

**课 程 实 验 报 告**

**课程名称： C语言程序设计**

**专业班级：计算机科学与技术（校交）1601班**

**学 号： U201610136**

**姓 名： 朱晓光**

**指导教师： 甘早斌**

**报告日期： 2017.6.7**

**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 4](#_Toc484598829)

[1.1 实验目的 4](#_Toc484598830)

[1.2 实验内容 4](#_Toc484598831)

[1.2.1 源程序改错题 4](#_Toc484598832)

[1.2.2 源程序修改替换 5](#_Toc484598833)

[1.2.3 编程设计 5](#_Toc484598834)

[1.3 自设题 8](#_Toc484598835)

[1.3 实验小结 9](#_Toc484598836)

[2 流程控制实验 10](#_Toc484598837)

[2.1 实验目的 10](#_Toc484598838)

[2.2 实验内容 10](#_Toc484598839)

[2.2.1 源程序改错题 10](#_Toc484598840)

[2.2.2 源程序修改替换题 10](#_Toc484598841)

[2.2.3 编程设计题 11](#_Toc484598842)

[2.3 自设题 18](#_Toc484598843)

[2.4 实验小结 20](#_Toc484598844)

[3 函数与程序结构实验 21](#_Toc484598845)

[3.1 实验目的 21](#_Toc484598846)

[3.2 实验内容 21](#_Toc484598847)

[3.2.1 源程序改错题 21](#_Toc484598848)

[3.2.2 源程序修改替换题 21](#_Toc484598849)

[3.2.3 跟踪调试题 22](#_Toc484598850)

[3.2.4 编程设计题 23](#_Toc484598851)

[3.2.5 选做题 26](#_Toc484598852)

[3.3 自设题 26](#_Toc484598853)

[3.4 实验小结 27](#_Toc484598854)

[4 编译预处理实验 28](#_Toc484598855)

[4.1 实验目的 28](#_Toc484598856)

[4.2 实验题目及要求 28](#_Toc484598857)

[4.2.1 源程序改错题 28](#_Toc484598858)

[4.2.2 源程序修改替换题 28](#_Toc484598859)

[4.2.3 跟踪调试题 29](#_Toc484598860)

[4.2.4 编程设计题 30](#_Toc484598861)

[4.3 自设题 33](#_Toc484598862)

[4.4 实验小结 35](#_Toc484598863)

[5 数组实验 36](#_Toc484598864)

[5.1 实验目的 36](#_Toc484598865)

[5.2 实验内容及要求 36](#_Toc484598866)

[5.2.1 源程序改错 36](#_Toc484598867)

[5.2.2 源程序完善、修改、替换 36](#_Toc484598868)

[5.2.3 跟踪调试源程序 38](#_Toc484598869)

[5.2.4 程序设计 39](#_Toc484598870)

[5.2.5 选做题 44](#_Toc484598871)

[5.3 自设题 47](#_Toc484598872)

[5.4 实验小结 48](#_Toc484598873)

[6 指针实验 49](#_Toc484598874)

[6.1 实验目的 49](#_Toc484598875)

[6.2 实验题目及要求 49](#_Toc484598876)

[6.2.1 源程序改错题 49](#_Toc484598877)

[6.2.2 源程序完善、修改、替换题 49](#_Toc484598878)

[6.2.3 跟踪调试题 51](#_Toc484598879)

[6.2.4 编程设计题 52](#_Toc484598880)

[6.2.5 选做题 60](#_Toc484598881)

[6.3 自设题 63](#_Toc484598882)

[6.4 实验小结 64](#_Toc484598883)

[7 结构与联合实验 65](#_Toc484598884)

[7.1 实验目的 65](#_Toc484598885)

[7.2 实验题目及要求 65](#_Toc484598886)

[7.2.1 表达式求值的程序验证题 65](#_Toc484598887)

[7.2.2 源程序修改替换题 65](#_Toc484598888)

[7.2.3 编程设计题 67](#_Toc484598889)

[7.2.4 选做题 78](#_Toc484598890)

[7.3 实验小结 79](#_Toc484598891)

[8 文件实验 80](#_Toc484598892)

[8.1 实验目的 80](#_Toc484598893)

[8.2 实验题目及要求 80](#_Toc484598894)

[8.2.1 文件类型的程序验证题 80](#_Toc484598895)

[8.2.2 源程序修改替换题 81](#_Toc484598896)

[8.2.3 编程设计题 82](#_Toc484598897)

[8.3 实验小结 83](#_Toc484598898)

附

所有题目完整源码地址

<https://github.com/smdsbz/homework/tree/master/C>

说明

源码文件名开头数字为作业系统中作业号

报告中涉及的题目已归档至对应的文件夹中，文件夹名最后两位数字为实验序号

**该仓库下所有项目遵循MIT许可！**

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 源程序改错题

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。

**解答**

#include<stdio.h>

#define PI 3.14159; // #define PI 3.14159

voidmain( void ) // void main(void)

{

int f ;

short p, k ;

double c , r , s ;

/\* for task 1 \*/

printf(“Input Fahrenheit:” ) ;

scanf(“%d”, f ) ; // scanf(“%d”, &f);

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

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

/\* for task 2 \*/

printf("input the radius r:");

scanf("%f", &r); // scanf(“%lf”, &r);

s = PI \* r \* r;

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

// printf(“\nThe acreage is %.2f\n\n”, s);

/\* for task 3 \*/

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

scanf("%x %x", &k, &p ); // scanf(“%hx %hx”, &k, &p);

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

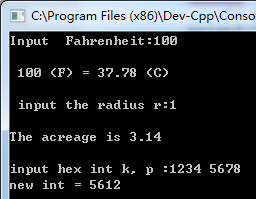
// short newint = (p&0xff00)|(k&0xff00)>>8;

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

// printf("new int = %hx\n\n",newint);

}

**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，请改用不用第三个变量的交换法实现。

**解答**

#include<stdio.h>

void main( )

{

int a, b, t;

printf(“Input two integers:”);

scanf(“%d %d”,&a,&b);

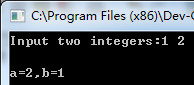
t=a, a=b, b=t;

// b += a, a = b - a, b -= a;

printf(“\na=%d,b=%d”,a,b);

}

**替换后运行结果：**

****

1.2.3 编程设计

（1）编写一个程序，输入无符号短整型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。

1）算法流程：

1. 输入数据组数；

2. 对每一组数据：

1. 输入x, m, m；

2. 根据m, n制作mask；

3. 输出x, mask与运算后结果；

4. 下一组数据；

**源程序清单：**

#include "stdio.h"

int main(void) {

int total\_count; // 数据总组数

unsigned short x, m, n; // 输入数据单元

unsigned short mask = 0; // mask单元

unsigned short i; // 循环控制变量

// 输入数据组数 - 并限制其不能小于0

scanf("%d", &total\_count);

if (total\_count < 0) { puts("negative input!"); return 0; }

for (; total\_count > 0; --total\_count) {

// 一行输入一行输出

scanf("%hu %hu %hu", &x, &m, &n);

// 限制m, n范围

if (m>15) { puts("m out of range!"); ++total\_count; continue; }

if (n>16-m) { puts("n out of range!"); ++total\_count; continue; }

// 不需要判断它们是否小于0，因为它们都是无符号的

// 制作mask

mask = 1U << m; // mask起点

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

mask |= mask << 1; // 一直拖到终点

} // endfor

// 一行输入一行输出

printf("%hu\n", (x&mask)<<(16-m-n));

} // endif

return 0;

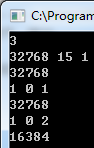
}

**测试：**

（a）测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例 | 程序输入 | | | 理论结果 |
| x | m | n |
| 用例1 | 32768 | 15 | 1 | 32768 |
| 用例2 | 1 | 0 | 1 | 32768 |
| 用例3 | 1 | 0 | 2 | 16384 |

（b）运行结果：



（2）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型表示。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。

1）算法流程：

1. 制作静态的mask；

2. 输入数据组数；

3. 对每一组数据：

1. 输入一个IP源数据；

2. 将IP数据要输出的部分移到mask的有效位；

3. 输出结果；

4. 下一组数据；

**源程序清单：**

#include "stdio.h"

#define LEN 32 // 无符号长整型位数（made static）

#define MASK 255UL // 即 00000000 00000000 00000000 11111111

int main(void) {

int total\_count; // 数据总组数

unsigned long current; // 存储当前输入数据

char i; // 循环控制变量

scanf("%d", &total\_count); // 限制 - 输入的数据组数必须大于0

if (total\_count < 0) { puts("negative count!"); return 0; }

for (; total\_count > 0; --total\_count) {

// 每次循环输入一次 - 一行输入一行输出

scanf("%ld", &current);

// 笨办法 - 从左至右依次输出

printf("%ld.", (current&MASK));

// 输出一次，加工一次数据 - mask不动

current >>= 8;

printf("%ld.", (current&MASK));

current >>= 8;

printf("%ld.", (current&MASK));

current >>= 8;

printf("%ld\n", (current&MASK));

}

return 0;

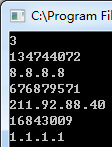
}

**测试：**

（a）测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| IP源数据 |
| 用例1 | 134744072 | 8.8.8.8 |
| 用例2 | 676879571 | 211.92.88.40 |
| 用例3 | 16843009 | 1.1.1.1 |

（b）运行结果



1.3 自设题

（1）自设实验题目，实型数的浮点表示误差的简单验证。

（2）实验目的：通过设计实验程序，理解实数的浮点数表示上的误差，体会浮点数不宜用于复杂计算场合（例如金融计算、科学计算等）的含义。

**源程序清单**

#include "stdio.h"

void main(void) {

puts("Here's a third: ");

float pocket = 1./3;

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

puts("\nnow times it with 7 and then subtracts it with 2 like craaaaazy!");

puts("(Note: in analog env, a third will still be a third)");

unsigned long times\_to\_repeat = -1;

printf("\nHow many times should I repeat this operation? ");

scanf("%ld", &times\_to\_repeat);

// start!

for (; times\_to\_repeat != 0; --times\_to\_repeat) {

pocket \*= 7;

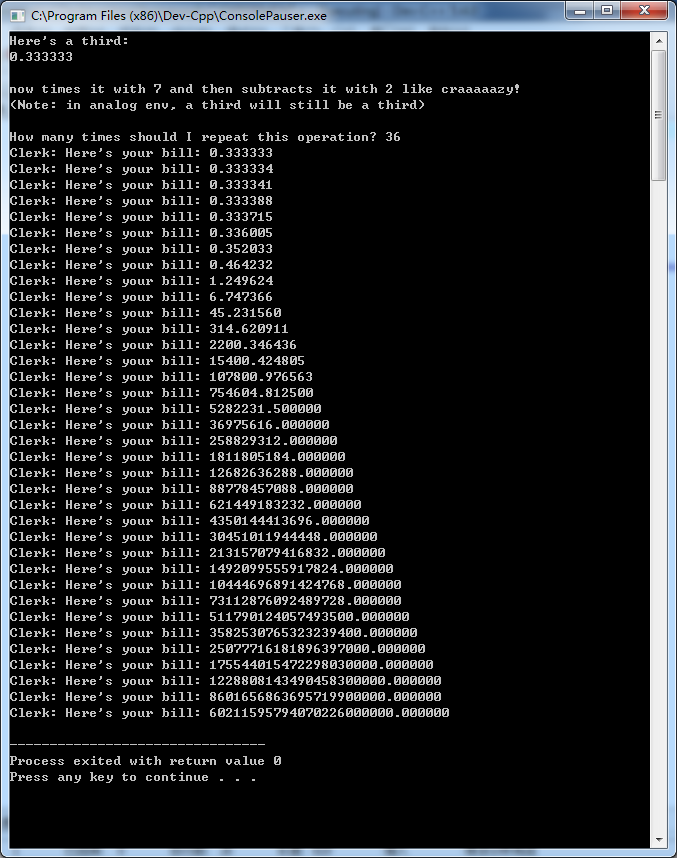
pocket -= 2;

printf("Clerk: Here's your bill: %f\n", pocket);

}

}

**运行结果截图**

****

**实验结论：**

浮点数不适合复杂计算场合。

1.3 实验小结

通过本次实验，我切身体会了从无到有、完整地编写C语言程序的过程。发现理论与实践还是有一定差距的，以后还需要多加练习。

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 源程序改错题

下面是计算s=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。例如，8！=40320。

#include <stdio.h>

void main(void)

{

int i,n,s=1;

printf("Please enter n:");

scanf("%d",n); // scanf(“%d”, &n);

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

s=s\*i;

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

}

2.2.2 源程序修改替换题

（1）修改第1题，分别用while和do-while语句替换for语句。

**源代码：**

1）使用while：

#include <stdio.h>

void main(void)

{

int i,n,s=1;

printf("Please enter n:");

scanf("%d", &n);

/\* using while \*/

i = 1;

while (i <= n) {

s \*= i;

++i;

}

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

}

2）使用do-while：

#include <stdio.h>

void main(void)

{

int i,n,s=1;

printf("Please enter n:");

scanf("%d", &n);

i = 1;

do {

s \*= i;

} while (++i, i <= n);

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

}

**运行结果：**

1）使用while：



2）使用do-while：



（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。

**源代码：**

#include <stdio.h>

void main(void)

{

int i,n,s=1;

printf("Please enter s:");

scanf("%d", &n);

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

s=s\*i;

printf("n=%d\n",i-1);

}

**运行结果：**



2.2.3 编程设计题

（1）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。

**算法流程：**

1. 输入形状大小size；

2. 输出第i行：

1. 使用组合公式计算第i行第j个数，并输出；

输出下一个数，直至该行结束；

2. 输出下一行，直至整个图形打印完毕；

3. 结束；

**源程序清单：**

#include "stdio.h"

/\* 函数原型：计算组合数 \*/

unsigned short combination(unsigned char, unsigned char);

void main(void) {

unsigned char size;

unsigned char i, j; // loop var

while (

scanf("%hhu", &size),

getchar(),

size != 0 && size <= 12

) {

// 第i行

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

// 第一个元素：1

printf("%\*d", (size-i)\*2+1, 1);

// 第一个元素之后的数字：用组合数公式计算

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

printf("%4hu", combination(i-1, j));

}

// 该行结束

putchar('\n');

}

// 两个图形之前空一行

putchar('\n');

}

}

unsigned short combination(unsigned char i, unsigned char j) {

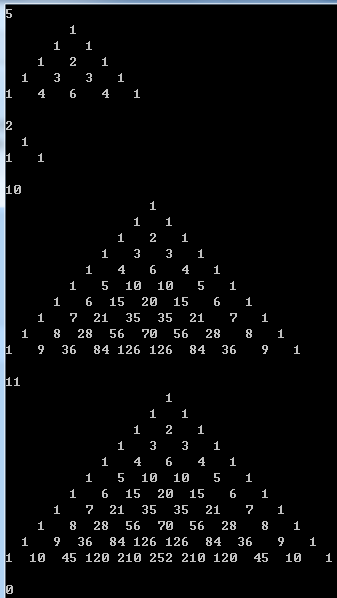
// printf("combination(%hhu,%hhu)\n", i, j);

if (j == 0) { return 1; }

else { return combination(i, j-1) \* (i-j+1) / j; }

}

**运行结果截图：**



（2）编写一个程序，将用户输入的任意正整数逆转，例如，输入1234，输出4321。

**算法流程：**

1. 输入正整数i，注册结果暂存器o；

2. 将i的低位依次存到o的高位上：

1. o×10把之前的结果推向高位，并空出最低位；

2. i%10得到当前最低位；

3. o的最低位存入刚才的计算所得的余数；

4. i÷10丢弃最低位，并进入下一轮循环；

3. 输出处理完成后的结果o；

4. 输入下一个数或退出；

**源程序清单：**

#include "stdio.h"

void main(void) {

int i, o;

while (

scanf("%d", &i),

getchar(),

i != 0

) {

// 每向o传递一位，i就消去一位

for (o = 0; i != 0; i/=10) {

// 将i的个位传递给o的个位

o = (o \* 10) + (i % 10);

// o之前的数字向高位挪

}

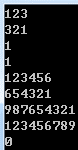
// 输出

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

}

}

**运行结果截图：**



（3）个人所得税。

**算法流程：**

1. 输入收入income；

2. 按照《规定》计算个人所得税：

if：从高到低依次计算每一个等级的税；

switch：对每一种情况单独进行判断；

3. 输入个人所得税；

**源程序清单：**

#include "stdio.h"

/\* 函数原型：使用if计算个人所得税 \*/

double using\_if(double);

/\* 函数原型：使用switch计算个人所得税 \*/

double using\_switch(double);

void main(void) {

double income;

while (

scanf("%lf", &income),

getchar(),

income != 0

) {

// 输出对应的个人所得税

printf("%lf\n", using\_switch(income));

}

}

double using\_if(double income) {

double tax = 0;

if (income > 5000) {

tax += (income-5000)\*0.25L;

income = 5000;

}

if (income > 4000) {

tax += (income-4000)\*0.20L;

income = 4000;

}

if (income > 3000) {

tax += (income-3000)\*0.15L;

income = 3000;

}

if (income > 2000) {

tax += (income-2000)\*0.10L;

income = 2000;

}

if (income > 1000) {

tax += (income-1000)\*0.05L;

income = 1000;

}

else {

tax = 0;

}

return tax;

}

double using\_switch(double income) {

double tax = 0;

switch ((int)(income/1000)) {

case 4: {

tax = (income-4000)\*0.20L + 1000\*0.15L + 1000\*0.10L + 1000\*0.05L;

break;

}

case 3: {

tax = (income-3000)\*0.15L + 1000\*0.10L + 1000\*0.05L;

break;

}

case 2: {

tax = (income-2000)\*0.10L + 1000\*0.05L;

break;

}

case 1: {

tax = (income-1000)\*0.05L;

break;

}

case 0: {

tax = 0;

break;

}

default: {

tax = (income-5000)\*0.25L + 1000\*0.20L + 1000\*0.15L + 1000\*0.10L + 1000\*0.05L;

break;

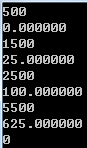
}

}

return tax;

}

**运行结果截图：**



（4）空格处理。

**算法流程：**

1. 注册输入字符串暂存区；

2. 从头开始处理字符：

前一个字符不为空格 -> 输出当前字符；

前一个字符为空格：

当前字符不为空格 -> 输出；

当前字符为空格 -> 不输出；

3. 下一组或结束；

**源程序清单：**

#include "stdio.h"

void main(void) {

unsigned char N; // total count

scanf("%hhu", &N);

getchar();

char str[100] = {'\0'}; // stores user input - 99 chars + '\0'

unsigned char fp = 0; // pointer

for (; N != 0; --N) {

// 用重写的 gets() 读入一行字符串

for (

fp = 0;

((str[fp]=getchar()) != '\n') && fp<99;

++fp

) {

continue;

}

// 字符串以空字符结尾

str[++fp] = '\0';

// 若用户什么也没输入就直接回车，则结束本次循环

if (str[0] != '\0') {

// 第一个字符总是要输出的

putchar(str[0]);

// 遍历整个字符串

for (fp = 1; str[fp] != '\0'; ++fp) {

// 如果前一个字符不是空格 -> 输出当前字符

if (str[fp-1] != ' ') {

putchar(str[fp]);

}

// 如果前一个字符是空格：

else {

// 1. 当前字符不是空格 -> 输出

if (str[fp] != ' ') {

putchar(str[fp]);

}

// 2. 当前字符是空格 -> 不输出

else {

continue;

}

}

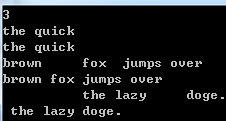
} // end for

} // end if - end of output loop

} // end for - end of main loop

}

**运行结果截图：**



（5）求方程的近似根。

**算法流程：**

按照书上所给的方法，写好原函数、导函数、迭代方法，然后开始计算。

**源程序清单：**

#include "stdio.h"

// 精度

#define EPSION 10e-6

/\* 原函数 \*/

#define original(x) ( 3\*(x)\*(x)\*(x) - 4\*(x)\*(x) - 5\*(x) + 13 )

/\* 原函数的导数 \*/

#define derivative(x) ( 9\*(x)\*(x) - 8\*(x) - 5 )

/\* 计算 X(k+1) \*/

#define yield(x) ( x - original((x)) / derivative((x)) )

double myAbs(double x) {

if (x > 0) { return x; }

else { return -x; }

}

void main(void) {

double a = -1.5; // start point

double previous, current = 0; // storage

previous = a;

for (; myAbs(previous - current) > EPSION;) {

// 当前数作为下一轮的比较值

previous = current;

// 用公式计算下一轮的比较值

current = yield(previous);

}

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

}

**运行结果截图：**



2.3 自设题

选择排序。

**算法流程：**

1. 输入；

2. 从第一个字符开始判断：

在这个元素之后是否有更小的？

有 → 交换；无 → 不交换；

判断下一个；

**源程序清单：**

#include "stdio.h"

// 需要排序的数字个数在这里修改

#define SIZE 5

void print\_array(int \* storage) {

unsigned char i = 0;

for (i; i < SIZE; ++i) {

printf("%d ", \*(storage+i));

}

putchar('\n');

}

void selectSort(int \* array) {

unsigned char i, j;

unsigned char min\_index;

for (i = 0; i < SIZE; ++i) {

min\_index = i;

// 获得剩余未排序序列中最小数的编号

for (j = i; j < SIZE; ++j) {

min\_index = (array[j] < array[min\_index]) ? j : min\_index;

}

// 如果之后的序列中有比当前元素更小的，则交换

if (min\_index > i) {

array[i] += array[min\_index];

array[min\_index] = array[i] - array[min\_index];

array[i] -= array[min\_index];

}

} // end of main loop

}

void main(void) {

int storage[SIZE] = {0};

unsigned char i; // loop var

// 输入

for (i = 0; i < SIZE; ++i) {

scanf("%d", storage+i);

getchar();

}

// 排序

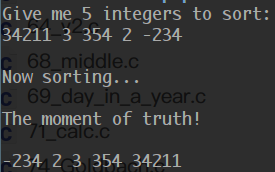
selectSort(storage);

// 输出

print\_array(storage);

}

**运行结果截图：**



2.4 实验小结

这次上机实验锻炼了我快速思考，并编程实现的能力。因之前对实验要求了解不足，漏掉了一题（编程题第五题）没写，然而我在5分钟之内就按照题目要求完成了任务，提交至作业系统并通过。以后还是要多加练习，多攒一些编程的经（tao）验（lu），提高自己编程的能力和程序的质量！

实验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 源程序改错题

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。

#include "stdio.h"

**// long sum\_fac(int);**

void main(void)

{

int k;

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

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

}

long sum\_fac(int n)

{

long s=0;

int i;

long fac;

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

fac\*=i;

s+=fac;

**/\***

**\* for(i=1,fac=1;i<=n;i++) {**

**\* fac\*=i;**

**\* s+=fac;**

**\* }**

**\*/**

return s;

}

3.2.2 源程序修改替换题

（1）修改第1题中sum\_fac函数，使其计算量最小。

**源程序：**

long sum\_fac(int n) {

static sum;

static tmp;

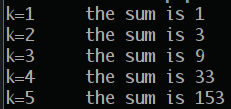
tmp \*= n;

sum += tmp;

return sum;

}

**运行结果截图：**



（2）修改第1题中sum\_fac函数，计算。

**源程序：**

#include "stdio.h"

double sum\_fac(int);

void main(void) {

int k;

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

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

}

}

double sum\_fac(int n) {

double sum = 0;

int i;

long fac = 1;

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

fac \*= i;

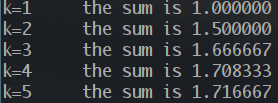
sum += (1./fac);

}

return sum;

}

**运行结果截图：**



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=&sum;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）。

#include “stdio.h”

long fabonacci(int);

void main(void)

{

int i,k;

long sum=0,\*p=&sum;

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？

**解答：**

（1）p = (long int \*) 0x28ff10; i = 2

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

（3）p = (long int \*) 0x28ff10; i = 2; sum = 0; n = 1

（4）2（上次遗留的值） -> 3（第一层） -> 2 -> 1 -> 3（回到第一层）

3.2.4 编程设计题

（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。

**递归算法思路：**

1. 定义全局变量result
2. 传入两个参数a, b
3. 使result为a, b中较小的那个数
4. 判断：

result是公约数：返回result

result不是公约数：result-1，再次调用判断函数

（这里，参与递归过程的函数只判断result是否是a, b的公约数）

**源程序清单：**

#include "stdio.h"

unsigned int result;

unsigned int commonDivisor(unsigned, unsigned);

void main(void) {

unsigned int a, b;

while (

scanf("%u %u", &a, &b),

getchar(),

a!=0 && b!=0

) {

result = 0;

printf("%u\n", commonDivisor(a, b));

}

}

unsigned int commonDivisor(unsigned a, unsigned b) {

if (a > b) {

b += a;

a = b - a;

b -= a;

}

if ( result == 0 ) { result = a; }

if ( !(b%result) && !(a%result) ) { return result; }

else { --result; return commonDivisor(a, b); }

}

**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

**算法流程：**

1. 输入范围
2. 对范围中每一个偶数current（程序中直接使用begin）

将循环变量i初始化为最小质数——2

判断i与current-i是否均为质数：

是：哥德巴赫猜想对偶数current正确

否：i+1，继续判断

**源程序清单：**

#include "stdio.h"

unsigned char isPrime(unsigned int);

void main(void) {

puts("GOLDBACH'S CONJECTURE:");

puts("Every even number n>=4 is the sum of two primes.");

unsigned int begin, end;

unsigned int i;

while (

scanf("%u %u", &begin, &end),

getchar(),

(begin!=0) && (begin>=6) && (end>=begin)

) {

if ( begin % 2 ) { begin = begin / 2 \* 2 + 2; }

for (; begin <= end; begin+=2) {

for (i = 2; i < begin; ++i) {

if ( isPrime(i) && isPrime(begin-i) ) {

printf("%u=%u+%u\n", begin, i, begin-i);

break;

}

}

}

putchar('\n');

}

}

unsigned char isPrime(unsigned x) {

unsigned i = 2;

unsigned max = x/2 + 1;

for (; i <= max; ++i) {

if ( !(x%i) ) { break; }

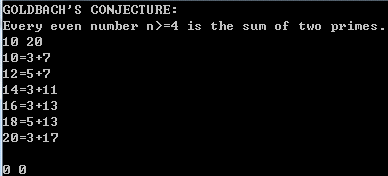
}

if (i==max+1) { return 1; }

else { return 0; }

}

**运行结果截图：**



3.2.5 选做题

设file1.c如下：

#include <stdio.h>

#include "file2.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);

}

file2.h如下：

#include <stdio.h>

extern int x,y;

extern char ch;

void func1(void);

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。

**运行结果截图：**



3.3 自设题

探索数组名与各元素地址之间的关系，并探索Linux下项目的编译和链接。

**源程序清单：**

**/\* first.c \*/**

#include "stdio.h"

#include "second.h"

int \* copy(int \*);

void main(void) {

int array[5] = {1, 2, 3, 4, 5};

printf("%p -> %d\n", &array[2], array[2]);

printf("%p -> %d\n",&copy(&array[1])[2], copy(&array[1])[2]);

}

**/\* second.h \*/**

int \* copy(int \*);

**/\* second.c \*/**

#include "stdio.h"

int \* copy(int \* name) {

return name;

}

**/\* makefile \*/**

main: first.c second.h second.c

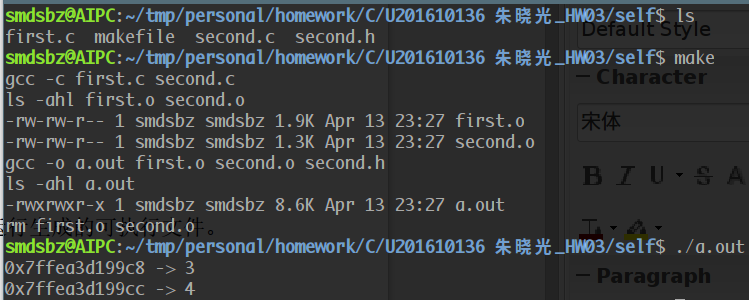
gcc -c first.c second.c

ls -ahl first.o second.o

gcc -o a.out first.o second.o second.h

ls -ahl a.out

rm first.o second.o

**运行结果截图：**

3.4 实验小结

通过本次实验，我学习了多文件编译、链接的基本方法。期间还发现了原题目中的一些错误，加深了对头文件作用的理解与对常见错误的认识。

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 实验题目及要求

4.2.1 源程序改错题

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。

**源程序修改**

#include "stdio.h"

#define SUM a+b // (a+b)

#define DIF a-b // (a-b)

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

void main // void main(void)

{

int b, t; // int a, b, t;

printf("Input two integers a, b:");

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

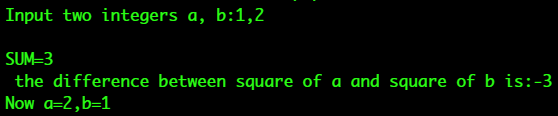
printf("\nSUM=%d\n the difference between square of a and square of b is:%d",SUM, SUM\*DIF);

SWAP(a,b);

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

}

**运行结果截图**



4.2.2 源程序修改替换题

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。

**源程序修改**

include <stdio.h>

int max(int, int, int);

float sum(float, float);

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;

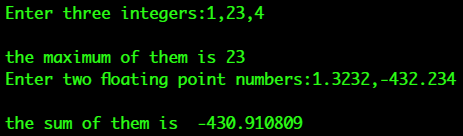
}

**源程序替换**

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

#define sum(d, e) ( d + e )

**运行结果截图**



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integer\_fraction时watch窗口中s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。

**解答**

1. 在文件首加上如下语句即可：

#include <stdio.h>

#include <assert.h>

int integer\_fraction(float);

1. 当输入为2时：

进入函数integer\_fraction()时watch窗口中s为12.5663605

返回main时，watch窗口中i为12

1. 运行时并未出现assertion failed错误。。。。

**watch窗口截图（返回main时）：**



4.2.4 编程设计题

（1）三角形的面积是，其中，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。

**算法流程**

按照题目要求定义两个宏：一个求s，另一个求area（见程序开头3、4行）；随后即可以像调用函数一样使用。

**源程序清单**

#include <stdio.h>

#include <math.h>

**#define S(a, b, c) ( (a)+(b)+(c) ) >> 1**

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

void main(void) {

unsigned a, b, c;

while (

scanf("%u %u %u", &a, &b, &c) == 3

) {

getchar();

printf(

"%u %lf\n",

**S(a, b, c),**

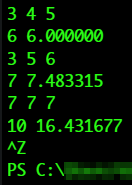
**AREA(S(a, b, c), a, b, c)**

);

}

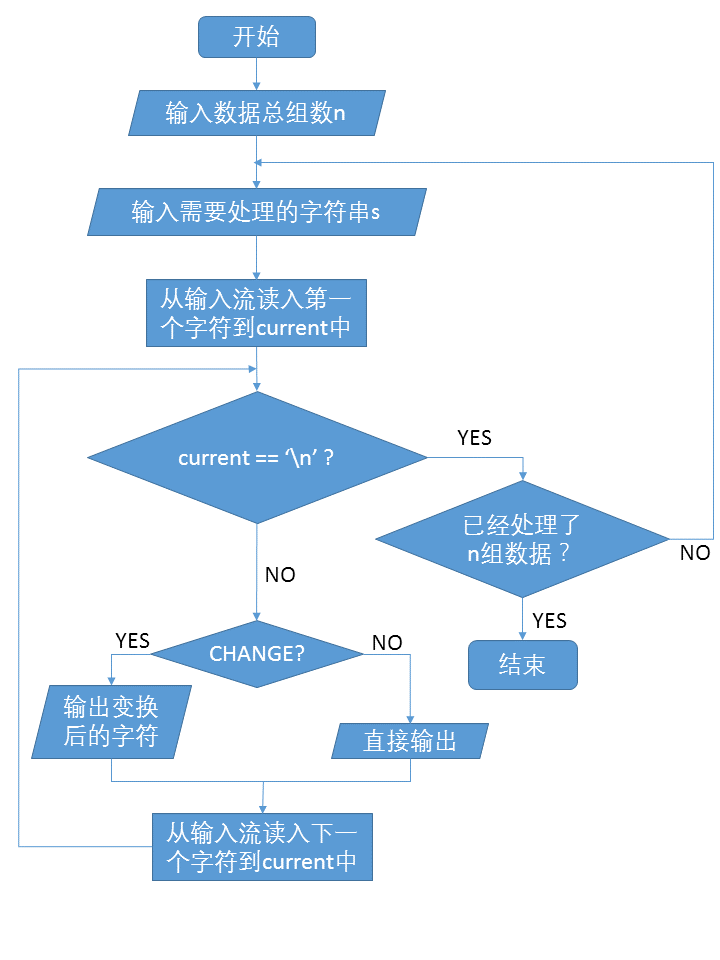
}

**运行结果截图**



（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。

**算法流程**



**源程序清单**

#include <stdio.h>

// CTest2 OJ ver.

// #define CHANGE(c) ( (c) % 2 )

// report ver. - 1

#define CHANGE(c) 1

// report ver. – 0

#define CHANGE(c) 0

char alterCase(char);

/\* alterCase:

\* lower case --> upper case

\* upper case --> lower case

\* non-alphabetic --> original

\*/

void main(void) {

unsigned N;

// OJ ver. required total data count

scanf("%u", &N); getchar();

char current;

unsigned short change\_flag;

for (; N > 0; --N) {

// get flag from first char

current = getchar();

change\_flag = CHANGE(current);

// dealing with every char

for (; current != '\n'; current = getchar()) {

switch (change\_flag) {

case 0: {

putchar(current);

break;

}

case 1: {

putchar(alterCase(current));

break;

}

default: {

puts("This message should not appear!");

return;

}

} // end of switch

}

// output the final \n

putchar('\n');

} // end of main loop

}

char alterCase(char c) {

if (c<='Z' && c>='A') {

return c - 'A' + 'a';

}

else if (c<='z' && c>='a') {

return c - 'a' + 'A';

}

else {

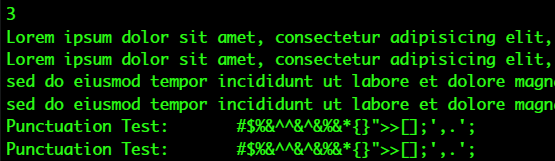
return c;

}

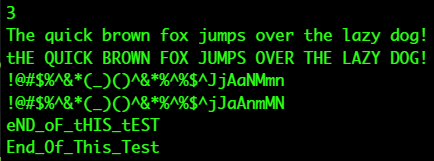
}

**运行结果截图**

CHANGE = 0：



CHANGE = 1：



4.3 自设题

编写一个小程序，加强对空宏定义与assert宏使用的理解。

**算法流程**

用NDEBUG宏控制调试环境与发布环境的切换

**源程序清单**

#include <stdio.h>

// **TODO:** comment the line below to start debugging

#define NDEBUG

#include <assert.h>

unsigned char RUNTIME\_STATUS;

void main(void) {

// published release

#ifdef NDEBUG

puts("Logged in as user-group!\n");

#else

puts("Welcome back, hacker!\n");

#endif

while (1) {

printf("Enter RUNTIME\_STATUS (1 for good, 0 for bad, otherwise generates UB): ");

scanf("%hhu", &RUNTIME\_STATUS); getchar();

// alpha verson

#if !(defined(NDEBUG))

puts("\n============================================================");

printf("[DEBUG\_LOG] RUNTIME\_STATUS = %hhu\n", RUNTIME\_STATUS);

puts("============================================================\n");

#endif

switch(RUNTIME\_STATUS) {

case 1: {

puts("Rolling al' well!\n");

break;

}

case 0: {

puts("Somethings gone wrong...\n");

break;

}

default: {

// will show debug log

assert(RUNTIME\_STATUS==1 || RUNTIME\_STATUS==0);

// will not show debug log

puts("Sorry to inform you that the programme has crashed...");

return;

// the following will not exec

puts("Aftermath will be avoided! So this will not show!");

}

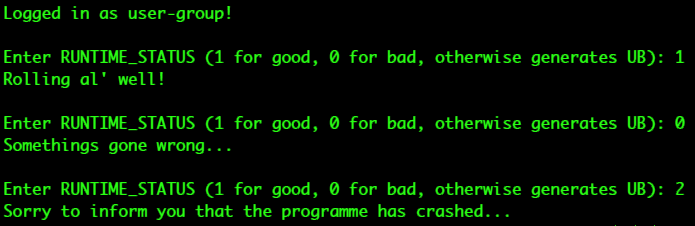
}

}

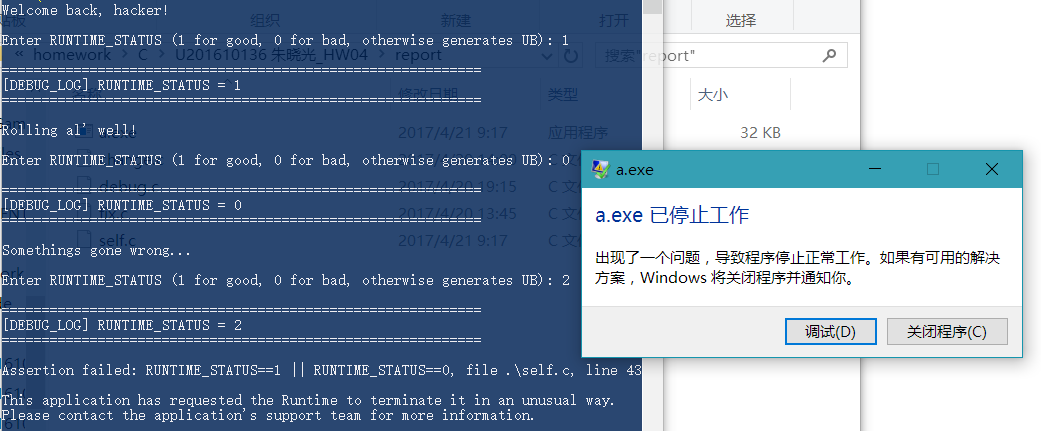
}

**运行结果截图**

发布版本：



测试版本：



4.4 实验小结

通过这次实验，我接触了宏定义与assert宏，认识到了其在调试中的重要意义，并在自设题中进行了更为深入的实践探索。发现assert宏对使用“文本编辑器+命令行+gcc”开发环境的程序员十分友好，以后会在实际开发过程中运用起来。

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 实验内容及要求

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); **// sort(a, 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 }

**运行结果截图**



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;

}

**运行结果截图**



(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。因此，请采用做标记的办法修改（1）中的程序，并使修改后的程序与（1）中的程序具有相同的功能。

**源程序**

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

int a[M];

int i, j, k; /\* 这里j为报数，k用来给标记计数，i用来做遍历变量 \*/

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

a[i] = i + 1;

}

i = 0, j = 1;

while (k != M) {

if (a[i]) { /\* a[i]非零 -> 仍在圈里 \*/

if (j == N) { /\* 数到3 \*/

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

j = 1; /\* j重置 \*/

a[i] = 0; /\* 做标记 \*/

++k; /\* 计数器++ \*/

}

else {

++j;

}

}

/\* 循环变量处理 - 不管当前元素在不在圈内，指针都要往前走 \*/

if (++i == M) { i = 0; }

}

putchar('\n');

return 0;

}

**运行结果截图（同上）**

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？

当光条落在for语句所在行时，i为23；

当光条落在strncat函数块结束标记所在行时，s为字符串a首地址（字符串未改变），t为字符串b首地址。

截图：





（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。

在遍历s的时候，i走过了’\0’，所以’\0’仍保留在串中间。

**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++]) ;

**// --i;**

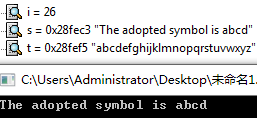
for(j = 0; j < n && t[j];)

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

s[i] = '\0';

}

**运行结果截图**



5.2.4 程序设计

编写并上机调试运行能实现以下功能的程序。

（1）编写一个程序,从键盘读取数据，对一个3×4矩阵进行赋值，求其转置矩阵，然后输出原矩阵和转置矩阵。

**算法流程**

1. entrMatrix()输入矩阵
2. reprOriginal()打印原矩阵
3. reprTransposed()打印转置后矩阵

（其中，reprOriginal()在打印时先循环第二个下标，reprTransposed()则先循环第一个下标）

**源程序**

#include <stdio.h>

#define ROW 3

#define COL 4

int matrix[ROW][COL] = {0};

void entrMatrix(int [ROW][COL]);

void reprOriginal(int [ROW][COL]);

void reprTransposed(int [ROW][COL]);

void main(void) {

entrMatrix(matrix);

reprOriginal(matrix);

putchar('\n');

reprTransposed(matrix);

return;

}

void entrMatrix(int m[ROW][COL]) {

int i, j;

for (j = 0; j < ROW; ++j) {

for (i = 0; i < COL; ++i) {

scanf("%d", \*(m + j\*COL + i));

} // end of ROW

} // end of all

return;

}

void reprOriginal(int m[ROW][COL]) {

int i, j;

for (j = 0; j < ROW; ++j) {

for (i = 0; i < COL; ++i) {

printf("%5d", \*\*(m + j\*COL + i));

} // end of ROW

putchar('\n');

} // end of all

return;

}

void reprTransposed(int m[ROW][COL]) {

int i, j;

for (i = 0; i < COL; ++i) {

for (j = 0; j < ROW; ++j) {

printf("%5d", \*\*(m + j\*COL + i));

} // end of COL

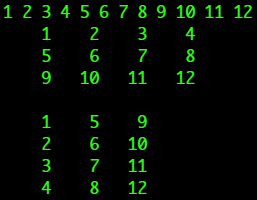
putchar('\n');

} // end of all

return;

}

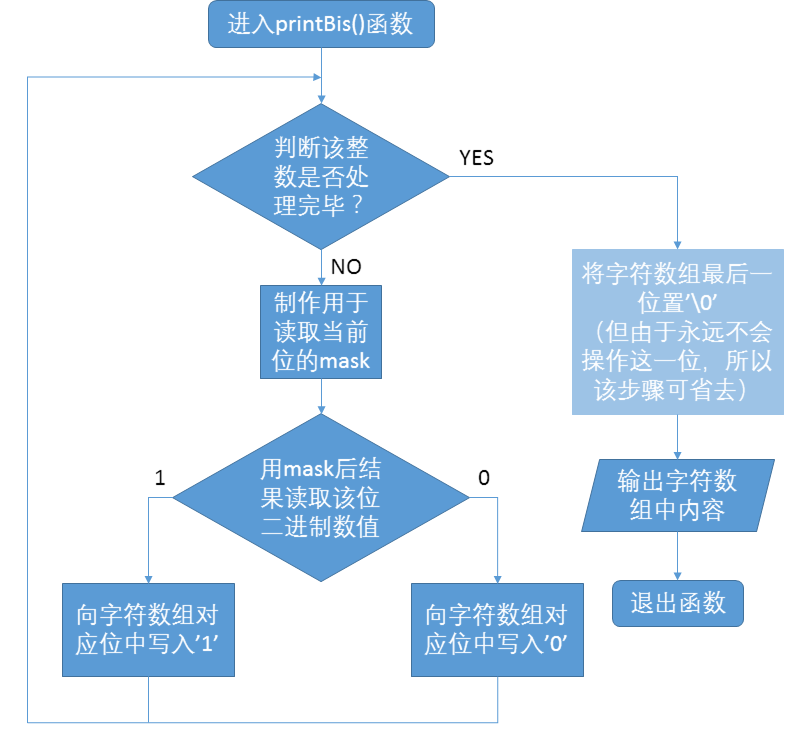
**运行结果截图**



（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。

**算法流程**

这里只给出关键函数算法：

****

**源程序**

#include <stdio.h>

void printBits(int, char \*);

void main(void) {

unsigned N;

scanf("%u", &N); getchar();

int nums[120] = {0};

char strNums[sizeof(int)\*8+1] = {'\0'};

unsigned i;

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

scanf("%d", (nums + i));

}

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

printBits(nums[i], strNums);

}

}

void printBits(int dec, char \*str) {

int cur;

for (cur = sizeof(int)\*8-1; cur >= 0; --cur) {

str[sizeof(int)\*8-1-cur] = ( ( dec & (1<<cur) ) ? '1' : '0' );

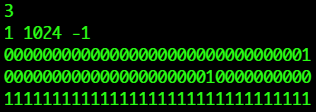
}

str[sizeof(int)\*8] = 0;

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

}

**运行结果截图**



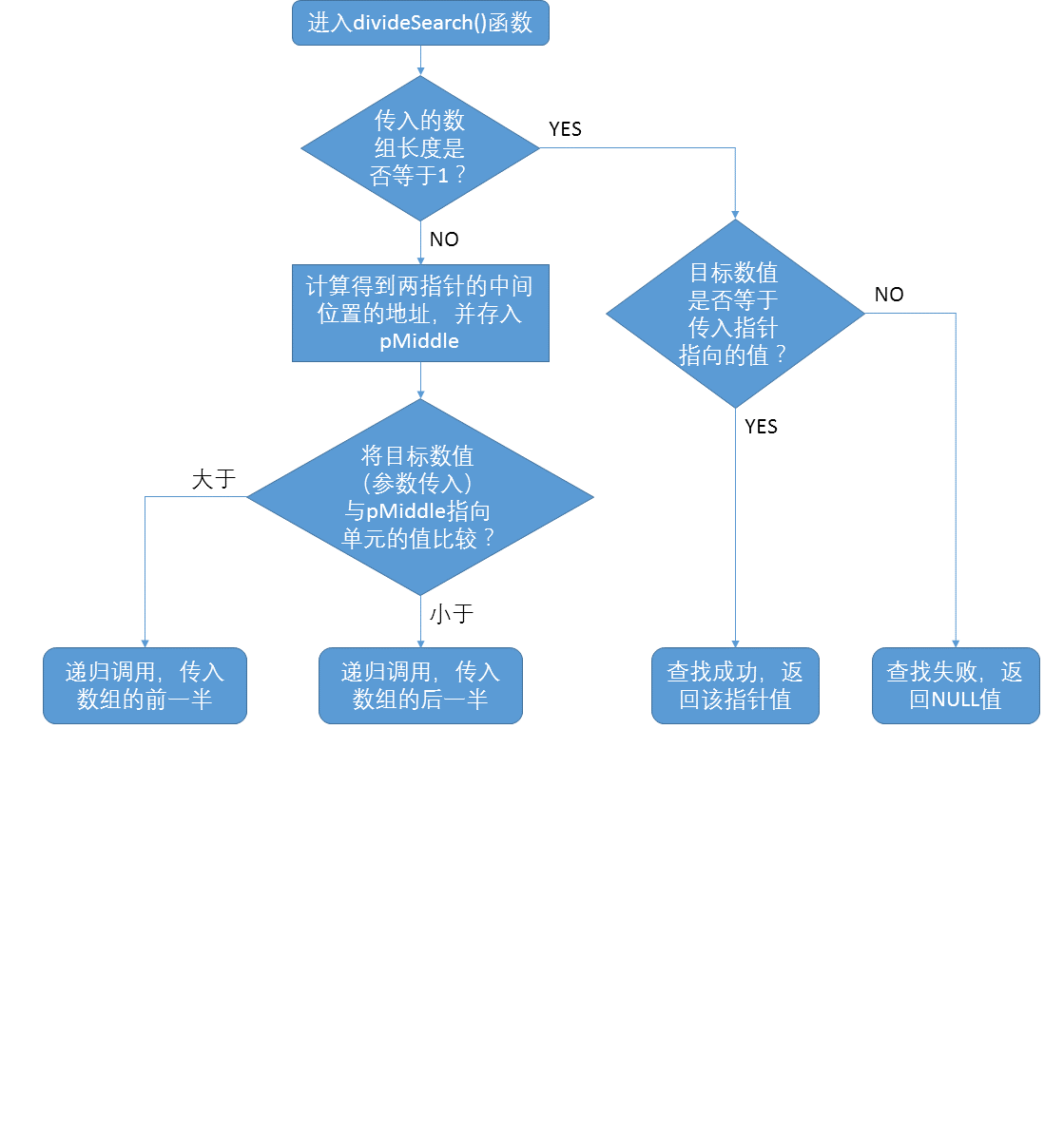
（3）编写一个程序, 其功能要求是：输入n个学生的姓名和C语言课程的成绩，将成绩按从高到低的次序排序，姓名同时作相应调整，输出排序后学生的姓名和C语言课程的成绩。然后，输入一个C语言课程成绩值，用二分查找进行搜索。如果查找到有该成绩，输出该成绩同学的姓名和C语言课程的成绩；否则输出提示“not found!”。

**算法流程**

（注：这里将学生姓名与其C语言成绩分别存在两个相互独立的数组中，但相同学生的信息在两个数组中下标相同。输入和在排序过程中交换数据时，两个数组对应的数据都需要交换）

（排序使用冒泡法进行排序）

二分查找算法（注：此时成绩已经过从大到小的排序）：



**源程序**

#include <stdio.h>

#include <string.h>

unsigned char \* divideSearch(unsigned char target, unsigned char \*start, unsigned count);

char Student\_name[100][21] = {'\0'};

unsigned char Student\_grade[100] = {0};

void main(void) {

unsigned N;

scanf("%u", &N); getchar();

unsigned idx, n; // loop var

char ctmp[21]; unsigned char uctmp;

for (idx = 0; idx < N; ++idx) {

scanf("%s %hhu", \*(Student\_name+idx), (Student\_grade+idx));

getchar(); // swallow '\n'

}

for (idx = 0; idx < N; ++idx) {

for (n = 0; n < N-1-idx; ++n) {

if (Student\_grade[n] < Student\_grade[n+1]) {

uctmp = Student\_grade[n+1];

Student\_grade[n+1] = Student\_grade[n];

Student\_grade[n] = uctmp;

strcpy(ctmp, \*(Student\_name+n+1));

strcpy(\*(Student\_name+n+1), \*(Student\_name+n));

strcpy(\*(Student\_name+n), ctmp);

}

}

}

for (idx = 0; idx < N; ++idx) {

printf("%-20s %hhu\n", Student\_name[idx], Student\_grade[idx]);

}

putchar('\n');

////////////////////////////////

unsigned M;

unsigned char search\_cont[100] = {0};

scanf("%u", &M); getchar();

if (M > N) {

puts("There isn't that much students here!");

return;

}

for (idx = 0; idx < M; ++idx) {

scanf("%hhu", (search\_cont+idx));

getchar();

}

for (n = 0; n < M; ++n) {

unsigned char \* cur = divideSearch(search\_cont[n], Student\_grade, N);

if (cur != NULL) {

printf("%-20s %hhu\n", Student\_name[cur-Student\_grade], \*cur);

}

else {

puts("Not found!");

}

} // end of search loop

return;

}

unsigned char \* divideSearch(unsigned char target, unsigned char \*start, unsigned count) {

unsigned char \* pMiddle = start + (int)count/2;

if (target == \*pMiddle) { return pMiddle; }

else if (count == 1) { return NULL; }

else if (target > \*pMiddle) { return divideSearch(target, start, count/2); }

else if (target < \*pMiddle) { return divideSearch(target, pMiddle, count - count/2); }

else {

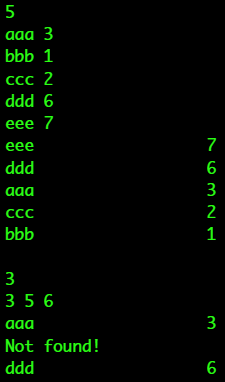
puts("Unexpected operation!");

return NULL;

}

}

**运行结果截图**



5.2.5 选做题

（1）编写函数strnins(s,t,n)，其功能是：可将字符数组t中的字符串插入到字符数组s中第n个字符的后面。

**算法流程**

strnins()功能实现：

1. 通过将数组片段整体后移，将需要占用的空间腾出来

（注：移动过程中采用从后向前的顺序，否则片段前面的数据可能会冲掉片段后面的数据，导致后面的数据丢失、错误）

1. 向腾出的空间写入要插入的字符

**源程序**

#include <stdio.h>

unsigned myStrlen(char s[]) {

unsigned i;

for (i = 0; s[i]; ++i) ;

return i;

}

char \* strnins(char s[], char t[], unsigned n) {

if (n > myStrlen(s)) { return NULL; }

unsigned i, j = myStrlen(t);

// 挪动s中n之后的元素

for (i = myStrlen(s)-1; i >= n; --i) {

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

}

// 在挪出来的空位中填充t

for (i = n; t[i-n]; ++i) {

s[i] = t[i-n];

}

return s;

}

void main(void) {

char s[20] = "abcdefg", t[20] = "123";

puts(strnins(s, t, 2));

}

**运行结果截图**



（2）编写一个实现八皇后问题的程序，即：在8×8方格国际象棋盘上放置8个皇后，任意两个皇后不能位于同一行、同一列或同一斜线（正斜线或反斜线）上，并输出所有可能的放法。

**算法流程**

这里采用模拟八进制数加一的方式来列举出每一种情况（也可以用实际的数值代替每一种情况，但是这里没有使用这种实现方法）。

checkerboard数组中每一位数代表该列（行）上，皇后所在的行（列）。这样省去了判断是否有棋子在同一列（行）上的步骤，因为数组的一个元素单元中只能存放一个数值。

printCheckerBoard()函数用于打印棋盘

judge()函数用于判断是否符合“八皇后”的情况。

plusOne()函数用于在checkerboard数组上模拟八进制加一的情况，传入的参数为需要加一的位置（checkerboard的下标）：

1. 如果传入的值为-1，则表明已经遍历了所有情况，返回False（0）；
2. 如果指定位的值不为8，则加一操作合法，加一后返回True（1）；
3. 如果指定位的值为8，则当前位置初始值（1），递归调用，对当前位的前一位进行加一操作。

**源程序**

#include <stdio.h>

int checkerboard[8] = {1, 1, 1, 1, 1, 1, 1, 1};

// no need for judging row - only one 'i' in a row

void printCheckerboard(void) {

int i, j;

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

for (j = 1; j <= 8; ++j) {

putchar((checkerboard[i] == j) ? 'i' : '\_');

putchar(' ');

}

putchar('\n');

}

putchar('\n');

}

int judge(void) {

int i, j;

// column

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

for (j = 0; j < 8; ++j) {

if (checkerboard[i] == checkerboard[j] && i != j) {

return 0;

}

}

}

// pos\_diagonal

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

for (j = 0; j < 8; ++j) {

if (checkerboard[i]+j == checkerboard[j]+i && i != j) {

return 0;

}

}

}

// neg\_diagonal

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

for (j = 0; j < 8; ++j) {

if (checkerboard[j]+j == checkerboard[i]+i && i != j) {

return 0;

}

}

}

return 1;

}

// simulating octal +1 and carry

int plusOne(int cur) {

// exit status

if (cur == -1) { return 0; }

// no carry

if (checkerboard[cur] != 8) {

checkerboard[cur]++;

return 1;

}

// carry

else if (checkerboard[cur] == 8) {

checkerboard[cur] = 1;

// pass value directly

switch (plusOne(cur-1)) {

case 0: { return 0; }

case 1: { return 1; }

default: { puts("This should not show!"); return -1; }

}

}

}

void main(void) {

int count = 0;

// no need to judge the first case

while (plusOne(7)) {

if (judge()) {

printCheckerboard();

++count;

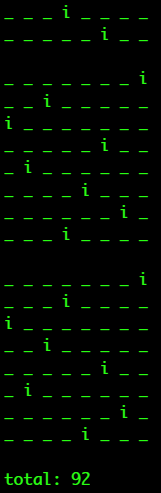
}

}

printf("total: %d case(s)\n", count);

}

**运行结果截图**



5.3 自设题

实现简单的列表切片功能：静态数组，正向切片可循环

语法：直接输入数字可获取对应元素；r"[(0-9):(0-9)]"获得两数之间的所有元素。

**算法流程**

用取余运算实现绕回。

getFromArray()函数中有简单的语义分析：

输入的第一个字符是否是'['：

1. 是：切片操作。继续读取，':'之前的数为开始位置，之后直到']'的数为结束位置
2. 否：直接用下标取元素

**源程序**

#include <stdio.h>

unsigned char str2int(char c) {

return c - '0';

}

void getFromArray(char oper[], int \* array) {

unsigned char start = 0, end = 0;

unsigned i;

if (oper[0] != '[') {

for (i = 0; oper[i]; ++i) {

start += str2int(oper[i]);

}

printf("%4d\n", array[start]);

return;

}

for (i = 1; oper[i] != ':'; ++i) {

start += str2int(oper[i]);

}

for (++i; oper[i] != ']'; ++i) {

end += str2int(oper[i]);

}

for (i = start; i < end-1; ++i) {

printf("%4d", \*(array + i%5) );

}

printf("%4d\n", \*(array + i%5));

return;

}

void main(void) {

int array[5] = {1, 2, 3, 4, 5};

char oper[10] = {'\0'};

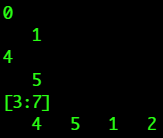
while (scanf("%s", oper), getchar(), oper[0] != 'q') {

getFromArray(oper, array);

}

}

**运行结果截图**



5.4 实验小结

在本次实验中，发现程序设计题第3题在提交至作业系统的时候，我并没有使用二分法，于是在实验课上修改。在重写查找算法的过程中，我充分认识到了认真思考递归结束条件的重要性，必须要全面、不遗漏任何一种情况，否则就会core dump！在以后的编程中，在碰到递归的时候要多加小心！

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 实验题目及要求

6.2.1 源程序改错题

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。

**源程序**

#include "stdio.h"

void main(void)

{

**// float f;**

float \*p; **// float \*p = &f;**

scanf("%f",p);

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

}

运行结果截图



6.2.2 源程序完善、修改、替换题

（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 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>5);

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;

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

i=0;

result= (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.

**源程序**

#include "stdio.h"

#include "string.h"

void main(void)

{

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

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>5);

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;

**for (i = 0; \*c = getchar(), \*c != '\n'; ++i) { a[i] = \*c; }**

**a[i] = 0;**

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

i=0;

**for (i = 0; \*c = getchar(), \*c != '\n'; ++i) { b[i] = \*c; }**

**b[i] = 0;**

result=p(a,b);

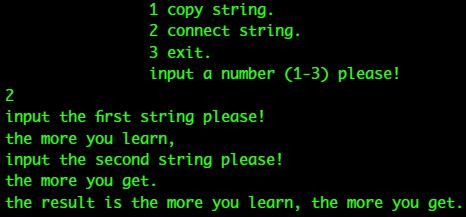
printf("the result is %s\n",result);

down:

;

}

**运行结果截图**



6.2.3 跟踪调试题

#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？

（1）





（2）

**源程序**

...

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

{

// TODO: added line below

**char \*original = s;**

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

;

// TODO: modified line below

return (**original**);

}

...

**运行结果截图**



（3）（gdb在调试时显示地址值）在strcpy函数的循环中printf出s的地址或者在while语句行添加断点。

6.2.4 编程设计题

（1）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。试从该长整型变量的高字节开始，依次取出每个字节的高4位和低4位并以数字字符的形式进行显示。

**算法流程**

1. 定义储存长整形的容器num和用于取字节的ruler指针
2. 依次用ruler指向并取出num的4个字节
3. 对ruler取出的值进行处理：
4. printHex()函数以十六进制的格式打印ruler中的值
5. num2Hexstr()函数将值转换为十六进制数对应的字符

**源程序**

#include <stdio.h>

void num2Hexstr(char c) {

if (c<10 && c>=0) {

putchar(c+'0');

}

else if (c>9 && c<16) {

putchar(c+'A'-10);

}

else {

puts("This should not show!!!");

printf("%hhd", c);

}

return;

}

void printHex(char one\_two) {

num2Hexstr(

(one\_two >> 4) & 0x0F

);

putchar(' ');

num2Hexstr(

(one\_two & 0x0F)

);

return;

}

void main(void) {

unsigned N;

scanf("%u", &N); getchar();

long num; // \*\*\*\*-\*\*\*\* \*\*\*\*-\*\*\*\* \*\*\*\*-\*\*\*\* \*\*\*\*-\*\*\*\*

char \* ruler; // \*\*\*\*-\*\*\*\*

unsigned char i;

for (; N; --N) {

scanf("%ld", &num); getchar();

for (i = 4, ruler = (char \*)&num+3; i; --i, --ruler) {

printHex(\*ruler);

if (i != 1) { putchar(' '); }

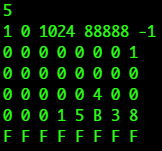
else { putchar('\n'); }

}

}

}

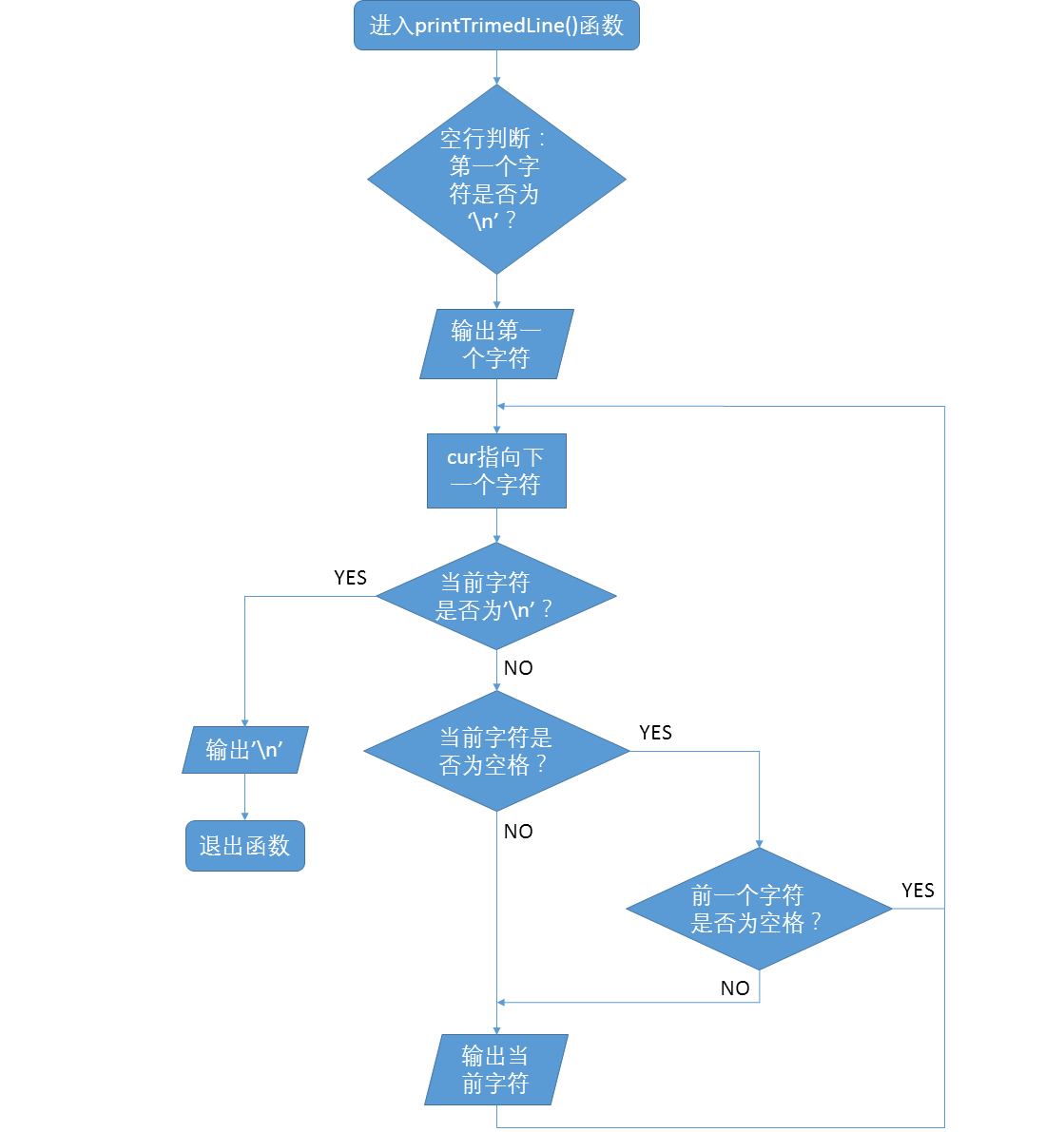
**运行结果截图**



（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。

**算法流程**

给出关键函数printTrimedLine()算法：



**源程序**

#include <stdio.h>

#include <stdlib.h>

void printTrimedLine(char \* cur) {

if (\*cur == '\n') {

return;

}

putchar(\*cur);

for (cur += 1; \*cur && \*cur!='\n'; ++cur) {

if (\*cur != ' ') {

putchar(\*cur);

}

else {

if (\*(cur-1) != ' ') {

putchar(\*cur);

}

}

}

putchar('\n');

return;

}

void main(void) {

unsigned num\_of\_lines = 0;

char \* lines[100];

unsigned char i;

while (

scanf("%u", &num\_of\_lines),

getchar(),

num\_of\_lines

) {

for (i = 0; i < num\_of\_lines; ++i) {

lines[i] = (char \*)malloc(81);

fgets(lines[i], 80, stdin);

}

for (i = 0; i < num\_of\_lines; ++i) {

printTrimedLine(lines[i]);

}

putchar('\n');

for (i = 0; i < num\_of\_lines; ++i) {

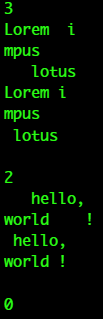
free(lines[i]);

}

}

}

**运行结果截图**



（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）。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：

a.计算每个学生各门课程平均成绩；

b.计算全班每门课程的平均成绩；

c.分别统计低于全班各门课程平均成绩的人数；

d.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。）

**算法流程**

这里使用了结构。

**源程序**

#include <stdio.h>

#define M 5 // course count (of each student)

#define N 5 // student count

//////////////////////////////////////

typedef struct {

char name[81];

unsigned char grade[M]; // this should be kept align with courses

double avg;

} Student;

typedef struct {

char name[81];

double avg;

} Course;

//////////////////////////////////////

Course courses[M];

Student students[N];

//////////////////////////////////////

/\* Student Operations \*/

/\* 计算每个学生各门课程的平均成绩 \*/

double getAvgOfStudent(Student Sally) {

unsigned i = 0;

for (Sally.avg = 0.; i < M; ++i) {

Sally.avg += Sally.grade[i];

}

Sally.avg /= M;

printf("Average score of %s is %.2lf\n", Sally.name, Sally.avg);

return Sally.avg;

}

/\* Class Operations \*/

/\* 计算全班每门课程的平均成绩 \*/

void getAvgOfClass(void) {

unsigned each\_course = 0, each\_student = 0;

for (; each\_course < M; ++each\_course) {

courses[each\_course].avg = 0.;

for (each\_student = 0; each\_student < N; ++each\_student) {

courses[each\_course].avg += students[each\_student].grade[each\_course];

} // student loop

courses[each\_course].avg /= M;

printf("Average score of %s is %.2lf\n",

courses[each\_course].name,

courses[each\_course].avg);

} // course loop

}

/\* 统计低于平均分的人数 \*/

void getStudentCountBelowAvg(void) {

unsigned student\_count = 0, each\_student = 0, each\_course = 0;

for (; each\_course < M; ++each\_course, student\_count = 0) {

for (each\_student = 0; each\_student < N; ++each\_student) {

if (

students[each\_student].grade[each\_course] < courses[each\_course].avg

) { ++student\_count; }

}

printf("Number of students lower than avg of %s is %u\n",

courses[each\_course].name, student\_count);

} // course loop

}

/\* 统计不及格人数 \*/

void getStudentCountFailed(void) {

unsigned student\_count = 0, each\_student = 0, each\_course = 0;

for (; each\_course < M; ++each\_course, student\_count = 0) {

for (each\_student = 0; each\_student < N; ++each\_student) {

if (

students[each\_student].grade[each\_course] < 60.0

) { ++student\_count; }

}

printf("Number of students %s fail is %u\n",

courses[each\_course].name, student\_count);

} // course loop

}

/\* 统计dalao人数 \*/

void getStudentCountPerfect(void) {

unsigned student\_count = 0, each\_student = 0, each\_course = 0;

for (; each\_course < M; ++each\_course, student\_count = 0) {

for (each\_student = 0; each\_student < N; ++each\_student) {

if (

students[each\_student].grade[each\_course] >= 90.0

) { ++student\_count; }

}

printf("Number of students %s perfect is %u\n",

courses[each\_course].name, student\_count);

} // course loop

}

/////////////////////////////////////////////////////////////

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

unsigned i, j;

// input

for (i = 0; i < M; ++i) {

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

}

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

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

for (j = 0; j < M; ++j) {

scanf("%hhu\n", &students[i].grade[j]);

}

}

// output

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

getAvgOfStudent(students[i]);

}

getAvgOfClass();

getStudentCountBelowAvg();

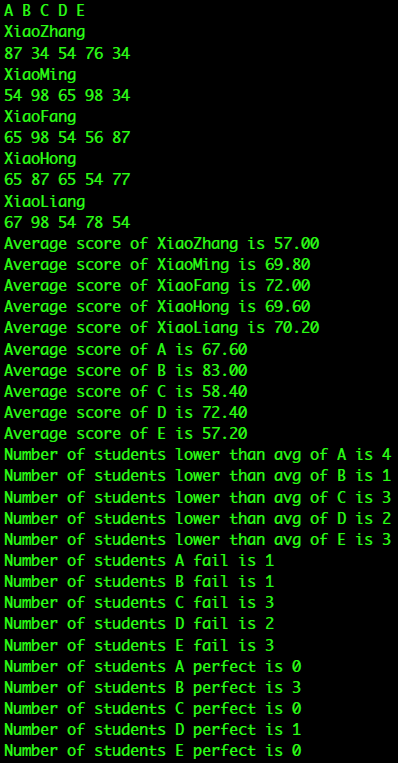
getStudentCountFailed();

getStudentCountPerfect();

return 0;

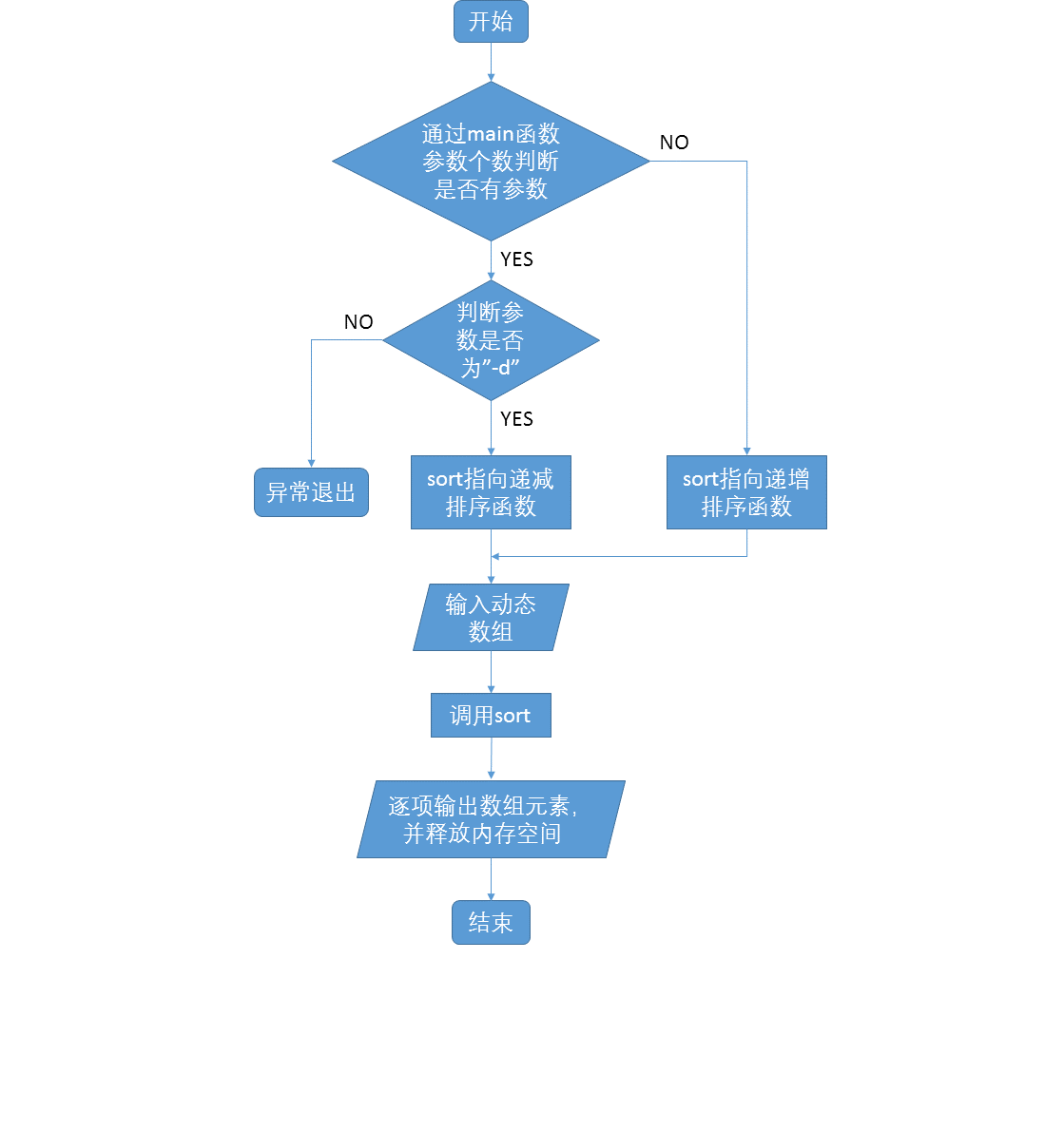
}

**运行结果截图**



（4）编写一个程序，输入n个整数，排序后输出。排序的原则由命令行可选参数-d决定，并且有参数-d时按递减顺序排序，否则按递增顺序排序。要求将排序算法定义成函数，利用指向函数的指针使该函数实现递增或递减排序。

**算法流程**



**源程序**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

void \_sort\_asc(int \*pArr, int n) {

int i, j;

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

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

if (pArr[j] > pArr[j+1]) {

pArr[j+1] += pArr[j];

pArr[j] -= pArr[j+1];

pArr[j] \*= -1;

pArr[j+1] -= pArr[j];

}

}

}

}

void \_sort\_des(int \*pArr, int n) {

int i, j;

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

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

if (pArr[j] < pArr[j+1]) {

pArr[j+1] += pArr[j];

pArr[j] -= pArr[j+1];

pArr[j] \*= -1;

pArr[j+1] -= pArr[j];

}

}

}

}

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

int n;

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

int \*pArr = (int \*)malloc(n\*sizeof(int));

if (pArr == NULL) { exit(-1); }

int i;

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

scanf("%d", pArr+i); getchar();

}

void (\*sort)(int \*, int) = \_sort\_asc;

if (argc > 1) {

if ( !strcmp("-d", argv[1]) ) {

sort = \_sort\_des;

}

else { exit(-1); }

}

sort(pArr, n);

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

printf("%8d", pArr[i]);

}

putchar('\n');

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

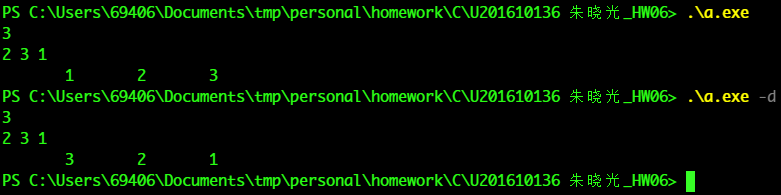
free(pArr+i);

}

return 0;

}

**运行结果截图**



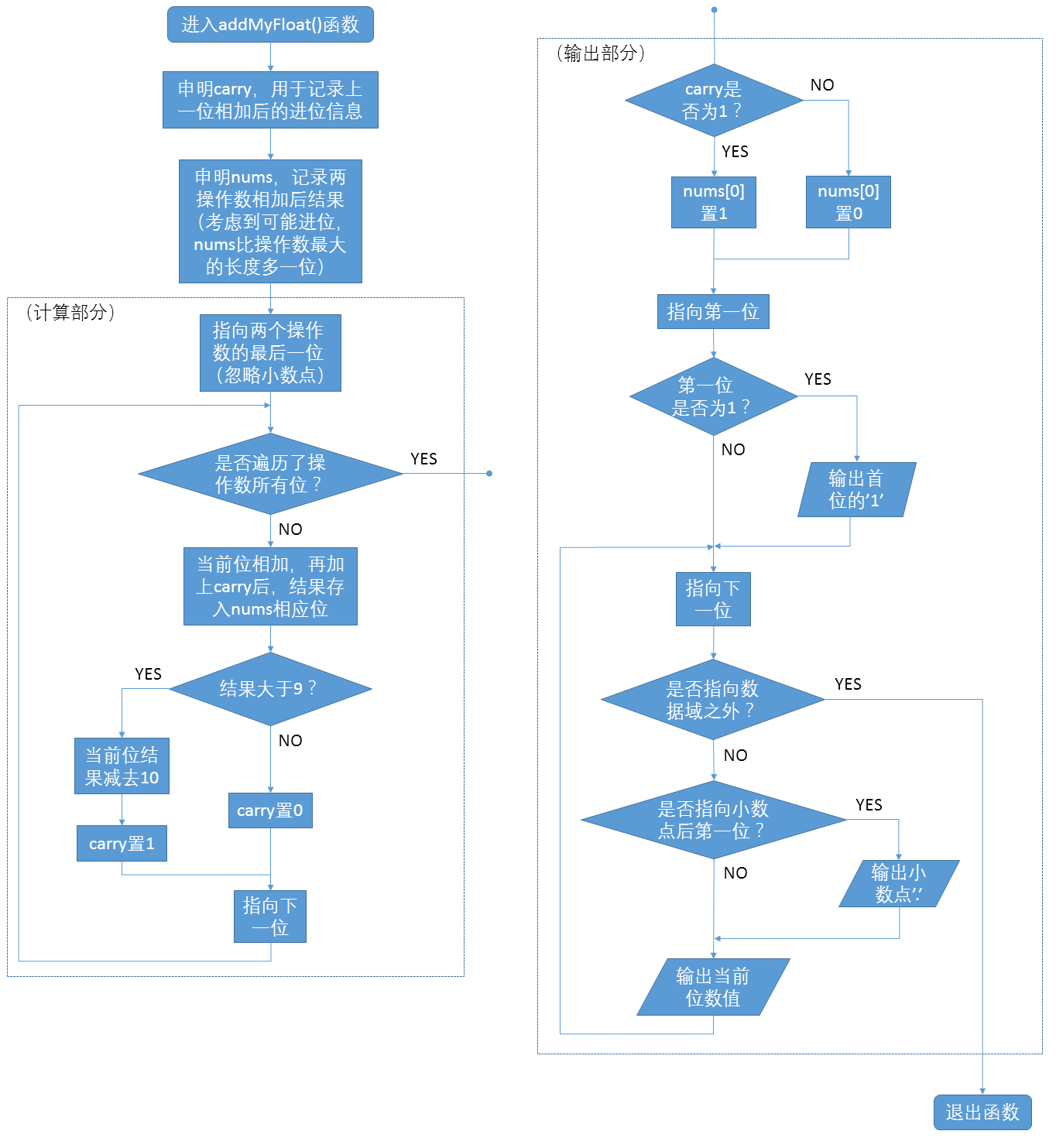
6.2.5 选做题

（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

**算法流程**

只给出关键函数addMyFloat()的算法：



**源程序**

#include <stdio.h>

#define N 20 // real

#define M 10 // frac

//////////////////////////////////////

typedef struct {

char num[N+M]; // 数据

// unsigned magnitude; // 小数点位数：已确定，不需要

} MyFloat;

//////////////////////////////////////

/\* NOTE: 由于题目声明了数据位数相同，且占满给定位数，故不考虑其他情况 \*/

/\* 如：不需要考虑数位对齐 \*/

void getMyFloat(MyFloat \*num) {

unsigned i = 0;

char buf; // 防止吞'\n'时干扰别的内存区域

while ((buf = getchar() - '0') != '\n' - '0') {

num->num[i] = buf;

// 用下一位数据覆盖小数点

if (num->num[i] == '.' - '0') { continue; }

else { ++i; }

}

}

void addMyFloat(MyFloat a, MyFloat b) {

char nums[N+M+1] = {0};

int i = M + N - 1;

char carry = 0;

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

nums[i+1] = a.num[i] + b.num[i] + carry;

if (nums[i+1] >= 10) {

carry = 1;

nums[i+1] -= 10;

}

else {

carry = 0;

}

}

if (carry) { nums[0] = 1; }

else { nums[0] = 0; }

// output

i = 0;

if (nums[0] == 0) { ++i; }

for (; i < M+N+1; ++i) {

printf("%hhd", nums[i]);

if (i == N) { putchar('.'); }

}

putchar('\n');

}

//////////////////////////////////////

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

MyFloat a, b;

getMyFloat(&a);

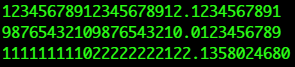
getMyFloat(&b);

addMyFloat(a, b);

return 0;

}

**运行结果截图**



（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。

**源程序**

#include <stdio.h>

#include <string.h>

char \* strstrpp(const char \*a, const char \*b) {

return strstr(a, b) + 1;

}

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

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

p some\_func;

some\_func[0] = strstr;

some\_func[1] = strstrpp;

char a[] = "Hello";

char b[] = "ll";

printf("%s\n", some\_func[0](a, b));

printf("%s\n", some\_func[1](a, b));

return 0;

}

**运行结果截图**



6.3 自设题

使用C语言中的结构与复杂指针类型，模仿出C++中类的部分功能。

（这里实现的有类初始化方法、变量赋值方法以及打印变量方法）

**源程序**

#include <stdio.h>

//////////////////////////////////////////////////

/\*\*\*\*\* Class Definition \*\*\*\*\*/

/\*\*\* Class Members \*\*\*/

typedef struct MyStruct {

int \_some\_number;

void (\*f)(struct MyStruct \*);

struct MyStruct \* self;

} MyClass;

/\*\*\* Class Functions \*\*\*/

MyClass \_\_init\_MyClass\_\_(

int a, void (\*func)(MyClass \*)

) {

/\* Class Init Function

Returns the desired class object

\*/

MyClass obj;

obj.\_some\_number = a;

obj.f = func;

return obj;

}

void func(MyClass \*self) {

/\* Some Other Function \*/

// set var

scanf("%d", &(self->\_some\_number));

puts("Hello, Wolrd!");

// put var

printf("self.\_some\_number = %d\n", self->\_some\_number);

return;

}

///////////////////////////////////////////////////

void main(void) {

/\* Instantiating Class Object \*/

MyClass Object = \_\_init\_MyClass\_\_(42, func);

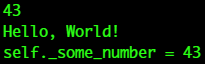
Object.self = &Object;

// Instantiation Finished

Object.f(&Object);

}

**运行结果截图**



6.4 实验小结

通过这次实验，我更加深入地研究了复杂类型的定义，基本弄清楚了函数与函数指针之间的关系，并对函数指针有了更加深刻地理解，可以更加灵活地运用函数指针（见自设题）。

在解决跟踪调试题时，尝试使用了gdb的命令行，了解了一些简单的gdb命令行操作。

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 实验题目及要求

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。

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 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 \*/

**// create\_list(&head, s);**

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)

**// 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 \*/

**// \*headp = loc\_head;**

}

**运行结果截图**



（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。

...

void create\_list(struct s\_list \*\*tailp, int \*data) {

struct s\_list \*prev;

if (data[0] == 0) { \*tailp = NULL; return; }

else {

if ((\*tailp = (struct s\_list \*)malloc(sizeof(struct s\_list))) == NULL) { exit(-1); }

(\*tailp)->data = data[0];

(\*tailp)->next = NULL;

for (prev = \*tailp, ++data; \*data; ++data, prev = \*tailp){

if (((\*tailp) = (struct s\_list \*)malloc(sizeof(struct s\_list))) == NULL) { exit(-1); }

(\*tailp)->next = prev;

(\*tailp)->data = \*data;

}

return;

}

}

...

**运行结果截图**



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);

}

**算法流程**

1. 将p\_isr中每一项赋值为对应的函数指针
2. 通过ISR\_REG.all可以一次性给所有的规则赋值
3. 通过ISR\_REG.biti取得对应的规则，然后可以进行相应的操作

**源程序清单**

#include <stdio.h>

#include <stdlib.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(void) { puts("The Interrupt Service Routine isr0 is called!"); }

void isr1(void) { puts("The Interrupt Service Routine isr1 is called!"); }

void isr2(void) { puts("The Interrupt Service Routine isr2 is called!"); }

void isr3(void) { puts("The Interrupt Service Routine isr3 is called!"); }

void isr4(void) { puts("The Interrupt Service Routine isr4 is called!"); }

void isr5(void) { puts("The Interrupt Service Routine isr5 is called!"); }

void isr6(void) { puts("The Interrupt Service Routine isr6 is called!"); }

void isr7(void) { puts("The Interrupt Service Routine isr7 is called!"); }

void main(void) {

void (\*p\_isr[8])(void);

p\_isr[0] = isr0; p\_isr[1] = isr1; p\_isr[2] = isr2; p\_isr[3] = isr3;

p\_isr[4] = isr4; p\_isr[5] = isr5; p\_isr[6] = isr6; p\_isr[7] = isr7;

unsigned n, i;

scanf("%u", &n); getchar();

union ISR\_REG \*flags = (union ISR\_REG \*)malloc(n \* sizeof(union ISR\_REG));

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

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

}

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

**/\*\*\*\* CORE CODE \*\*\*\*/**

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

if (flags[i].bit.bit0) { p\_isr[0](); }

if (flags[i].bit.bit1) { p\_isr[1](); }

if (flags[i].bit.bit2) { p\_isr[2](); }

if (flags[i].bit.bit3) { p\_isr[3](); }

if (flags[i].bit.bit4) { p\_isr[4](); }

if (flags[i].bit.bit5) { p\_isr[5](); }

if (flags[i].bit.bit6) { p\_isr[6](); }

if (flags[i].bit.bit7) { p\_isr[7](); }

putchar('\n');

**/\*\*\*\* END \*\*\*\*/**

}

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

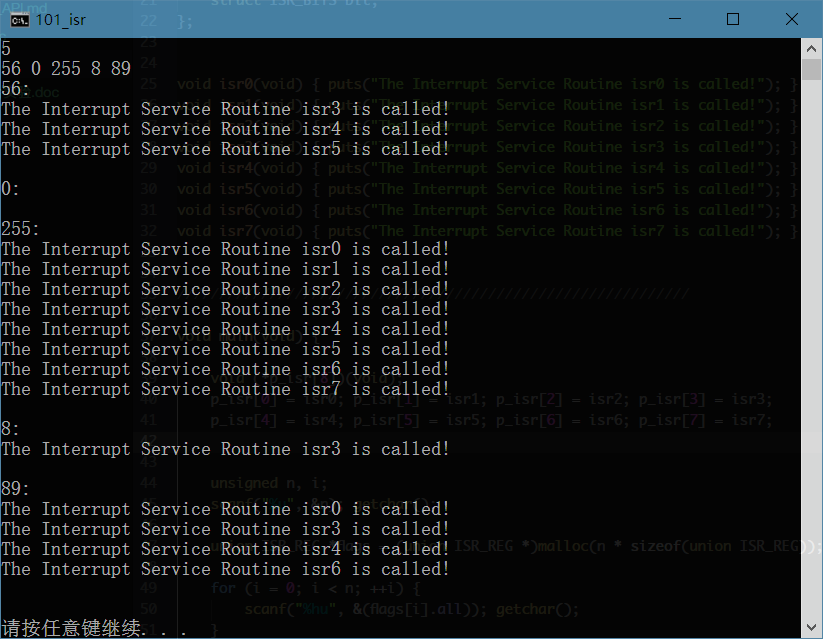
free(flags+i);

}

return;

}

**运行结果截图**



（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。

**算法流程**

本例中采用的数据结构与API一览：







其中交换数据域的排序方法sortChainSwapData()采用冒泡排序；交换Node节点顺序的排序方法sortChainChangeRoute()中交换节点采用的具体算法，同教材上第一种交换节点顺序的排序方法。

**源程序清单**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define True 1

//////////////////////////////////

enum Courses {

English = 0,

Math,

Physics,

C

};

int getCourseCode(char const \*name) {

if (!strcmp(name, "English")) { return 0; }

else if (!strcmp(name, "Math")) { return 1; }

else if (!strcmp(name, "Physics")) { return 2; }

else if (!strcmp(name, "C")) { return 3; }

else { return -1; }

}

#define COURSE\_CNT 4

//////////////////////////////////

/\*\*\*\* Student Class \*\*\*\*/

typedef struct \_Student {

char id[15];

char name[20];

float grades[COURSE\_CNT];

double avg;

} Student;

Student \* \_initStudent(void) {

/\* create a Student record

RETURN the pointer of the record

\*/

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

scanf("%s", stu->id); getchar();

scanf("%s", stu->name); getchar();

for (int i = 0; i < COURSE\_CNT; ++i) {

scanf("%f", &(stu->grades[i])); getchar();

}

return stu;

}

void \_printStudent(Student stu) {

/\* print out a formated Student record

NO RETURN

\*/

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",

stu.id, stu.name,

stu.grades[English],

stu.grades[Math],

stu.grades[Physics],

stu.grades[C]);

return;

}

void \_printStudentStat(Student \*stu) {

/\* ID NAME SUM AVG

\*/

double sum = 0;

for (int i = 0; i < COURSE\_CNT; ++i) {

sum += stu->grades[i];

}

stu->avg = sum / COURSE\_CNT;

printf("%-15s%-20s%-10.2f%-10.2f\n",

stu->id, stu->name,

sum, (sum / COURSE\_CNT));

return;

}

double \_compareStudent(Student Alice, Student Bob) {

return (Alice.avg - Bob.avg);

}

void \_modifyStudent(Student \*stu,

char const \*course\_name,

float new\_score) {

stu->grades[getCourseCode(course\_name)] = new\_score;

return;

}

void \_freeStudent(Student \*stu) {

free(stu);

}

//////////////////////////////////////////////////

/\*\*\*\* Node Class \*\*\*\*/

// index of Student records

typedef struct \_Node {

Student \*data;

struct \_Node \*next;

} Node;

Node \* initStudents(int cnt) {

/\* create $cnt Student records wrapped by Node

RETURN pointer of the chain's head

\*/

Node \*head = (Node \*)malloc(sizeof(Node));

// init first Student

head->data = \_initStudent();

head->next = NULL;

--cnt;

// init following Students

for (Node \*subs = head; cnt; --cnt) {

subs->next = (Node \*)malloc(sizeof(Node)); // create next Node

subs = subs->next; // move to next Node

subs->next = NULL;

subs->data = \_initStudent(); // toggle data input

}

return head;

}

void printStudents(Node \*cur) {

/\* print out all Student records

\*/

printf("%-15s%-20s%-10s%-10s%-10s%-10s\n",

"ID", "Name", "English", "Math", "Physics", "C");

for (; cur; cur = cur->next) {

\_printStudent(\*(cur->data));

// if (cur->next == NULL) { break; }

}

putchar('\n');

}

void printStudentsStat(Node \*head) {

printf("%-15s%-20s%-10s%-10s\n",

"ID", "Name", "SUM", "AVG");

for (; head; head = head->next) {

\_printStudentStat(head->data);

}

putchar('\n');

return;

}

void printStudentsAvg(Node \*head) {

printf("%-15s%-20s%-10s\n",

"ID", "Name", "AVG");

for (; head; head = head->next) {

printf("%-15s%-20s%-10.2f\n",

head->data->id,

head->data->name,

head->data->avg);

// pass

}

putchar('\n');

return;

}

Node \* getNodeById(Node \*head, char const \*id) {

Node \*n = head;

for (; n; n = n->next) {

if (!strcmp(id, n->data->id)) { break; } // node found

if (n->next == NULL) { n = NULL; break; } // end of chain

}

return n;

}

void modifyChain(Node \*head) {

/\* given the head of the chain

will toggle input action

\*/

char id[15]; char course\_name[20]; float new\_score;

scanf("%s %s %f", id, course\_name, &new\_score); getchar();

\_modifyStudent(getNodeById(head, id)->data,

course\_name,

new\_score);

return;

}

double compareNode(Node A, Node B) {

return \_compareStudent(\*(A.data), \*(B.data));

}

void sortChainSwapData(Node \*prev, int len) {

int i, j;

Node \*head = prev;

Node \*subs;

for (i = 0; i < len-1; ++i, prev = prev->next) {

for (j = 0, subs = head; j < len-i-1; ++j, subs = subs->next) {

if (compareNode(\*subs, \*(subs->next)) > 0) {

Student \*tmp = subs->data; // ye only have to swap the data pointer

subs->data = subs->next->data;

subs->next->data = tmp;

// puts("===================");

// printStudentsAvg(head);

// puts("===================");

}

}

}

return;

}

// HACK: got this piece completely from the book, need to study on it

void sortChainChangeRoute(Node \*\*headp) {

Node \*prev1, \*p1, \*prev2, \*p2, \*tmp; // notice the sequence here

// deal with headp - pointer to head pointer

p1 = \*headp;

p2 = p1->next; prev2 = p1; // p2 is the current num

for (; p2; prev2 = p2, p2 = p2->next) {

if (compareNode(\*p1, \*p2) > 0) { // swap spaces p1 & p2 are pointing to

prev2->next = p1;

tmp = p1->next;

p1->next = p2->next;

\*headp = p2; // update head pointer

p2->next = tmp;

p1 = p2;

}

}

// finished with headp

prev1 = \*headp; p1 = prev1->next;

for (; p1->next; prev1 = p1, p1 = p1->next) {

p2 = p1->next; prev2 = p1; // p2 is the current num

for (; p2; prev2 = p2, p2 = p2->next) {

if (compareNode(\*p1, \*p2) > 0) {

tmp = p2->next;

prev1->next = p2;

prev2->next = p1;

p2->next = p1->next;

p1->next = tmp;

p1 = p2;

}

}

}

}

void freeStudents(Node \*cur) {

Node \*pnode;

while (cur) {

pnode = cur;

cur = cur->next;

\_freeStudent(pnode->data); // free data storage

free(pnode); // free Node

}

return;

}

///////////////////////////////////////////////////////////////////////

void main(void) {

int cnt;

scanf("%d", &cnt); getchar();

Node \*chain = initStudents(cnt);

printStudents(chain);

int change\_cnt;

scanf("%d", &change\_cnt); getchar();

for (; change\_cnt; --change\_cnt) {

modifyChain(chain);

}

puts("Alter:");

printStudents(chain);

puts("SumAndAvg:");

printStudentsStat(chain);

puts("Sort:");

// sortChainSwapData(chain, cnt);

sortChainChangeRoute(&chain);

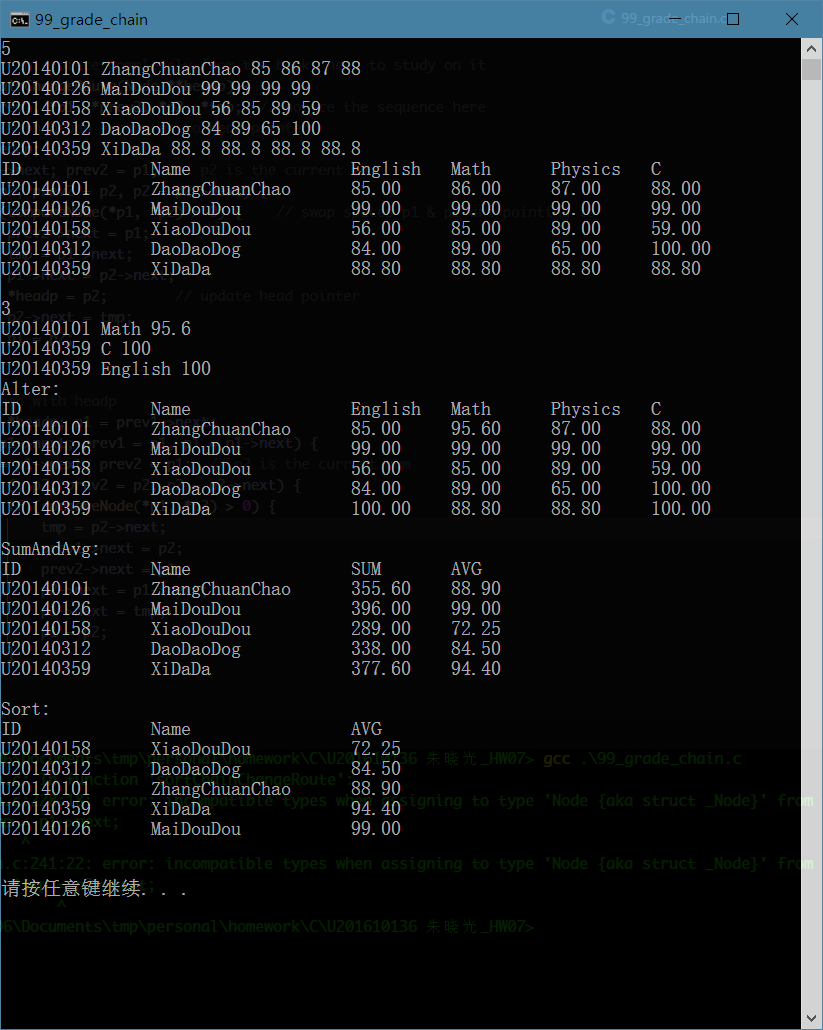
printStudentsAvg(chain);

freeStudents(chain);

return;

}

**运行结果截图**



7.2.4 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。

（源程序清单中sortChainSwapData(...)函数）

（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。

（源原题程序清单中sortChainChangeRoute(...)函数）

（3）采用双向链表重做编程设计题第（2）题。

NULL

7.3 实验小结

通过这次实验，我更加深入地理解了函数编程与面向对象编程之间在程序设计哲学层面上的差别；同时也以肌肉记忆的方式，将malloc与free联系起来，养成内存管理的意识。

实验8 文件实验

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

8.2.1 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。

（下面只考虑小端序机器上的运行情况）

（考虑到程序兼容性，在编译时已将文件目录均改为当前目录）

**预期输出结果**

?%}{

253f 7b7d

**实际运行结果截图**



1. 将两处sizeof(short)均改为sizeof(char)结果有什么不同，为什么？

输出结果中第一行变为：?}



原因：fwrite()函数的参数限制最多能够写进的单位数量为1。因此更改写入时解释数据用的单位大小后，多余的位数被舍去，只留下低位。

1. 将fprintf(fp2,"%hx %hx",a,b) 改为 fprintf(fp2,"%d %d",a,b)结果有什么不同。

输出结果的第二行变为：9535 31613



8.2.2 源程序修改替换题

1. 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

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 \*/

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

putchar(ch); /\* 向显示器中写字符 \*/

fclose(fp); /\* 关闭filename \*/

return 0;

}

**运行结果截图**



1. 用输入输出重定向freopen改写上述源程序中的main函数。

对上一题中第11行做如下修改：

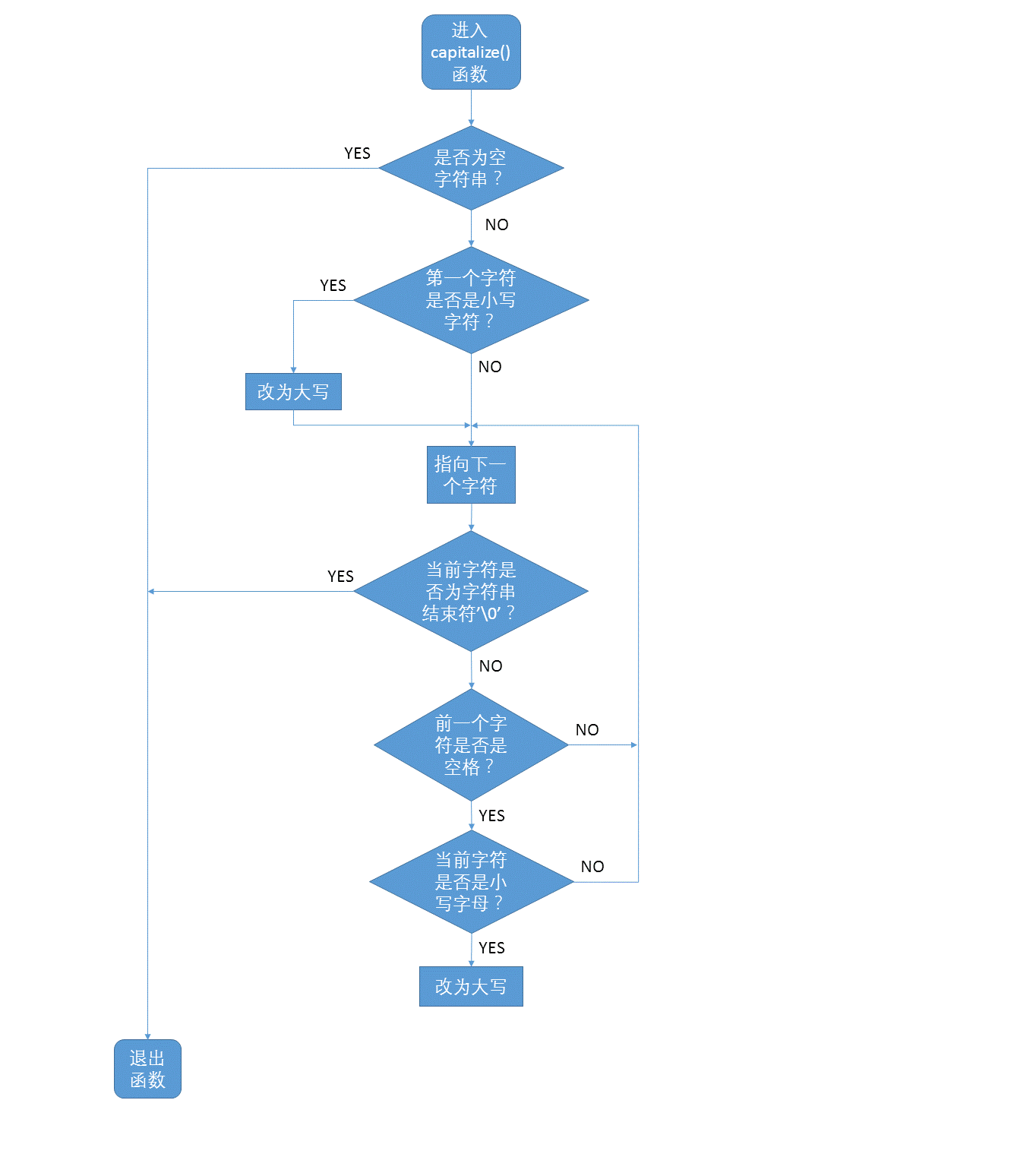
if((fp=freopen(argv[1], ”r”, stdin)) == NULL) {

8.2.3 编程设计题

（1）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。

**算法流程**

给出首字母大写方法capitalize()函数的算法：



**源程序清单**

#include <stdio.h>

#include <stdlib.h>

#define isLowerCase(c) ( ((c)<='z' && (c)>='a') ? 1 : 0 )

void lowerToUpper(char \*c) { \*c = \*c + 'A' - 'a'; }

void capitalize(char \*str) {

if (!str) { return; }

if (isLowerCase(\*str)) { lowerToUpper(str); }

for (++str; \*str; ++str) {

if (\*(str-1)==' ' && isLowerCase(\*str)) { lowerToUpper(str); }

}

return;

}

void main(void) {

char str[81] = {0};

char \*buf = str;

for (; (\*buf=getchar()) != '\n'; ++buf) ;

\*buf = 0;

capitalize(str);

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

FILE \*fp;

#ifdef \_\_WIN32\_\_

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

#elif linux

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

#else

puts("OS type not recognized!");

exit(-1);

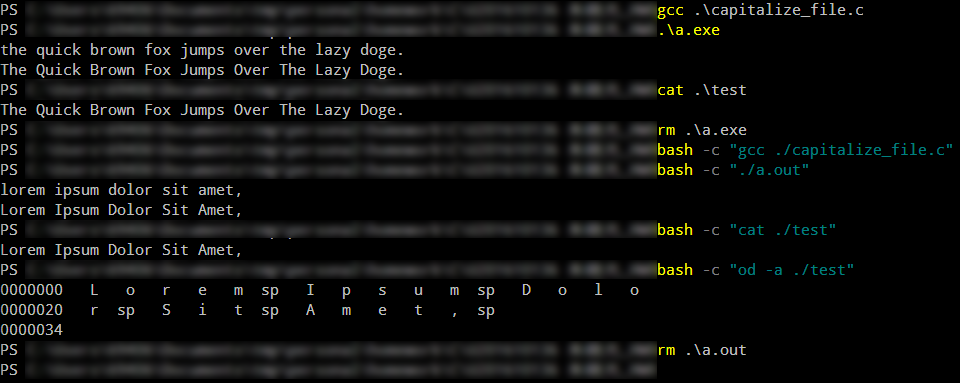
#endif

fputs(str, fp);

fclose(fp);

}

**运行结果截图**



8.3 实验小结

通过这次实验，我熟悉了C语言中基本的文件操作，但使用还不是很熟练，有些概念还不是太清楚，需要在课后加强。

另外，我也尝试编写了通过宏定义来同时适配Windows和\*nix环境的程序（编程设计题第一小题）。