

}

if(flag==1) printf("%d是合数", x);

else printf("%d不是合数", x);

}

return 0;

}

1. 文本

   描述已自动生成修改后运行效果

图2-2 程序修改题1的运行结果图

（2）修改实验2-1程序，将for循环改用do-while循环。

**解答：**

1. 修改后完整代码

#include <stdio.h>

int main()

{

int i, x, k, flag = 0;

printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");

if (scanf("%d", &x) == EOF) {

return 0;

}

do {

flag = 0;

for(i=2,k=x>>1;i<=k;i++) {

if (!(x%i)) {

flag = 1;

break;

}

}

if(flag==1) printf("%d是合数", x);

else printf("%d不是合数", x);

} while (scanf("%d", &x) != EOF);

return 0;

}

1. 文本

   描述已自动生成修改后运行效果

图2-3 程序修改题2的运行结果图

（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：**

1. 修改后完整代码

#include <stdio.h>

int composite(int x) {

int flag = 0, k = x / 2;

for(int i=2; i<=k;i++) {

if (!(x%i)) {

flag = 1;

break;

}

}

return flag;

}

int main()

{

for (int i = 100; i <= 999; i++) {

if (composite(i) && composite(i / 10) && composite(i / 100))

printf("%d ", i);

}

return 0;

}

1. 电脑键盘

   中度可信度描述已自动生成修改后运行效果

图2-4 程序修改题3的运行结果图

**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：**

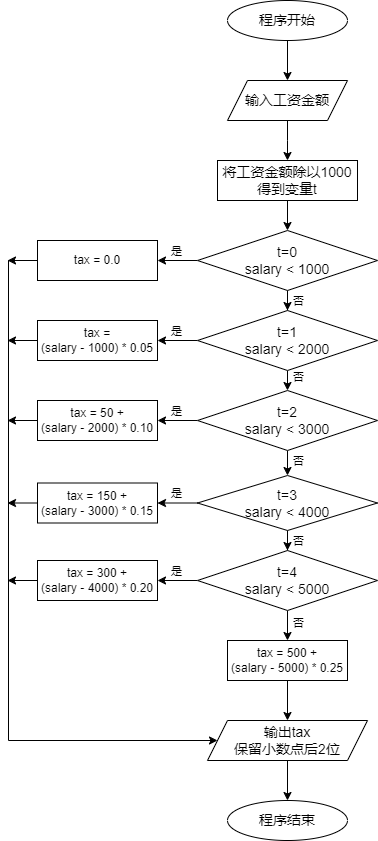
1. 算法流程

图2-5 编程题1的程序流程图

1. 使用if的代码清单

#include <stdio.h>

int main() {

double salary;

printf("请输入工资金额：");

scanf("%lf", &salary);

double tax = 0.0;

int t = salary / 1000;

if (t == 0) {

tax = 0.0;

} else if (t == 1) {

tax = (salary - 1000) \* 0.05;

} else if (t == 2) {

tax = 50 + (salary - 2000) \* 0.1;

} else if (t == 3) {

tax = 150 + (salary - 3000) \* 0.15;

} else if (t == 4) {

tax = 300 + (salary - 4000) \* 0.20;

} else {

tax = 500 + (salary - 5000) \* 0.25;

}

printf("收税金额为：%.2lf元", tax);

}

1. 使用switch的代码清单

#include <stdio.h>

int main() {

double salary;

printf("请输入工资金额：");

scanf("%lf", &salary);

double tax = 0.0;

int t = salary / 1000;

switch (t) {

case 0:

tax = 0.0;

break;

case 1:

tax = (salary - 1000) \* 0.05;

break;

case 2:

tax = 50 + (salary - 2000) \* 0.1;

break;

case 3:

tax = 150 + (salary - 3000) \* 0.15;

break;

case 4:

tax = 300 + (salary - 4000) \* 0.20;

break;

default:

tax = 500 + (salary - 5000) \* 0.25;

break;

}

printf("收税金额为：%.2lf元", tax);

}

1. 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 | 运 行 结 果 |
| 用例1 | 4500 | 400.00 | 400.00 |
| 用例2 | 1000 | 0.00 | 0.00 |

1. 文本

   描述已自动生成文本

   描述已自动生成运行效果

图2-6 编程题1的运行结果图

以下（2）至（4）题对应Educoder教学平台“C语言实验”课程，实验2，第4关实验2-1、第5关实验2-2，以及第6关实验2-3。

(2) 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。

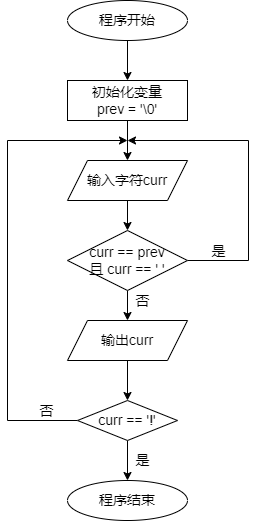
1. 算法流程

图2-7 编程题2的程序流程图

1. 源程序清单

#include<stdio.h>

int main() {

char prev = '\0', curr;

while (1) {

scanf("%c", &curr);

if (curr == prev && curr == ' ') {

continue;

}

printf("%c", curr);

if (curr == '!') {

break;

}

prev = curr;

}

return 0;

}

1. 测试数据

表2-2 编程题2的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序输入 | 理 论 结 果 | 运 行 结 果 |
| 用例1 | 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! | 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! |
| 用例2 | Never gonna give you up.  Never gonna let you down.  Never gonna desert you! | Never gonna give you up.  Never gonna let you down.  Never gonna desert you! | Never gonna give you up.  Never gonna let you down.  Never gonna desert you! |

1. 文本

   描述已自动生成文本

   描述已自动生成运行效果

图2-8 编程题2的运行结果图

（3）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。

1. 图示

   描述已自动生成算法流程

图2-9 编程题3的程序流程图

1. 源程序清单

#include<stdio.h>

int main() {

int n;

int arr[10][10];

arr[0][0] = 1;

scanf("%d", &n);

for (int i = 0; i <= n; i++) {

printf("%\*s", n + (n - i) \* 2, "");

for (int j = 0; j <= i; j++) {

if (j == 0 || j == i) {

arr[i][j] = 1;

} else {

arr[i][j] = arr[i][j - 1] \* (i - j + 1) / j;

}

printf("%-4d", arr[i][j]);

}

printf("\n");

}

}

1. 测试数据

表2-3 编程题3的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序输入 | 理 论 结 果 | 运 行 结 果 |
| 用例1 | 6 | 1  1 1  1 2 1  1 3 3 1  1 4 6 4 1  1 5 10 10 5 1  1 6 15 20 15 6 1 | 1  1 1  1 2 1  1 3 3 1  1 4 6 4 1  1 5 10 10 5 1  1 6 15 20 15 6 1 |
| 用例2 | 5 | 1  1 1  1 2 1  1 3 3 1  1 4 6 4 1  1 5 10 10 5 1 | 1  1 1  1 2 1  1 3 3 1  1 4 6 4 1  1 5 10 10 5 1 |

1. 图片包含 图示

   描述已自动生成形状, 箭头

   描述已自动生成运行效果

图2-10 编程题3的运行结果图

（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。

1. 图示

   描述已自动生成算法流程

图2-11 编程题4的程序流程图

1. 源程序清单

#include<stdio.h>

int main() {

for (int i = 100; i <= 999; i++) {

if (i \* i % 1000 == i) {

printf("%d\n", i);

}

}

}

1. 测试数据

表2-4 编程题4的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序输入 | 理 论 结 果 | 运 行 结 果 |
| 用例1 | 无输入 | 376  625 | 376  625 |

1. 图形用户界面, 应用程序

   中度可信度描述已自动生成运行效果

图2-12 编程题4的运行结果图

2.3 实验小结

1. 在本次实验中，我学会并练习了复合语句的应用，学习了if-else语句、switch语句、for循环、while循环、do-while循环的基本应用。
2. 在本次实验中，我对各种数据类型又有了新的认识，并且逐渐学会了如何使用格式控制语句对不同类型的数据进行输入输出。
3. 在本次实验中，我尝试了使用集成开发环境中自带的debug功能对程序进行单步调试等，提高了开发的效率。