

**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2003班**

**学 号： U202010851**

**姓 名： 侯皓斐**

**指导教师： 唐赫**

**报告日期： 2020.11.03**

**软件工程**

**目 录**

[**2 流程控制实验 1**](#_Toc404837920)

[2.1 实验目的 1](#_Toc404837921)

[2.2 实验内容 1](#_Toc404837922)

[2.3 实验小结 1](#_Toc404837923)7

# 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

**2.2.1 源程序改错**

**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\*/

#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) {

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;

}

**解答：**

（1）错误修改：

1) 第7行的while循环内flag应该每次置为0，正确形式为：

flag = 0;

2) 第9行的k = x >> 1运算数量明显偏多，正确形式为：

for (i = 2, k = (int)sqrt(x); i <= k; i++)

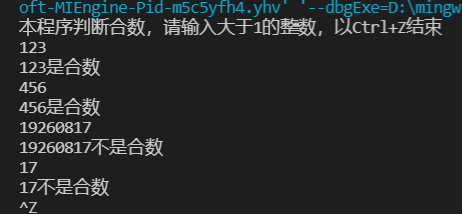
3) 第10行的!运算优先级高，产生错误运算结果，正确形式为：

if (!(x % i))

4) 第14行的判断语句应使用==，正确形式为：

if (flag == 1)

（2）错误修改后运行结果：



**2.2.2 源程序修改替换**

（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：**

For循环中的break非结构语句破坏了单出口结构，将flag的判断放入for循环的条件控制语句，这样中途如果出现一个模数使其取模为0，即其为合数，便退出循环。

/\* 实验2-1改错题程序：合数判断器\*/

/\* 程序修改替换：改单出口 \*/

#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;//需添加，否则出现一次合数，之后都是合数

for (i = 2, k = (int)sqrt(x); (!flag) && i <= k; i++)

if (!(x % i))

flag = 1;

if (flag == 1)//if (flag = 1)

printf("%d是合数\n", x);//printf("%d是合数", x);

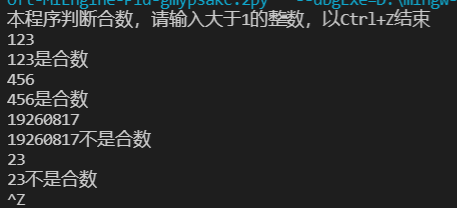
else

printf("%d不是合数\n", x);//printf("%d不是合数", x);

}

return 0;

}



（2）修改实验2-1程序，将for循环改用do-while循环。

**解答：**

将for循环的判断语句放入while的判断语句，若如是，可将flag取消。因为当我们i一直自加到sqrt(x)后，仍未出现一个模数与x取模为0，即可判定其为质数，若i<floor(sqrt(x))+1便可已知其中途退除了循环，发现了其某个因数，便其为合数。

/\* 实验2-1改错题程序：合数判断器\*/

/\* 程序修改替换：改用dowhile循环 \*/

#include <stdio.h>

int main() {

int i, x, k;

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) {

i = 1; k = (int)sqrt(x);

do {

i++;

}while((x % i) && i <= k);

if (i != k+1)//if (flag = 1)

printf("%d是合数\n", x);//printf("%d是合数", x);

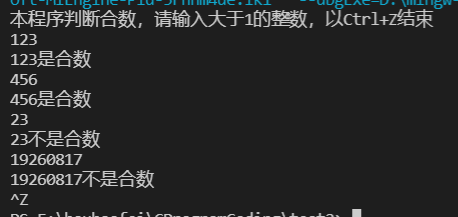
else

printf("%d不是合数\n", x);//printf("%d不是合数", x);

}

return 0;

}



（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：**

我们依次枚举X所有的三位数，即从100到999，每个数判断其是否为纯粹合数。我们设立一个表示其为纯粹合数的标志变量pureComFlag，一开始设为1。定义临时变量x=X。然后利用上述代码判断x是否为合数，若为合数则pureComFlag = 0，退出循环。否则x自除10，直到x等于0为止，这样我们就判断了X是否是纯粹合数。

算法流程图如下：

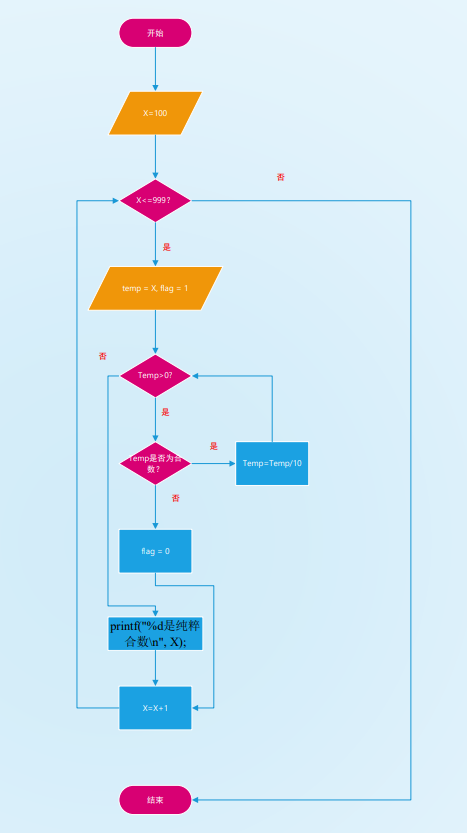


图2-1 源程序修改替换3的程序流程图

/\* 实验2-1改错题程序：合数判断器\*/

/\* 程序修改替换：纯粹合数求解器 \*/

#include <stdio.h>

#include <math.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");

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

x = X;

int pureComFlag = 1;

while(x) {

flag = 0;

for (i = 2, k = (int)sqrt(x); i <= k; i++)

if (!(x % i)) {

flag = 1;

break;

}

if(!flag) {

pureComFlag = 0;

break;

}

x = x/10;

}

if (pureComFlag == 1)

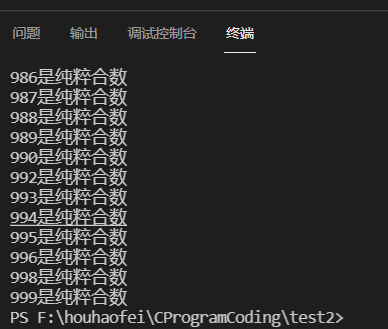
printf("%d是纯粹合数\n", X);

}

return 0;

}

部分程序输出展示如下：



**2.2.3 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） 本题，可使用累加的计算方法从而避免手动计算大量的公式，简单而直接。

若使用if语句：我们便注意到：

x>=5000的部分(x-4999)应该征收25%的税金，为保证阶梯计税，x=4999

x>=4000的部分(x-3999)应该征收20%的税金，为保证阶梯计税，x=3999

x>=3000的部分(x-2999)应该征收15%的税金，为保证阶梯计税，x=2999

x>=2000的部分(x-1999)应该征收10%的税金，为保证阶梯计税，x=1999

x>=1000的部分(x-999)应该征收5%的税金，为保证阶梯计税，x=999

若使用switch语句：我们注意到case必须是整数值，则可利用int(x/1000)作为参量值，使用deflaut语句避免x>=6000，的情况，我们仍然要使用累加的方法，所以我们从大到小排列case，并不break，令其一路运算到底。

2）源程序清单

/\*salaryCalc\_ifVersion\*/

#include <stdio.h>

int main () {

int x = 0; double tax = 0;

scanf("%d", &x);

if(x >= 5000) {

tax += 0.25\*(x - 4999);

x = 4999;

}

if(x >= 4000) {

tax += 0.2\*(x - 3999);

x = 3999;

}

if(x >= 3000) {

tax += 0.15\*(x - 2999);

x = 2999;

}

if(x >= 2000) {

tax += 0.1\*(x - 1999);

x = 1999;

}

if(x >= 1000) {

tax += 0.05\*(x - 999);

x = 999;

}

printf("tax = %lf\n", tax);

return 0;

}

/\* salaryCalc\_switchVersion \*/

#include <stdio.h>

int main () {

int x = 0; double tax = 0;

scanf("%d", &x);

switch (x/1000) {

default:

case 5:tax += 0.25\*(x - 4999);x = 4999;

case 4:tax += 0.2\*(x - 3999); x = 3999;

case 3:tax += 0.15\*(x - 2999);x = 2999;

case 2:tax += 0.1\*(x - 1999); x = 1999;

case 1:tax += 0.05\*(x - 999); x = 999;

case 0: break;

}

printf("tax = %lf\n", tax);

return 0;

}

3）测试

（a） 测试数据：

600

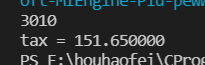
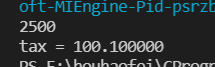
2500

3010

6000

（b） 对应测试数据的运行结果截图

/\*salaryCalc\_ifVersion\*/



/\* salaryCalc\_switchVersion \*/



（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。

**解答：**

1)解题思路：

1.定义一个字符数组s用以存储读入的字符，数字n作为短文长度，flag表示前一个是否是空格。

2.如果n==0或者s[n-1]!=’!’，执行2.1，否则跳转3

2.1 读入一个字符ch

2.2 如果字符为空格，执行2.2.1，否则执行2.3

2.2.1若flag=1,跳转至2，开始新一轮工作流程。否则跳转2.2.2

2.2.2 s[n++]=ch，flag=1；

2.3 s[n++]=ch，flag=0；

3. 显示输出s；

4. 结束

2）程序清单

#include <stdio.h>

int main () {

char s[1000];int top=0;int flag = 0;

while(s[top - 1]!='!' || top == 0) {

char now = getchar();

if(now == ' ') {

if(flag)

continue;

else {

flag = 1;

s[top++] = now;

}

}

else {

s[top++] = now;

if(flag)

flag = 0;

}

}

s[top] = 0;

printf("%s",s);

return 0;

} 3）测试

（a） 测试数据：

根据题目要求以及各种可能出现的错误情况构造数据如表1-1所示。

表2-1 编程题2的测试数据（以’#’表示空格符号）

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 | 运 行 结 果 |
| 输入文本 |
| 用例1 | I#love###you  Just##like#fish####like  ####water  ! | I#love#you  Just#like#fish#like  #water  ! | I#love#you  Just#like#fish#like  #water  ! |
| 用例2 | Mary#had#a#little#lamb! | Mary#had#a#little#lamb! | Mary#had#a#little#lamb! |
| 用例3 | Mary#had####a#little#lamb.  ####It#is#a#good#pet#for#Mary###! | Mary#had#a#little#lamb.  #It#is#a#good#pet#for#Mary#! | Mary#had#a#little#lamb.  #It#is#a#good#pet#for#Mary#! |

（b） 对应测试测试用例1的运行结果如图2-2所示。

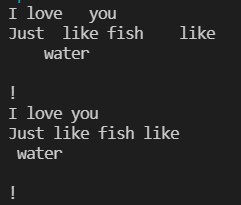


图2-2 编程题2的测试用例一的运行结果

对应测试测试用例2的运行结果如图2-3所示。



图2-3 编程题3的测试用例二的运行结果

对应测试测试用例3的运行结果如图2-4所示。

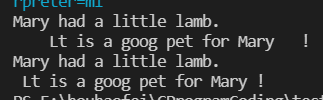


图2-4 编程题3的测试用例三的运行结果

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。

（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

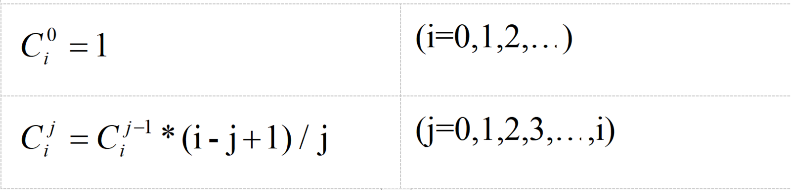
1 6 15 20 15 6 1

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C表示，C的计算如下：



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，每个数字占位4格，例如，1位数后补3个空格。第N行行首是N个空格（每向上一行，行首空格数对应增加）。每行末尾是换行符。

**解答：**

1) 解题思路：

1.输入N.

2.枚举输出的第i行，从0到N:

2.1 输出3N-2i个空格

2.2 定义Cnow =1，并按照加上本身4个格定位输出

2.3 枚举j从1到i

2.3.1 Cnow = Cnow \* (i-j+1) / j;

2.3.2 按照加上本身4个格定位输出

3.结束

2）程序清单

#include <stdio.h>

int main () {

int N;

scanf("%d", &N);

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

for(int j = 1; j <= i; j++)

printf(" ");

for(int j = 1; j <= 3 \* (N - i); j++)

printf(" ");

int Cnow = 1;

printf("%d ",Cnow);

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

Cnow = Cnow \* (i-j+1) / j;

printf("%d",Cnow);

if(Cnow < 10) printf(" ");

else if(Cnow < 100) printf(" ");

else printf(" ");

}

printf("\n");

}

return 0;

} 3）测试

（a） 测试数据：

根据题目要求以及各种可能出现的错误情况构造数据如表1-2所示。

表2-2 编程题3的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 输 入 | 理 论 结 果 | 运 行 结 果 |
| N |
| 例1 | 0 | 1 | 1 |
| 例2 | 3 | 1  1 1  1 2 1  1 3 3 1 | 1  1 1  1 2 1  1 3 3 1 |
| 例3 | 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 |

（b） 对应测试测试用例1的运行结果如图2-5所示。



图2-5 编程题3的测试用例一的运行结果

对应测试测试用例2的运行结果如图2-6所示。

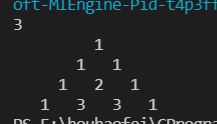


图2-6 编程题3的测试用例二的运行结果

对应测试测试用例3的运行结果如图2-7所示。

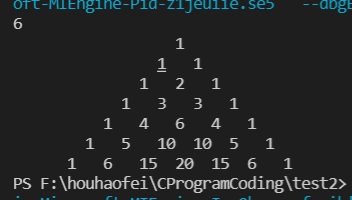


图2-7 编程题3的测试用例三的运行结果

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。

（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. 解题思路：算法流程可用算法流程图表示

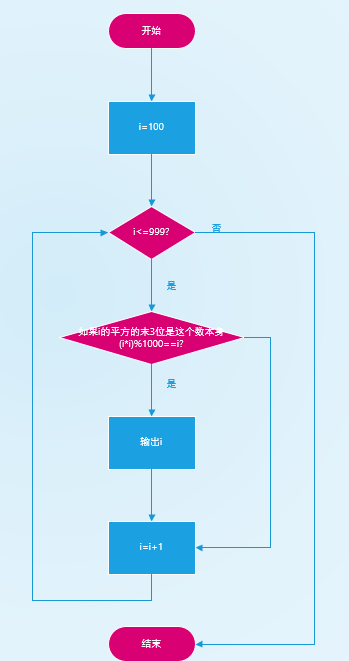


图2-8 编程题4的程序流程图

2）程序清单

#include <stdio.h>

int main () {

for(int i = 100; i < 1000; i++)

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

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

return 0;

}

3）测试

（a） 测试数据：

由数学推导已知，这类数的个数是有限的，可演算得仅2个三位数满足条件，即376和625。

（b） 对应测试的运行结果如图2-8所示。



图2-9 编程题4的测试的运行结果

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。

## 2.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在实验1源程序改错中，采用逐步调试的方法成功找出了语法上的错误，在分析算法复杂度的时候确定了其运算次数过高的第4个问题。最终解决了所有改错。

在修改题1中不知道如何修改实验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。和同学交流过后，明白了单出口的含义，For循环中的break非结构语句破坏了单出口结构，将flag的判断放入for循环的条件控制语句，这样中途如果出现一个模数使其取模为0，即其为合数，便退出循环。既没有破坏单出口的性质，又保证了运算的效率。

在编程题3中，学会了仔细阅读题面然后逐渐分析得出空格的个数解决了问题，明白了学会审题的重要意义。