# 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：**

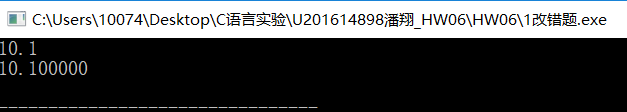
（1）错误修改：

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);

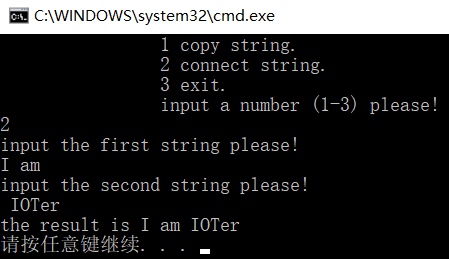
result = p(a, b);

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';

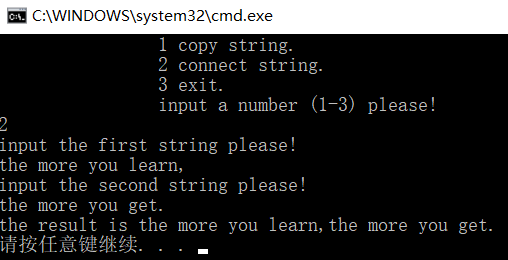
result = p(a, b);

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);

}

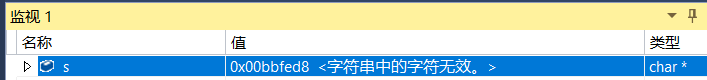
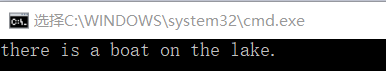
（1）单步执行。进入strcpy时watch窗口中s为何值？返回main时, watch窗口中s为何值？

（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));

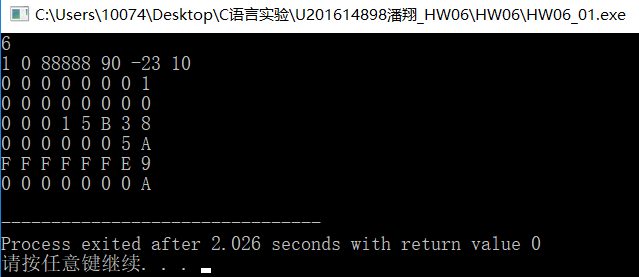
}

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++;

}

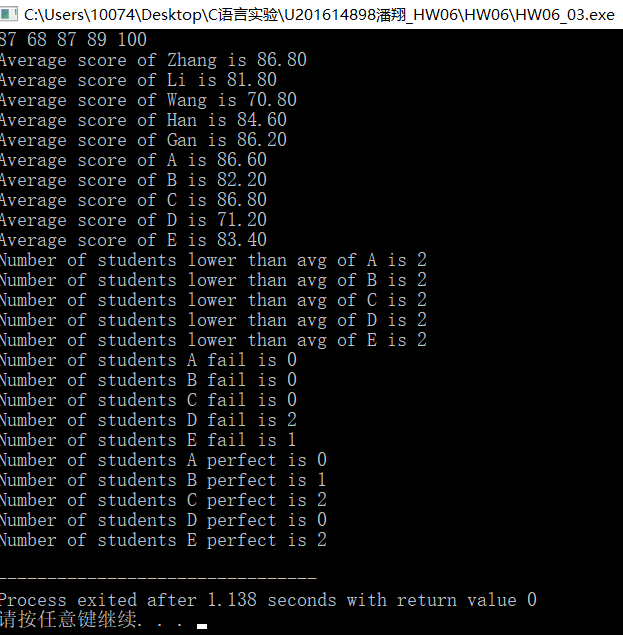
}

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");

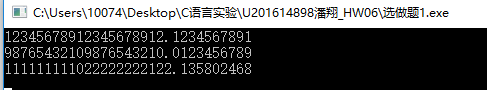
}

}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));

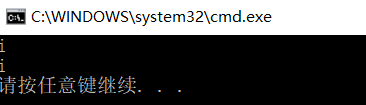
printf("%c\n", \*(\*p[1])(test1, test2));

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++)

{

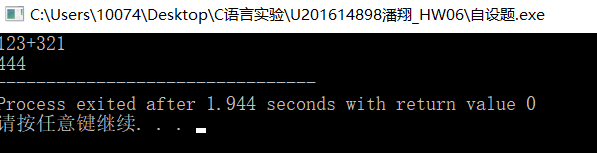
num2[i] = s[i] - '0';

\*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