

**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**

[1 表达式和标准输入输出实验 1](#_Toc123182083)

[1.1 实验目的 1](#_Toc123182084)

[1.2 实验内容 2](#_Toc123182085)

[1.3 实验小结 9](#_Toc123182086)

[2 流程控制实验 9](#_Toc123182087)

[2.1 实验目的 9](#_Toc123182088)

[2.2 实验内容 10](#_Toc123182089)

[2.3 实验小结 25](#_Toc123182090)

[3 函数与程序结构实验 26](#_Toc123182091)

[3.1 实验目的 26](#_Toc123182092)

[3.2 实验内容 26](#_Toc123182093)

[3.3 实验小结 42](#_Toc123182094)

[4 编译预处理实验 43](#_Toc123182095)

[4.1 实验目的 43](#_Toc123182096)

[4.2 实验内容 43](#_Toc123182097)

[4.3 实验小结 52](#_Toc123182098)

[5 数组实验 52](#_Toc123182099)

[5.1 实验目的 52](#_Toc123182100)

[5.2 实验内容及要求 52](#_Toc123182101)

[5.3 实验小结 67](#_Toc123182102)

[6 指针实验 67](#_Toc123182103)

[6.1 实验目的 67](#_Toc123182104)

[6.2 实验题目及要求 67](#_Toc123182105)

[6.3 实验小结 85](#_Toc123182106)

[7 结构与联合实验 86](#_Toc123182107)

[7.1 实验目的 86](#_Toc123182108)

[7.2 实验题目及要求 86](#_Toc123182109)

[7.3 实验小结 104](#_Toc123182110)

[8 文件操作实验 105](#_Toc123182111)

[8.1 实验目的 105](#_Toc123182112)

[8.2 实验题目及要求 105](#_Toc123182113)

[8.2 实验小结 110](#_Toc123182114)

# 1 表达式和标准输入输出实验

## 1.1 实验目的

（1）熟练掌握各种运算符的运算功能，操作数的类型，运算结果的类型及运算过程中的类型转换，重点是C语言特有的运算符，例如位运算符，问号运算符，逗号运算符等。

（2）掌握getchar, putchar, scanf 和printf 函数的用法。

（3）掌握简单C程序的编写方法。

（4）熟悉C语言程序的开发环境，并学会调试程序的方法。

## 1.2 实验内容

**1、程序改错与跟踪调试**

下面的实验1-1程序用来完成以下任务：

（1）输入华氏温度f，将它转换成摄氏温度c后输出。

（2）输入圆的半径值ｒ，计算并输出圆的面积ｓ。

（3）将ｋ的高字节作为结果的低字节，ｐ的高字节作为结果的高字节，拼成一个新的整数后输出。

在这个程序中存在若干语法和逻辑错误，要求先编译程序改正语法错误，再采用单步执行的方式调试程序找出逻辑错误。在单步执行程序的过程中，观察以下变量值：

1. 执行完c = 5/9 \* (f-32)，c的值为多少？
2. 执行完scanf(“%f”, &r)，r的值为多少？
3. 执行完newint = p&0xff00|k>>8， newint的值是多少？表达式k>>8的值是多少？

根据观察结果分析代码并修改程序，使之能够正确完成指定任务。

/\*实验1-1程序改错与跟踪调试题源程序\*/

1 #include<stdio.h>

2 #define PI 3.14159;

3 int main( void )

4 {

5 int f ;

6 short p, k ;

7 double c, r, s ;

/\* 任务1 \*/

8 printf("Input Fahrenheit: " ) ;

9 scanf("%d", f ) ;

10 c = 5/9 \* (f-32) ;

11 printf( "\n %d (F) = %.2f (C)\n\n ", f, c ) ;

/\* 任务2 \*/

12 printf("input the radius r:");

13 scanf("%f", &r);

14 s = PI \* r \* r;

15 printf("\nThe acreage is %.2f\n\n",&s);

/\* 任务3 \*/

16 k = 0xa1b2, p = 0x8432;

17 newint = p&0xff00|k>>8;

18 printf("newint = %#x\n\n",newint);

19 return 0;

20 }

解答：

1. 语法错误

1）第二行#define PI 3.14159; 正确形式为

#define PI 3.14159

2）第九行scanf(“%d”, f ); 正确形式

scanf(“%d”, &f );

3）第十五行printf("\nThe acreage is %.2f\n\n",&s); 正确形式为

printf("\nThe acreage is %.2f\n\n",s);

4）第十七行newint = p&0xff00|k>>8; 正确形式为

short newint = p&0xff00|k>>8;

1. 修改完语法错误后，在单步执行程序的过程中，观察到以下变量值：

1）执行完c = 5/9 \* (f-32)，c的值为0

2）执行完scanf(“%lf”, &r)，r的值为0

3）执行完newint = p&0xff00|k>>8，newint的值是0xffffffa1.表达式k>>8的值是0xa1.

（3）逻辑错误

1）第十行c = 5/9 \* (f-32) ; 正确应改为

c = 5.0/9 \* (f-32);

2）第十三行scanf("%f", &r); 正确形式为

scanf("%lf", &r);

3）第十七行short newint = p&0xff00|k>>8; 正确形式为

unsigned short newint = p&0xff00|k>>8;

4）第六行short p, k ; 正确形式为

unsigned short p, k ;

（4）错误修改后运行结果：

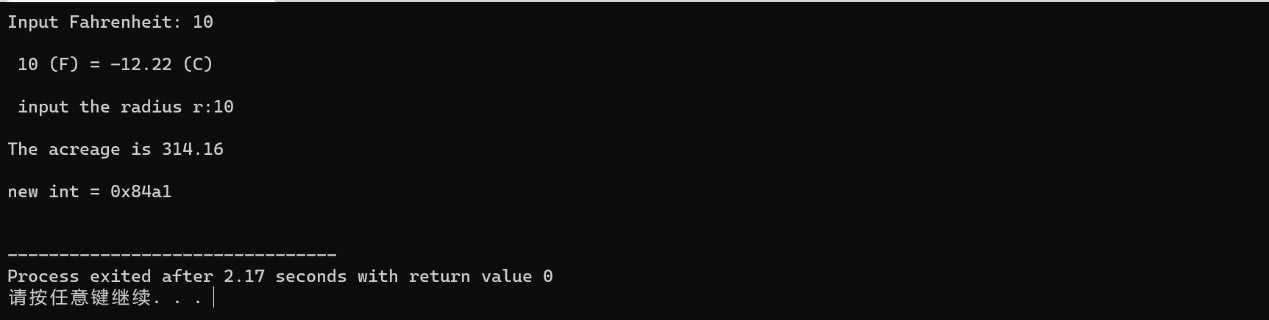


图1-1 实验1-1修改后运行结果

**2、 程序设计**

（1）输入字符ｃ，如果ｃ是大写字母，则将ｃ转换成对应的小写，否则ｃ的值不变，输入Ctrl+Z程序结束。要求①用条件表达式；②字符的输入输出用getchar和putchar函数。程序应能循环接受用户的输入，直至输入Ctrl+Z程序结束。

解答：

1. 算法流程如图2-1所示。

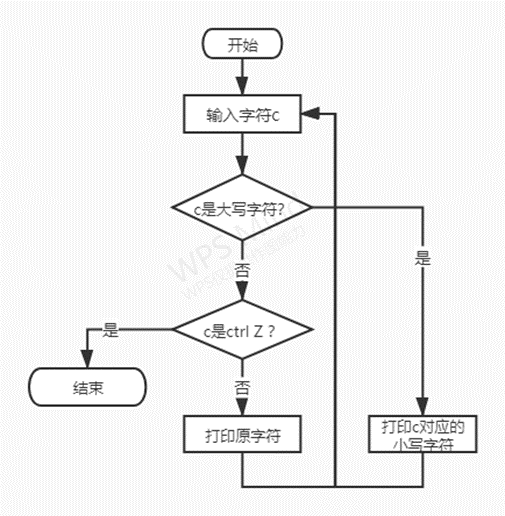


图2-1 程序设计（1）程序流程图

1. 源程序清单

#include<stdio.h>

int main(){

char c = 0;

while(1){

c = getchar();

if (c >= 65 && c <= 90){

putchar(c+32);

}

else{

if (c == EOF)

break;

else

putchar(c);

}

}

return 0;

}

1. 测试
2. 测试数据：

测试数据如表2-1所示：

表2-1程序设计（1）测试数据

|  |  |  |
| --- | --- | --- |
| 输入数据 | 预计输出 | 实际输出 |
| A | A | A |
| A | a | a |
| Ctrl + Z | 结束 | 结束 |

1. 对应测试数据的运行结果如图2-2所示：

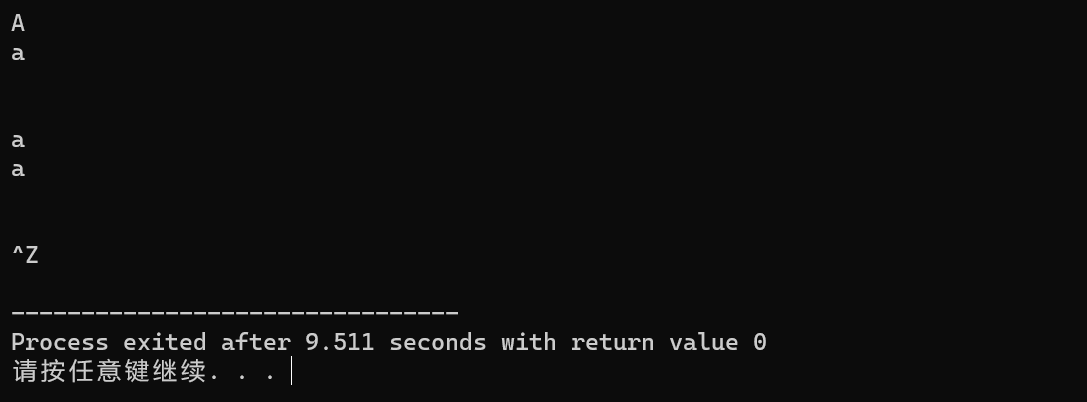


图2-2程序设计（1）运行结果

（2）输入无符号短整数x，ｍ，ｎ（0 ≤ｍ≤ 15, 1 ≤ ｎ≤ 16-ｍ）,取出x从第ｍ位开始向左的ｎ位（ｍ从右至左编号为0～15），并使其向左端（第15位）靠齐。要求：①检查m和n的范围；②x的值以十六进制输入，m和n以十进制输入；③结果以十六进制输出。

解答：

1. 解题思路如图2-3所示：

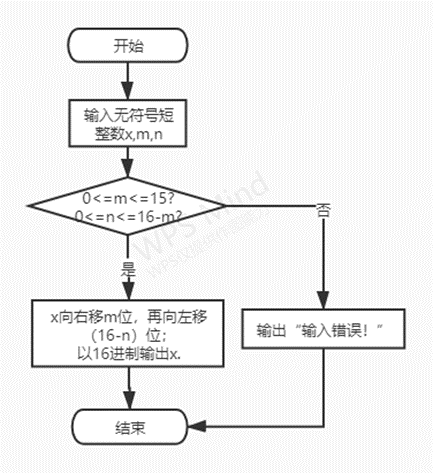


图2-3程序设计（2）程序流程图

1. 源程序清单

#include<stdio.h>

int main(){

unsigned short x,m,n,q;

scanf("%hx %hd %hd",&x,&m,&n);

int a = 16-m;

if ((m > 15 || m < 0) || (n < 1 || n > a)){

printf("error");

}

else{

printf("%hx",x<<=(16-m-n));

}

return 0;

}

1. 测试
2. 测试数据

测试数据如表2-2所示：

表2-2程序设计（2）测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | | | 理 论 结 果 |
| X | m | n |
| 用例1 | 0100 0110 1000 0000（4680） | 7 | 4 | 1101 0000 0000 0000 即d000 |
| 用例2 | d583 | 16 | 3 | error（m值超范围） |
| 用例3 | d583 | 13 | 5 | error（n值超范围） |

1. 对应测试数据的运行结果如图2-4所示：



图2-4程序设计（2）运行结果

（3）IP地址通常是4个用句点分隔的小整数（即点分十进制），但这些地址在机器中是用一个无符号长整型数表示的。例如3232235876，其机内二进制表示就是11000000 10101000 00000001 01100100，按照8位一组用点分开，该IP地址就写成192.168.1.100。

读入无符号长整型数表示的互联网IP地址，对其译码，以常见的点分十进制形式输出。要求循环输入和输出，直至输入Ctrl+Z结束。

解答：

1. 解题思路如图2-5所示：

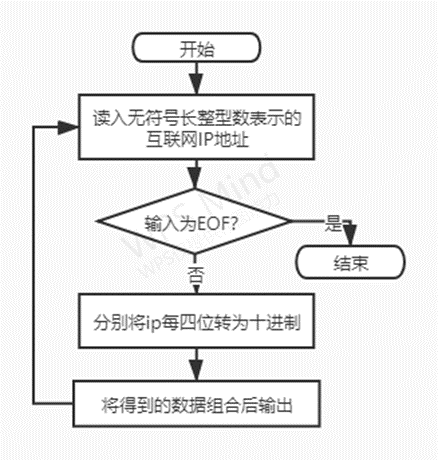


图2-5程序设计（3）程序流程图

1. 源程序清单

#include<stdio.h>

int main(){

unsigned long ip,p1,p2,p3,p4;

while(scanf("%lu",&ip) != EOF){

p1 = ip >> 24;

p2 = (ip & 0x00ff0000) >> 16;

p3 = (ip & 0x0000ff00) >> 8;

p4 = ip & 0x000000ff;

printf("%d.%d.%d.%d\n",p1,p2,p3,p4);

}

return 0;

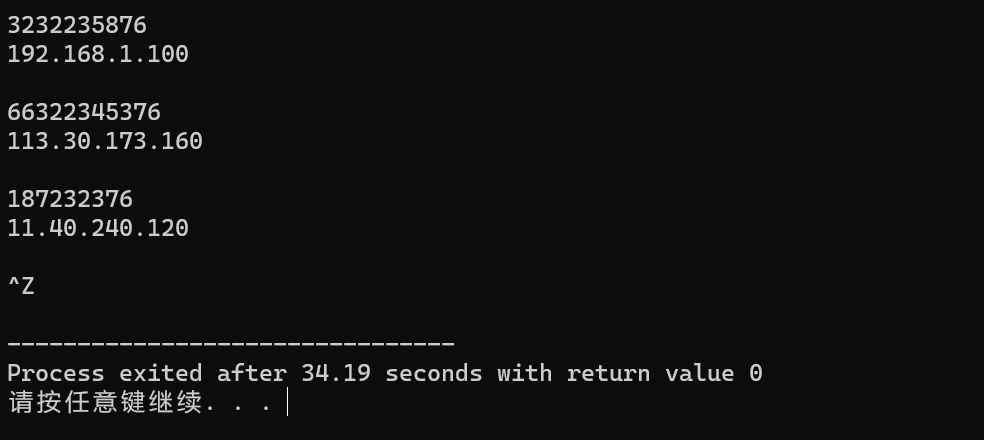
}

1. 测试
2. 测试数据如表2-3所示：

表2-3程序设计（3）测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 实际输出 |
| 用例1 | 3232235876 | 192.168.1.100 |
| 用例2 | 66322345376 | 113.30.173.160 |
| 用例3 | 187232376 | 11.40.240.120 |
| 用例4 | ^Z | 结束 |

（b）对应测试数据的运行结果如图2-6所示：

图2-6程序设计（3）运行结果

## 1.3 实验小结

本次实验主要练习了输入输出函数的用法，并学习了几种表达式的运用（如getchar()、putchar()、scanf()、printf()及一些位运算等），在C语言编程学习过程中有重要的意义。

本次实验遇到的问题主要在于不了解位运算知识，导致程序编写一度难以推进，后来通过翻看《C语言程序设计》学习了相关内容，问题亦迎刃而解。

本次实验最大的体会是，将程序实现内容与需求有条理地记录下来是极为重要的（具体体现在绘制流程图），并且每一次实验都需要提前预习，以给程序编写、报告撰写、整理反思留出更多时间。

# 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

**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);
  15. }
  16. return 0;
  17. }

解答：

1. 语法错误：
   * + 1. 第8行 if (!x%i)，正确形式为：

if (!(x%i))

* + - 1. 第12行 if(flag=1)，正确形式为：

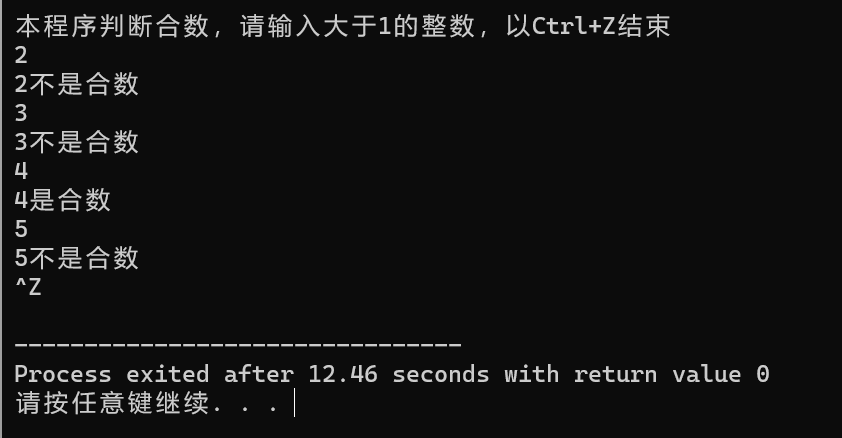
if(flag==1)

1. 逻辑错误：

第14行，应加上变量归零语句：

flag = 0;

1. 修改后运行结果如下：



**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。

替换后的程序如下：

#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;(x%i)&&(i<=k);i++){}

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

if(flag==1) printf("%d是合数", x);

else printf("%d不是合数", x);

flag = 0;

}

return 0;

}

运行结果如下：



（2）修改实验2-1程序，将for循环改用do-while循环。

替换后的程序如下：

#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) {

i = 2;

k = x >>1;

if(x==2) printf("%d不是合数", x);

else{

do{

if(!(x%i)){

flag = 1;

break;

}

i++;

}while(i<=k);

if(flag==1) printf("%d是合数", x);

else printf("%d不是合数", x);

}

flag = 0;

}

return 0;

}

运行结果如下：



（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。

替换后的程序如下：

#include <stdio.h>

int main( )

{

int num,a,b,c,d,i;

printf("本程序输出所有三位纯粹合数，如下：\n");

for(num = 100;num <= 999;num++){

//divide num

a = num/100;

b = (num-(100\*a))/10;

//is num composite number?

for(i=2,c=num>>1;i<=c;i++){

if((num%i)==0){

//leave a,b

d = (10\*a)+b;

for(i=2,c=d>>1;i<=c;i++){

if((d%i)==0){

//leave a

for(i=2,c=a>>1;i<=c;i++){

if((a%i)==0){

printf("%d ",num);

break;

}

}

break;

}

}

break;

}

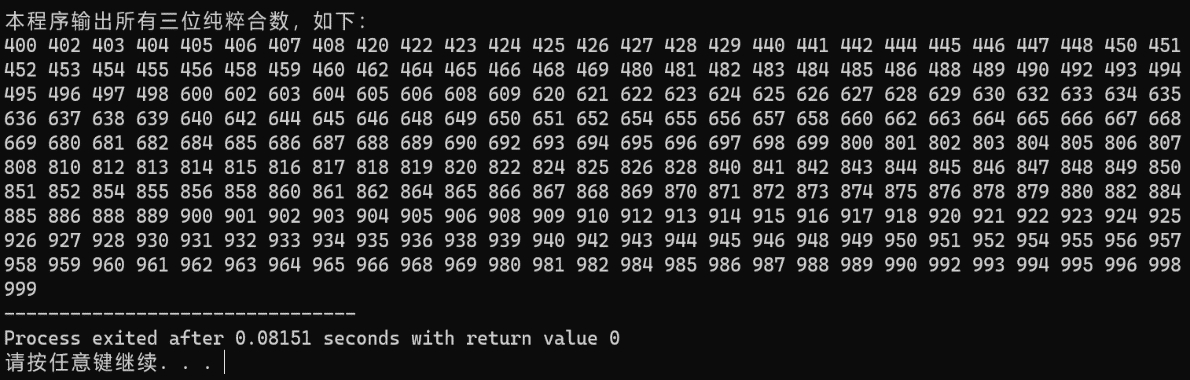
}

}

return 0;

}

运行结果如下：



**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：

a）算法流程如图3-1所示：

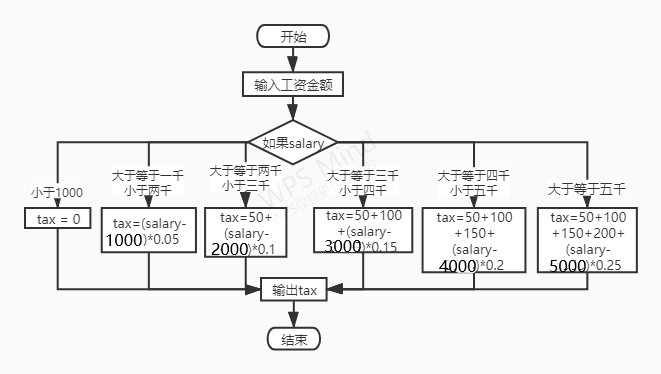


图3-1程序设计（1）流程图

b）源程序清单

if语句源程序代码：

#include<stdio.h>

int main(){

float wage,a;

printf("请输入工资：");

scanf("%f",&wage);

if(wage<1000){

printf("应收税金0元");

}

else{

if(wage<2000){

a = (wage-1000)\*0.05;

printf("应收税金%.1f元",a);

}

else{

if(wage<3000){

a = (wage-2000)\*0.1+50;

printf("应收税金%.1f元",a);

}

else{

if(wage<4000){

a = (wage-3000)\*0.15+50+100;

printf("应收税金%.1f元",a);

}

else{

if(wage<5000){

a = (wage-4000)\*0.2+50+100+150;

printf("应收税金%.1f元",a);

}

else{

a = (wage-5000)\*0.25+50+100+150+200;

printf("应收税金%.1f元",a);

}

}

}

}

}

}

switch语句源程序代码：

#include<stdio.h>

int main(){

float wage=0,tax=0;

char a;

scanf("%f",&wage);

if(wage<1000) a='a';

if(wage<2000&&wage>=1000) a='b';

if(wage<3000&&wage>=2000) a='c';

if(wage<4000&&wage>=3000) a='d';

if(wage<5000&&wage>=4000) a='e';

if(wage>=5000) a='f';

switch(a){

case 'a':

tax = 0;

printf("应收税金%f元",tax);

break;

case 'b':

tax = (wage-1000)\*0.05;

printf("应收税金%f元",tax);

break;

case 'c':

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

printf("应收税金%f元",tax);

break;

case 'd':

tax = (wage-3000)\*0.15+50+100;

printf("应收税金%f元",tax);

break;

case 'e':

tax = (wage-4000)\*0.2+50+100+150;

printf("应收税金%f元",tax);

break;

case 'f':

tax = (wage-5000)\*0.25+50+100+150+200;

printf("应收税金%f元",tax);

break;

default:

printf("error");

}

}

c）测试

i．测试用例如表3-1所示：

表3-1 程序设计（1）测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | 900 | 0 |
| 用例2 | 4500 | 400.0 |
| 用例3 | 6000 | 750.0 |

ii．对应测试数据的运行结果如图3-2所示：





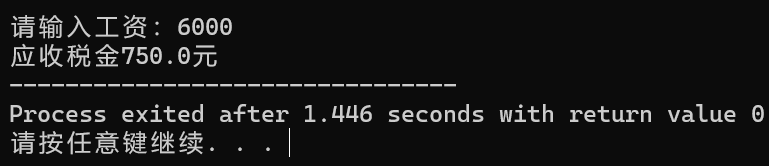


图3-2 程序设计（1）运行结果

(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. 算法流程如图3-3所示：

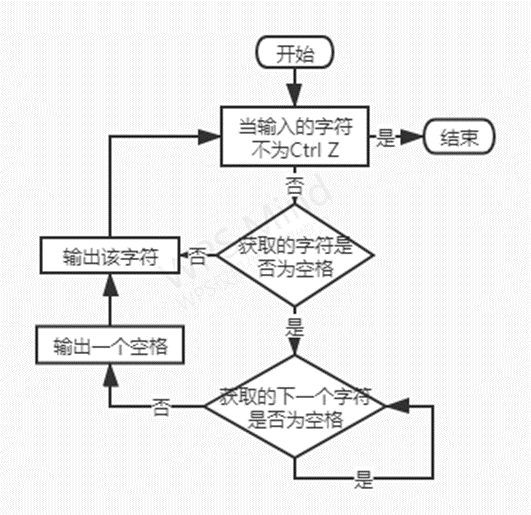


图3-3程序设计（2）流程图

1. 源程序清单

#include <stdio.h>

int main(){

int i;

char input;

while((input = getchar()) != EOF){

if (input != ' '){

i = 0;

putchar(input);

}

else if(input == ' '){

if(i == 0){

putchar(input);

i++;

}

}

}

return 0;

}

1. 测试
2. 测试数据：

测试用例如表3-2所示：

表3-2 程序设计（2）测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | □□□abc□□□12 | □abc□12 |
| 用例2 | □□□□□□□a | □a |
| 用例3 | 1□□□2□□□3 | 1□2□3 |
| 用例4 | ^Z | 结束 |

1. 对应测试用例的运行结果如图3-4所示：

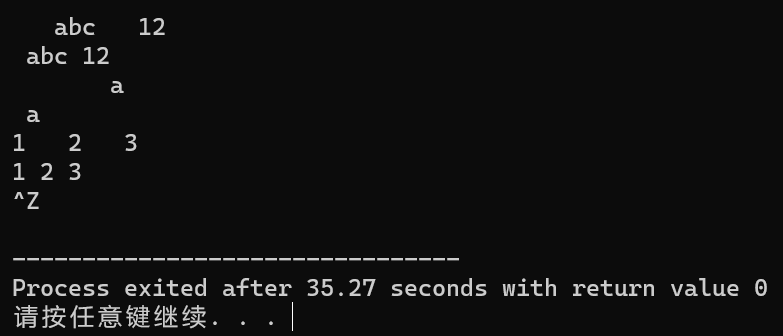


图3-4 程序设计（2）运行结果

（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（每向上一行行首增加两个空格）。每行末尾是3个空格和换行。

解答：

1. 算法流程如图3-5所示：

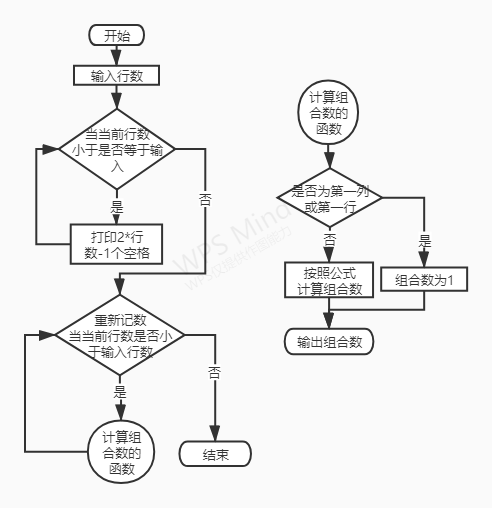


图3-5 程序设计（3）流程图

1. 源程序清单

#include<stdio.h>

int ij(int i,int j);

int blank(int N,int i);

int main() {

int i=0,j=0,N,n;

scanf("%d",&N);

while(i<=N){

while(n<blank(N,i)){

printf(" ");

n++;

}

n = 0;

while(j<=i){

if(ij(i,j)<10){

printf("%d ",ij(i,j));

}

else if(ij(i,j)<100){

printf("%d ",ij(i,j));

}

else if(ij(i,j)<1000){

printf("%d ",ij(i,j));

}

j++;

}

j = 0;

i++;

printf("\n");

}

return 0;

}

int blank(int N,int i) {

int sum,n;

sum = 2\*(N-i)+N;

return sum;

}

int ij(int i,int j) {

int output,mol=1,den=1,a;

a = j;

if(i == 0){

output = 1;

}

else{

while(a>0){

mol \*= i;

i--;

a--;

}

}

if(j == 0){

output = 1;

}

else{

while(j>0){

den \*= j;

j--;

}

output = mol/den;

}

return output;

}

1. 测试
2. 测试数据

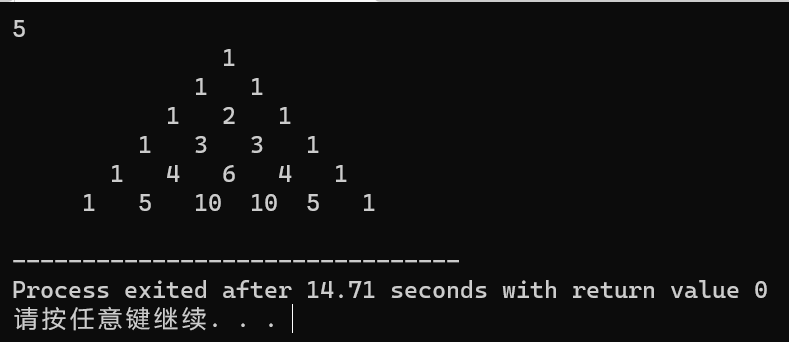
测试用例如表3-3所示：

表3-3 程序设计（3）测试数据

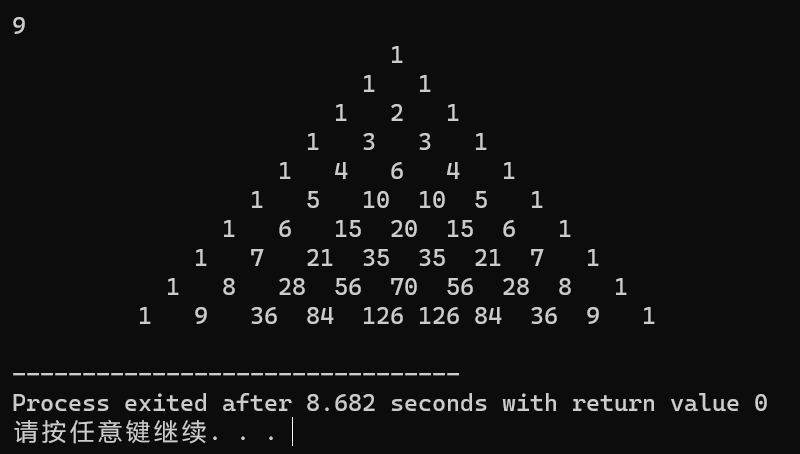
|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试数据 | 预计输出 |
| 用例1 | 5 |  |
| 用例2 | 9 |  |
| 用例3 | 10 |  |

1. 对应测试用例的运行结果如下图所示：

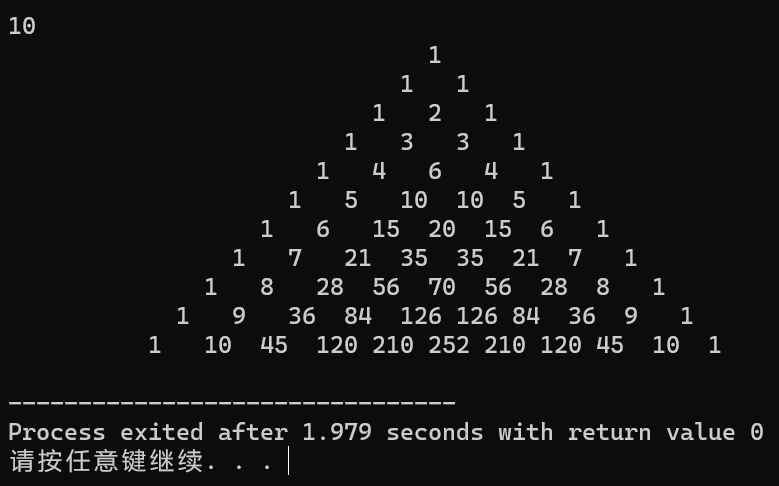
用例1：



用例2：



用例3：



（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3-6所示：

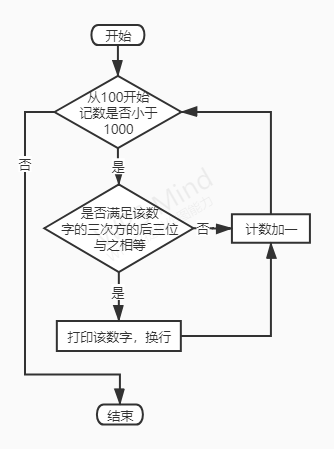


图3-6 程序设计（4）流程图

1. 源程序清单

#include<stdio.h>

int main(){

long end,square;

for(int num=100;num<1000;num++){

square = num\*num;

end = square % 1000;

if(end == num) {

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

}

}

return 0;

}

1. 测试

预期输出结果：

376

625

对应预期输出结果的运行结果如图3-7所示：

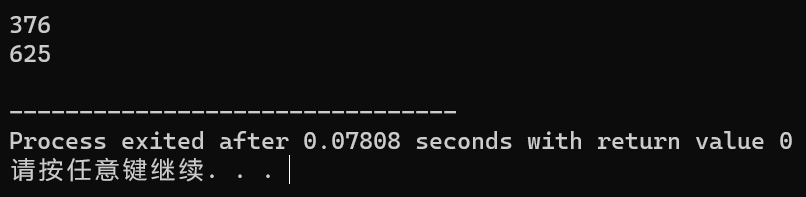


图3-7 程序设计（4）运行结果

2.3 实验小结

通过此次实验，我充分认识到了自定义函数的必要性与便利性，也弥补了自身一些C语言基础知识的缺失，更明白了自己多层if嵌套结构极差的可读性，并下定决心从基层做起养成良好的编程习惯和优雅的编程风格。

另外，我还学习到了通过处理数据来进行switch分层分段的手法，收益颇丰。

# 3 函数与程序结构实验

3.1 实验目的

（1）熟悉和掌握函数的定义、声明；函数调用与参数传递，函数返回值类型的定义和返回值使用。

（2）熟悉和掌握不同存储类型变量的使用。

（3）练习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。

3.2 实验内容

**1．程序改错题**

下面是计算s=1!+2!+3!+…+n!的源程序(n<20)。在这个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，使之能够输出如下结果：

k=1 the sum is 1

k=2 the sum is 3

k=3 the sum is 9

……

k=20 the sum is 2561327494111820313

/\*实验3-1改错题程序：计算s=1!+2!+3!+…+n!\*/

1. #include <stdio.h>
2. int main(void)
3. {
4. int k;
5. for(k=1;k<=20;k++)
6. printf("k=%d\tthe sum is %ld\n",k,sum\_fac(k));
7. return 0;
8. }
9. long sum\_fac(int n)
10. {
11. long s=0;
12. int i,fac;
13. for(i=1;i<=n;i++)
14. fac\*=i;
15. s+=fac;
16. return s;
17. }

解答：

1. 语法错误：
   1. 第二行应声明自定义函数，正确形式为：

long sum\_fac(int n);

1. 逻辑错误：
   1. 第六行占位符使用错误，正确形式为：

printf("k=%d\tthe sum is %lld\n",k,sum\_fac(k));

* 1. 第十行自定义函数返回类型有误，正确形式为：

long long sum\_fac(int n)

* 1. 第十二行定义s应为静态变量，且变量类型有误，正确形式为：

static long long s=0;

* 1. 第十三行定义fac的变量类型有误，正确形式为：

long long fac；

1. 错误修改后的运行结果如图1-1所示：

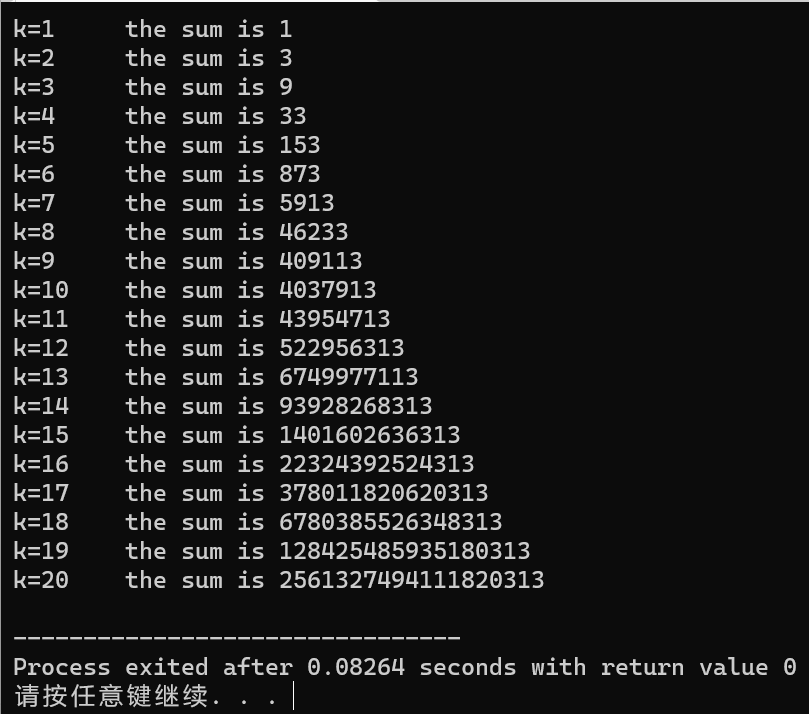


图1-1 程序改错题1修改后运行结果

**2．程序修改替换题**

（1）根据将实验3-1改错题程序中sum\_fac函数修改为一个递归函数，用递归的方式计算。

解答：修改后的sum\_fac函数如下：

long long sum\_fac(int n)

{

long long fac=1;

int i=1;

for(i=1;i<=n;i++)

fac\*=i;

if(n!=1){

fac += sum\_fac(n-1);

}

return fac;

}

替换后程序运行结果如图2-1所示：

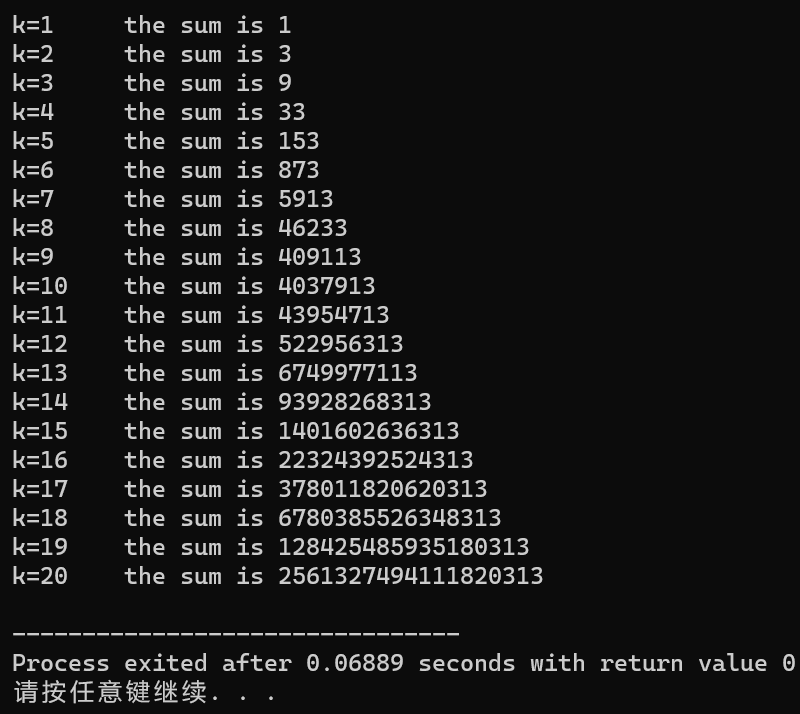


图2-1 sum\_fac函数替换后运行结果

（2）下面是计算的源程序，其中x是浮点数，n是整数。从键盘输入x和n，然后计算s的值。修改该程序中的sum和fac函数，使之计算量最小。

/\*实验3-2程序修改替换第(2)题程序：根据公式计算 s\*/

#include<stdio.h>

double mulx(double x,int n);

long fac(int n);

double sum(double x,int n)

{

int i;

double z=1.0;

for(i=1;i<=n;i++)

{

z=z+mulx(x,i)/fac(i);

}

return z;

}

double mulx(double x,int n)

{

int i;

double z=1.0;

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

{

z=z\*x;

}

return z;

}

long fac(int n)

{

int i;

long h=1;

for(i=2;i<=n;i++)

{

h=h\*i;

}

return h;

}

int main()

{

double x;

int n;

printf("Input x and n:");

scanf("%lf%d",&x,&n);

printf("The result is %lf:",sum(x,n));

return 0;

}

解答：

（1）修改后的程序源代码清单如下：

#include<stdio.h>

double mulx(double x,int n);

long fac(int n);

double sum(double x,int n) {

int i;

double z=1.0,y=1.0;

y = mulx(x,1);

for(i=1; i<=n; i++) {

z=z+y/fac(i);

y\*=x;

}

return z;

}

double mulx(double x,int n) {

int i;

double z=1.0;

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

{

z=z\*x;

}

return z;

}

long fac(int n) {

static long h=1;

h \*= n;

return h;

}

int main() {

double x;

int n;

printf("Input x and n:");

scanf("%lf%d",&x,&n);

printf("The result is %lf:",sum(x,n));

return 0;

}

（2）测试数据如表2-1所示：

表2-1 程序改错与替换题2测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序输入 | | 预计输出 |
| x | n |
| 用例1 | 3 | 3 | 13.000000 |
| 用例2 | 4 | 5 | 42.866667 |

（3）对应测试数据的运行结果如图2-2所示：



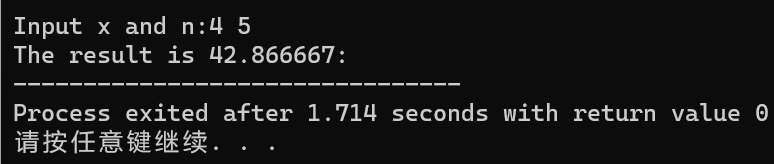


图2-2 程序改错与替换题2运行结果

**3．跟踪调试题**

下面是计算fabonacci数列前n项和的源程序，现要求单步执行该程序，在watch窗口中观察Ik,sum,n值。具体操作如下：

（1）设输入5，观察刚执行完“scanf("%d",&k);”语句时，sum、k的值是多少？

（2）在从main函数第一次进入fabonacci函数前的一刻，观察各变量的值是多少？返回后光条停留在哪个语句上？

（3）在从main函数第一次进入fabonacci函数后的一刻，观察光条从main函数“sum+=fabonacci(i);”语句调到了哪里？

（4）在fabonacci函数内部单步执行，观察函数的递归执行过程。体会递归方式实现的计算过程是如何完成数计算的，并特别注意什么时刻结束递归，然后直接从第一个return语句返回到了哪里？

（5）在fabonacci函数递归执行过程中观察参数n的变化情况，并回答为什么k、sum在fabonacci函数内部不可见？

**/\*实验3-3跟踪调试题程序：**计算fabonacci数列前n项和**\*/**

#include<stdio.h>

int main(void)

{

int i,k;

long sum=0,fabonacci(int n);

printf("Inut n:");

scanf("%d",&k);

for(i=1;i<=k;i++){

sum+=fabonacci(i);

printf("i=%d\tthe sum is %ld\n",i,sum);

}

return 0;

}

long fabonacci(int n)

{

if(n==1 || n==2)

return 1;

else

return fabonacci(n-1)+fabonacci(n-2);

}

解答：

1. 输入5，刚执行完“scanf("%d",&k);”语句时，sum的值是0，k的值是5；
2. 在从main函数第一次进入fabonacci函数前的一刻，各变量的值分别为：sum=0，k=5，i=1，返回后光条停留在sum+=fabonacci(i);语句上；
3. 在从main函数第一次进入fabonacci函数后的一刻，观察光条从main函数“sum+=fabonacci(i);”语句调到了if(n==1 || n==2)上；
4. 在fabonacci函数内部单步执行，观察函数的递归执行过程，注意到n=1或2时结束递归，然后直接从第一个return语句返回到了printf("i=%d\tthe sum is %ld\n",i,sum);上；
5. 在fabonacci函数递归执行过程中观察参数n的变化情况， k、sum在fabonacci函数内部不可见是因为k、sum是定义在main函数中的变量，其作用域只在main函数内，在fabonacci函数中未作外部变量引用声明。

**4．程序设计**

以下（1）至（3）题对应Educoder教学平台 “C语言实验”课程，实验3，第7关实验3-1、第8关实验3-2，以及第9关实验3-3。

（1）编程验证歌德巴赫猜想：任何一个大于等于4的偶数都是两个素数之和。要求设计一个函数，接受形参n，以“n=n1+n2”的形式输出结果，若有多种分解情况，取n1最小的一个输出。

例如：n=6，输出“6=3+3”。

main函数循环接收从键盘输入的整数n，如果n是大于或等于4的偶数，调用上述函数进行验证。

解答：

* + 1. 解题思路：

程序设计（1）流程图如图4-1所示：

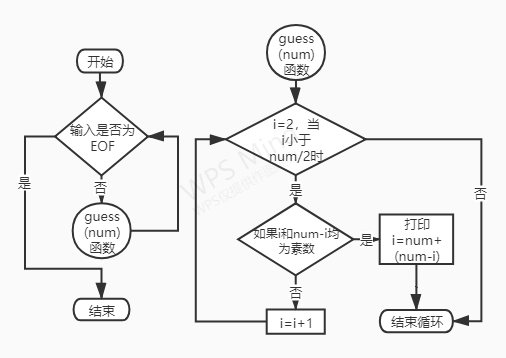


图4-1 程序设计（1）程序流程图

* + 1. 源程序清单

#include <stdio.h>

int IsPrimeNumber(int num);

int main()

{

int input,i,remain;

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

if(input<4){

printf("error\n");

}

else{

for(i=2;i<=input>>1;i++){

remain = input - i;

if(IsPrimeNumber(i)+IsPrimeNumber(remain)==0){

printf("%d=%d+%d\n",input,i,remain);

break;

}

}

}

}

return 0;

}

int IsPrimeNumber(int num){

int i,lim,judge=0;

for(i=2,lim=num>>1;i<=lim;i++){

if((num%i)==0){

judge=1;

break;

}

}

return judge;

}

* + 1. 测试

1. 测试数据：

程序设计（1）测试数据如表4-1所示：

表4-1 程序设计（1）测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 预计输出 |
| 用例1 | 6 | 6=3+3 |
| 用例2 | 8 | 8=3+5 |
| 用例3 | ^Z | 结束 |

1. 对应测试数据的运行结果如图4-2所示：

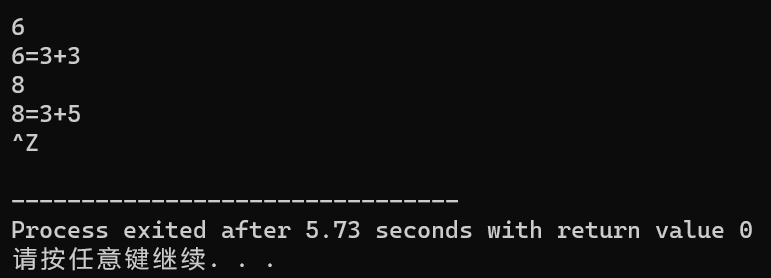


图4-2 程序设计（1）运行结果

（2）完全数（Perfect number），又称完美数或完备数，特点是它的所有真因子（即除了自身以外的约数，包括1）之和恰好等于它本身。例如6=1+2+3，28=1+2+4+7+14等。

编程寻找10000以内的所有完全数。

要求设计一个函数，判定形参n是否为完全数，如果是，返回1，否则返回0。在main函数中调用该函数求10000以内的所有完全数，并以完全数的真因子之和的形式输出结果，例如“6=1+2+3”。程序输出中，每个完全数单独占一行。

解答：

1. 解题思路：

程序设计（2）流程图如图4-3所示：

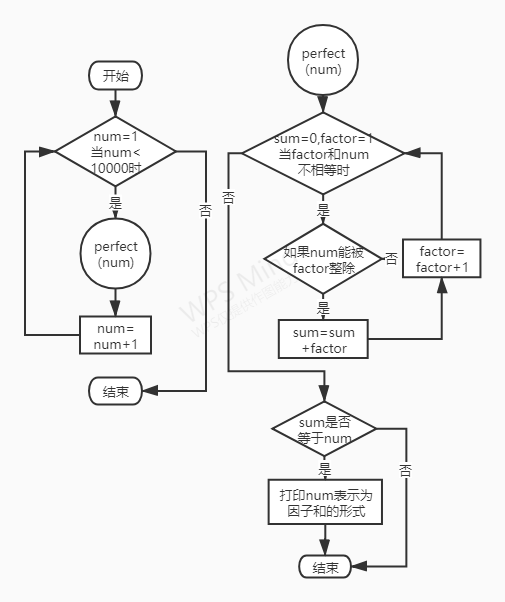


图4-3 程序设计（2）流程图

1. 源程序清单

#include<stdio.h>

int IsPerfectNumber(int num);

int IsPerfectNumber(int num){

int sum=0,factor;

for(factor=1;factor<num;factor++){

if(num%factor==0) sum+=factor;

}

if(num==sum) return 1;

else return 0;

}

int main(){

int num,factor,i=0;

for(num=2;num<=10000;num++){

if(IsPerfectNumber(num)==1){

if(i==0) {

printf("%d=1",num);

i++;

}

else printf("\n%d=1",num);

for(factor=2;factor<num;factor++){

if(num%factor==0) printf("+%d",factor);

}

}

}

return 0;

}

1. 测试

程序设计（2）运行结果如图4-4所示：

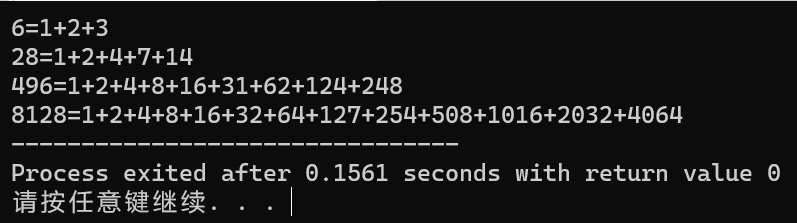


图4-4 程序设计（2）运行结果

（3）自幂数是指一个n位数，它的每个位上的数字的n次幂之和等于它本身。水仙花数是3位的自幂数，除此之外，还有4位的四叶玫瑰数、5位的五角星数、6位的六合数、7位的北斗星数、8位的八仙数等。

编写一个函数，判断其参数n是否为自幂数，如果是，返回1；否则，返回0。要求main函数能反复接收从键盘输入的整数k，k代表位数，然后调用上述函数求k位的自幂数，输出所有k位自幂数，并输出相应的信息，例如“3位的水仙花数共有4个153，370，371，407”。当k=0时程序结束执行。

解答：

1. 解题思路：

程序设计（3）流程图如图4-5所示：

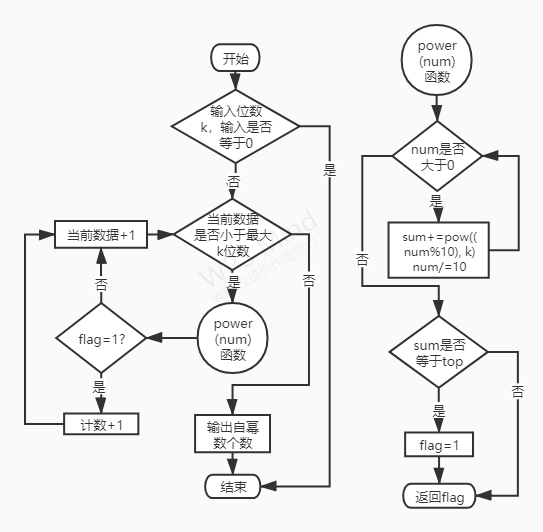


图4-5 程序设计（3）流程图

1. 源程序清单

#include<stdio.h>

int IsSelfPoweredNumber(int n,int num);

int IsSelfPoweredNumber(int n,int num){

int sum=0,alter1=num,alter2=num,i,j,power=1;

for (i=1;i<=n;i++){

alter1 %= 10;

for (j=1;j<=n;j++){

power \*=alter1;

}

sum += power;

power = 1;

alter2 /= 10;

alter1 = alter2;

}

if (sum==num) return 1;

else return 0;

}

int main(){

int num,n,count=0,i,min=1,judge=0;

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

if(n==3) printf("3位的水仙花数共有");

else if(n==4) printf("4位的四叶玫瑰数共有");

else if(n==5) printf("5位的五角星数共有");

else if(n==6) printf("6位的六合数共有");

else if(n==7) printf("7位的北斗星数共有");

else if(n==8) printf("8位的八仙数共有");

else if(n==0) break;

for (i=1;i<n;i++){

min \*=10;

}

for (num=min;num<10\*min;num++){

if (IsSelfPoweredNumber(n,num)==1){

count++;

}

}

printf("%d个",count);

for (num=min;num<10\*min;num++){

if (IsSelfPoweredNumber(n,num)==1){

if (judge == 0){

printf("%d",num);

judge++;

}

else printf(",%d",num);

}

}

min = 1;

judge = 0;

count = 0;

printf("\n");

}

return 0;

}

1. 测试
2. 测试数据

程序设计（3）测试用例如表4-2所示：

表4-2 程序设计（3）测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 预计输出 |
| 用例1 | 3 | 3位的自幂数共有4个:153 370 371 407 |
| 用例2 | 4 | 4位的自幂数共有3个:1634 8208 9474 |
| 用例3 | 0 | 结束 |

1. 对应测试用例的运行结果如图4-6所示：

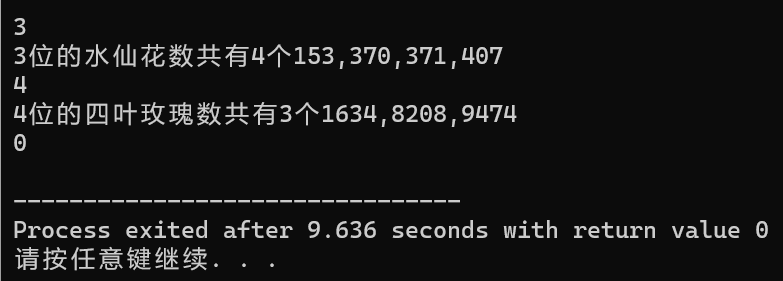


图4-6 程序设计（3）运行结果

3.3 实验小结

在减少计算量时，可以考虑使用static静态变量减少重复计算；通过自定义函数，将不同功能模块进行封装，可以减少编写代码的时间；必要时可以通过断点调试判断某些语句或某个自定义函数中出现的编程错误，方便快捷；另外，通过灵活运用递归不仅可以提高代码可读性，还可以提高程序运行效率。

# 4 编译预处理实验

## 4.1 实验目的

（1）掌握文件包含、宏定义、条件编译和assert宏的使用；

（2）练习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。

（3）熟悉多文件编译技术

4.2 实验内容

**1．程序改错**

下面是用宏来计算平方差、交换两数的源程序.在这个源程序中存在若干错误，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4-1改错与跟踪调试题程序：计算平方差、将换两数\*/

1. #include<stdio.h>
2. #define SUM a+b
3. #define DIF a-b
4. #define SWAP(a,b) a=b,b=a
5. int main()
6. {
7. int a,b;
8. printf("Input two integers a, b:");
9. scanf("%d%d", &a,&b);
10. printf("\nSUM=%d\n the difference between square of a and square of b is:%d",SUM, SUM\*DIF);
11. SWAP(a,b);
12. printf("\nNow a=%d,b=%d\n",a,b);
13. return 0;
14. }

解答：

1. 第2行及第3行，正确形式为：

#define SUM (a+b)

#define DIF (a-b)

1. 第4行，正确形式为：

#define SWAP(a,b) t=a,a=b,b=t

1. 第7行，正确形式为：

int a,b,t;

4）如要使平方差以正数形式表示，可以添加条件语句，不做赘述。

**2．程序修改替换**

下面是用函数实现求三个数中最大数、计算两浮点数之和的程序。在这个源程序中存在若干语法和逻辑错误。

要求：（1）对这个例子程序进行调试修改，使之能够正确完成指定任务；

（2）用带参数的宏替换函数max，来实现求最大数的功能。

/\*实验4-2程序修改替换题程序\*/

1. #include<stdio.h>
2. int main(void)
3. {
4. int a, b, c;
5. float d, e;
6. printf("Input three integers:");
7. scanf("%d %d %d",&a,&b,&c);
8. printf("\nThe maximum of them is %d\n",max(a,b,c));
9. printf("Input two floating point numbers:");
10. scanf("%f %f",&d,&e);
11. printf("\nThe sum of them is %f\n",sum(d,e));
12. return 0;
13. }
14. int max(int x, int y, int z)
15. {
16. int m=z;
17. if (x>y)
18. if(x>z) m=x;
19. else
20. if(y>z) m=y;
21. return m;
22. }
23. float sum(float x, float y)
24. {
25. return x+y;
26. }

解答：

1. 程序修改：
2. 源代码缺少对自定义max函数和sum函数的声明，需在第1行与第2行间加上int max(int x, int y, int z);float sum(float x, float y);
3. 第19行应把if语句用{}括起来，阻断它和第20行else的关联
4. 函数替换：

max函数可替换为 #define max(a,b,c) (a>b)?((a>c)? a:c):((b>c)? b:c)

**3．跟踪调试**

下面程序利用R计算圆的面积s，以及面积s的整数部分。现要求：

（1）修改程序，使程序编译通过且能运行；

（2）单步执行。进入函数integerl\_fraction时，watch窗口中x为何值？在返回main时, watch窗口中i为何值？

（3）修改程序，使程序能输出面积s值的整数部分（要求四舍五入），不会输出错误信息assertion failed。

/\*实验4-3跟踪调试题程序利用R计算圆的面积s\*/

1. #define R
2. int main(void)
3. {
4. float r, s;
5. int s\_integer=0;
6. printf ("Input a number: ");
7. scanf("%f",&r);
8. #ifdef R
9. s=3.14159\*r\*r;
10. printf("Area of round is: %f\n",s);
11. s\_integer=integer\_fraction(s);
12. assert((s-s\_integer)<0.5);
13. printf("The integer fraction of area is %d\n", s\_integer);
14. #endif
15. return 0;
16. }
17. int integer\_fraction(float x)
18. {
19. int i=x;
20. return i;
21. }

解答：

1. 应在第1行前加上头文件引用及自定义函数声明：

#include <stdio.h>

#include <assert.h>

int integer\_fraction(float x);

1. x等于s，而i等于s的整数部分
2. 只需在第11行与第12行之间加入条件判断语句：

if ((s - s\_integer) > 0.5) s\_integer++;

**4．程序设计**

（1）三角形的面积是，其中，a,b,c为三角形的三边，要求编写程序用带参数的宏来计算三角形的面积。定义两个带参数的宏，一个用来求s，另一个用来求area。

解答：

1. 解题思路

利用math.h头文件中的pow()函数进行开方计算；为保证精确度，计算半周长s时应采用浮点数计算以保证精确度。

1. 源程序清单

#include <stdio.h>

#include <math.h>

#define S(a, b, c) (a + b + c)

#define Area(s, a, b, c) (s \* (s - a) \* (s - b) \* (s - c))

int main()

{

float a, b, c;

float s, area;

while (scanf("%f %f %f", &a, &b, &c)!=EOF){

s = S(a, b, c) / 2;

area = sqrt(Area(s, a, b, c));

printf("%f\n", area);

}

return 0;

}

1. 测试
2. 测试数据

程序设计（1）的测试数据如表4-1所示

表4-1 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | 3，4，5 | 6.000000 |
| 用例2 | 1，1，2 | 0.000000 |
| 用例3 | ^Z | 结束 |

1. 对应测试用例的运行结果如图4-1所示

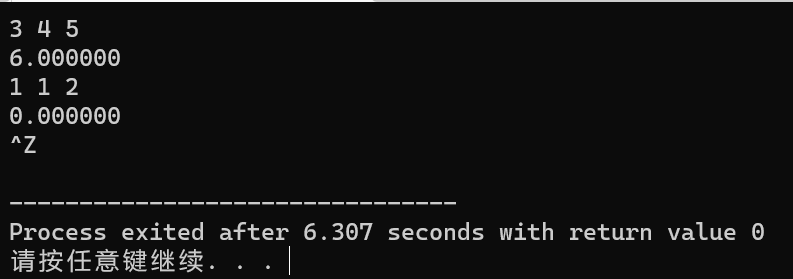


图4-1 程序设计（1）运行结果

1. 用条件编译方法来编写程序。输入一行英文字符序列，可以任选两种方式之一输出：一为原文输出；二为变换字母的大小写后输出。例如小写‘a’变成大写‘A’，大写‘D’变成小写‘d’，其他字符不变。用#define命令控制是否变换字母的大小写。例如，#define CHANGE 1 则输出变换后的文字，若#define CHANGE 0则原文输出。

解答：

1. 解题思路

程序流程图如图4-2所示

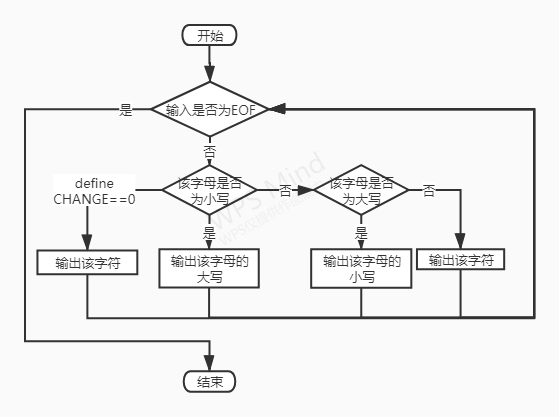


图4-2 程序设计（2）流程图

1. 源程序清单

#include <stdio.h>

#define CHANGE 0

int main()

{

char input;

printf("Please input a sentence:");

while ((input = getchar()) != EOF)

{

#if CHANGE

putchar(input);

#else

if (input >= 65 && input <= 90)

input += 32;

else if (input >= 97 && input <= 122)

input -= 32;

putchar(input);

#endif

}

return 0;

}

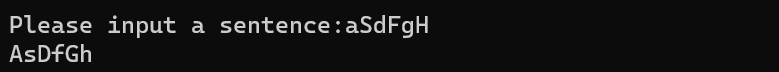
1. 测试
2. 测试数据

程序设计（2）测试数据如表4-2所示

表4-2 程序设计（2）测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | aSdFgH  (CHANGE=1) | AsDfGh |
| 用例2 | ASdFgH  (CHANGE=0) | ASdFgH |
| 用例3 | ^Z | 结束 |

1. 对应测试数据的运行结果如图4-3所示



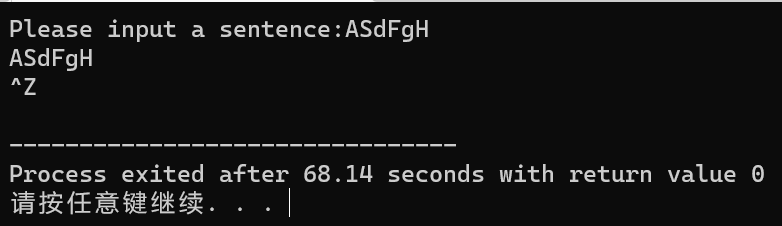


图4-3 程序设计（2）运行结果

（3）假设一个C程序由file1.c和file2.c两个源文件及一个file.h头文件组成，file1.c、file2.c和file.h的内容分别如下所述。试编辑该多文件C程序，补充file.h头文件内容，然后编译和链接。然后运行最后生成的可执行文件。

/\*源文件file1.c的内容\*/

#include "file.h"

int x,y; /\* 外部变量的定义性说明 \*/

char ch; /\* 外部变量的定义性说明 \*/

int main(void)

{

x=10;

y=20;

ch=getchar();

printf("in file1 x=%d,y=%d,ch is %c\n",x,y,ch);

func1();

return 0;

}

/\*源文件file2.c的内容为：\*/

#include "file.h"

void func1(void)

{

x++;

y++;

ch++;

printf("in file2 x=%d,y=%d,ch is %c\n",x,y,ch);

}

解答：

1. 解题思路：

在Dev C++中生成一个项目文件，通过头文件链接的方式链接外部 参数和函数。

1. 源程序清单

file.h代码如下：

#include <stdio.h>

extern void func1(void);

extern int x,y;

extern char ch;

1. 测试

输入字符a，预计输出：

in file1 x=10,y=20,ch is a

in file2 x=11,y=21,ch is b

对应测试用例的运行结果如图4-4所示

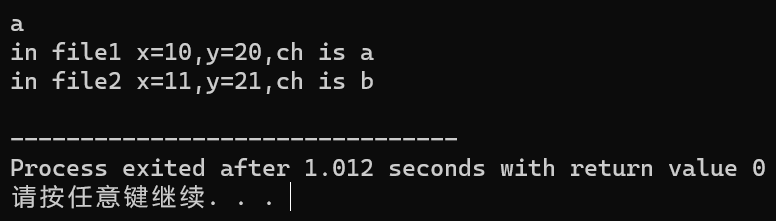


图4-4 程序设计（3）运行结果

4.3 实验小结

本次实验主要练习了编译预处理的相关内容，包括头文件的声明，自定义文件的链接，宏定义，带参数的宏定义。头文件声明是C/C++程序的重要组成部分，许多库函数为程序编写提供了很大的方便；自定义文件链接为开发者自行编写库，对大型项目进行模块化设计和开发管理提供了可能；宏定义在一定程度上简化了重复出现参数，同时可以更加清楚的表示其含义。

5 数组实验

5.1 实验目的

（1）掌握数组的说明、初始化和使用。

（2）掌握一维数组作为函数参数时实参和形参的用法。

（3）掌握字符串处理函数的设计，包括串操作函数及数字串与数之间转换函数实现算法。

（4）掌握基于分治策略的二分查找算法和选择法排序算法的思想，以及相关算法的实现。

5.2 实验内容及要求

**1、 源程序改错与跟踪调试**

在下面所给的源程序中，函数strcate(t,s)的功能是将字符串s连接到字符串t的尾部；函数strdelc(s,c)的功能是从字符串s中删除所有与给定字符c相同的字符，程序应该能够输出如下结果：

Programming Language

ProgrammingLanguage Language

ProgramingLnguage

跟踪和分析源程序中存在的问题，排除程序中的各种逻辑错误，使之能够输出正确的结果。

1. 单步执行源程序。进跟踪进入strcate时，观察字符数组t和s中的内容，分析结果是否正确。当单步执行光条刚落在第二个while语句所在行时，i为何值？t[i]为何值？分析该结果是否存在问题。当单步执行光条落在strcate函数块结束标记即右花括号“}”所在行时，字符数组t和s分别为何值？分析是否实现了字符串连接。

（2）跟踪进入函数strdelc时，观察字符数组s中的内容和字符c的值，分析结果是否正确。单步执行for语句过程中，观察字符数组s, j和k值的变化，分析该结果是否存在问题。当单步执行光条落在strdelc函数块结束标记“}”所在行时，字符串s为何值？分析是否实现了所要求的删除操作。

/\*实验5程序改错与跟踪调试题程序\*/

#include<stdio.h>

void strcate(char [],char []);

void strdelc(char [],char );

int main(void)

{

char a[]="Language", b[]="Programming";

printf("%s %s\n", b,a);

strcate(b,a); printf("%s %s\n",b,a);

strdelc(b, 'a'); printf("%s\n",b);

return 0;

}

void strcate(char t[],char s[])

{

int i = 0, j = 0;

while(t[i++]) ;

while((t[i++] = s[j++] )!= '\0');

}

void strdelc(char s[], char c)

{

int j,k;

for(j=k=0; s[j] != '\0'; j++)

if(s[j] != c) s[k++] = s[j];

}

解答：

1. 跟踪进入strcate时，字符数组t为”Programming”，s为”Language”;当单步执行光条刚落在第二个while语句所在行时，i为12; t[i]为’\000’;结果存在问题,故应在第二个while语句前加上 i--; 语句;当单步执行光条落在strcate函数块结束标记即右花括号“}”所在行时，字符数组t为”ProgrammingLanguage”，s为”Language”;增加 i--; 语句后实现了字符串连接。
2. 跟踪进入函数strdelc时，字符数组s为”ProgrammingLanguage”，字符c为’a’，结果正确。单步执行for语句过程中，字符数组s为”ProgrammingLanguage”, j和k均加一，该结果不存在问题。当单步执行光条落在strdelc函数块结束标记“}”所在行时，字符串s为”ProgrmmingLngugeage”;未实现所要求的删除操作;修改方法：在函数strdelc最后加上for (; s[k] != '\0'; k++) s[k] = '\000';清除多余位。

**2、 源程序完善和修改替换**

(1) 下面的源程序用于求解瑟夫问题：M个人围成一圈，从第一个人开始依次从1至N循环报数，每当报数为N时报数人出圈，直到圈中只剩下一个人为止。①请在源程序中的下划线处填写合适的代码来完善该程序。

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

int a[M], b[M]; /\* 数组a存放圈中人的编号，数组b存放出圈人的编号 \*/

int i, j, k;

for(i = 0; i < M; i++) /\* 对圈中人按顺序编号1—M \*/

a[i] = i + 1;

for(i = M, j = 0; i > 1; i--){

/\* i表示圈中人个数，初始为M个，剩1个人时结束循环；j表示当前报数人的位置 \*/

for(k = 1; k <= N; k++) /\* 1至N报数 \*/

if(++j > i - 1) j = 0;/\* 最后一个人报数后第一个人接着报，形成一个圈 \*/

b[M-i] = j ? a[j - 1] : a[i - 1]; /\* 将报数为N的人的编号存入数组b \*/

if(j)

for(k = --j; k < i; k++) /\* 压缩数组a，使报数为N的人出圈 \*/

a[k] = a[k + 1];

}

for(i = 0;i < M-1; i++) /\* 按次序输出出圈人的编号 \*/

printf(“%6d”, b[i]);

printf(“%6d\n”, a[0]); /\* 输出圈中最后一个人的编号 \*/

return 0;

}

②上面的程序中使用数组元素的值表示圈中人的编号，故每当有人出圈时都要压缩数组，这种算法不够精炼。如果采用做标记的办法，即每当有人出圈时对相应数组元素做标记，从而可省掉压缩数组的时间，这样处理效率会更高一些。请采用做标记的办法修改程序，并使修改后的程序与原程序具有相同的功能。

修改后程序源码：

#include <stdio.h>

#define N 3

#define M 10

int is\_out[11];

int result[11];

void decline()

{

int i = 0, index = 0, count = 1;

while (count <= M)

{

i++;

index++;

if (index >= M + 1)

index = 1;

while (is\_out[index] == 1)

index = (index + 1 >= M + 1) ? 1 : index + 1;

if (i >= N)

{

result[count] = index;

is\_out[index] = 1;

count++;

i = 0;

}

}

}

int main()

{

decline();

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

{

printf("%d ", result[j]);

}

return 0;

}

**3. 程序设计**

以下（1）至（3）题对应Educoder教学平台 “C语言实验”课程，实验5，第10关实验5-1、第11关实验5-2，以及第12关实验5-3。

（1）输入一个整数，将它在内存中二进制表示的每一位转化成对应的数字字符并且存放到一个字符数组中，然后输出该整数的二进制表示。

解答：

1. 算法流程如图3-1所示：

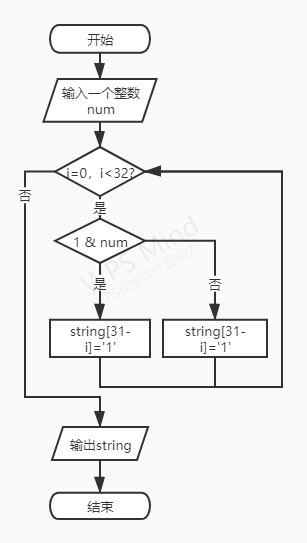


图3-1 程序设计（1）流程图

1. 源程序清单

#include <stdio.h>

int main()

{

unsigned num, i;

scanf("%d", &num);

int a[32] = {0}, power = 2, remainer, divisor;

for (i = 31; i >= 0; i--)

{

remainer = num % power;

a[i] = remainer;

num /= power;

if (num == 0)

break;

}

for (i = 0; i <= 31; i++)

{

printf("%d", a[i]);

}

return 0;

}

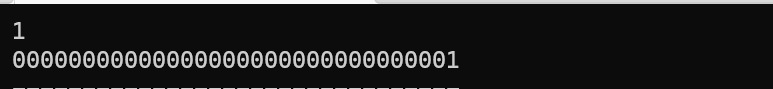
1. 测试
2. 测试数据：

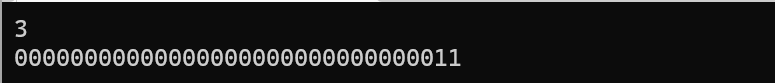
编程设计题1的测试数据如表3-1所示：

表3-1 程序设计（1）测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | 1 | 00000000000000000000000000000001 |
| 用例2 | 7 | 00000000000000000000000000000111 |
| 用例3 | 3 | 00000000000000000000000000000011 |

1. 对应测试用例的运行结果如图3-2所示：





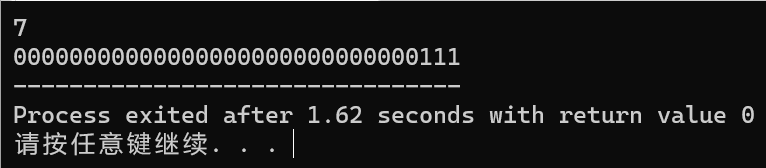


图3-2 程序设计（1）运行结果

（2）编写一个C程序，要求采用模块化程序设计思想，将相关功能用函数实现，并提供菜单选项。该程序具有以下功能：

①“成绩输入”，输入n个学生的姓名和C语言课程的成绩。

②“成绩排序”，将成绩按从高到低的次序排序，姓名同时进行相应调整。

③“成绩输出”，输出排序后所有学生的姓名和C语言课程的成绩。

④“成绩查找”，输入一个C语言课程成绩值，用二分查找进行搜索。如果查找到有该成绩，则输出该成绩学生的姓名和C语言课程的成绩；否则，输出提示“not found!”。

解答：

1. 解题思路：

1.分别输入学生成绩和姓名；

2.按照成绩从高到低进行选择排序；

3.输出指令菜单：输入指令以选择输出姓名成绩或按成绩查询姓名；

4.若按成绩查询姓名：利用二分查找查询某一成绩对应的学生姓名（之一），再分别向前向后查找同成绩的学生，打印对应姓名，若成绩不相同则停止查找；

1. 源程序清单

#include <stdio.h>

#include <string.h>

#define SWAPGRADE(a, b) s = a, a = b, b = s

struct

{

char name[10];

int grade;

} data[10];

int n;

void insert()

{

scanf("%d\n", &n);

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

{

scanf("%s %d\n", &data[i].name, &data[i].grade);

}

printf("%d records were input!\n", n);

}

void reorder()

{

int s;

char t[10];

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

{

for (int j = 0; j < n - i; j++)

{

if (data[j].grade < data[j + 1].grade)

{

SWAPGRADE(data[j].grade, data[j + 1].grade);

strcpy(t, data[j].name);

strcpy(data[j].name, data[j + 1].name);

strcpy(data[j + 1].name, t);

}

}

}

printf("Reorder finished!\n");

}

void output()

{

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

{

printf("%s %d\n", data[i].name, data[i].grade);

}

}

int middle;

void dichotomy(int grade, int start, int end)

{

middle = (end + start) / 2;

if (grade == data[middle].grade)

{

printf("%s %d\n", data[middle].name, data[middle].grade);

return;

}

else if (start == end)

{

printf("not found!\n");

return;

}

else if (grade > data[middle].grade)

{

if (middle == start + 1)

middle = start;

dichotomy(grade, start, middle);

}

else if (grade < data[middle].grade)

{

if (middle == end - 1)

middle = end;

dichotomy(grade, middle, end);

}

}

void query()

{

int grade;

scanf("%d\n", &grade);

dichotomy(grade, 0, n - 1);

}

int main()

{

int opt;

while (scanf("%d\n", &opt) != EOF)

{

if (opt == 1)

insert();

else if (opt == 2)

reorder();

else if (opt == 3)

output();

else if (opt == 4)

query();

else if (opt == 0)

break;

}

return 0;

}

1. 测试
2. 测试数据如表3-2所示：

表3-2 程序设计（2）测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入 | 预计输出 |
| 用例1 | 1  6  Jack 95  Mike 90  Joe 75  Andy 95  Rose 89  Sophia 77  2  3  4  89  4  88  0 | 6 records were input!  Reorder finished!  Jack 95  Andy 95  Mike 90  Rose 89  Sophia 77  Joe 75  Rose 89  not found!  （程序退出） |

1. 对应测试用例的运行结果如图3-3所示：

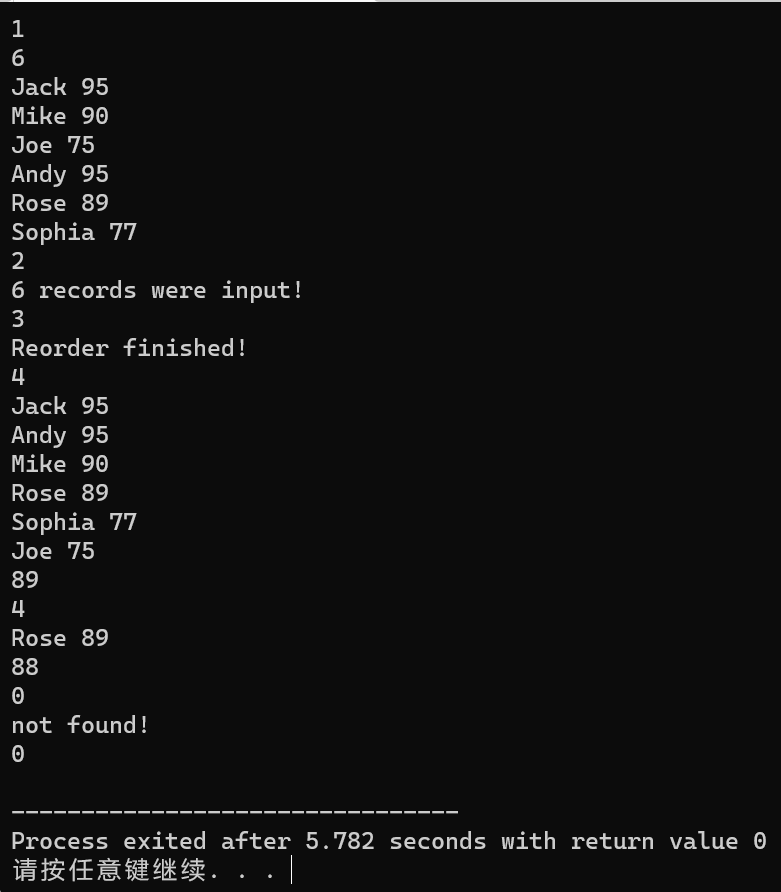


图3-3 程序设计（2）运行结果

（3）求解N皇后问题，即在N×N的棋盘上摆放N个皇后，要求任意两个皇后不能在同一行、同一列、同一对角线上。输入棋盘的大小N（N取值1-10），如果能满足摆放要求，则输出所有可能的摆放法的数量，否则输出“无解。”

解答：

1. 解题思路：

将N皇后定义为一个queen[n] 一维数组，通过取值来表示每一行的皇后所在位置，然后通过递归运算最大效率遍历所有可能性。

1. 源程序清单

#include <stdio.h>

int queen[11] = {1};

int count = 0;

int abs(int input)

{

return (input >= 0) ? input : (-input);

}

int checkup(int row)

{

int i;

for (i = 1; i <= row; i++)

{

if ((abs(queen[row] - queen[row - i]) == i) || (queen[row] == queen[row - i]))

{

return 0;

}

}

return 1;

}

void move(int row, int n)

{

if (row > n-1)

return;

for (queen[row] = 1; queen[row] <= n; queen[row]++)

{

if (row < n-1)

{

if (checkup(row))

move(row + 1, n);

}

else

{

if (checkup(row))

count++;

}

}

}

int main()

{

int n;

scanf("%d", &n);

move(0, n);

if (count > 0)

printf("%d", count);

else

printf("无解");

return 0;

}

1. 测试：
2. 测试用例如表3-3所示：

表3-3 程序设计（3）测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入 | 预计输出 |
| 测试用例1 | 1 | 1 |
| 测试用例2 | 8 | 92 |

1. 对应测试用例的运行结果如图3-4所示：



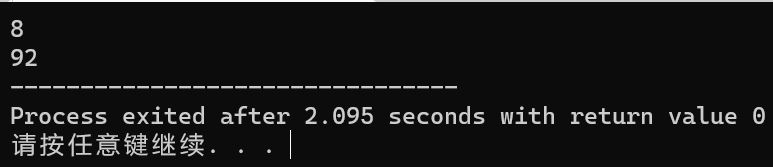


图3-4 程序设计（3）测试结果

## 5.3 实验小结

本次实验主要练习了数组的使用，涉及数组的初始化，数组元素的赋值，作为函数参数的数组，字符数组的使用，作为函数参数的多维数组等。

其中，我对作为函数参数的多维数组的使用体会最深。在C语言中，传入函数的多维数组要求有确定的非第一唯长度，因此对多维数组的使用存在很多限制。在学习过程中，了解到了数组和指针的统一性。这一点希望可以在下一次实验课上获得更深的体会。

6 指针实验

6.1 实验目的

（1）熟练掌握指针的说明、赋值、使用。

（2）掌握用指针引用数组的元素，熟悉指向数组的指针的使用。

（3）熟练掌握字符数组与字符串的使用，掌握指针数组及字符指针数组的用法。

（4）掌握指针函数与函数指针的用法。

（5）掌握带有参数的main函数的用法。

6.2 实验题目及要求

**1、源程序改错题**

在下面所给的源程序中，函数strcopy(t, s)的功能是将字符串s复制给字符串t，并且返回串t的首地址。请单步跟踪程序，根据程序运行时出现的现象或观察到的字符串的值，分析并排除源程序的逻辑错误，使之能按照要求输出如下结果：

Input a string:

programming↙ （键盘输入）

programming

Input a string again:

language↙ （键盘输入）

language

1. #include<stdio.h>
2. char \*strcopy(char \*, const char \*);
3. int main(void)
4. {
5. char \*s1, \*s2, \*s3;
6. printf("Input a string:\n", s2);
7. scanf("%s", s2);
8. strcopy(s1, s2);
9. printf("%s\n", s1);
10. printf("Input a string again:\n", s2);
11. scanf("%s", s2);
12. s3 = strcopy(s1, s2);
13. printf("%s\n", s3);
14. return 0;
15. }
16. /\*将字符串s复制给字符串t，并且返回串t的首地址\*/
17. char \* strcopy(char \*t, const char \*s)
18. {
19. while(\*t++ = \*s++);
20. return (t);
21. }

解答：

1. 第六行及第十行 printf("Input a string:\n", s2); 正确形式为 printf("Input a string:\n");
2. 第七行及第十一行 scanf("%s", s2); 正确形式应为 scanf("%s", &s2);
3. 第五行 char \*s1, \*s2, \*s3; 正确形式为 char s1[100], s2[100], \*s3;
4. 第十二行 s3 = strcopy(s1, s2); 正确形式为 strcopy(s1, s2);
5. 第十三行 printf("%s\n", s3); 正确形式为 printf("%s\n", s1);
6. 第二十一行可以直接去掉

**2、源程序完善、修改替换题**

（1）下面程序中函数strsort用于对字符串进行升序排序，在主函数中输入N个字符串（字符串长度不超过49）存入通过malloc动态分配的存储空间，然后调用strsort对这N个串按字典序升序排序。

①请在源程序中的下划线处填写合适的代码来完善该程序。

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define N 4

/\*对指针数组s指向的size个字符串进行升序排序\*/

void strsort(char \*s[], int size)

{

char \*temp;

int i, j;

for(i=0; i<size-1; i++)

for (j=0; j<size-i-1; j++)

if (strcmp(s[j],s[j+1]) > 0)

{

temp = s[j];

s[j] = s[j+1];

s[j+1] = temp;

}

}

int main()

{

int i;

char \*s[N], t[50];

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

{

gets(t);

s[i] = (char \*)malloc(strlen(t)+1);

strcpy(s[j] , t);

}

strsort(s , N);

for (i=0; i<N; i++) {puts(s[i]); free(s[i]);}

return 0;

}

②数组作为函数参数其本质类型是指针。例如，对于形参char \*s[]，编译器将其解释为char \*\*s，两种写法完全等价。请用二级指针形参重写strsort函数，并且在该函数体的任何位置都不允许使用下标引用。

解答：

修改后的 strsort 方法如下：

void strsort(char \*\*s,int size){

char \*temp;

int i,j;

for(i=0; i<size-1; i++)

for (j=0; j<size-i-1; j++)

if (strcmp(\*(s+j),\*(s+j+1))>0) {

temp = \*(s+j);

\*(s+j)= \*(s+j+1);

\*(s+j+1) = temp;

}

}

（2）下面源程序通过函数指针和菜单选择来调用库函数实现字符串操作；串复制strcpy、串连接strcat或串分解strtok。

①请在源程序中的下划线处填写合适的代码来完善该程序，使之能按照要求输出下面结果：

1 copy string.

2 connect string.

3 parse string.

4 exit.

input a number (1-4) please!

2↙ （键盘输入）

input the first string please!

the more you learn,↙ （键盘输入）

input the second string please!

the more you get. ↙ （键盘输入）

the result is the more you learn, the more you get.

# include<stdio.h>

# include<string.h>

int main (void)

{

char \*(\*p)(char \*,const char \*);

char a[80], b[80], \*result;

int choice;

while(1)

{

do

{

printf("\t\t1 copy string.\n");

printf("\t\t2 connect string.\n");

printf("\t\t3 parse string.\n");

printf("\t\t4 exit.\n");

printf("\t\tinput a number (1-4) please.\n");

scanf("%d", &choice);

}while(choice<1 || choice>4);

switch(choice)

{

case 1: p = strcpy; break;

case 2: p = strcat; break;

case 3: p = strtok; break;

case 4: goto down;

}

getchar();

printf("input the first string please!\n");

gets(a);

printf("input the second string please!\n");

gets(b);

result = p(a, b);

printf("the result is %s\n", result);

}

down:

return 0;

}

②函数指针的一个用途是用户散转程序，即通过一个转移表（函数指针数组）来实现多分枝函数处理，从而省去了大量的if语句或者switch语句。转移表中存放了各个函数的入口地址（函数名），根据条件的设定来查表选择执行相应的函数。请使用转移表而不是switch语句重写以上程序。

解答：

switch相关语句修改为：

char \*(\*c[])(char\*,const char\*)={strcpy,strcat,strtok};

if(choice == 4) goto down;

//else p = (choice==1)?strcpy:(choice==2)?strcat:(choice==3)?strtok:0;

else p = c[choice-1];

**3、编程设计题**

（1）指定main函数的参数

在IDE（比如DevC++）中，选择“**运行**”｜“**参数**”菜单，在 “**传递给主程序的参数**”文本框中输入main函数的参数arg1 arg2 arg3，只输入命令行中文件名后的参数，文件名不作为参数输入，参数间以空格隔开。编写程序在命令行输出这三个参数。（注意不同IDE输入参数的方式不相同，可参考各个IDE的使用手册。）

解答：

源代码如下：

#include<stdio.h>

int main(int argc,char \*argv[]){

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

printf("%s ",argv[i]);

}

return 0;

}

以下（2）至（5）题对应Educoder 教学平台“C语言实验”课程，实验6，第13关实验6-1、第14关实验6-2、第15关实验6-3, 以及第16关实验6-4。

（2）一个长整型变量占4个字节，其中每个字节又分成高4位和低4位。输入一个长整型变量，要求从高字节开始，依次取出每个字节的高4位和低4位并以十六进制数字字符的形式进行显示，通过指针取出每字节。

样例输入：15

样例输出：0000000F

解答：

1. 解题思路：

程序设计（2）解题思路如图3-1所示：

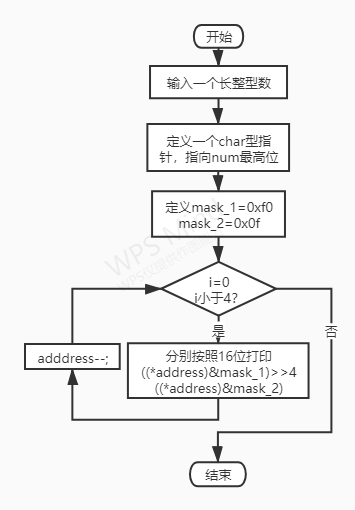


图3-1 程序设计（2）流程图

1. 源程序清单

#include <stdio.h>

int main()

{

unsigned long input;

scanf("%ld", &input);

unsigned char \*p = (unsigned char \*)&input;

for (int i = 3; i >= 0; i--)

{

printf("%02X", (p[i]));

}

return 0;

}

1. 运行结果：

程序设计（2）运行结果如图3-2所示：

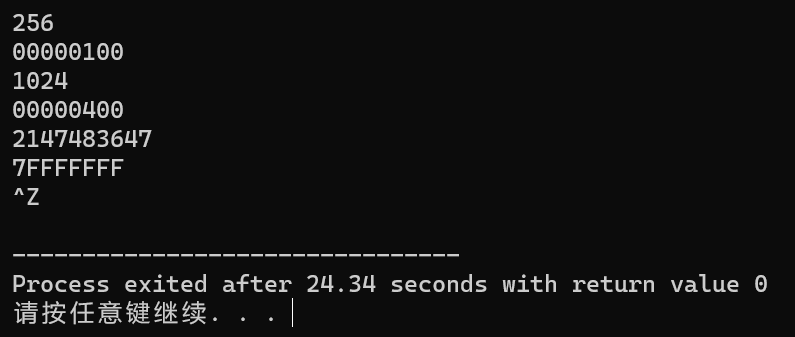


图3-2 程序设计（2）运行结果

（3）旋转是图像处理的基本操作，编程实现一个将一个图像逆时针旋转90°。提示：计算机中的图像可以用一个矩阵来表示，旋转一个图像就是旋转对应的矩阵。将旋转矩阵的功能定义成函数，通过使用指向数组元素的指针作为参数使该函数能处理任意大小的矩阵。要求在main函数中输入图像矩阵的行数n和列数m，接下来的n行每行输入m个整数，表示输入的图像。输出原始矩阵逆时针旋转90°后的矩阵。

样例输入：

2 3

1 5 3

3 2 4

样例输出：

3 4

5 2

1 3

解答：

1. 解题思路：

程序设计（3）解题思路如图3-3所示：

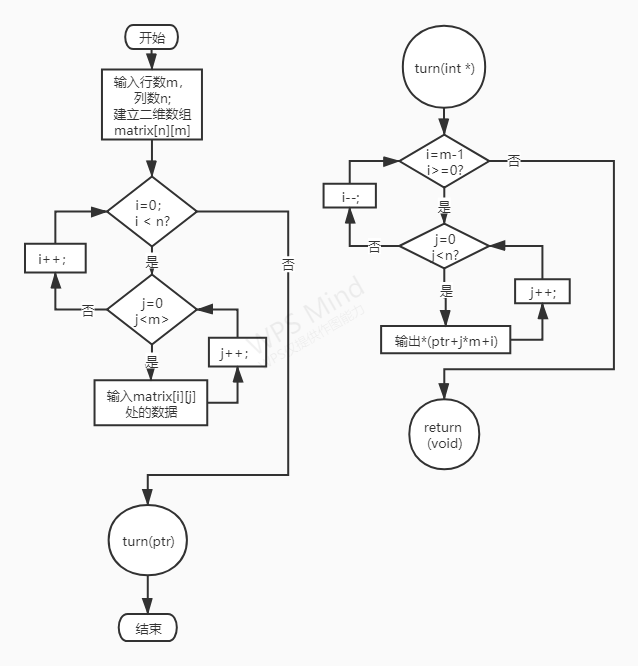


图3-3 程序设计（3）流程图

1. 源程序清单：

#include <stdio.h>

int matrix[10][10];

int spinned[10][10];

int n, m;

int main()

{

scanf("%d %d", &n, &m);

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

{

for (int j = 0; j < m; j++)

{

scanf("%d", &matrix[i][j]);

}

}

spin();

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

printf("%d", spinned[i][j]);

if(j!=n-1)

printf(" ");

}

printf("\n");

}

}

void spin()

{

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

{

for (int j = 0; j < m; j++)

{

spinned[m - 1 - j][i] = matrix[i][j];

}

}

}

1. 测试
2. 测试数据：

测试数据如表3-1所示：

表3-1 程序设计（3）测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | 2 3  1 5 3  3 2 4 | 3 4  5 2  1 3 |
| 用例2 | 5 4  3 0 9 7  8 9 6 3  0 4 2 7  2 9 5 6  9 6 5 3 | 7 3 7 6 3  9 6 2 5 5  0 9 4 9 6  3 8 0 2 9 |
| 用例3 | 1 4  11 76 88 45 | 45  88  76  11 |

1. 对应测试用例的运行结果如图3-4所示：





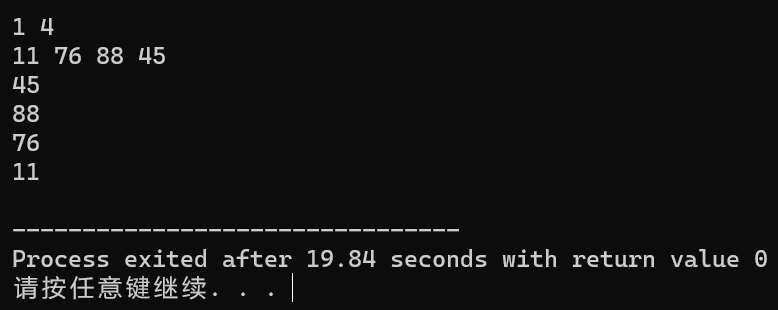


图3-4 程序设计（3）运行结果

1. 输入n行文本，每行不超过80个字符，用字符指针数组指向键盘输入的n行文本，且n行文本的存储无冗余，删除每一行中的前置空格（' '）和水平制表符（'\t'）。要求：将删除一行文本中前置空格和水平制表符的功能定义成函数，在main函数中输出删除前置空格符的各行。

解答：

1. 解题思路

程序设计（4）解题思路如图3-5所示：

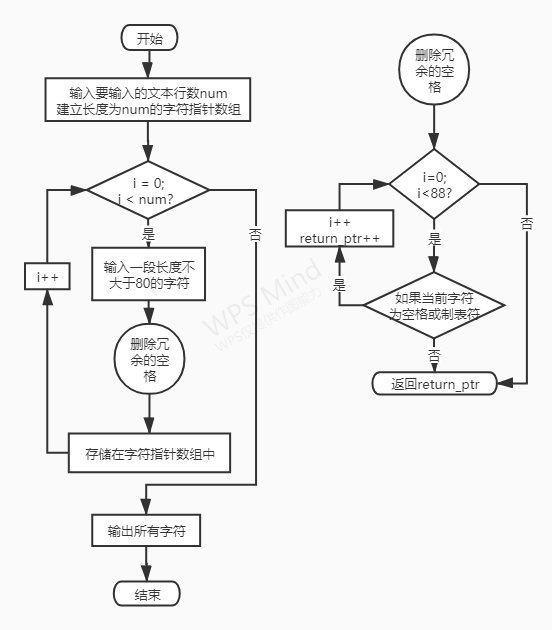


图3-5 程序设计（4）流程图

1. 源程序清单：

#include<stdio.h>

#include<string.h>

char a[10][80];

int i=0;

void decline(){

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

for (int l=0;a[j][l]==' ';){

for(int n=0;a[j][l+n+1]!='\000';n++){

a[j][l+n] = a[j][l+1+n];

}

}

for(int l=0;a[j][l]!='\000';l++){

if(a[j][l]=='\n') a[j][l]='\0';

if(a[j][l]=='\t'){

for(int n=0;a[j][l+n+1]!='\000';n++){

a[j][l+n] = a[j][l+1+n];

}

}

}

}

}

int main(){

while(fgets(a[i],80,stdin)!=NULL){

i++;

}

decline();

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

printf("%s\n",a[r]);

}

}

1. 测试
2. 测试数据：

程序设计（4）测试数据如表3-2所示：

表3-2 程序设计（4）测试数据

|  |  |  |
| --- | --- | --- |
| 用例 | 实际输入 | 预计输出 |
| 用例1 | A a  Aaa aa  A a | A a  Aaa aa  A a |

1. 对应测试用例的运行结果如图3-6所示：

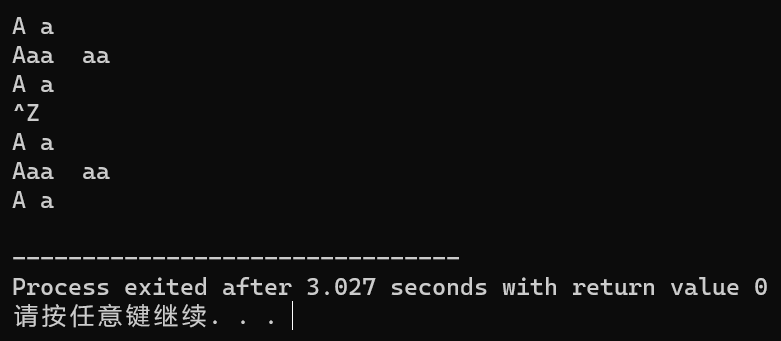


图3-6 程序设计（4）运行结果

（5）编写8个任务函数，一个scheduler调度函数和一个execute执行函数。仅在main函数中调用scheduler函数，scheduler函数要求用最快的方式调度执行用户指定的任务函数。

①先设计task0, task1, task2, task3, task4, task5, task6, task7共8个任务函数，每个任务函数的任务就是输出该任务被调用的字符串。例如，第0个任务函数输出“task0 is called!”，第1个任务函数输出“task1 is called!”，以此类推。

②scheduler函数根据键盘输入的数字字符的先后顺序，一次调度选择对应的任务函数。例如，输入：1350并回车，则scheduler函数一次调度选择task1, task3, taks5, task0，然后以函数指针数组和任务个数为参数将调度选择结果传递给execute函数并调用execute函数。

③execute函数根据scheduler函数传递的指针数组和任务个数为参数，按照指定的先后顺序依此调用执行选定的任务函数。

例如，当输入13607122并回车，程序运行结果如下：

task1 is called!

task3 is called!

task6 is called!

task0 is called!

task7 is called!

task1 is called!

task2 is called!

task2 is called!

解答：

1. 解题思路：
2. 声明存储8个任务函数地址的指针
3. 以字符串形式输入指令
4. 依据指令长度声明储存即将执行的任务的地址的指针
5. 调用
6. 源程序清单：

#include <stdio.h>

void task0(){

printf("task0 is called!\n");

}

void task1(){

printf("task1 is called!\n");

}

void task2(){

printf("task2 is called!\n");

}

void task3(){

printf("task3 is called!\n");

}

void task4(){

printf("task4 is called!\n");

}

void task5(){

printf("task5 is called!\n");

}

void task6(){

printf("task6 is called!\n");

}

void task7(){

printf("task7 is called!\n");

}

void execute(int count,void (\*\*fun)()){

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

fun[i]();

}

}

void scheduler(){

void (\*function[8])()={

task0,task1,task2,task3,task4,task5,task6,task7};

char input[10];

scanf("%s",&input);

int count = strlen(input);

void (\*fun[count])();

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

fun[i]=function[(int)input[i]-48];

}

execute(count,fun);

}

int main(){

scheduler();

return 0;

}

1. 测试：
2. 测试数据：

程序设计（5）的测试数据如表3-3所示：

表3-3 程序设计（5）测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 实际输入 | 预计输出 |
| 用例1 | 13607122 | task1 is called!  task3 is called!  task6 is called!  task0 is called!  task7 is called!  task1 is called!  task2 is called!  task2 is called! |

1. 对应测试用例的运行结果如图3-7所示：

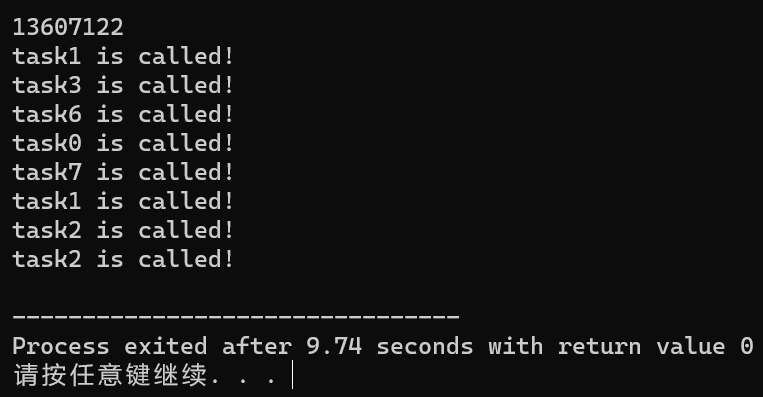


图3-7 程序设计（5）运行结果

## 6.3 实验小结

指针是C语言学习和使用的重难点。本次实验主要练习了指针的使用，包括指针变量的声明和引用，利用指针引用所指变量，作为函数参数和返回值的指针的使用，复杂类型指针的使用等。

指针常常被称为C语言的精华，这显示出了它的重要意义。借由指针，C语言可以对计算机内存进行直接操作，更加贴近底层，大大提高了执行效率。但同时，这也要求设计程序时格外重视对指针的使用，防止出现野指针、空指针等情况。

# 7 结构与联合实验

## 7.1 实验目的

1．通过实验，熟悉和掌握结构的说明和引用、结构的指针、结构数组、以及函数中使用结构的方法。

2．通过实验，掌握动态储存分配函数的用法，掌握自引用结构，单向链表的创建、遍历、结点的增删、查找等操作。

3．了解字段结构和联合的用法。

## 7.2 实验题目及要求

**1．表达式求值的程序验证题**

设有说明：

char u[]="UVWXYZ";

char v[]="xyz";

struct T{

int x;

char c;

char \*t;

}a[]={{11,ˊAˊ,u},{100, ˊBˊ,v}},\*p=a;

请先自己计算下面表达式的值，然后通过编程计算来加以验证。(各表达式相互无关)

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **表达式** | **计算值** | **验证值** |
| 1 | (++p)->x | 100 | 100 |
| 2 | p++,p->c | B | B |
| 3 | \*p++->t,\*p->t | x,U | x,U |
| 4 | \*(++p)->t | x | x |
| 5 | \*++p->t | V | V |
| 6 | ++\*p->t | V | V |

**2．源程序修改替换题**

给定一批整数，以0作为结束标志且不作为结点，将其建成一个先进先出的链表，先进先出链表的指头指针始终指向最先创建的结点（链头），先建结点指向后建结点，后建结点始终是尾结点。

1. 源程序中存在什么样的错误（先观察执行结果）？对程序进行修改、调试，使之能够正确完成指定任务。

源程序如下：

#include "stdio.h"

#include "stdlib.h"

struct s\_list{

int data; /\* 数据域 \*/

struct s\_list \*next; /\* 指针域 \*/

} ;

void create\_list (struct s\_list \*headp,int \*p);

void main(void)

{

struct s\_list \*head=NULL,\*p;

int s[]={1,2,3,4,5,6,7,8,0}; /\* 0为结束标记 \*/

create\_list(head,s); /\* 创建新链表 \*/

p=head; /\*遍历指针p指向链头 \*/

while(p){

printf("%d\t",p->data); /\* 输出数据域的值 \*/

p=p->next; /\*遍历指针p指向下一结点 \*/

}

printf("\n");

}

void create\_list(struct s\_list \*headp,int \*p)

{

struct s\_list \* loc\_head=NULL,\*tail;

if(p[0]==0) /\* 相当于\*p==0 \*/

;

else { /\* loc\_head指向动态分配的第一个结点 \*/

loc\_head=(struct s\_list \*)malloc(sizeof(struct s\_list));

loc\_head->data=\*p++; /\* 对数据域赋值 \*/

tail=loc\_head; /\* tail指向第一个结点 \*/

while(\*p){ /\* tail所指结点的指针域指向动态创建的结点 \*/

tail->next=(struct s\_list \*)malloc(sizeof(struct s\_list));

tail=tail->next; /\* tail指向新创建的结点 \*/

tail->data=\*p++; /\* 向新创建的结点的数据域赋值 \*/

}

tail->next=NULL; /\* 对指针域赋NULL值 \*/

}

headp=loc\_head; /\* 使头指针headp指向新创建的链表 \*/

}

解答：

1. create\_list定义应改为struct s\_list\* create\_list(int \*p);
2. create\_list最后应改为return loc\_head;
3. create\_list调用应改为head = create\_list(s);

（2）修改替换create\_list函数，将其建成一个后进先出的链表，后进先出链表的头指针始终指向最后创建的结点（链头），后建结点指向先建结点，先建结点始终是尾结点。

解答：

修改后源代码清单如下：

#include "stdio.h"

#include "stdlib.h"

struct s\_list

{

int data; /\* 数据域 \*/

struct s\_list \*next; /\* 指针域 \*/

};

struct s\_list \*create\_list(int \*p);

void main(void)

{

struct s\_list \*head = NULL, \*p;

int s[] = {1, 2, 3, 4, 5, 6, 7, 8, 0}; /\* 0为结束标记 \*/

head = create\_list(s); /\* 创建新链表 \*/

p = head; /\*遍历指针p指向链头 \*/

while (p)

{

printf("%d\t", p->data); /\* 输出数据域的值 \*/

p = p->next; /\*遍历指针p指向下一结点 \*/

}

printf("\n");

}

struct s\_list \*create\_list(int \*p)

{

struct s\_list \*tail = NULL, \*loc\_head;

if (p[0] == 0) /\* 相当于\*p==0 \*/

;

else

{ /\* tail指向动态分配的第一个结点 \*/

tail = (struct s\_list \*)malloc(sizeof(struct s\_list));

tail->data = \*p++; /\* 对数据域赋值 \*/

loc\_head = tail; /\* loc\_head指向第一个结点 \*/

tail->next = NULL;

while (\*p)

{ /\* loc\_head所指结点的指针域指向动态创建的结点 \*/

loc\_head = (struct s\_list \*)malloc(sizeof(struct s\_list));

loc\_head->next = tail; /\* loc\_head指向新创建的结点 \*/

loc\_head->data = \*p++; /\* 向新创建的结点的数据域赋值 \*/

tail = loc\_head;

}

}

return loc\_head;

}

**3 程序设计**

以下（1）至（3）题对应Educoder 教学平台“C语言实验”课程，实验7，第17关实验7-1、第18关实验7-2，以及第19关实验7-3。

（1）本关任务：用单向链表建立一张班级成绩单，包括每个学生的学号、姓名、英语、高等数学、普通物理、C语言程序设计四门课程的成绩。用菜单实现下列功能：

① 输入每个学生的各项信息。

② 输出每个学生的各项信息。

③ 修改指定学生的指定数据项的内容。

④ 统计每个同学的平均成绩（保留2位小数）。

⑤ 输出各位同学的学号、姓名、四门课程的总成绩和平均成绩。

解答：

1. 解题思路：

程序设计（1）流程如图3-1所示：

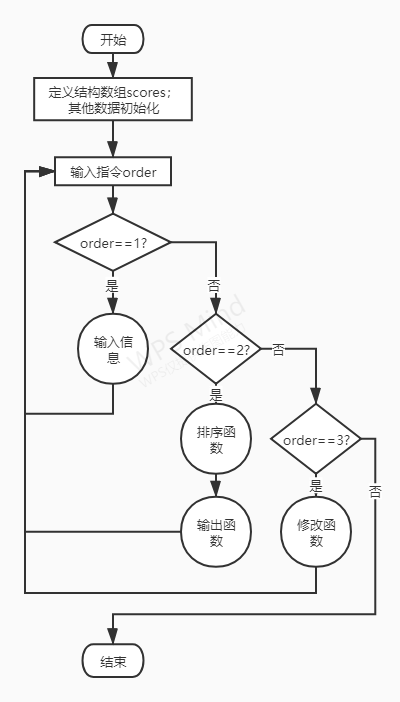


图3-1 程序设计（1）流程图

1. 源代码清单：

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

struct data

{

int id;

char name[10];

int Egrade;

int Mgrade;

int Pgrade;

int Cgrade;

};

struct data\_list

{

struct data content;

struct data\_list \*next;

};

struct data\_list \*create\_list(int n)

{

int id, Egrade, Mgrade, Pgrade, Cgrade;

char name[10];

struct data data;

struct data\_list \*head, \*tail;

head = (struct data\_list \*)malloc(sizeof(struct data\_list));

scanf("%d %s %d %d %d %d",

&data.id, data.name,

&data.Egrade, &data.Mgrade,

&data.Pgrade, &data.Cgrade);

head->content = data;

tail = head;

for (int i = 0; i < n - 1; i++)

{

scanf("%d %s %d %d %d %d",

&data.id, data.name,

&data.Egrade, &data.Mgrade,

&data.Pgrade, &data.Cgrade);

tail->next = (struct data\_list \*)malloc(sizeof(struct data\_list));

tail = tail->next;

tail->content = data;

}

tail->next = NULL;

return head;

}

void query(struct data\_list \*head)

{

while (head)

{

printf("%d %s %d %d %d %d\n",

head->content.id, head->content.name,

head->content.Egrade, head->content.Mgrade,

head->content.Pgrade, head->content.Cgrade);

head = head->next;

}

}

void modify(struct data\_list \*head)

{

int id, order, grade,

Egrade, Mgrade, Pgrade, Cgrade;

char name[10];

struct data \*modify;

scanf("%d %d", &id, &order);

while (head)

{

if (id == head->content.id)

{

modify = &head->content;

break;

}

head = head->next;

}

if (order == 0)

{

scanf("%s", name);

strcpy(modify->name, name);

}

else

{

scanf("%d", &grade);

if (order == 1)

modify->Egrade = grade;

else if (order == 2)

modify->Mgrade = grade;

else if (order == 3)

modify->Pgrade = grade;

else

modify->Cgrade = grade;

}

printf("%d %s %d %d %d %d\n",

modify->id, modify->name,

modify->Egrade, modify->Mgrade,

modify->Pgrade, modify->Cgrade);

}

void average(struct data\_list \*head)

{

double avergrade;

while (head)

{

avergrade = (head->content.Cgrade +

head->content.Egrade +

head->content.Mgrade +

head->content.Pgrade) /

4;

printf("%d %s %.2lf\n",

head->content.id, head->content.name,

avergrade);

head = head->next;

}

}

void grade\_pro(struct data\_list \*head)

{

int totalgrade;

double avergrade;

while (head)

{

totalgrade = head->content.Cgrade +

head->content.Egrade +

head->content.Mgrade +

head->content.Pgrade;

avergrade = totalgrade / 4.0;

printf("%d %s %d %.2lf\n",

head->content.id, head->content.name,

totalgrade, avergrade);

head = head->next;

}

}

int main()

{

int option, n;

struct data\_list \*head;

if (scanf("%d", &option) == 1)

{

scanf("%d", &n);

head = create\_list(n);

printf("完成了%d位同学的成绩输入。\n", n);

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

{

if (option == 2)

query(head);

else if (option == 3)

modify(head);

else if (option == 4)

average(head);

else if (option == 5)

grade\_pro(head);

else if (option == 0)

break;

}

}

else

printf("No Data!");

return 0;

}

1. 测试：
2. 测试数据：

程序设计（1）测试数据如表3-1所示：

表3-1 程序设计（1）测试数据

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 第一次指令 | 第二次指令 | 第三次指令 | 第四次指令 | 第五次指令 |
| 用例1 | 1  3  01 aaa 89 89 90 90  02 sss 78 98 90 80  03 ddd 86 96 85 97 | 5 | 3  01 1 100 | 2 | 0 |
| 预计输出 | 完成了3位同学的成绩输入。 | 输出三名学生的总成绩和平均成绩 | 学生sss的数学成绩改为100，并输出该生成绩 | 输出修改后三名学生的各科成绩 | 程序结束 |

1. 测试结果：

对应测试用例的运行结果如图3-2所示：

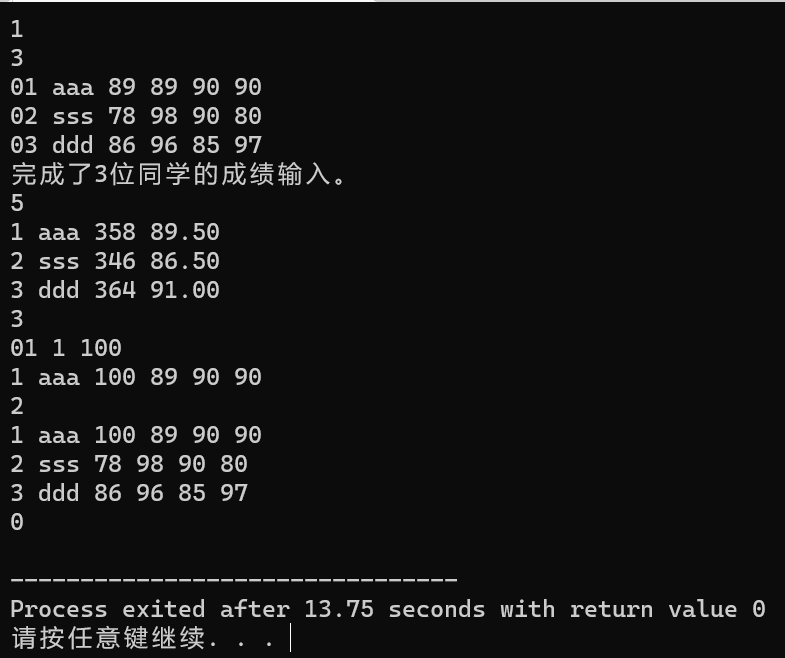


图 3-2 程序设计（1）运行结果

（2）本关任务：对程序设计题第（1）题的程序，⑥增加按照平均成绩进行升序排序的函数，写出用交换结点数据域的方法升序排序的函数，排序可用选择法或冒泡法。

解答：

1. 解题思路：

在程序设计（1）的基础上增加一个排序函数，该函数接收链表结点数n与链表头指针head，内部使用冒泡排序交换结点的数据域以达到排序效果。

1. 源程序清单：

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

struct data

{

int id;

char name[10];

int Egrade;

int Mgrade;

int Pgrade;

int Cgrade;

int totalgrade;

double avergrade;

};

struct data\_list

{

struct data content;

struct data\_list \*next;

};

struct data\_list \*create\_list(int n)

{

int id, Egrade, Mgrade, Pgrade, Cgrade;

char name[10];

struct data data;

struct data\_list \*head, \*tail;

head = (struct data\_list \*)malloc(sizeof(struct data\_list));

scanf("%d %s %d %d %d %d",

&data.id, data.name,

&data.Egrade, &data.Mgrade,

&data.Pgrade, &data.Cgrade);

head->content = data;

tail = head;

for (int i = 0; i < n - 1; i++)

{

scanf("%d %s %d %d %d %d",

&data.id, data.name,

&data.Egrade, &data.Mgrade,

&data.Pgrade, &data.Cgrade);

tail->next = (struct data\_list \*)malloc(sizeof(struct data\_list));

tail = tail->next;

tail->content = data;

}

tail->next = NULL;

return head;

}

void query(struct data\_list \*head)

{

while (head)

{

printf("%d %s %d %d %d %d\n",

head->content.id, head->content.name,

head->content.Egrade, head->content.Mgrade,

head->content.Pgrade, head->content.Cgrade);

head = head->next;

}

}

void modify(struct data\_list \*head)

{

int id, order, grade,

Egrade, Mgrade, Pgrade, Cgrade;

char name[10];

struct data \*modify;

scanf("%d %d", &id, &order);

while (head)

{

if (id == head->content.id)

{

modify = &head->content;

break;

}

head = head->next;

}

if (order == 0)

{

scanf("%s", name);

strcpy(modify->name, name);

}

else

{

scanf("%d", &grade);

if (order == 1)

modify->Egrade = grade;

else if (order == 2)

modify->Mgrade = grade;

else if (order == 3)

modify->Pgrade = grade;

else

modify->Cgrade = grade;

}

printf("%d %s %d %d %d %d\n",

modify->id, modify->name,

modify->Egrade, modify->Mgrade,

modify->Pgrade, modify->Cgrade);

}

void order(int n, struct data\_list \*head)

{

struct data\_list \*headflag;

struct data medium;

headflag = head;

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

{

for (int j = 0; j < n - i; j++)

{

if (head->content.avergrade > head->next->content.avergrade)

{

medium = head->content;

head->content = head->next->content;

head->next->content = medium;

}

head = head->next;

}

head = headflag;

}

head = headflag;

while (head)

{

printf("%d %s %.2lf\n",

head->content.id, head->content.name,

head->content.avergrade);

head = head->next;

}

}

void grade\_pro(int n, int option, struct data\_list \*head)

{

struct data\_list \*headflag;

headflag = head;

while (head)

{

head->content.totalgrade = head->content.Cgrade +

head->content.Egrade +

head->content.Mgrade +

head->content.Pgrade;

head->content.avergrade = head->content.totalgrade / 4.0;

if (option == 4)

printf("%d %s %.2lf\n",

head->content.id,

head->content.name,

head->content.avergrade);

else if (option == 5)

printf("%d %s %d %.2lf\n",

head->content.id,

head->content.name,

head->content.totalgrade,

head->content.avergrade);

head = head->next;

}

head = headflag;

if (option == 6)

{

order(n, head);

}

}

int main()

{

int option, n;

struct data\_list \*head;

scanf("%d", &option);

if (option == 1)

{

scanf("%d", &n);

head = create\_list(n);

printf("完成了%d位同学的成绩输入。\n", n);

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

{

if (option == 2)

query(head);

else if (option == 3)

modify(head);

else if (option == 4 || option == 5 || option == 6)

grade\_pro(n, option, head);

else if (option == 0)

break;

}

}

else

printf("No Data!");

return 0;

}

1. 测试：
2. 测试数据：

程序设计（2）测试数据如表3-2所示：

表3-2 程序设计（2）测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 第一次指令 | 第二次指令 | 第三次指令 | 第五次指令 |
| 用例1 | 1  3  01 aaa 89 89 90 90  02 sss 78 98 90 80  03 ddd 86 96 85 97 | 5 | 6 | 0 |
| 预计输出 | 完成了3位同学的成绩输入 | 输出三名学生的总成绩和平均成绩 | 将三名同学成绩按平均成绩升序排列并输出 | 程序结束 |

1. 运行结果：

对应测试用例的运行结果如图3-3所示：

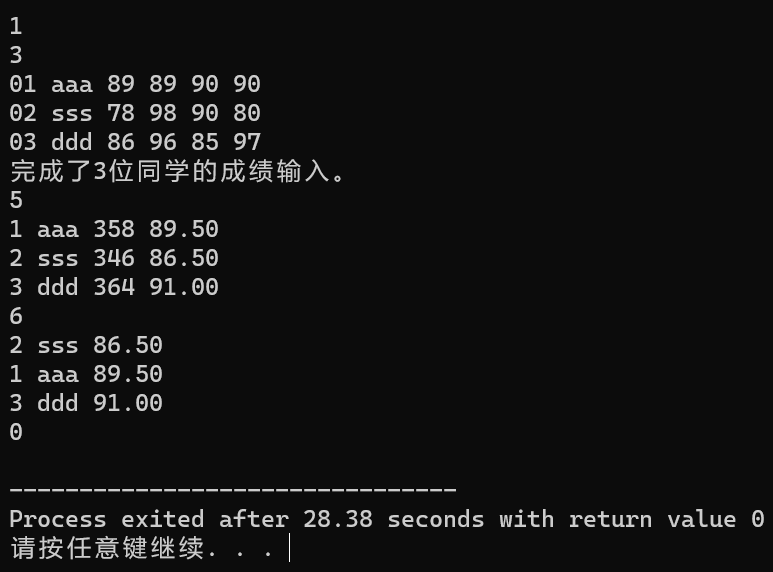


图 3-3 程序设计（2）运行结果

## 7.3 实验小结

本次实验主要学习了结构体、结构体指针的使用，链表的声明和使用。结构体是C语言中一种重要的构造类型，借助结构体，可以实现链表等多种复杂的数据类型。同时，链表的空间不连续性确保了程序的运行效率，相较于普通数组有更大优势。

在本次实验中，我充分认识到了利用指针对链表进行插入结点、删除结点与交换结点等操作的奇妙，也曾因为在交换指针域时错误地忽略了对指针取值从而耗费了大量时间与精力去检查程序错误原因，我认识到很小的疏忽也会在检查程序时增大极其庞大的工作量，程序员就应该“细致入微”。

8 文件操作实验

8.1 实验目的

（1）熟悉文本文件和二进制文件在磁盘中的存储方式；

（2）熟练掌握流式文件的读写方法。

8.2 实验题目及要求

1．文件类型的程序验证题

设有程序：

#include <stdio.h>

int main(void)

{

short a=0x253f,b=0x7b7d;

char ch;

FILE \*fp1,\*fp2;

fp1=fopen("d:\\abc1.bin","wb+");

fp2=fopen("d:\\abc2.txt","w+");

fwrite(&a,sizeof(short),1,fp1);

fwrite(&b,sizeof(short),1,fp1);

fprintf(fp2,"%hx %hx",a,b);

rewind(fp1); rewind(fp2);

while((ch = fgetc(fp1)) != EOF)

putchar(ch);

putchar('\n');

while((ch = fgetc(fp2)) != EOF)

putchar(ch);

putchar('\n');

fclose(fp1);

fclose(fp2);

return 0;

}

1. 请思考程序的输出结果，然后通过上机运行来加以验证。
2. 将两处sizeof(short)均改为sizeof(char)结果有什么不同，为什么？
3. 将fprintf(fp2,"%hx %hx",a,b) 改为 fprintf(fp2,"%d %d",a,b)结果有什么不同。

解答：

1. 输出结果：

?%}{

253f 7b7d

1. 输出结果变为：

?}

253f 7b7d

因为在写入文件fp1时，a与b只取了一个字符大小的前一部分（Windows是大端法系统，高字节储存在低地址中）。

1. 输出结果变为：

?%}{

9535 31613

同（2）因为写入文件fp2时，a与b按整数形式写入，而读取时按字符读取。

**2．源程序修改替换题**

将指定的文本文件内容在屏幕上显示出来，命令行的格式为：

type filename

1. 源程序中存在什么样的逻辑错误（先观察执行结果）？对程序进行修改、调试，使之能够正确完成指定任务。

#include<stdio.h>

#include<stdlib.h>

int main(int argc, char\* argv[])

{

char ch;

FILE \*fp;

if(argc!=2){

printf("Arguments error!\n");

exit(-1);

}

if((fp=fopen(argv[1],"r"))==NULL){ /\* fp 指向 filename \*/

printf("Can't open %s file!\n",argv[1]);

exit(-1);

}

while(ch=fgetc(fp)!=EOF) /\* 从filename中读字符 \*/

putchar(ch); /\* 向显示器中写字符 \*/

fclose(fp); /\* 关闭filename \*/

return 0;

}

解答：

只需将while(ch=fgetc(fp)!=EOF)改为while((ch=fgetc(fp))!=EOF)即可

1. 用输入输出重定向freopen改写main函数。

解答：

改写后的main函数如下：

int main(int argc, char \*argv[])

{

FILE \*fp;

char ch;

fp = freopen(argv[1], "r", stdin);

if (argc != 2)

{

printf("Arguments error!\n");

exit(-1);

}

if (fp == NULL)

{

printf("Can't open %s file!\n", argv[1]);

exit(-1);

}

while((ch = fgetc(fp))!=EOF){

putchar(ch);

}

fclose(fp)

return 0;

}

**3．程序设计**

（1）编写一个程序，用给定的字符串替换文件中的目标字符串，并显示输出替换的个数。

注意：读取的文件路径请使用experiment/src/step8/source.txt

若文件为

`There are moments in life when you miss someone so much that you just want to pick them from your dreams and hug them for real!`

样例输入：`you they`

样例输出：

`3`

`There are moments in life when they miss someone so much that they just want to pick them from theyr dreams and hug them for real!`

解答：

1. 算法流程图如图3-1所示：

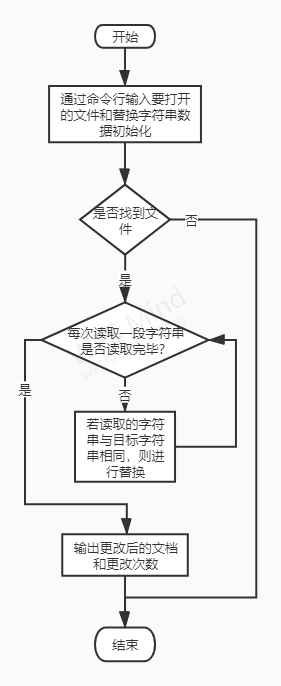


图3-1 程序设计（1）流程图

1. 源程序清单：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main()

{

int count = 0, flag = 0;

FILE \*fp, \*new;

fpos\_t position;

fp = fopen("D:/abc.txt", "r");

new = fopen("D:/abc1.txt", "w+");

char target[10], alter[10], tempc;

scanf("%s%s", target, alter);

while ((tempc = fgetc(fp)) != EOF)

{

if (tempc == target[0])

{

for (int i = 1; i < strlen(target); i++)

{

if (i == 1)

{

fgetpos(fp, &position);

}

if ((tempc = fgetc(fp)) == target[i])

{

flag++;

}

else

{

break;

}

}

if (flag == strlen(target) - 1)

{

fprintf(new, "%s", alter);

count++;

}

else

{

fsetpos(fp, &position);

fprintf(new, "%c", target[0]);

}

}

else

{

fprintf(new, "%c", tempc);

}

flag = 0;

}

fclose(fp);

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

rewind(new);

while ((tempc = fgetc(new)) != EOF)

{

putchar(tempc);

}

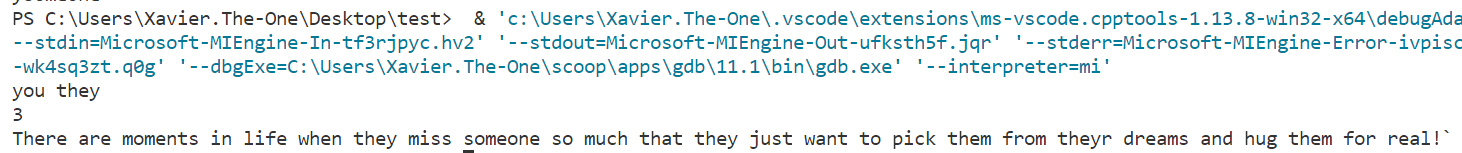
fclose(new);

remove("D:/abc.txt");

rename("D:/abc1.txt","D:/abc.txt");

}

1. 测试：
2. 测试用例如题干所示
3. 对应测试用例的运行结果如图3-2所示：



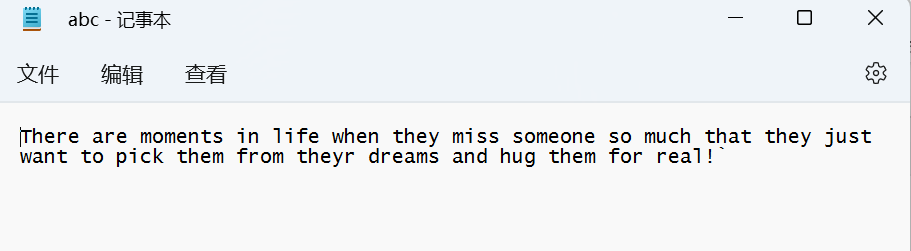


图3-2 程序设计（1）运行结果

8.2 实验小结

本次实验主要学习了C语言中流式文件的使用和文件的存储方式。文件在磁盘中以二进制或字符形式存储，在程序中可以通过函数以特定的格式打开或读写。格式决定了程序能对文件进行的操作。

通过实验，我还认识到了虽然文件指针地址不随fgetc函数的使用而变化，但文件内部其实存在fpos\_t类型的文件内部指针，决定了当前fgetc函数会读取哪一位置的字符。可以通过fgetpos与fsetpos函数进行对文件内部指针的操作。

借由文件，程序可以进行更广泛的数据读取，不再局限于键盘的输入。流式文件的使用尽管复杂，但更贴近底层，固定的打开格式也在一定程度上保护了文件的读写安全。