

**课 程 实 验 报 告**

**课程名称： C语言程序设计**

**专业班级： IOT1601**

**学 号： U201614898**

**姓 名： 潘翔**

**指导教师： 刘芳**

**报告日期： 2017.6.8**

**计算机科学与技术学院**

**目 录**

[**1 表达式和标准输入与输出实验 5**](#_Toc484702386)

[**1.1 实验目的 5**](#_Toc484702387)

[**1.2 实验内容 5**](#_Toc484702388)

[**1.3 自设题 14**](#_Toc484702389)

[**1.4 实验小结 15**](#_Toc484702390)

[**2 流程控制实验 16**](#_Toc484702391)

[**2.1 实验目的 16**](#_Toc484702392)

[**2.2 实验内容 16**](#_Toc484702393)

[**2.3 自设题 30**](#_Toc484702394)

[**2.4 实验小结 32**](#_Toc484702395)

[**3 函数和程序结构实验 33**](#_Toc484702396)

[**3.1 实验目的 33**](#_Toc484702397)

[**3.2 实验内容 33**](#_Toc484702398)

[**3.3 自设题 42**](#_Toc484702399)

[**1.4 实验小结 44**](#_Toc484702400)

[**4 编译预处理实验 45**](#_Toc484702401)

[**4.1 实验目的 45**](#_Toc484702402)

[**4.2 实验内容 45**](#_Toc484702403)

[**4.2.1 源程序改错 45**](#_Toc484702404)

[**4.2.2 源程序修改替换 46**](#_Toc484702405)

[**4.2.3 跟踪调试题 48**](#_Toc484702406)

[**4.2.4 程序设计 50**](#_Toc484702407)

[**4.3 自设题 54**](#_Toc484702408)

[**4.4 实验小结 57**](#_Toc484702409)

[**5 数组实验 58**](#_Toc484702410)

[**5.1 实验目的 58**](#_Toc484702411)

[**5.2 实验内容 58**](#_Toc484702412)

[**5.2.1 源程序改错 58**](#_Toc484702413)

[**5.2.2 源程序完善、修改、替换 59**](#_Toc484702414)

[**5.2.3 跟踪调试源程序 60**](#_Toc484702415)

[**5.2.4 程序设计 61**](#_Toc484702416)

[**5.3 选做题 68**](#_Toc484702417)

[**5.4 自设题 69**](#_Toc484702418)

[**5.5 实验小结 71**](#_Toc484702419)

[**6 指针实验 72**](#_Toc484702420)

[**6.1 实验目的 72**](#_Toc484702421)

[**6.2 实验内容 72**](#_Toc484702422)

[**6.2.1 源程序改错 72**](#_Toc484702423)

[**6.2.2 源程序完善、修改、替换 72**](#_Toc484702424)

[**6.2.3 跟踪调试源程序 75**](#_Toc484702425)

[**6.2.4 程序设计 75**](#_Toc484702427)

[**6.3 选做题 83**](#_Toc484702428)

[**6.4 自设题 86**](#_Toc484702429)

[**6.5 实验小结 89**](#_Toc484702430)

[**7 结构与联合实验 90**](#_Toc484702431)

[**7.1 实验目的 90**](#_Toc484702432)

[**7.2 实验内容 90**](#_Toc484702433)

[**7.2.1 表达式求值的程序验证题 90**](#_Toc484702434)

[**7.2.2 源程序修改替换 91**](#_Toc484702435)

[**7.2.3 编程设计题 93**](#_Toc484702436)

[**7.3 自设题 102**](#_Toc484702437)

[**7.4 选做题 106**](#_Toc484702438)

[**7.4 实验小结 108**](#_Toc484702439)

[**8 指针实验 109**](#_Toc484702440)

[**8.1 实验目的 109**](#_Toc484702441)

[**8.2实验内容及要求 109**](#_Toc484702442)

[**8.2.1. 文件类型的程序验证 109**](#_Toc484702443)

[**8.2.2 源程序修改替换 110**](#_Toc484702444)

[**8.2.3 程序设计 111**](#_Toc484702445)

[**8.3 实验小结 112**](#_Toc484702447)

# 实验1 表达式和标准输入与输出实验

## 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；熟记运算符的优先级和结合性。

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

（3）掌握简单C程序（顺序结构程序）的编写方法。

## 1.2 实验内容

**1.2.1 源程序改错**

下面给出了一个简单C语言程序例程，用来完成以下工作：

（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）输入短整数ｋ、ｐ，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，然后输出；

在这个例子程序中存在若干语法和逻辑错误。要求参照1.3和1.4的步骤对下面程序进行调试修改，使之能够正确完成指定任务。

1. #include<stdio.h>
2. #define PI 3.14159;

3 voidmain( void )

4 {

5 int f ;

6 short p, k ;

7 double c , r , s ;

8 /\* for task 1 \*/

9 printf(“Input Fahrenheit:” ) ;

10 scanf(“%d”, f ) ;

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

12 printf( “ \n %d (F) = %.2f (C)\n\n ”, f, c ) ;

13 /\* for task 2 \*/

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

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

16 s = PI \* r \* r;

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

18 /\* for task 3 \*/

19 printf("input hex int k, p :");

20 scanf("%x %x", &k, &p );

21 newint = (p&0xff00)|(k&0xff00)<<8;

22 printf("new int = %x\n\n",newint);

}

**解答：**

（1）错误修改：

1) 第2行的符号常量定义后不能有分号，正确形式为：

#define PI 3.14159

2) 第3行的voidmain，正确形式为：

void main

3) 第9行的“”，正确形式为：

“”

4) 第10行的“”，正确形式为：

“”

5) 第10行的f应该有&，正确形式为：

&f

6) 第11行的int运算有丢失应转成double，正确形式为：

(double) 5/9\*(f-32);

7) 第12行的“”，正确形式为：

“”

8) 第17行的&s，正确形式为：

s

9) 第15行的f，正确形式为：

lf

10) 第20行的short输入应为%hx而不是x，正确形式为：

hx

11) 第21行的short右移应为>>，正确形式为：

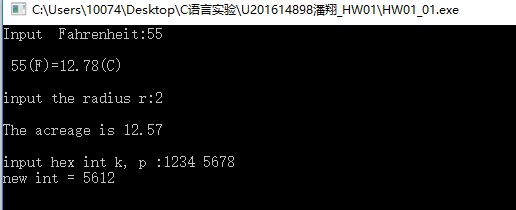
hx

12) 第22行的short输出应为%.4hx

，正确形式为：

%.4hx

（2）错误修改后运行结果：



**1.2.2 源程序修改替换**

下面的程序利用常用的中间变量法实现两数交换，请改用不使用第3个变量的方法实现。该程序中t是中间变量，要求将定义语句中的t删除，修改下划线处的语句，使之实现两数对调的操作。

#include<stdio.h>

void main( )

{

int a, b, t;

printf(“Input two integers:”);

scanf(“%d %d”,&a,&b);

t=a ；a=b；b=t；

prinf(“\na=%d,b=%d”,a,b);

}

**解答：**

将t=a ；a=b；b=t；替换为a = b-a; b = b-a; a =b+a;替换后的程序如下所示：

#include<stdio.h>

void main( )

{

int a, b;

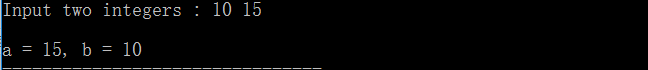
printf(“Input two integers:”);

scanf(“%d %d”,&a,&b);

a = b-a; b = b-a; a =b+a;

prinf(“\na=%d,b=%d”,a,b);

}



**1.2.3 程序设计**

**（1）**编写一个程序，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，最后输出ｃ。

**解答：**

1） 算法流程如图1.1所示。



图1-1 编程题1的程序流程图

2）源程序清单

#include<stdio.h>

int main()

{

char c;

c=getchar();

c=c>='A'&&c<='Z'?c+32:c;

putchar(c);

return 0;

}

3）测试

（a） 测试数据：

A a B b C c

（b） 对应测试数据的运行结果截图

（2）编写一个程序，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。

**解答：**

1) 解题思路：

1.输入x，m，n，为了方便分析测试结果，x的输入采用16进制

2.如果0 ≤ｍ≤ 15, 1 ≤ ｎ≤ 16-ｍ，转2.1，否则转3.

2.1 首先x>>m，将要处理的n位移动到最右；

2.2 再将上一步的结果左移16-n位，即： x=x<<(16-n)

2.3 用16进制输出结果并转4.

3. 显示输入错误信息；

4. 结束



图1-2 编程题2的程序流程图

2）程序清单

#include<stdio.h>

int main()

{

unsigned short x, m, n;

printf("输入x（16进制）、m（0~15）和n（1~16-m）：\n");

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

if (0 <= m && m <= 15 )

{

if (1 <= n && n <= 16 - n)

{

x = x >> m << (15 - n + 1);

printf("ans=%hx\n", x);

}

else

{

printf("n值超范围");

}

}

else

{

printf("m值超范围");

}

return 0;

}

3）测试

（a） 测试数据：

叙述选择测试数据的方法。。。如表1-1所示。

表1-1 编程题2的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 | 计算结果1101 0000 0000 0000 即D000 |
| 用例2 | 1101 0101 1000 0011（D583） | 16 | 1 | 输入错误（m值超范围） |
| 用例3 | 1101 0101 1000 0011（D583） | 13 | 5 | 输入错误（n值超范围） |

（b） 对应测试测试用例1的运行结果如图1-2所示。

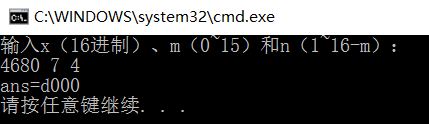


图1-2 编程题2的测试用例一的运行结果

对应测试测试用例2的运行结果如图1-3所示。

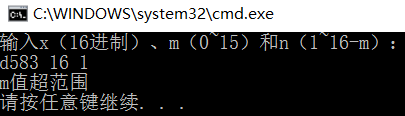


图1-3 编程题2的测试用例二的运行结果

对应测试测试用例3的运行结果如图1-4所示。

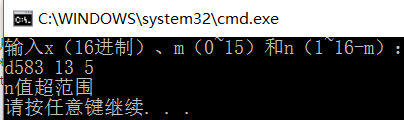


图1-4 编程题2的测试用例三的运行结果

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）**IP地址通常是4个用句点分隔的小整数，如32.55.1.102。这些地址在机器中用无符号长整形表示。编写一个程序，以机器存储的形式读入一个32位的互联网IP地址，对其译码，然后用常见的句点分隔的4部分的形式输出。

**解答：**

1. 解题思路：



1. 程序清单

#include<stdio.h>

int main()

{

int N;

scanf("%d", &N);

char str\_ip\_index[4] = { '\0' };

unsigned long int IP;

unsigned long int IP1;

unsigned long int IP2;

unsigned long int IP3;

unsigned long int IP4;

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

{

scanf("%lu", &IP);

IP1 = ((IP & 0xff000000) >> 24);

IP2 = ((IP & 0x00ff0000) >> 16);

IP3 = ((IP & 0x0000ff00) >> 8);

IP4 = ((IP & 0x000000ff));

printf("%lu.%lu.%lu.%lu\n", IP4, IP3, IP2, IP1);

}

return 0;

}

3）测试

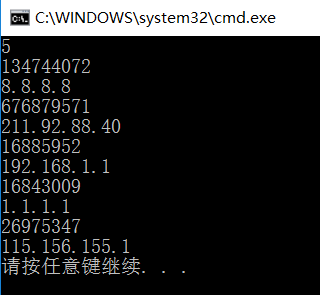
（a） 测试数据：

叙述选择测试数据的方法使数据覆盖尽量广泛 各节相同和不同如表1-2所示。

表1-2 编程题3的测试数据

|  |  |  |
| --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 |
| X |
| 用例1 | 134744072 | 8.8.8.8 |
| 用例2 | 676879571 | 211.92.88.40 |
| 用例3 | 16885952 | 192.168.1.1 |
| 用例4 | 16843009 | 1.1.1.1 |
| 用例5 | 26975347 | 115.156.115.1 |

（b） 对应测试测试用例的运行结果如图1-2所示。



## 1.3 自设题

**（1）** 自设实验题目：编写一个程序, 对用户录入的产品信息进行格式化, 程序运行后需要有以下会话:

INPUT:

GroupNumber:1

Enter item number: 583

Enter unit price: 13.5

Enter purchase date(mm/dd/yy): 10/24/95

OUTPUT:

Item      Unit Price      Purchase Date

583        $ 13.50          10/24/95

其中, 数字项和日期项单位价格左对齐, 美元数量最大取值为9999.99

**（2）** 实验目的：通过设计实验程序，熟练标准输入输出，掌握输出表格的方法

**（3）**实验程序：

#include <stdio.h>

int main()

{

int GroupNumber=0;

int number[9] = {0};

float price[9] = { 0 };

int year[9] = { 0 };

int month[9] = { 0 };

int day[9] = { 0 };

printf("GroupNumber:");

scanf("%d", &GroupNumber) ;

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

{

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

scanf("%f", &price[i]);

scanf("%d/%d/%d", &month[i], &day[i], &year[i]);

}

printf("Item \t Unit\t Purchase\n");

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

{

printf("%-9d$ %-9.2f%02d/%02d/%02d\n", number[i], price[i], month[i], day[i], year[i]);

}

getchar();

return 0;

}

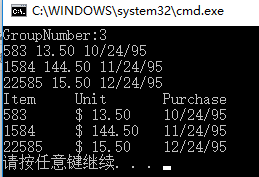
**（4）**实验用例：

3

583 13.50 10/24/95

1584 144.50 11/24/95

22585 15.50 12/24/95



**（5）**实验结论：

可以使用格式化输入输出制作表格，使打印清晰

## 1.4 实验小结

**（1）**对于可能工程中不太常见的输入输出格式不熟悉，通过练习得到了巩固。

**（2）**对于输入需要输出进行检测避免后续处理错误的输入数据导致DEBUG困难。

**（3）**学习使用测试用例和有效的测试方法。

**（4）**注重细节，培养习惯才能在工程代码中少出错。

# 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Turbo C 2.0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**

下面给出了一个简单C语言程序例程，用来完成以下工作：

（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）输入短整数ｋ、ｐ，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，然后输出；

在这个例子程序中存在若干语法和逻辑错误。要求参照1.3和1.4的步骤对下面程序进行调试修改，使之能够正确完成指定任务。

1. #include<stdio.h>

2 void main(void)

3 {

4 int i,n,s=1;

5 printf("Please enter n:");

6 scanf("%d",n);

7 for(i=1,i<=n,i++)

8 s=s\*i;

9 printf("%d! = %d",n,s);

10 }

**解答：**

（1）错误修改：

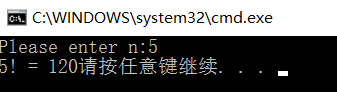
1) 第6行的scanf("%d",n);，正确形式为：

scanf("%d",&n);

2) 第7行的for(i=1,i<=n,i++)，正确形式为：

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

（2）错误修改后运行结果：



**2.2.2 源程序修改替换**

（1）修改第1题，分别用while和do-while语句替换for语句。

**解答：**

将for (i = 1; i <= n; i++) s = s\*i;

替换为

i = 1;

while (i <= n)

{

s = s\*i;

i++;

}替换后的程序如下所示：

#include <stdio.h>

void main(void)

{

int i, n, s = 1;

printf("Please enter n:");

scanf("%d", &n);

i = 1;

while (i <= n)

{

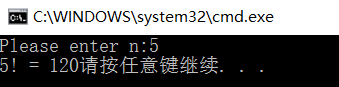
s = s\*i;

i++;

}

printf("%d! = %d", n, s);

}



**解答：**

#include <stdio.h>

void main(void)

{

int i, n, s = 1;

printf("Please enter n:");

scanf("%d", &n);

i = 1;

do

{

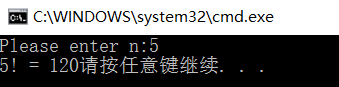
s = s\*i;

i++;

} while (i <= n);

printf("%d! = %d", n, s);

}



（2）修改第1题，输入改为“整数S”，输出改为“满足n！≥S的最小整数n”。例如输入整数40310，输出结果为n=8。

**解答：**

#include <stdio.h>

void main(void)

{

int i, n, s = 1;

int S;

printf("Please enter S:");

scanf("%d", &S);

i = 1;

while (s < S)

{

s = s\*i;

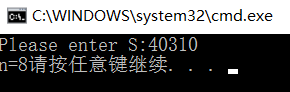
i++;

}

i--;

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

}



**2.2.3 程序设计**

（1）求方程的近似根

**解答：**

1） 算法流程如图2.1所示。



图2-1 编程题1的程序流程图

2）源程序清单

#include<stdio.h>

#include<math.h>

int a, b, c, d;

float Solve(float a, float b, float c, float d);

int main()

{

a = 3; b = -4; c = -5; d = 13;

printf("%lf\n", Solve(a, b, c, d));

return 0;

}

float Solve(float a, float b, float c, float d)

{

float x = 1, x0=2, f, f1;

while (fabs(x - x0) >= 1e-5)

{

x0 = x;

f = ((a\*x0 + b)\*x0 + c)\*x0+d;

f1 = (3 \* a\*x0 + 2 \* b)\*x0 + c;

x = x0 - f / f1;

}

return x;

}

3）测试



（2）打印如下杨辉三角形。

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

每个数据值可以由组合计算（表示第i行第j列位置的值），而的计算如下：

 (i=0,1,2,…)

 (j=0,1,2,3,…,i)

本程序中为了打印出金字塔效果，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，程序编制过程中要注意区分。

**解答：**

1） 算法流程如图2.1所示。



图2-2 编程题2的程序流程图

2）源程序清单

#include<stdio.h>

void YangHui(int n);

void CalculateYH(int n);

void PrintSpace(int n);

int DigitNum(int n);

int a[13][13];

int main()

{

int n ;

scanf("%d", &n);

while (n != 0)

{

YangHui(n);

printf("\n");

scanf("%d", &n);

}

return 0;

}

void CalculateYH(int n)

{

int i, j;

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

{

a[i][1] = 1; /\*第一列全置为一\*/

}

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

{

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

{

a[i][j] = a[i - 1][j - 1] + a[i - 1][j];

}

}

}

void YangHui(int n)

{

CalculateYH(n);

int i, j;

for (i = 1; i<=n; i++) /\*输出杨辉三角\*/

{

PrintSpace(2 \* (n - i));

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

{

if ((j == i))

{

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

}

else

{

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

PrintSpace(4 - DigitNum(a[i][j + 1]));

}

}

printf("\n");

}

}

void PrintSpace(int n)

{

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

{

printf(" ");

}

}

int DigitNum(int n)

{

int sum=0;

while (n)

{

sum++;

n /= 10;

}

return sum;

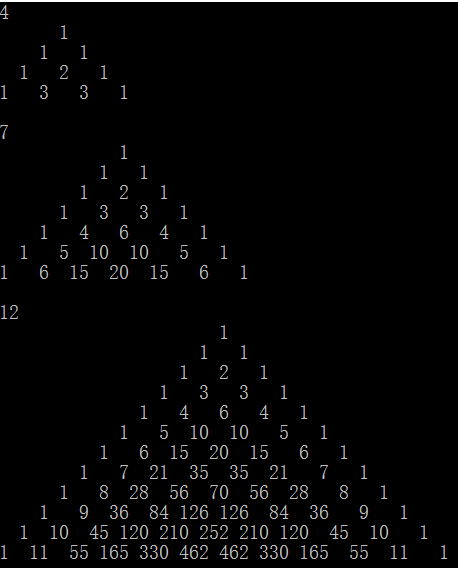
}

3）测试

（a） 测试数据：

4 7 12

（b） 对应测试数据的运行结果截图



（3）正整数逆转，输入任意正整数逆转。

输入：多组输入，每组一个正整数m，输入为0结束

输出：每行对应一个输出

**解答：**

1) 解题思路：



图2-3 编程题3的程序流程图

2）程序清单

/\*整数翻转\*/

#include<stdio.h>

void NumReverse(int n);

int main()

{

int n;

scanf("%d", &n);

while (n != 0)

{

NumReverse(n);

scanf("%d", &n);

}

return 0;

}

void NumReverse(int n)

{

int ModNum = 0;// 余数

int result = 0;

while (n / 10 != 0) //一位数跳出循环

{

ModNum = n % 10;

n = n / 10;

result = result \* 10 + ModNum;

}

result=result \* 10 + n;

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

}

3）测试

（a） 测试数据

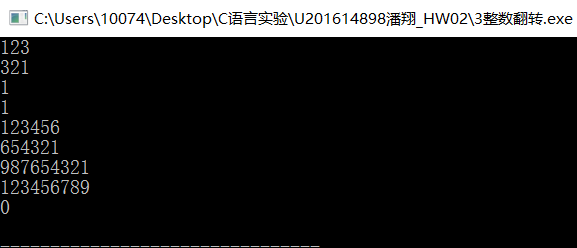
123

1

123456

987654321

（b） 对应测试测试用例1的运行结果如图1-2所示。



**（4）个人所得税。**

**解答：**

1. 解题思路：

使用多层if实现分段函数

1. 程序清单

/\*个人所得税\*/

#include<stdio.h>

#include<stdbool.h>

#include<math.h>

void CountTax(double salary);

bool IsZero(double salary);

int main()

{

double salary;

scanf("%lf", &salary);

while (!IsZero(salary))

{

CountTax(salary);

scanf("%lf", &salary);

}

return 0;

}

void CountTax(double salary)

{

//printf("%.6lf\n", salary);

int n;

double Tax;

if (salary <= 1000)

{

n = 1;

}

if (1000 < salary&&salary <= 2000)//50

{

n = 2;

}

if (2000 < salary&&salary <= 3000)//100

{

n = 3;

}

if (3000 < salary&&salary <= 4000)//150

{

n = 4;

}

if (4000 < salary&&salary <= 5000)//200

{

n = 5;

}

if (5000 < salary)

{

n = 6;

}

switch (n)

{

case 1:Tax = 0; break;

case 2:Tax = (salary-1000)\*0.05; break;

case 3:Tax = 50+(salary - 2000)\*0.10; break;

case 4:Tax = 150 + (salary - 3000)\*0.15; break;

case 5:Tax = 300 + (salary - 4000)\*0.20; break;

case 6:Tax = 500 + (salary - 5000)\*0.25; break;

default:

break;

}

printf("%.6lf\n", Tax);

}

bool IsZero(double salary)

{

if (fabs(salary - 0) < 0.000001)

{

return true;

}

else

{

return false;

}

}

3）测试

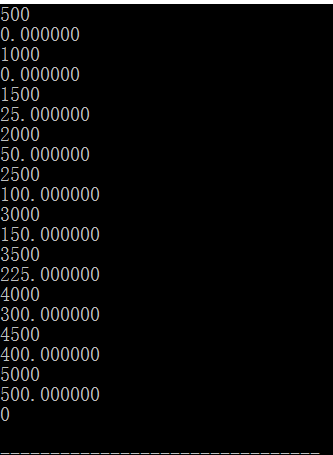
（a） 测试数据：

叙述选择测试数据的方法使数据覆盖尽量广泛 各节相同和不同如表1-2所示。

表1-1 编程题4的测试数据

|  |  |  |
| --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 |
| X |
| 用例1 | 500 | 0.000000 |
| 用例2 | 1000 | 0.000000 |
| 用例3 | 1500 | 25.000000 |
| 用例4 | 2000 | 50.000000 |
| 用例5 | 2500 | 100.000000 |
| 用例6 | 3000 | 150.000000 |
| 用例7 | 3500 | 225.000000 |
| 用例8 | 4000 | 300.000000 |
| 用例9 | 4500 | 400.000000 |
| 用例10 | 5000 | 500.000000 |

（b） 对应测试测试用例的运行结果如图1-2所示。



**（5）个人所得税。**

**解答：**

1. 解题思路：



图2-4 编程题5的程序流程图

1. 程序清单

#include<stdio.h>

void fun();

int main()

{

int N;

scanf("%d", &N);

getchar();

//char input[100] = " ";

//fgets(input, 100, stdin);

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

{

fun();

}

}

void fun()

{

char c = 0, flag = 0;

while ((c = getchar()) != '\n')

{

if (c == ' ')

{

if (flag == 0)

{

flag = 1;

}

else

{

continue;

}

}

else

{

flag = 0;

}

printf("%c", c);

}

printf("\n");

}

3）测试

（a） 测试数据：

选择不同的空格数目和位置

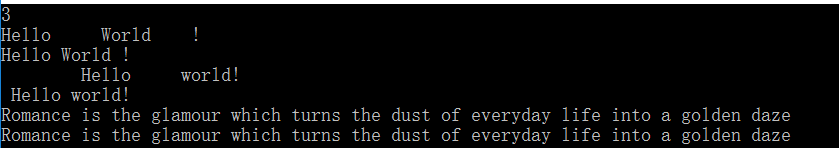
3

Hello World !

Hello world!

Romance is the glamour which turns the dust of everyday life into a golden daze

（b） 对应测试测试用例的运行结果如图1-2所示。



## 2.3 自设题

**（1）** 自设实验题目：

描述：

给定一个由不同的小写字母组成的字符串，输出这个字符串的所有全排列。 我们假设对于小写字母有'a' < 'b' < ... < 'y' < 'z'，而且给定的字符串中的字母已经按照从小到大的顺序排列。

输入：

输入只有一行，是一个由不同的小写字母组成的字符串，已知字符串的长度在1到6之间。

输出：

输出这个字符串的所有排列方式，每行一个排列。要求字母序比较小的排列在前面。字母序如下定义：  
  
已知S = s1s2...sk , T = t1t2...tk，则S < T 等价于，存在p (1 <= p <= k)，使得  
s1 = t1, s2 = t2, ..., sp - 1 = tp - 1, sp < tp成立。

其中, 数字项和日期项单位价格左对齐, 美元数量最大取值为9999.99

**（2）** 实验目的：结合算法设计，熟悉流程实现方法

**（3）**实验程序：

#include <stdio.h>

#include <string>

string array;

int len;

void swap(int a, int b) {

char temp = array[b];

array[b] = array[a];

array[a] = temp;

}

void reverse(int a, int b) {

while (a < b) {

swap(a, b);

a++;

b--;

}

}

void permutation() {

int i,j;

while (true) {

printf("%s\n",array.c\_str());

for (i = len - 2; i >= 0; i--) {

if (array[i] < array[i+1]) {

break;

}

if (i == 0) {

return;

}

}

for (j = len - 1; j > i ; j--) {

if (array[j] > array[i]) {

break;

}

}

swap(i,j);

reverse(i+1,len-1);

}

}

int main(){

char data[10];

while(scanf("%s",&data) != EOF ){

array = data;

len = array.size();

if (len == 1) {

printf("%s\n",array.c\_str());

printf("\n");

continue;

}

permutation();

printf("\n");

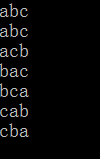
}

return 0;

}

**（4）**实验用例**：**

abc



**（5）**实验结论：

字典序输出的流程控制，需要在每一层逻辑清楚

## 2.4 实验小结

**（1）**对于测试格式的严格要求，当测试不通过的时候检查空格换行等错误

**（2）**对于scanf后getchar()吞回车符的应用

**（3）**学习使用测试用例和有效的测试方法。

**（4）**注重细节，培养习惯才能在工程代码中少出错。

**（5）**结合算法和OJ题目，提高程序编写能力。

# 实验3 函数和程序结构实验

## 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方法；以及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）熟悉多文件编译技术。

## 3.2 实验内容

**3.2.1 源程序改错**

下面是计算s=1!+2!+3!+…+n!的源程序，在这个源程序中存在若干语法和逻辑错误。要求在计算机上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。

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

**解答：**

（1）错误修改：

1) 第1行的“stdio.h”，正确形式为：

< stdio.h >

2) 在1和2之间加入long sum\_fac(int n)的函数声明：

long sum\_fac(int n);

3) 第12行的fac初始化，正确形式为：

Long fac=1;

4) 第15行的s+=fac;加入for循环体下面，正确形式为：

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

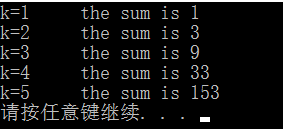
{

fac \*= i;

s += fac;

}

（2）错误修改后运行结果：



**3.2.2 源程序修改替换**

（1）修改第1题中sum\_fac函数，使其计算量最小。

（2）修改第1题中sum\_fac函数，计算C:\Users\10074\AppData\Local\Temp\ksohtml\wpsCAC8.tmp.png。

**解答：**

（1）

#include <stdio.h>

long sum\_fac(int n);

int k;

void main(void)

{

for (k = 1; k<6; k++)

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

}

long s = 0;

long fac = 1;

long sum\_fac(int n)

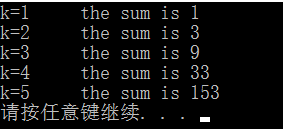
{

fac \*= k;

s += fac;

return s;

}



（2）修改第1题中sum\_fac函数，计算C:\Users\10074\AppData\Local\Temp\ksohtml\wpsCAC8.tmp.png。

#include <stdio.h>

float sum\_fac(int n);

void main(void)

{

int k;

for (k = 1; k<6; k++)

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

}

float sum\_fac(int n)

{

float s = 0;

int i;

float fac = 1;

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

{

fac \*= (float)1 / i;

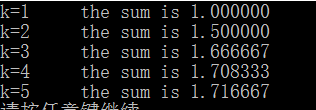
//printf("%lf\n", fac);

s += fac;

}

return s;

}



**3.2.3 跟踪调试题**

计算fabonacci数列前n项和的程序如下：

其中，long sum=0,\*p=∑声明p为长整型指针并用&sum取出sum的地址对p初始化。\*p表示引用p所指的变量（\*p即sum）。

void main(void)

{

int i,k;

long sum=0,\*p=∑

scanf("%d",&k);

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

sum+=fabonacci(i);

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

}

}

long fabonacci(int n)

{

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

return 1;

else

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

}

单步执行程序，观察p,i,sum,n值。

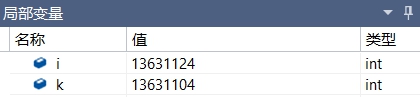
（1）刚执行完scanf("%d",&k);语句，p,i值是多少？

（2）从fabonacci函数返回后光条停留在哪个语句上？

（3）进入fabonacci函数，watch窗口显示的是什么？

（4）当i=3，从调用fabonacci函数到返回，n值如何变化？

（1）



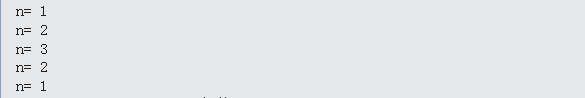
（2）



（3）

选择监视的变量值

（4）



**3.2.4 程序设计**

（1）编程让用户输入两个整数，计算两个数的最大公约数并且输出之（要求用递归函数实现求最大公约数）。同时以单步方式执行该程序，观察递归过程。

**解答：**

1） 算法流程如图3.1所示。



图3-1 编程题1的程序流程图

2）源程序清单

#include<stdio.h>

int MaxComDivisor(int a, int b);

int main()

{

int a;

int b;

scanf("%d", &a);

while (a != 0)

{

scanf("%d", &b);

printf("%d\n", MaxComDivisor(a, b));

scanf("%d", &a);

}

return 0;

}

int MaxComDivisor(int a, int b)

{

if (b == 0)

{

return a;

}

else

{

return MaxComDivisor(b, a%b);

}

}

3）测试

（a） 测试数据：

1 6

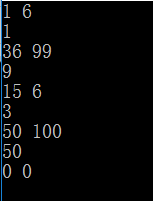
36 99

15 6

50 100

0 0

（b） 对应测试数据的运行结果截图



（2）编程验证歌德巴赫猜想：一个大于等于4的偶数都是两个素数之和。

编写一个程序证明对于在符号常量BEGIN和END之间的偶数这一猜测成立。例如，如果BEGIN为10，END为20，程序的输出应为：

GOLDBACH'S CONJECTURE:

Every even number n>=4 is the sum of two primes.

10=3+7

12=5+7

……

20=3+17

**解答：**

1) 解题思路



图3-2 编程题2的程序流程图

2）程序清单

#include<stdio.h>

#include<math.h>

#include<stdbool.h>

void Goldbach(int k);

bool IsPrime(int k);

bool IsEven(int i);

int main()

{

int begin, end;

scanf("%d", &begin);

while (begin!=0)

{

scanf("%d", &end);

for (int i = begin; i<= end; i++)

{

if (IsEven(i))

{

Goldbach(i);

}

}

printf("\n");

scanf("%d", &begin);

}

return 0;

}

void Goldbach(int k)

{

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

{

if (IsPrime(i) && IsPrime(k - i))

{

printf("%d=%d+%d\n", k, i, k - i);

break;

}

}

}

bool IsPrime(int k)

{

int m = (int)sqrt(k);

for (int j = 2; j <= m; j++)

{

if (k%j == 0)

{

return false;

}

}

return true;

}

bool IsEven(int i)

{

if (i % 2 == 0)

{

return true;

}

else

{

return false;

}

}

3）测试

（a） 测试数据：

6 10

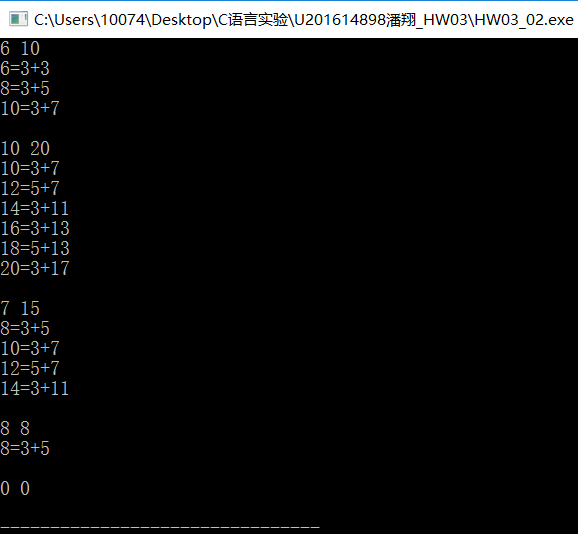
10 20

7 15

8 8

0 0

（b） 对应测试测试用例的运行结果如图所示。



**3.2.5 选做s题**

1、设file1.c如下：

#include <stdio.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 \*/

void main(void)

{

x=10;

y=20;

ch=getchar();

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

func1();

}

file2.c如下：

extern int x,y; /\* 外部变量的引用性说明 \*/

extern char ch; /\* 外部变量的引用性说明 \*/

void func1(void)

{

x++;

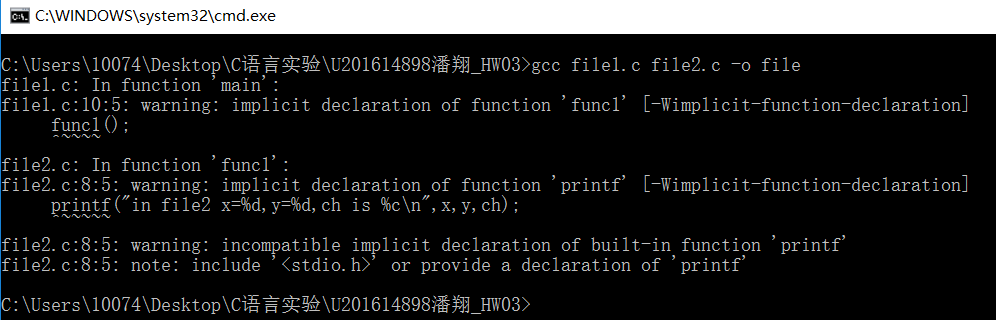
y++;

ch++;

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

}

试用TCC进行多文件编译和链接。然后在DOS环境下运行生成的可执行文件。



## 3.3 自设题

**（1）** 自设实验题目：

描述

在一个给定形状的棋盘（形状可能是不规则的）上面摆放棋子，棋子没有区别。要求摆放时任意的两个棋子不能放在棋盘中的同一行或者同一列，请编程求解对于给定形状和大小的棋盘，摆放k个棋子的所有可行的摆放方案C。

输入

输入含有多组测试数据。  
每组数据的第一行是两个正整数，n k，用一个空格隔开，表示了将在一个n\*n的矩阵内描述棋盘，以及摆放棋子的数目。 n <= 8 , k <= n  
当为-1 -1时表示输入结束。  
随后的n行描述了棋盘的形状：每行有n个字符，其中 # 表示棋盘区域， . 表示空白区域（数据保证不出现多余的空白行或者空白列）。

输出

对于每一组数据，给出一行输出，输出摆放的方案数目C （数据保证C<2^31）。

样例输入

2 1

#.

.#

4 4

...#

..#.

.#..

#...

-1 -1

样例输出

2

1

**（2）** 实验目的：通过设计实验程序，熟练函数和程序结构设计

**（3）**实验程序：

#include<stdio.h>

#include<string.h>

int n,k,ans;

char map[12][12];//棋盘

int vis[12];

int DFS(int i,int cur)

{

if(cur>=k) //

{

ans++; //方案数

return 0;

}

int x,y;

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

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

if(!vis[y] && map[x][y]=='#')

{

vis[y]=1;//标记

DFS(x+1,cur+1);//递归

vis[y]=0;

}

return 0;

}

int main()

{

while(scanf("%d%d",&n,&k)&&n!=-1)

{

ans=0;

memset(map,0,sizeof(map));

memset(vis,0,sizeof(vis));

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

scanf("%s",map[i]);

DFS(0,0);

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

}

return 0;

}

**（4）**实验用例：

2 1

#.

.#

4 4

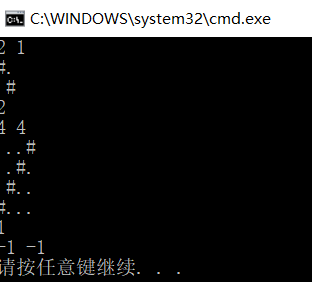
...#

..#.

.#..

#...

-1 -1



**（5）**实验结论：

可以使用函数块设计，使程序清晰

## 1.4 实验小结

**（1）**对于养成结构化设计习惯，自顶向下设计，使结构清晰。

**（2）**对于单步调试DEBUG的使用。

**（3）**学习使用测试用例和有效的测试方法。

**（4）**注重细节，培养习惯才能在工程代码中少出错。

# 实验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、assert宏的使用；
2. 练习带参数的宏定义、条件编译的使用；
3. 练习assert宏的使用；
4. 使用Turbo C 2.0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。

## 4.2 实验内容

4.2.1 源程序改错

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语法和逻辑错误。要求在计算机上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。

1. #include "stdio.h"
2. #define SUM a+b
3. #define DIF a-b
4. #define SWAP(a,b) a=b,b=a
5. void main
6. {
7. int b, t;
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
11. is:%d",SUM, SUM\*DIF);
12. SWAP(a,b);
13. Printf("\nNow a=%d,b=%d\n",a,b);
14. }

**解答：**

（1）错误修改：

1) 第1行的"stdio.h"，正确形式为：

<stdio.h>

2) 第4行的a=b,b=a，正确形式为：

t=a;a=b;b=t

3) 第5行的void main，正确形式为：

void main()

4)第6和7行添加a定义，正确形式为：

int a;

5) 第9行%d,%d中间无逗号，正确形式为：

%d%d

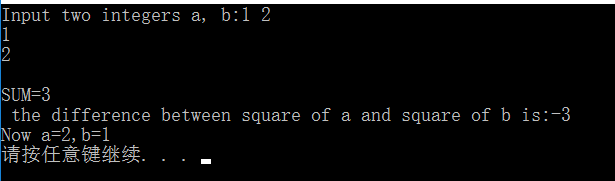
6) 第11行SUM DIF需加括号，正确形式为：

(SUM)\*(DIF)

7) 第13行Printf，正确形式为：

printf;

（2）错误修改后运行结果：



4.2.2 源程序修改替换

（1）下面是用函数实现求三个数中最大数、计算两数之和的程序，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。

void main(void)

{

int a, b, c;

float d, e;

printf("Enter three integers:");

scanf("%d,%d,%d",&a,&b,&c);

printf("\nthe maximum of them is %d\n",max(a,b,c));

printf("Enter two floating point numbers:");

scanf("%f,%f",&d,&e);

printf("\nthe sum of them is %f\n",sum(d,e));

}

int max(int x, int y, int z)

{

int t;

if (x>y)

t=x;

else

t=y;

if (t<z)

t=z;

return t;

}

float sum(float x, float y)

{

return x+y;

}

**解答：**

1. 修改：

#include<stdio.h>

int max(int x, int y, int z);

float sum(float x, float y);

float f;

void main(void)

{

int a, b, c;

float d, e;

printf("Enter three integers:");

scanf("%d%d%d", &a, &b, &c);

printf("\nthe maximum of them is %d\n", max(a, b, c));

printf("Enter two floating point numbers:");

scanf("%f%f", &d, &e);

printf("\nthe sum of them is %f\n", sum(d, e));

}

int max(int x, int y, int z)

{

int t;

if (x>y)

t = x;

else

t = y;

if (t<z)

t = z;

return t;

}

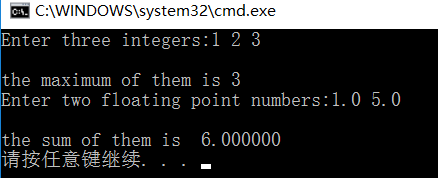
float sum(float x, float y)

{

return x+y;

}

运行结果：



**2）解答：**

#include<stdio.h>

#define MAX(a,b,c) (a>b?(a>c?a:c):(b>c?b:c))

float sum(float x, float y);

float f;

void main(void)

{

int a, b, c;

float d, e;

int m;

printf("Enter three integers:");

scanf("%d%d%d", &a, &b, &c);

printf("\nthe maximum of them is %d\n", MAX(a, b, c));

printf("Enter two floating point numbers:");

scanf("%f%f", &d, &e);

printf("\nthe sum of them is %f\n", sum(d, e));

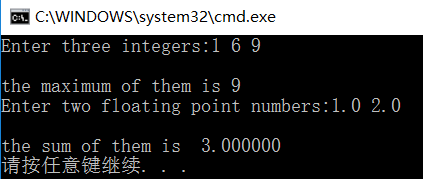
}

float sum(float x, float y)

{

return x+y;

}



4.2.3 跟踪调试题

下面程序利用R计算圆的面积s，以及面积s的整数部分。

#define R

void main(void)

{

float r, s;

int s\_integer=0;

printf ("input a number: ");

scanf("%f",&r);

#ifdef R

s=3.14159\*r\*r;

printf("area of round is: %f\n",s);

s\_integer= integer\_fraction(s);

printf("the integer fraction of area is %d\n", s\_integer);

assert((s-s\_integer)<1.0);

#endif

}

int integer\_fraction(float x)

{

int i=x;

return i;

}

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decimal\_fraction时watch窗口中x为何值？在返回main时, watch窗口中i为何值？

3）排除错误，使程序能正确输出面积s值的整数部分，不会输出错误信息assertion failed。

1）

#include<stdio.h>

#include<assert.h>

#define R

int integer\_fraction(float x);

int main(void)

{

float r, s;

int s\_integer=0;

printf("input a number: ");

scanf("%f",&r);

#ifdef R

s=3.14159\*r\*r;

printf("area of round is: %f\n",s);

s\_integer=integer\_fraction(s);

printf("the integer fraction of area is %d\n", s\_integer);

assert((s-s\_integer)<1.0);

#endif

}

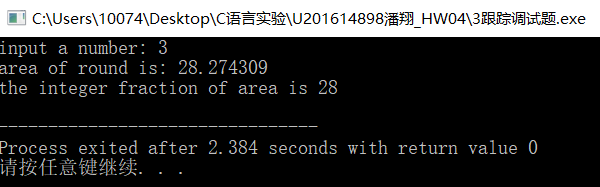
int integer\_fraction(float x)

{

int i = x;

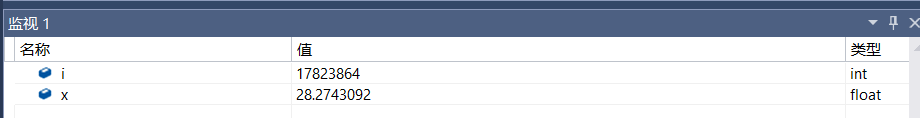
return i;

}



2）

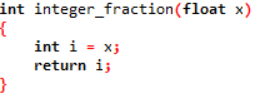
**进入**：



**出来：**



3）

进行转换 满足小于1.0条件

4.2.4 程序设计

（1）求三角形的面积是C:\Users\10074\AppData\Local\Temp\ksohtml\wpsAC24.tmp.png，其中C:\Users\10074\AppData\Local\Temp\ksohtml\wpsAC25.tmp.png，a,b,c为三角形的三边，定义两个带参数的宏，一个用来求s，另一个用来求area。编写程序，用带参数的宏来计算三角形的面积。

**解答：**

1） 算法流程如图4.1所示。



图4-1 编程题1的程序流程图

2）源程序清单

#include<stdio.h>

#include<math.h>

#define s(a,b,c) (a+b+c)/2

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

int main()

{

double a, b, c;

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

{

int s = s(a, b, c);

double area = area(a, b, c);

printf("%d ", s);

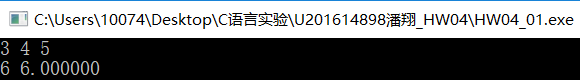
printf("%lf\n", area);

}

return 0;

}

3）测试



（2）用条件编译方法来编写程序。输入一行电报文字，可以任选两种输出：一为原文输出；二为变换字母的大小写（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） 算法流程如图4.2所示。



图4-2 编程题2的程序流程图

2）源程序清单

#include<stdio.h>

#include<string.h>

#define CHANGE c%2

void Conversion();

char s[100];

int len;

int count = 0;

int i;

int main()

{

int N;

scanf("%d", &N);

getchar();

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

{

fgets(s, 100, stdin);

len = strlen(s)-1;

//printf("%d\n", len);

char c = s[0];

if (CHANGE != 0)

{

Conversion();

}

else

{

printf("%s", s);

}

}

}

void Conversion()

{

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

{

if ('a' <= s[i] && s[i] <= 'z')

{

s[i] = s[i] - 32;

}

else if ('A' <= s[i] && s[i] <= 'Z')

{

s[i] = s[i] + 32;

}

}

printf("%s", s);

}

3）测试

（a） 测试数据：

hello !

world !

hello world!

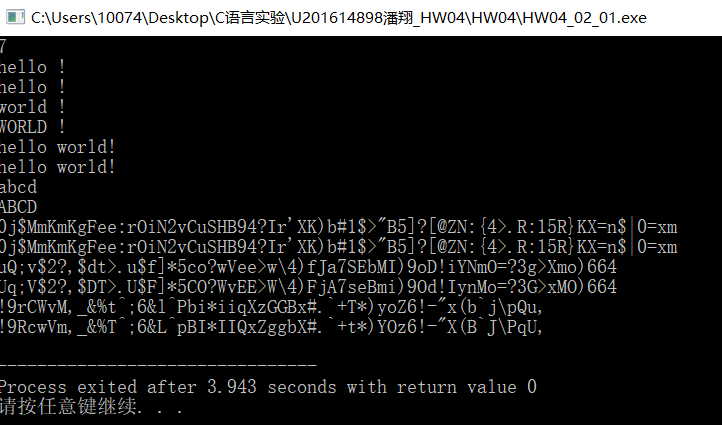
abcd

0j$MmKmKgFee:rOiN2vCuSHB94?Ir'XK)b#1$>"B5]?[@ZN:{4>.R:15R}KX=n$|0=xm

uQ;v$2?,$dt>.u$f]\*5co?wVee>w\4)fJa7SEbMI)9oD!iYNmO=?3g>Xmo)664

!9rCWvM,\_&%t^;6&l^Pbi\*iiqXzGGBx#.`+T\*)yoZ6!-"x(b`j\pQu,

（b） 对应测试数据的运行结果截图



## 4.3 自设题

**（1）**自设实验题目：

描述

T. Chur教大学的各类学生。每个U-student都有一个独特的学生识别号码（SIN）。 SIN s是范围为0≤s≤MaxSIN，MaxSIN = 106-1的整数。 T. Chur发现这个SIN的范围太大，无法识别她的团体。 对于每个组，她想要找到最小的正整数m，使得在组内，所有SIN减小模m是唯一的。

输入

在输入的第一行是单个正整数N，告诉测试用例（组）的数量。 每个案例从包含整数G（1≤G≤300）的一行开始：组中的学生人数。 以下G行各自包含一个SIN。 一组内的SIN是不同的，但不一定是排序的。

输出

对于每个测试用例，输出包含最小模数m的一行，使得模m减少的所有SIN都不同。

样例输入

2

1

124866

3

124866

111111

987651

样例输出

1

8

**（2）**实验目的：结合算法设计，熟悉宏定义memset实现方法

#define MEM(a,x) memset(a,x,sizeof(a))

**（3）**实验程序：

#include <iostream>

#include <cstdio>

#include <cstring>

#include <cstdlib>

#define MEM(a,x) memset(a,x,sizeof(a))

using namespace std;

const int MAXN = 1000010;

bool visit[MAXN];

bool remainder[MAXN];

int main()

{

int t;

int n;

int num[310];

scanf("%d",&t);

while(t--)

{

scanf("%d",&n);

memset(visit, 0, sizeof(visit));

int maxNum = 0;

int i;

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

{

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

}

int j;

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

{

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

{

visit[abs(num[i] - num[j])] = true;

}

}

for(i = 1; ; i++)

{

if(!visit[i])

{

// cout<<"i= "<<i<<endl;

MEM(remainder,0);

bool flag = true;

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

{

if(remainder[num[j] % i] == true)

{

flag = false;

break;

}

remainder[num[j] % i] = true;

}

if(flag == true)

{

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

break;

}

}

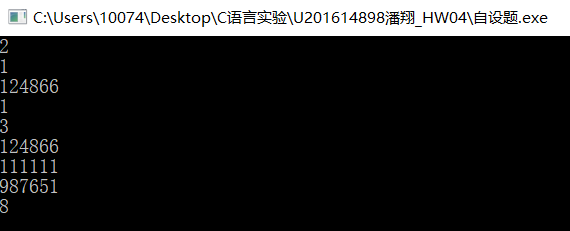
}

}

return 0;

}

**（4）**实验用例**：**



**（5）**实验结论：

对于多层FOR循环 和常用的内存初始化函数宏定义的熟悉 使结构清晰

#define FOR(i,f\_start,f\_end) for(int i=f\_start;i<=f\_end;++i)

#define MEM(x,i) memset(x,i,sizeof(x))

## 4.4 实验小结

**（1）**恰当使用宏定义使常用的常量不输入出错和结构清晰

**（2）**学习使用测试用例和有效的测试方法。

**（3）**注重细节，培养习惯才能在工程代码中少出错。

**（4）**结合算法和OJ题目，提高程序编写能力。

# 实验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

5.2.1 源程序改错

下面是用来将数组a中元素按升序排序后输出的源程序。分析源程序中存在的问题，并对源程序进行修改，使之能够正确完成任务。

源程序

1 #include<stdio.h>

2 int main(void)

3 {

4 int a[10] = {27, 13, 5, 32, 23, 3, 17, 43, 55, 39};

5 void sort(int [],int);

6 int i;

7 sort(a[0],10);

8 for(i = 0; i < 10; i++)

9 printf("%6d",a[i]);

10 printf("\n");

11 return 0;

12 }

13 void sort(int b[], int n)

14 {

15 int i, j, t;

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

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

18 if(b[j] < b[j+1])

19 t = b[j], b[j] = b[j+1], b[j+1] = t;

20 }

**解答：**

（1）错误修改：

1) 第5行函数声明应在main函数外。

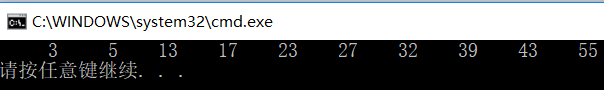
2) 第7行函数参数应为数组名，正确形式为：

sort(a,10);

3) 第18行<应为>号，正确形式为：

if(b[j] >b[j+1])

（2）错误修改后运行结果：



5.2.2 源程序完善、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。请在源程序中的下划线处填写合适的代码来完善该程序。

源程序：

#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]; ;

}

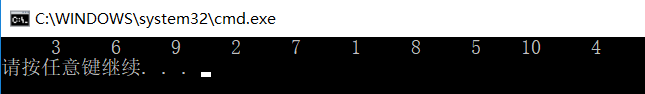
for(i = 0;i < M – 1; i++) /\* 按次序输出出圈人的编号 \*/

printf(“%6d”, b[i]);

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

return 0;

}



(2)使用标记法

#include<stdio.h>

#include<stdbool.h>

#define M 10

#define N 3

bool visited[M] = {0};

int main(void)

{

int i, j, k;

k = 0;

j = 0;

i = M;

do

{

++j;//圈中的所有位置

if (j > M)

j = 1;

if (!visited[j])

k++;

if (k == N)//当前报的数是M 重头开始

{

k = 0;

printf("% 6d", j);//输出被访问编号

visited[j] = true;//此处人已访问，设置为空

i--;

}

} while (i != 0);//直到所有人都访问过

}

5.2.3 跟踪调试源程序

在下面所给的源程序中，函数strncat(s,t,n)本来应该将字符数组t的前n个字符连接到字符数组s中字符串的尾部。但函数strncat在定义时代码有误，不能实现上述功能。请按下面的要求进行操作，并回答问题和排除错误。

（1） 单步执行源程序。进入函数strncat后观察表达式s、t和i。当光条落在for语句所在行时，i为何值？当光条落在strncat函数块结束标记（右花括号 }）所在行时, s、t分别为何值？

（2）分析函数出错的原因，排除错误，使函数正确实现功能，最后写出程序的输出结果。

源程序：

#include<stdio.h>

void strncat(char [],char [],int);

int main(void)

{

char a[50]="The adopted symbol is ",b[27]="abcdefghijklmnopqrstuvwxyz";

strncat(a, b, 4);

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

return 0;

}

void strncat(char s[],char t[], int n)

{

int i = 0, j;

while(s[i++]) ;

for(j = 0; j < n && t[j];)

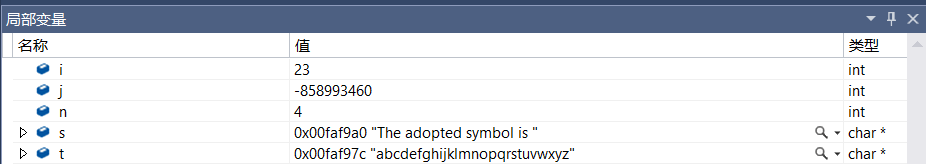
s[i++] = t[j++];

s[i] = '\0';

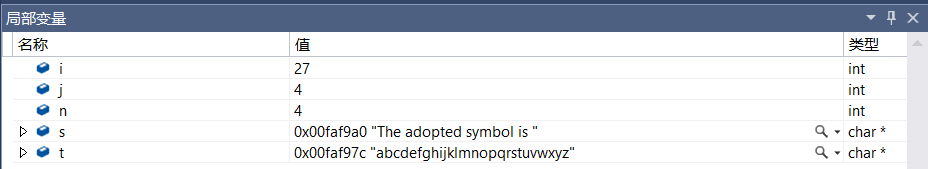
}

(1)单步调试

进入for循环：



**结束：**



(2)分析函数出错的原因，排除错误，使函数正确实现功能，最后写出程序的输出结果。

#include<stdio.h>

void strncat(char[], char[], int);

int main(void)

{

char a[50] = "The adopted symbol is ", b[27] = "abcdefghijklmnopqrstuvwxyz";

strncat(a, b, 4);

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

return 0;

}

void strncat(char s[], char t[], int n)

{

int i = 0, j;

while (s[i++]);

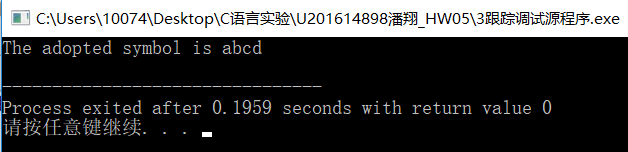
i--;

for (j = 0; j < n && t[j];)

s[i++] = t[j++];

s[i] = '\0';

}



5.2.4 程序设计

（1）编写一个程序，其功能要求是：输入n个学生的姓名（不超过100个学生，姓名字符数不大于20）和C语言课程的成绩，将成绩按照从高到低的次序排序，姓名同时作相应调整，输出排序后学生的姓名和C语言课程的成绩。然后，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，输出该成绩同学的姓名和C语言课程成绩；否则输出提示“Not found!”。

输入：第一行为正整数n，接下来的n行是n个学生和对应的成绩信息；然后是一个正整数N，表示需要查询N次，接下来的一行是N个需要查询的成绩。

输出：首先输出排序后的姓名和成绩信息，姓名和成绩用空格隔开，且姓名的宽度为20（左对齐，空格补齐）；输出一个空行；然后是N行查询结果，若查询到则输出其姓名和成绩信息，格式与前者一致，否则输出“Not found!”。

注意：为了方便查找，成绩为正整数，且无重复成绩。

**解答：**

1）算法流程如图5.1所示。



图5-1 编程题1的程序流程图

2）源程序清单

#include<stdio.h>

int Resort(int begin, int end);

int Inquire(int score, int begin, int end);

struct Student

{

char name[15];

int score;//成绩

};

struct Student stu[20];

int main()

{

int n;

scanf("%d", &n);

getchar();

int result;

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

{

scanf("%s%d",stu[i].name,&stu[i].score);

getchar();

}

Resort(1, n);

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

{

printf("%-20s %d\n", stu[i].name, stu[i].score);

}

printf("\n");

int N;

int temp\_score;

scanf("%d", &N);

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

{

scanf("%d", &temp\_score);

Inquire(temp\_score,1,n);

}

return 0;

}

int Resort(int begin, int end)

{

if (begin == end)

{

return 0;

}

int maxi=begin;

struct Student temp;

for (int i = begin; i <= end; i++)

{

if (stu[i].score > stu[maxi].score)

{

maxi = i;

}

}

temp = stu[begin];

stu[begin] = stu[maxi];

stu[maxi] = temp;

return Resort(begin+1, end);

}

int Inquire(int score,int begin,int end)

{

int flag=0;

int half = (begin + end) / 2;

if (begin >= end || stu[half].score == score)

{

if (score == stu[half].score)

{

printf("%-20s %d\n", stu[half].name, stu[half].score);

return half;

}

else

{

printf("Not found!\n");

return -1;

}

}

if (begin < end)

{

if (stu[half].score>score)

{

return Inquire(score, half + 1, end);

}

else

{

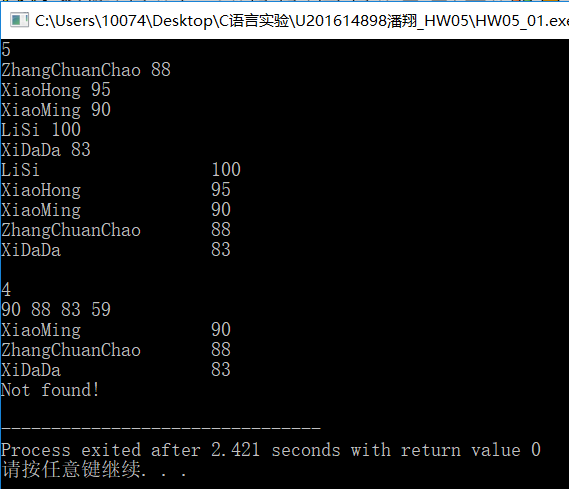
return Inquire(score, begin, half);

}

}

}

3）测试



（2）

|  |
| --- |
| 编写一个程序，其功能要求是：输入一个整数，将它在内存中二进制表示的每一位转换成为对应的数字字符，存放到一个字符数组中，然后输出该整数的二进制表示。  输入：第一行为正整数N，表示有N组输入；第二行为N个int型整数，即N组输入。  输出：对每组输入的整数，输出其二进制表示，每组一行 |

**解答：**

1） 算法流程如图5.2所示。



图5-2 编程题2的程序流程图

2）源程序清单

#include<stdio.h>

void ToBin(int n);

int N;

int i;

int BIN[100][33];

int main()

{

int n = 0;

scanf("%d", &n);

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

{

scanf("%d", &N);

ToBin(N);

}

return 0;

}

void ToBin(int N)

{

int t;

t = 1;

t = t << 31;

for (int j = 1; j <= 32; j++)

{

BIN[i][j]=( N&t ? '1' : '0');

printf("%c",BIN[i][j]);

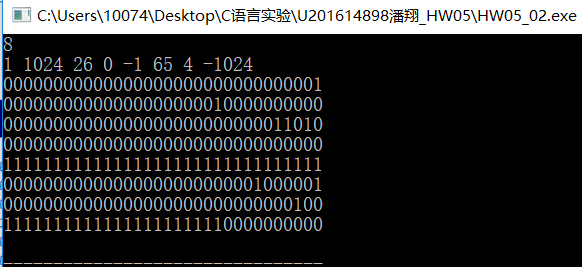
N = N<< 1;

}

printf("\n");

}

3）测试



（3）编写一个程序，其功能要求是：输入一个整数，将它在内存中二进制表示的每一位转换成为对应的数字字符，存放到一个字符数组中，然后输出该整数的二进制表示。

写一个程序，从键盘读取数据，对一个3\*4矩阵进行赋值，求其转置矩阵，然后分别输出原矩阵和转置矩阵。

输入：一个3\*4的整数矩阵。

输出：输出原矩阵和转置矩阵，每个数的宽度为5（右对齐，空格补齐），原矩阵和转置矩阵用空行隔开

1. 算法流程如图下所示。
   1. 取数组a[4][5]从a[1][1]开始存取先行后列
   2. 输出数组先列后行
2. 源程序清单

#include<stdio.h>

void transpose();

int a[4][5];

int main()

{

int i, j;

for (i = 1; i <=3; i++)

{

for (j = 1; j <=4; j++)

{

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

}

}

for (i = 1; i <=3; i++)

{

for (j = 1; j<=4; j++)

{

printf("%5d",a[i][j]);

}

printf("\n");

}

printf("\n");

transpose();

return 0;

}

void transpose()

{

int i, j;

for (j = 1; j <= 4; j++)

{

for (i = 1; i <= 3; i++)

{

printf("%5d", a[i][j]);

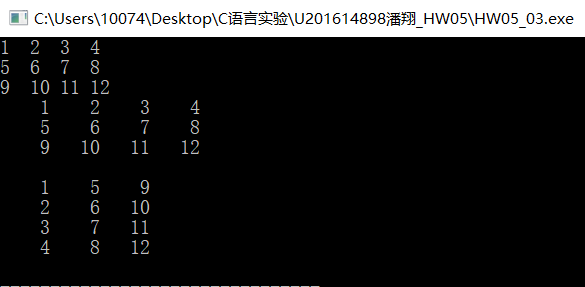
}

printf("\n");

}

}

1. 测试



## 5.3 选做题

（1）字符串中间插入函数

1）源程序清单

#include<stdio.h>

#include<string.h>

#include<stdbool.h>

bool strnins(char s[], char t[], int n);

int main()

{

char a[50] = "The adopted symbol is ", b[27] = "abcdefghijklmnopqrstuvwxyz";

strnins(a, b, 5);

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

return 0;

}

bool strnins(char s[], char t[], int n)

{

char temp[100];

if (n < 0 || n>strlen(s))

{

printf("Error!\n");

return false;

}

strcpy(temp, s + n);

strcpy(s+n, t);//t+n之后添加到s结尾处

strcat(s, temp);

return true;

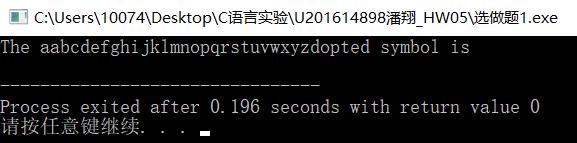
}

2）测试用例：

The adopted symbol is

abcdefghijklmnopqrstuvwxyz

3）测试结果:

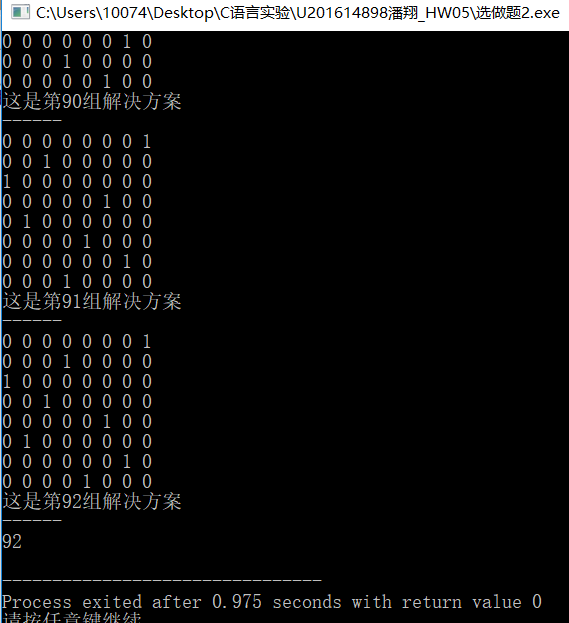


(2)八皇后

1）源程序清单：

2）测试用例：无

3）测试结果：



## 5.4 自设题

1. 自设实验题目

描述：

输入一个n维矩阵 求它的行列式

输入：

维数为n的数组

输出：

它的行列式

1. 实验目的：熟悉数组的使用方法，考虑矩阵运算库的编写
2. 实验程序：

#include<stdio.h>

#include<math.h>

int countA(int a[100][100], int n);

int main()

{

int a[100][100];

int i, j;

int n;

scanf("%d", &n);

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

{

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

{

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

}

}

printf("%d\n", countA(a,n));

return 0;

}

int countA(int a[100][100], int n)

{

int b[100][100] = { { 0 } };

int i = 0, j = 0, sum = 0; //i,j为行与列,sum为行列式的值

int x = 0, c = 0, p = 0; //用x判断加与减符号

if (n == 1)

{

return a[0][0];

}

for (i = 0; i < n; i++) //余子式存储

{

for (c = 0; c < n - 1; c++)

{

for (j = 0; j < n - 1; j++)

{

if (c < i) //判断行列式的移动方向

{

p = 0; //当p=0时,行列式只向左移,即消去对应的第一列的数

}

else

{ //否则行列式左移后再上移

p = 1;

}

b[c][j] = a[c + p][j + 1];

}

}

if (i % 2 == 0) { //i+j（此时j=0）为偶数,加法

x = 1;

}

else //i+j为奇数,减法

{

x = (-1);

}

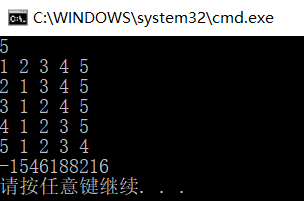
sum += a[i][0] \* countA(b,n-1) \* x; //计算行列式的值

}

return sum;

}

1. 实验用例：



1. 实验结论：

编写考虑输入判错，并使接口清晰

## 5.5 实验小结

**（1）**注意数组下标的起始位置，考虑从0开始符合人类思考习惯

**（2）**对多层for语句的输出逻辑清楚

**（3）**数组指针间接访问的掌握

# 实验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 实验内容

6.2.1 源程序改错

下面程序是否存在错误？如果存在，原因是什么？如果存在错误，要求在计算机上对这个例子程序进行调试修改，使之能够正确执行。

#include "stdio.h"

void main(void)

{

float \*p;

scanf("%f",p);

printf("%f\n",\*p);

}

**解答：**

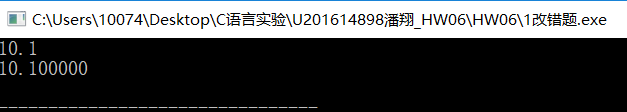
（1）错误修改：

添加指针所指向的元素，防止野指针

float a;

p = &a;

（2）错误修改后运行结果：



6.2.2 源程序完善、修改、替换

（1）下面的程序通过函数指针和菜单选择来调用字符串拷贝函数或字符串连接函数，请在下划线处填写合适的表达式、语句、或代码片段来完善该程序。

#include "stdio.h"

#include "string.h"

void main(void)

{

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

char a[80], b[80], c[160], \*result = c;

int choice, i;

do {

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

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

printf("\t\t3 exit.\n");

printf("\t\tinput a number (1-3) please!\n");

scanf("%d", &choice);

} while (choice<1 || choice>3);

switch (choice) {

case 1:

p = strcpy;

break;

case 2:

p = strcat;

break;

case 3:

goto down;

}

getchar();

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

i = 0;

scanf("%s", a);

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

i = 0;

scanf("%s", b);

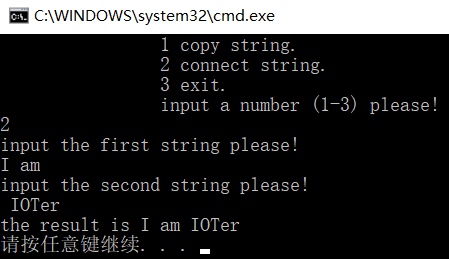
result = p(a, b);

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

down:

;

}



（2）为了使程序不受scanf、getchar、gets等函数输入后回车符的影响，请修改第（1）题程序，按要求输出下面结果：（（输入）表示该数据是键盘输入数据）

1 copy string.

2 connect string.

3 exit.

input a number (1-3) please!

2 （输入）

input the first string please!

the more you learn, （输入）

input the second string please!

the more you get. （输入）

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

#include "stdio.h"

#include "string.h"

void main(void)

{

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

char a[80], b[80], c[160], \*result = c;

int choice, i;

do {

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

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

printf("\t\t3 exit.\n");

printf("\t\tinput a number (1-3) please!\n");

scanf("%d", &choice);

} while (choice<1 || choice>3);

switch (choice) {

case 1:

p = strcpy;

break;

case 2:

p = strcat;

break;

case 3:

goto down;

}

getchar();

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

i = 0;

while ((a[i++] = getchar()) != '\n');

a[i - 1] = '\0';

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

i = 0;

while ((b[i++] = getchar()) != '\n');

b[i - 1] = '\0';

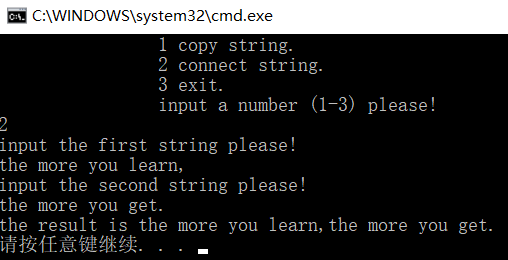
result = p(a, b);

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

down:

;

}



6.2.3 跟踪调试源程序

#include "stdio.h"

char \*strcpy(char \*,char \*);

void main(void)

{

char a[20],b[60]="there is a boat on the lake.";

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

}

char \*strcpy(char \*s,char \*t)

{

while(\*s++=\*t++)

;

return (s);

}

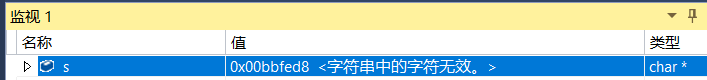
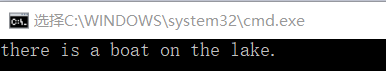
（1）单步执行。进入strcpy时watch窗口中s为何值？返回main时, watch窗口中s为何值？

（2）排除错误，使程序输出结果为：

there is a boat on the lake.

（3）选做：由于watch窗口中只显示s所指串的值，不显示s中存储的地址值，怎样才能观察到s值的变化呢？

解答

1. 
2. 
3. 在VS2015中后附显示，或定义一个char变量，然后监视

6.2.4 程序设计

（1）知一个长整型变量占4个字节，其中每个字节又分成高4位和低4位。试从该整型变量的高字节开始，依次取出每个字节的高四位和低四位并以其值的十六进制字符（大写）形式进行显示。

输入：第一行为正整数N，表示有N组输入；每组输入为一个整数，第二行即为N个整数。

输出：每组输入对应一行输出，每两个字符用空格隔开，每行最后一个字符后无空格

**解答：**

1）算法流程如图6.1所示。



图6-1 编程题1的程序流程图

2）源程序清单

#include<stdio.h>

void Address(long int \*p);

int main()

{

long int L;

long int\* p\_L=&L;

int N;

scanf("%d", &N);

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

{

scanf("%ld", &L);

Address(&L);

printf("\n");

}

return 0;

}

void Address(long int \*p)

{

unsigned mask = 0xf0000000;

printf("%X ", (unsigned )(mask&\*p) >> 28);

for (int i = 1; i <= 6; i++)

{

mask >>= 4;

printf("%X ", (unsigned)(mask&\*p) >> (28 - 4 \* i));

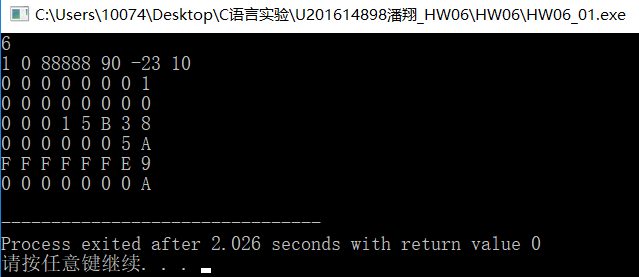
}

mask >>= 4;

printf("%X", (unsigned)(mask&\*p));

}

3）测试



（2）

|  |
| --- |
| 利用大小为n的指针数组指向用gets函数输入的n行，每行不超过80个字符。  编写一个函数，它将每一行中连续的多个空格字符压缩为一个空格字符。在调用函数中输出压缩空格后的各行，空行不予输出。 |

**解答：**

1） 算法流程如图6.2所示。



图6-2 编程题2的程序流程图

2）源程序清单

#include<stdio.h>

#include<string.h>

void fun(char len[][80], int i);

int main()

{

int n;

char len[20][80];

int i;//代表行数

while (scanf("%d", &n), n)

{

getchar();

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

{

fgets(len[i], 500, stdin);

if (len[i][0] != '0'&& len[i][0] != '\n')

{

fun(len, i);

}

}

printf("\n");

}

return 0;

}

void fun(char len[][80],int i)

{

int length=strlen(len[i])-1;

char c = 0, flag = 0;

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

{

c = len[i][j];

if (c == ' ')

{

if (flag == 0)

{

flag = 1;

}

else

{

continue;

}

}

else

{

flag = 0;

}

printf("%c", c);

}

printf("\n");

}

3）测试



（3）设某个班有N个学生，每个学生修了M门课程（用#define定义N、M，M=N=5）。

输入M门课程的名称。然后依次输入N个学生中每个学生所修的M门课程的成绩并且都存放到相应的数组中。编写下列函数：

（1）计算每个学生各门课程平均成绩；

（2）计算全班每门课程的平均成绩；

（3）分别统计低于全班各门课程平均成绩的人数；

（4）分别统计全班各门课程不及格的人数和90分以上（含90分）的人数；

在调用函数中输出上面各函数的计算结果。

（要求都用指针操作，不得使用下标操作）

输入：第一行为5门课程的课程名；接下来每两行为一组，第一行为学生姓名，第二行为该学生对应的5门课程成绩。

输出：输出统计结果，具体格式见样例输出

**解答：**

1. 算法流程下所示。

利用结构体指针访问并统计各项数据

1. 源程序清单

#include<stdio.h>

#define M 5

#define N 5

typedef struct Student

{

char name[20];

int scores[M+1];

} Student;

void AveOfStudent(Student\* p\_stu);

void AveOfClasses(Student\* p\_stu, char classes[M+1][20], float \*aveOfClasses);

void LowerThanAve(float \*aveOfClasses, Student \*p\_stu, char classes[M + 1][20]); //低于课程的平均分的人数

void FailNum(Student \*p\_stu, char classes[M + 1][20]);

void PerfectNum(Student \*p\_stu, char classes[M + 1][20]);

int main()

{

Student stu[N+1];

Student\* p\_stu = stu;

char classes[M+1][20];

float aveOfClass[M+1];//课程成绩

float\* p\_aveOfClass = aveOfClass;

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

{

scanf("%s", classes[i]);

}

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

{

scanf("%s", stu[i].name);

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

{

scanf("%d", &stu[i].scores[j]);

}

}

for (int i = 1; i <= N; i++)//每一个学生平均成绩

{

AveOfStudent(p\_stu +i);

}

AveOfClasses(p\_stu, classes, aveOfClass);

LowerThanAve(aveOfClass, p\_stu, classes);

FailNum(p\_stu, classes);

PerfectNum(p\_stu, classes);

return 0;

}

void AveOfStudent(Student\* p\_stu)

{

int sum=0;

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

{

sum = sum + p\_stu->scores[k];

}

printf("Average score of %s is %.2f\n", (\*p\_stu).name, (double)sum / M);

}

void AveOfClasses(Student\* p\_stu,char classes[M + 1][20], float \*aveOfClasses)

{

int sum;

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

{

sum = 0;

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

{

sum = sum + (\*(p\_stu+j)).scores[i];

}

aveOfClasses[i] = (double)sum / M;

printf("Average score of %s is %.2f\n", classes[i], aveOfClasses[i]);

}

}

void LowerThanAve(float \*aveOfClasses, Student \*p\_stu,char classes[M + 1][20]) //低于课程的平均分的人数

{

int sum;

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

{

sum = 0;

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

{

if ((\*(p\_stu + j)).scores[i] < aveOfClasses[i])

{

sum++;

}

}

printf("Number of students lower than avg of %s is %d\n", classes[i],sum);

}

}

void FailNum(Student \*p\_stu, char classes[M + 1][20])

{

int sum;

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

{

sum = 0;

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

{

if ((\*(p\_stu + j)).scores[i] < 60)

{

sum++;

}

}

printf("Number of students %s fail is %d\n", classes[i], sum);

}

}

void PerfectNum(Student \*p\_stu, char classes[M + 1][20])

{

int sum;

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

{

sum = 0;

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

{

if ((\*(p\_stu+j)).scores[i] > 90)

{

sum++;

}

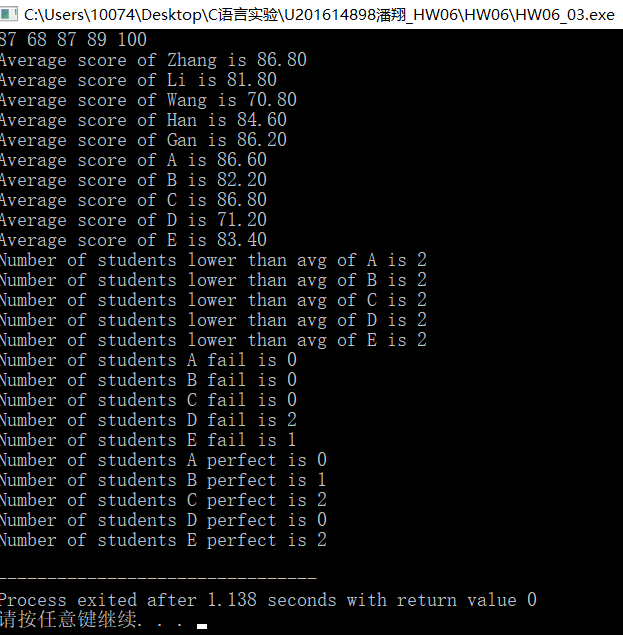
}

printf("Number of students %s perfect is %d\n", classes[i],sum);

}

}

3） 测试



## 6.3 选做题

（1）设有N位整数和M位小数（N=20，M=10）的数据a,b。编程计算a+b并输出结果。

如：12345678912345678912.1234567891 + 98765432109876543210.0123456789

1）源程序清单

#include<stdio.h>

#include<string.h>

char s1[110], s2[110];//存放字符串形式

int num1[220], num2[220];//存放数字形式

char result[220];

void init();

void CharToNum(char \*s, int \*a);

void sum(char \*result, int\* num1, int\* num2);

void print();

int main()

{

while (scanf("%s %s", s1, s2) != EOF)

{

init(); //每计算一次初始化数组

CharToNum(s1, num1);

CharToNum(s2, num2);

sum(result, num1,num2); //进位求和过程

print();

}

return 0;

}

void init()//对数组初始化处理

{

memset(num1, 0, sizeof(num1));

memset(num2, 0, sizeof(num2));

memset(result, 0, sizeof(result));

}

void CharToNum(char \*s, int \*a)

{

int len = strlen(s);

int k;

if (strchr(s, '.') != NULL) //判断有无小数点

k = strchr(s, '.') - s; //标记小数点的位置

else

k = len; //无小数点

for (int i = k + 1, j = 99; i<len; i++, j--) //小数点后面的存到前

a[j] = s[i] - '0';

for (int i = k - 1, j = 100; i >= 0; i--, j++) //小数点前面的存放到后

a[j] = s[i] - '0';

}

void sum(char \*result, int\* num1,int\* num2)

{

int s, v = 0;

for (int i = 0; i<220; i++)

{

s = num1[i] + num2[i] + v;

result[i] = s % 10;

v = s / 10;

}

}

void print()

{

int i = 220;

while (result[i] == 0 && i >= 100)

{

i--;

}

int j = 0;

while (result[j] == 0 && j<100)

{

j++;

}

if (i == 99 && j == 100) //全是零的情况

printf("0\n");

else

{

for (; i >= 100; i--)

{

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

}

if (j != 100)

{

printf(".");

}

for (int i = 99; i >= j; i--)

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

printf("\n");

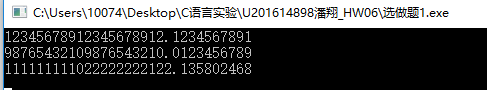
}

}2）测试用例：

12345678912345678912.123456789

98765432109876543210.0123456789

3）测试结果:



（2）编写使用复杂声明char \*(\*p[2])(const char \*,const char \*);的程序。

提示：p中元素可为strcmp、strstr等函数名。

1. 源程序清单：

/\*因为编译器原因 对申明稍加更改\*/

#include <stdio.h>

#include <string.h>

int main()

{

char\* (\_\_cdecl \*p[2])(char \* , char const\* );

p[0] = strcpy;

p[1] = strstr;

char test1[100] = "I AM AN IOTER";

const char test2[100] = "i am an ioter";

printf("%c\n", \*(\*p[0])(test1, test2));

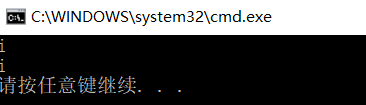
printf("%c\n", \*(\*p[1])(test1, test2));

return 0;

}

2）测试用例：无

3）测试结果：



## 6.4 自设题

1. 自设实验题目

描述：

对一个字符串形式的输入处理

输入：

123+321

输出：

444

1. 实验目的：熟悉函数数组指针的使用方法
2. 实验程序：

#include<stdio.h>

#include<string.h>

#include<math.h>

void init();

void CharToNum(char \*s, int \*a, int \*n1, int \*n2);

int add(int a, int b);

int sub(int a, int b);

int mul(int a, int b);

int div(int a, int b);

int symbol;

int(\*a[4])(int a, int b) = { add,sub,mul,div };

char s[110];

int num1[220], num2[220];//存放数字形式

int n1, n2;

char result[220];

int main()

{

init(); //每计算一次初始化数组

scanf("%s",s);

CharToNum(s, num1,&n1,&n2);

printf("%d", a[symbol](n1, n2));

return 0;

}

int add(int a, int b)

{

return a + b;

}

int sub(int a, int b)

{

return a - b;

}

int mul(int a, int b)

{

return a\*b;

}

int div(int a, int b)

{

if (b != 0)

{

return a / b;

}

else

{

return -1;

}

}

void init()//对数组初始化处理

{

memset(num1, 0, sizeof(num1));

memset(num2, 0, sizeof(num2));

memset(result, 0, sizeof(result));

}

void CharToNum(char \*s, int \*a, int \*n1, int \*n2)

{

int k = 0;

int len = strlen(s);

if (strchr(s, '+') != NULL)

{

k = strchr(s, '+') - s;

symbol = 0;

}

if (strchr(s, '-') != NULL)

{

k = strchr(s, '-') - s;

symbol = 1;

}

if (strchr(s, '\*') != NULL)

{

k = strchr(s, '\*') - s;

symbol = 2;

}

if (strchr(s, '/') != NULL)

{

k = strchr(s, '/') - s;

symbol = 3;

}

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

{

num1[i]= s[i] - '0';

\*n1 += (int)pow(10, k - 1 - i)\*num1[i];

}

for (int i = k+1; i < len; i++)

{

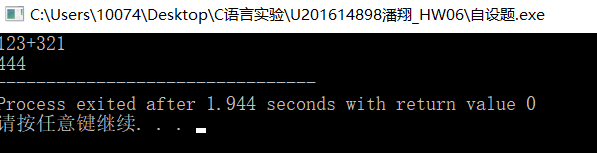
num2[i] = s[i] - '0';

\*n2 += (int)pow(10, len - 1 - i)\*num2[i];

}

}

1. 实验用例：



1. 实验结论：

使用函数结构指针可简化SWITCH

## 6.5 实验小结

**（1）**注意函数声明在不同的编译器的处理不同

**（2）**函数指针数组的使用

**（3）**高精度算法的掌握

# 实验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 实验内容

7.2.1 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;

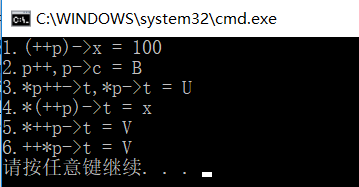
char c;

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 | x |
| 4 | \*(++p)->t | x | x |
| 5 | \*++p->t | V | V |
| 6 | ++\*p->t | V | V |



7.2.2 源程序修改替换

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。

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：**

**（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：**

**源代码清单：**

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

{

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

if (p[0] == 0) /\* 相当于\*p==0 \*/ //保存的数组为空

;

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 \*/

loc\_head->next = NULL;

tail = loc\_head; /\* tail指向第一个结点 \*/

while (\*p) //p还有值

{ /\* tail所指结点的指针域指向动态创建的结点 \*/

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

tail->next = loc\_head; /\* 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 \*/

loc\_head = tail;

}

//tail->next = NULL; /\* 对指针域赋NULL值 \*/

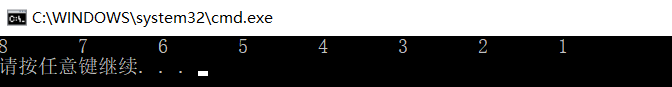
}

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

return headp;

}

测试结果：



7.2.3 编程设计题

（1）设计一个字段结构struct bits，它将一个8位无符号字节从最低位向最高位声明为8个字段，各字段依次为bit0, bit1, …, bit7，且bit0的优先级最高。同时设计8个函数，第i个函数以biti(i=0,1,2,…,7)为参数，并且在函数体内输出biti的值。将8个函数的名字存入一个函数指针数组p\_fun。如果bit0为1，调用p\_fun[0]指向的函数。如果struct bits中有多位为1，则根据优先级从高到低依次调用函数指针数组p\_fun中相应元素指向的函数。8个函数中的第0个函数可以设计为：

void f0(struct bits b)

{

Printf(“the function %d is called!\n”,b);

}

**解答：**

1） 算法流程如下所示

1.输入转换为八位二进制数

2.如为1 则输出 所在位中断

2）源程序清单

#include<stdio.h>

struct ISR\_BITS

{

unsigned int bit0 : 1;

unsigned int bit1 : 1;

unsigned int bit2 : 1;

unsigned int bit3 : 1;

unsigned int bit4 : 1;

unsigned int bit5 : 1;

unsigned int bit6 : 1;

unsigned int bit7 : 1;

unsigned int rsv : 8;

};

union ISR\_REG

{

unsigned short all;

struct ISR\_BITS bit;

};

void isr0()

{

printf("The Interrupt Service Routine isr0 is called!\n");

}

void isr1()

{

printf("The Interrupt Service Routine isr1 is called!\n");

}

void isr2()

{

printf("The Interrupt Service Routine isr2 is called!\n");

}

void isr3()

{

printf("The Interrupt Service Routine isr3 is called!\n");

}

void isr4()

{

printf("The Interrupt Service Routine isr4 is called!\n");

}

void isr5()

{

printf("The Interrupt Service Routine isr5 is called!\n");

}

void isr6()

{

printf("The Interrupt Service Routine isr6 is called!\n");

}

void isr7()

{

printf("The Interrupt Service Routine isr7 is called!\n");

}

void(\*p\_isr[8])() = { isr0,isr1,isr2,isr3,isr4,isr5,isr6,isr7 };

int main()

{

int N;

scanf("%d", &N);

union ISR\_REG irArray[10];

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

{

scanf("%hu", &irArray[i].all);

}

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

{

printf("%hu:\n", irArray[i].all);

if (irArray[i].bit.bit0)p\_isr[0]();

if (irArray[i].bit.bit1)p\_isr[1]();

if (irArray[i].bit.bit2)p\_isr[2]();

if (irArray[i].bit.bit3)p\_isr[3]();

if (irArray[i].bit.bit4)p\_isr[4]();

if (irArray[i].bit.bit5)p\_isr[5]();

if (irArray[i].bit.bit6)p\_isr[6]();

if (irArray[i].bit.bit7)p\_isr[7]();

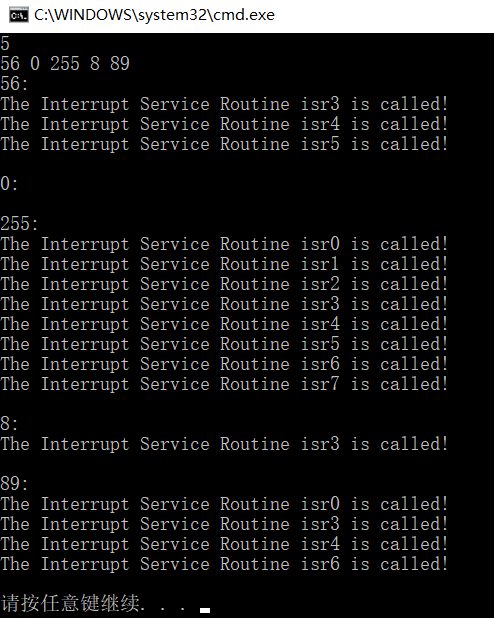
printf("\n");

}

return 0;

}

3）测试



（2）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。用函数编程实现下列功能：

(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. 算法流程：

利用Student \*next;进行访问和判断

2）源程序清单

#include<stdio.h>

#include<string.h>

#include<malloc.h>

#include <stdlib.h>

typedef struct Student

{

char id[15];

char name[15];

float scores[4];

float sum;

float avg;

struct Student \*next;

}Student;

Student\* Create(int n);

void PrintAll(Student \*head);

void AlterData(Student \*head, int n);

void SumAndAvg(Student \*head);

void swapDF(Student \*A, Student \*B);

void sawpPf(Student \*A, Student \*B, Student\* head);

Student \*Sort(Student \*pNode, int length);

int main()

{

Student\* head;

int n;

scanf("%d", &n);

int length = n;

head = Create(n);

PrintAll(head);

scanf("%d", &n);

AlterData(head, n);

SumAndAvg(head);

Sort(head, length);

}

Student\* Create(int n)

{

Student\* head;

Student\* cur\_1;

if ((head = (Student\*)malloc(sizeof(Student))) == NULL)

{

printf("无法开辟空间!");

return head;

}

head->next = NULL;

Student\* pre;

pre = head;

Student\* cur;

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

{

if ((cur = (Student\*)malloc(sizeof(Student))) == NULL)

{

printf("无法开辟空间!");

return head;

}

pre->next = cur;

scanf("%s %s %f %f %f %f", cur->id, cur->name, &cur->scores[0],&cur->scores[1], &cur->scores[2], &cur->scores[3]);

cur->sum = 0;

for (int j = 0;j < 4;j++)

{

cur->sum += cur->scores[j];

}

cur->avg = cur->sum / 4;

cur->next = NULL;

pre = cur;

}

cur\_1 = head->next;

free(head);

return cur\_1;

}

void PrintAll(Student \*head)

{

printf("%-15s%-20s%-10s%-10s%-10s%-10s\n", "ID", "Name", "English", "Math", "Physics", "C");

while (head != NULL)//遍历打印

{

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f", head->id,head->name, head->scores[0], head->scores[1],head->scores[2], head->scores[3]);

printf("\n");

head = head->next;

}

printf("\n");

}

void AlterData(Student \*head,int n)

{

int index;

Student\* cur = head;

char id[15];

char project[15];

float score;

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

{

scanf("%s %s %f", id, project, &score);

if (!strcmp(project, "English"))

{

index = 0;

}

else

{

if (!strcmp(project, "Math"))

{

index = 1;

}

else

{

if (!strcmp(project, "Physics"))

{

index = 2;

}

else

{

if (!strcmp(project, "C"))

{

index = 3;

}

}

}

}

while (strcmp(id, cur->id))

{

cur = cur->next;

}

//操作

cur->scores[index] = score;

cur = head;

}

printf("Alter:\n");

printf("%-15s%-20s%-10s%-10s%-10s%-10s\n", "ID", "Name", "English", "Math", "Physics", "C");

while (head != NULL)//遍历打印

{

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f", head->id, head->name, head->scores[0], head->scores[1], head->scores[2], head->scores[3]);

printf("\n");

head = head->next;

}

printf("\n");

}

void SumAndAvg(Student \*head)

{

printf("SumAndAvg:\n");

printf("%-15s%-20s%-10s%-10s\n", "ID","Name", "SUM", "AVG");

while (head != NULL)//遍历打印

{

head->sum = 0;

for (int j = 0; j < 4; j++)

{

head->sum += head->scores[j];

}

head->avg = head->sum / 4;

printf("%-15s%-20s%-10.2f%-10.2f", head->id, head->name, head->sum,head->avg);

printf("\n");

head = head->next;

}

printf("\n");

}

void swapDF(Student \*A,Student \*B)

{

Student\* T;

T = (Student\*)malloc(sizeof(Student));

strcpy(T->id, A->id);

strcpy(A->id, B->id);

strcpy(B->id, T->id);

strcpy(T->name, A->name);

strcpy(A->name, B->name);

strcpy(B->name, T->name);

for (int i = 0; i < 4; i++)

{

T->scores[i] = A->scores[i];

A->scores[i] = B->scores[i];

B->scores[i] = T->scores[i];

}

T->avg = A->avg;

A->avg = B->avg;

B->avg = T->avg;

free(T);

}

void sawpPf(Student \*A, Student \*B,Student\* head)

{

Student\* temp;

Student\* pre\_A;

Student\* pre\_B;

while (head != NULL)//遍历打印

{

if (head->next == A)

{

pre\_A = head;

}

if (head->next == B)

{

pre\_B = head;

}

head = head->next;

}

temp = B->next;

pre\_A->next = B;

B->next = A->next;

pre\_B->next = A;

A->next = temp;

printf("\n");

}

//链表实现冒泡排序

Student \*Sort(Student \*pNode,int length)

{

if (pNode == NULL)

{

printf("Error!");

return NULL;

}

else

{

Student \*pMove;

pMove = pNode;

for (int i = 0; i < length; i++) //需要进行(n-1)次遍历,控制次数

{

while (pMove->next != NULL)

{

if (pMove->avg > pMove->next->avg)

{

/\*sawpPf(pMove, pMove->next, pNode);\*/

/\*temp = pMove->avg;

pMove->avg = pMove->next->avg;

pMove->next->avg = temp;\*/

swapDF(pMove, pMove->next);

}

pMove = pMove->next;

}

//每次遍历结束，pMove重新移动到链表头部

pMove = pNode;

}

}

printf("Sort:\n");

printf("%-15s%-20s%-10s\n", "ID","Name", "AVG");

Student\* head = pNode;

while (head != NULL)//遍历打印

{

printf("%-15s%-20s%-10.2f", head->id, head->name,head->avg);

printf("\n");

head = head->next;

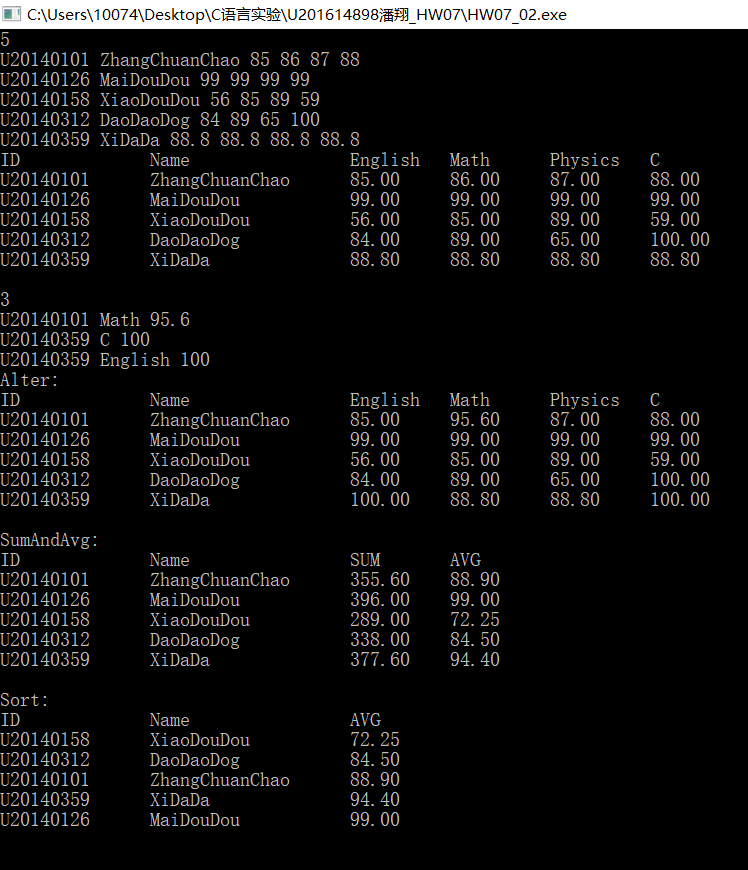
}

printf("\n");

return pNode;

}

3）测试



## 7.3 自设题

**（1）**自设实验题目：十字链表

**（2）**实验目的：熟练链表使用

**（3）**实验程序：

#include<stdio.h>

#include<string.h>

#include<malloc.h>

#include <stdlib.h>

typedef struct Student

{

char id[15];

char name[15];

struct Subjects \*sub\_head;

float sum;

float avg;

struct Student \*next;

}Student;

typedef struct Subjects

{

float score;

struct Subjects \*next;

}Subjects;

Student\* stu\_Create(int n);

Subjects\* sub\_Create(int n);

void PrintAll(Student \*head);

int main()

{

Student\* head;

int n;

scanf("%d", &n);

int length = n;

head = stu\_Create(n);

PrintAll(head);

return 0;

}

Student\* stu\_Create(int n)

{

Student\* head;

Student\* cur\_1;

if ((head = (Student\*)malloc(sizeof(Student))) == NULL)

{

printf("无法开辟空间!");

return head;

}

head->next = NULL;

Student\* pre;

pre = head;

Student\* cur;

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

{

if ((cur = (Student\*)malloc(sizeof(Student))) == NULL)

{

printf("无法开辟空间!");

return head;

}

pre->next = cur;

scanf("%s %s", cur->id, cur->name);

cur->sub\_head=sub\_Create(4);

Subjects \*sub\_cur = cur->sub\_head;

cur->sum = 0;

/\*for (int j = 0;j < 4;j++)

{

cur->sum += sub\_cur->score;

sub\_cur = sub\_cur->next;

}

cur->avg = cur->sum / 4;\*/

cur->next = NULL;

pre = cur;

}

cur\_1 = head->next;

free(head);

return cur\_1;

}

Subjects\* sub\_Create(int n)

{

Subjects\* head;

Subjects\* head\_1;

if ((head = (Subjects\*)malloc(sizeof(Subjects))) == NULL)

{

printf("无法开辟空间!");

return head;

}

head->next = NULL;

Subjects\* pre;

pre = head;

Subjects\* cur;

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

{

if ((cur = (Subjects\*)malloc(sizeof(Subjects))) == NULL)

{

printf("无法开辟空间!");

return head;

}

pre->next = cur;

scanf("%f",&cur->score);

cur->next = NULL;

pre = cur;

}

head\_1 = head->next;

free(head);

return head\_1;

}

void PrintAll(Student \*head)

{

printf("%-15s%-20s%-10s%-10s%-10s%-10s\n", "ID", "Name", "English", "Math", "Physics", "C");

while (head != NULL)//遍历打印

{

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f", head->id, head->name,

head->sub\_head->score,

head->sub\_head->next->score,

head->sub\_head->next->next->score,

head->sub\_head->next->next->next->score);

printf("\n");

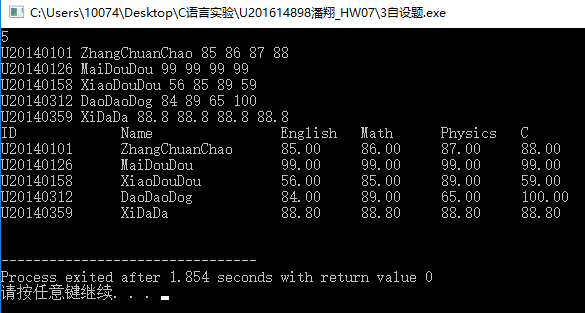
head = head->next;

}

printf("\n");

}

1. 实验用例**：**



1. 实验结论：

可通过十字链表减少内存使用 当存在选修课和课程数目不同的时候

## 7.4 选做题

（1）对编程设计题第（2）题的程序，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。

（2）对选做题第（1）题，进一步写出用交换结点指针域的方法升序排序的函数。

（3）采用双向链表重做编程设计题第（2）题。

**解答：**

**(1)(2)见程序设计**

**(3)双向链表：**

源程序清单:

结构体声明：

typedef struct Student

{

char id[15];

char name[15];

float scores[4];

float sum;

float avg;

struct Student \*next;

struct Student \*pre;

}Student;

交换指针域：

void sawpPf(Student \*A, Student \*B, Student\* head)

{

Student\* temp\_next;

Student\* temp\_pre;

Student\* pre\_A;

Student\* pre\_B;

while (head != NULL)//遍历打印

{

pre\_A = A->pre;

pre\_B = B->pre;

temp\_next = B->next;

temp\_pre = B->pre;

pre\_A->next = B;

B->next = A->next;

pre\_B->next = A;

A->next = temp\_next;

A->pre = temp\_pre;

}

创建链表：

Student\* Create(int n)

{

Student\* head;

Student\* cur\_1;

if ((head = (Student\*)malloc(sizeof(Student))) == NULL)

{

printf("无法开辟空间!");

return head;

}

head->pre = NULL;

head->next = NULL;

Student\* pre;

pre = head;

Student\* cur;

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

{

if ((cur = (Student\*)malloc(sizeof(Student))) == NULL)

{

printf("无法开辟空间!");

return head;

}

cur->pre = pre;

pre->next = cur;

scanf("%s %s %f %f %f %f", cur->id, cur->name, &cur->scores[0], &cur->scores[1], &cur->scores[2], &cur->scores[3]);

cur->sum = 0;

for (int j = 0; j < 4; j++)

{

cur->sum += cur->scores[j];

}

cur->avg = cur->sum / 4;

cur->next = NULL;

pre = cur;

}

cur\_1 = head->next;

free(head);

return cur\_1;

}

## 7.4 实验小结

**（1）**学习使用链表

**（2）**注意指针的赋值和头指针的处理

**（3）**注重细节，培养习惯才能在工程代码中少出错。

# 实验8 指针实验

## 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）熟悉掌握流式文件的读写方法。

## 8.2实验内容及要求

8.2.1. 文件类型的程序验证

下面程序是否存在错误？如果存在，原因是什么？如果存在错误，要求在计算机上对这个例子程序进行调试修改，使之能够正确执行。

设有程序：

#include <stdio.h>

int main()

{

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)后，结果有什么不同？为什么？

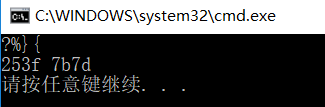
（3）将fprintf(fp2,”%hx%hx”,a,b)改为fprintf(fp2,”%d%d”,a,b)后结果有什么不同。

**解答：**

（1）

wb+二进制文件更新

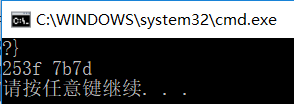
w+ 文本文件更新



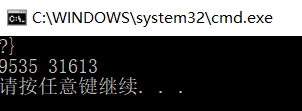
（2）

a为short 两个字节

改为char写入 则低字节写入 高字节舍去



（3）



转换为十进制写入

8.2.2 源程序修改替换

已知将制定文本文件内容在屏幕上显示出来的命令行格式为

type filename

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

**程序代码：**

#include <stdio.h>

#include <stdlib.h>

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

{

char ch;

FILE \*fp;

if(argc!=2){

printf("Argument 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\*/

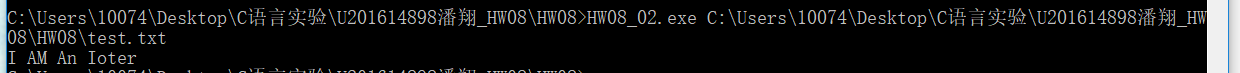
return 0;

}

（2）用输入输出重定向freopen改写上述源程序中的main函数。

**解答：**

（1）

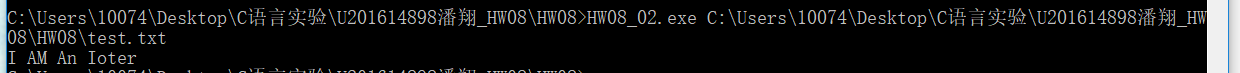


while ((ch = fgetc(fp))!= EOF)

此处需要使用括号来使ch读入

（2）

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



8.2.3 程序设计

编写一个程序，实现以下功能：从键盘输入一行英文句子，将每个单词的首字母换成大写字母，然后输出到一个磁盘文件“test”中保存。

解答

1）算法流程如图8.1所示。



图8-1 编程题1的程序流程图

2）源程序清单

#include <stdio.h>

#include <ctype.h> //用于判断字符类型 字符处理

int main()

{

char s[100];

for (int i = 0; i < 100; ++i)

{

scanf("%c", &s[i]);

if (s[i] == '\n')

{

break;

}

}

s[0] = toupper(s[0]); //将第一个字符变成大写

for (int i = 1; s[i] != '\n'; ++i)

{

if (s[i] == ' ')

{

s[i+1] = toupper(s[i+1]);

i++;

}

}

FILE \*fp;

fp = fopen("test.txt", "w+");

char ch;

for (int i = 0; (ch=s[i]) != '\n'; i++)

{

fputc(ch, fp); //写入文件

putchar(ch); //输出到屏幕

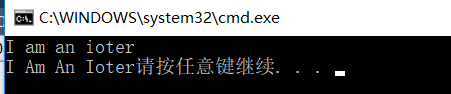
}

fclose(fp);

return 0;

}

3）测试



## 8.3 实验小结

**（1）**使用文件注意指针的初始位置

**（2）**注意文件的关闭

**（3）**不同文件打开方式的掌握