# 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;

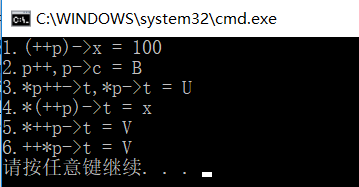
char c;

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 \*/

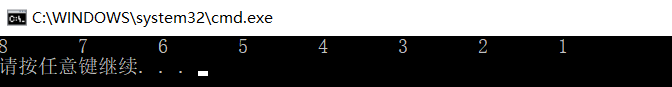
}

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]();

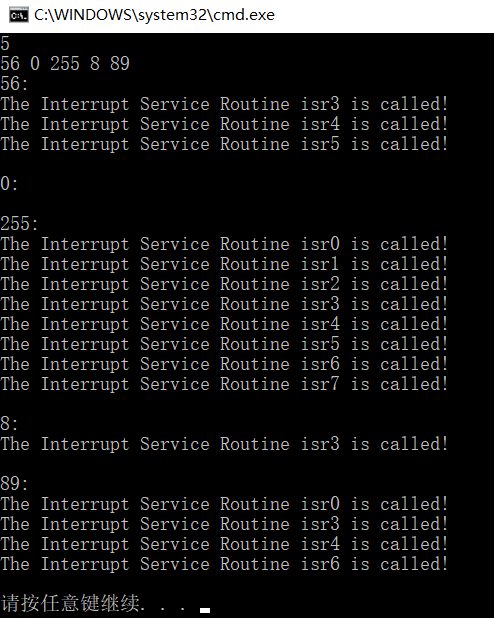
printf("\n");

}

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;

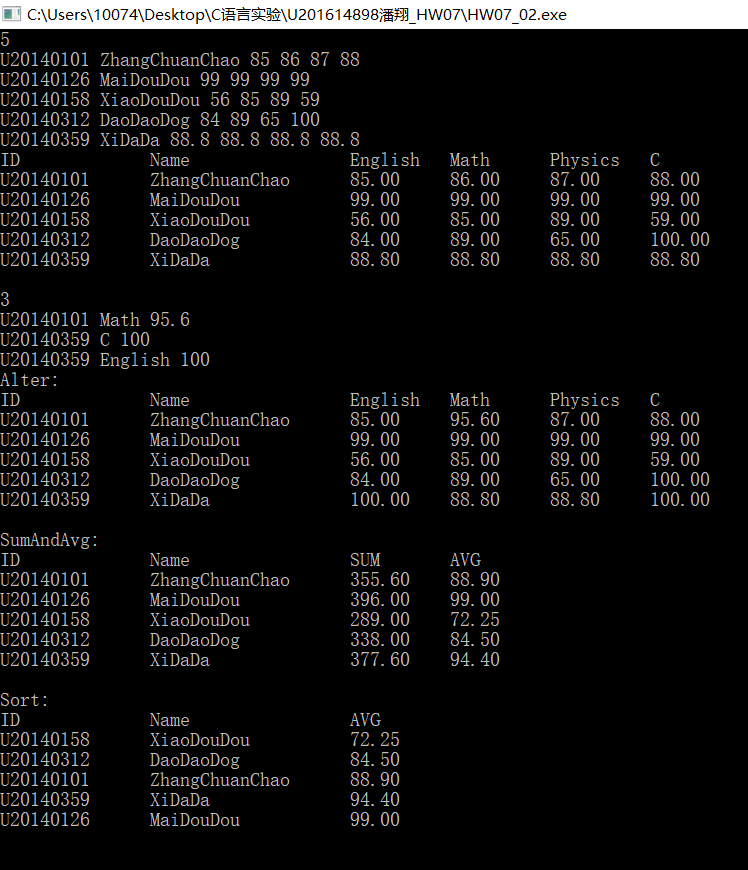
}

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");

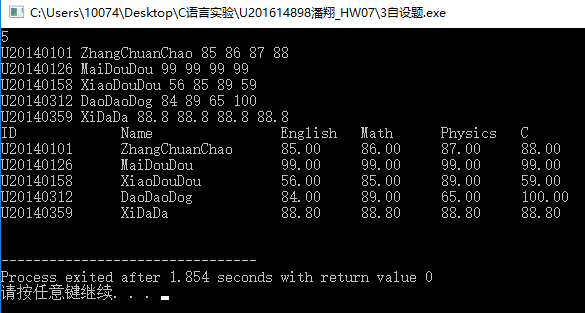
head = head->next;

}

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。