电子科技大学信息与软件工程学院

实 验 报 告

学　　号 20180916180008

姓　　名 袁昊男

（实验）课程名称 程序设计与算法基础 II

理论教师 曹晟

实验教师 文淑华

电 子 科 技 大 学

实　验　报　告

学生姓名：袁昊男　　学号：2018091618008　　指导教师：曹晟

实验地点：基础实验大楼 537　　实验时间：2019. 04. 20

一、 实验名称：学生课程成绩查询程序

二、 实验学时：8 学时

三、 实验目的：

1. 掌握结构体数组的定义和使用方法；
2. 掌握单链表的建立方法；
3. 掌握栈的定义与建立方法；
4. 掌握链式队列的定义与建立方法；
5. 掌握在结构体数组、单链表、栈及队列中对数据的查询、排序及输出；
6. 掌握 C 语言文件读写的一般方法；
7. 掌握 C 语言循环菜单界面的创建方法。

四、 实验原理：

1. **结构体：**结构是可能具有不同类型的值(成员)的集合，结构的元素(在 C 语言中的说法是结构的成员)可能具有不同的类型。而且每个结构成员都有名字，所以在选择特定的结构成员时需要指明结构成员的名字而不是它的位置。

2. **结构变量的声明：**当需要存储相关数据项的集合时，结构是一种合乎逻辑的选择。例如：假设需要记录存储在仓库中的零件，用来存储每种零件的信息可能包括零件的编号(整数)、零件的名称(字符串)以及现有零件的数量。为了产生一个可以存储全部三种数据项的变量，可以使用类似下面这样的声明：结构 part1 和 part2 中的成员 number 和成员 name 不会与结构 employee1 和 employee2 中的成员 number 和成员 name 冲突。

```
1. struct{
2. int number;
3. char name[NAME_LEN+1];
```

```
4. int on_hand;
5. } part1, part2;
```

每个结构变量都有三个成员：number(零件的编号)、name(零件的名称)和on_hand(现有数量)。注意这里的声明格式和 C 语言中其他变量的声明格式一样，struct{...}指明了类型，而 part1 和 part2 则是具有这种类型的变量。结构的成员在内存中是按照声明的顺序存储的。

3. **结构标记：**结构标记是用于标识某种特定结构的名字。下面的例子声明了名为 part 的结构标记：

```
1. struct part{
2. int number;
3. char name[NAME_LEN+1];
4. int on_hand;
5. };
```

一旦创建了标记 part ，就可以用它来声明变量了：

```
1. struct part part1, part2;
```

但是，不能通过漏掉单词 struct 来缩写这个声明，因为 part 不是类型名，如果没有单词 struct 的话，它没有任何意义。

4. **结构类型的定义：**除了声明结构标记，还可以用 typedef 来定义真实的类型名。例如，可以按照如下方式定义名为 Part 的类型：

```
1. typedef struct {
2. int number;
3. char name[NAME_LEN+1];
4. int on_hand;
5. } Part;
```

注意，类型 Part 的名字必须出现在定义的末尾，而不是在单词 struct 后。可以像内置类型那样使用 Part。

5. **结构体数组的基本操作函数：**

a) InputStruct (Struct s[], int n, FILE *fp)：将结构体数组 s 中的 n 个信息写入文件中；

b) OutputStruct (Struct s[], FILE *fp, int n)：将文件中的 n 个信息读入结构体数组 s 中；

c) PrintStruct (Struct s[], int n)：将结构体数组 s 中的 n 个信息输出到屏幕上；

d) SortStruct (Struct s[], int n)：将结构体数组 s 中一成员作为关键字进行冒泡排序算法进行升序或降序排序，排序结果仍存于结构体数组 s 中。

6. **链表存储结构：**在链式存储结构中，存储数据结构的存储空间可以不连续，各数据结点的存储顺序与数据元素之间的逻辑关系可以不一致，而数据元素之间的逻辑关系则是由指针域来确定的。链式存储方式既可以用于表示线性结构，也可用于表示非线性结构。一般来说，在线性表的链式存储结构中，各数据结点的存储符号是不连续的，并且各结点在存储空间中的位置关系与逻辑关系也不一致。对于线性链表，可以从头指针开始，沿各结点的指针遍历链表中的所有结点。

7. **链表结点的类型及定义：**链表结点的类型应该包括存储元素的数据域，一般用 data 表示，它的类型可以是前期定义的元素结构体；以及包括存储后续结点位置的指针域，一般用 next 表示，最后一个结点的 next 指针域为空(NULL)。例如下面定义一个学生信息的链表结点：

```
typedef struct student_node
{
    int sno;
    char sname[10];
    char sex[5];
    char major[20];
    struct student_node *next;
}student_node, StudentList, *SPtr;
typedef SPtr *StudentListPtr;
```

8. **单链表的头结点：**线性链表的链接存储中，为了方便在表头插入和删除结点的操作，经常在表头结点(存储第一个元素的结点)的前面增加一个结点，称之为头结点或表头附加结点。这样原来的表头指针由指向第一个元素的结点改为指向头结点，头结点的数据域为空，头结点的指针域指向第一个元素的结点。带头结点的单链表更易于修改。

9. **单链表的基本操作函数：**

   a) InitList(ListPtr L)：链表初始化，构造一个空的链表 L，并为 L 头结点分配空间；

   b) DestroyList(ListPtr L)：销毁链表 L，需要在链表 L 存在的情况下执行，主要工作为释放 L 每个结点的空间；

   c) CreateList(ListPtr L, Struct s[], int n)：将结构体数组 s 中的 n 个信息作为结点插入单链表中；

   d) PrintList(ListPtr L, int n)：将单链表 L 中的 n 个结点信息输出到屏幕上；

10. **顺序栈储存结构：**栈的逻辑结构与线性表相同，其特点在于运算受到了限制：只能在线性表的一端进行插入和删除操作，具有“后进先出(LIFO)”的特点。进行操作的一端为栈顶，用一个“栈顶指针”指示，其位置经常发生变化；而另一端是固定端，称为栈底。当栈中没有元素时，称为空栈。向栈中插入元素的操作称为入栈，从栈中删除元素的操作称为出栈。

11. **顺序栈的类型描述：**顺序栈中域 entry 用于存放数据元素。约定 top 为存放栈顶元素的位置，若 top=−1，表示该栈为空栈；若 top=MAXSIZE−1，表示栈满。例如，名为 Stack 的顺序栈类型描述如下：

```
#define MAXSIZE 100
typedef struct stack
{
    int top;
    Student A[MAXSIZE];
} Stack, *StackPtr;
```

12. **顺序栈的基本操作函数：**

    a) InitStack(Stack *s)：初始化栈 s；

    b) PushSatck(StackPtr s, ListPtr L, int n)：将单链表 L 中的 n 个结点信息压入栈 s 中；

    c) PopStack(StackPtr s, Struct t[], int n)：将栈 s 中 n 个元素弹栈，并将其存入结构体数组 s 中。

13. **链式队列存储结构：**队列也是一种操作受限的线性表，与栈“后进先出”的特性不同，队列具有“先进先出(FIFO)”的特性。在队列中，元素的插入操作都在表的一端进行，而删除操作在表的另一端进行。我们称允许插入的一端为队尾，允许删除的一端为队头，分别用队尾指针和队头指针指示。通常使用单链表来实现链式队列，但只有单链表的头指针不方便在表尾进行操作。为了操作方便，需要增加一个尾指针，指向链表的最后一个结点，于是一个链式队列就由一个头指针和一个尾指针唯一确定。

14. **链式队列的类型描述：**在实现链式队列存储结构时，我们将头指针和尾指针封装在同一个结构中。例如，名为 queue 的链式队列及其结点的类型描述如下：

```
#define MAXSIZE 100
typedef struct queue_node
{
    Connect Queue[MAXSIZE];
    struct queue_node *next;
} queue_node, *QueueNodePtr;
```

```
7.
8. typedef struct queue
9. {
10.     queue_node *front;
11.     queue_node *rear;
12.} queue, *QueuePtr;
```

15. **链式队列的基本操作函数：**

a) InitQueue(QueuePtr q)：初始化链式队列 q；

b) EnQueue(QueuePtr q, Struct s[], int n)：将结构体数组 s 中的 n 个元素入队；

c) DeQueue(QueuePtr q, Struct s[], int n)：将链式队列 q 中的 n 个元素出队，并将其存入结构体数组 s 中；

d) EmptyQueue(QueuePtr q)：链式队列 q 的判空。若 q 为空，返回 1；否则返回 0。

16. **系统选择界面：**可以通过一个 while 条件为 1 的循环来完成程序功能的循环调用，直至输入退出系统的命令。例如：

```
1. while(1)
2. {
3. switch(choice) {
4. case 1: Fucntion1;
5. break; ...
6. } }
```

17. **文件操作示例：**

```
1. char tempstring[30];
2. FILE *fp;           //定义文件操作的指针
3. //fopen 用于打开文件，接收两个参数，一个是文件的路径，另一个是文件打开的方式。例如 xxxxxx.txt 和该项目的可执行文件在同一目录下，则此处只需要所读取内容的文件名；r 代表以只读方式打开文件
4.     fp = fopen("xxxxxxx.txt", "r");
5.     //如果以 w 方式代表打开只写文件，若文件存在则长度清为 0，即该文件内容消失；若不存在则创建该文件，其余打开方式请自行查阅文档
6.     fp = fopen("xxxxxxx.txt", "w");
7.     //fscanf 用于从 txt 文件中读取内容，下例以字符串形式读入一段字符并存放在 tempstring 中
8.     fscanf(fp, "%s", tempstring);
9.     //或者以格式化的方式读入字符串
10.    fscanf(fp, "\t%s\n", tempstring);
11.    //fprintf 以格式化的方式向 txt 文件中写入内容
12.    fprintf(fp, "%s\t", tempstring);
13.   //检查文件内容是否已经读到文件结束符了
14.     while (!feof(fp)){……}
15.   //最后需要使用 fclose 关闭文件指针
```

```
16.fclose(fp);
```

五、 实验内容：

设有学生信息文件 student.dat，每个学生记录包括：学号 sno、姓名 sname、性别 sex、专业 major；课程信息文件 course.dat，课程记录包括：课程号 cno、课程名称 cname、课时数 classHours；课程成绩信息文件 courseGrade.dat，成绩记录包括学号 sno、课程号 cno、考试成绩 score；自学教材 61-64 页线性表的应用，设计应用程序完成如下功能：

1. 设计程序运行的功能菜单：

图 5.1.1 功能菜单界面

2. 输入 10 个学生记录，其中软件技术专业 5 人，人工智能专业 5 人，并存入文件 student.dat 中；

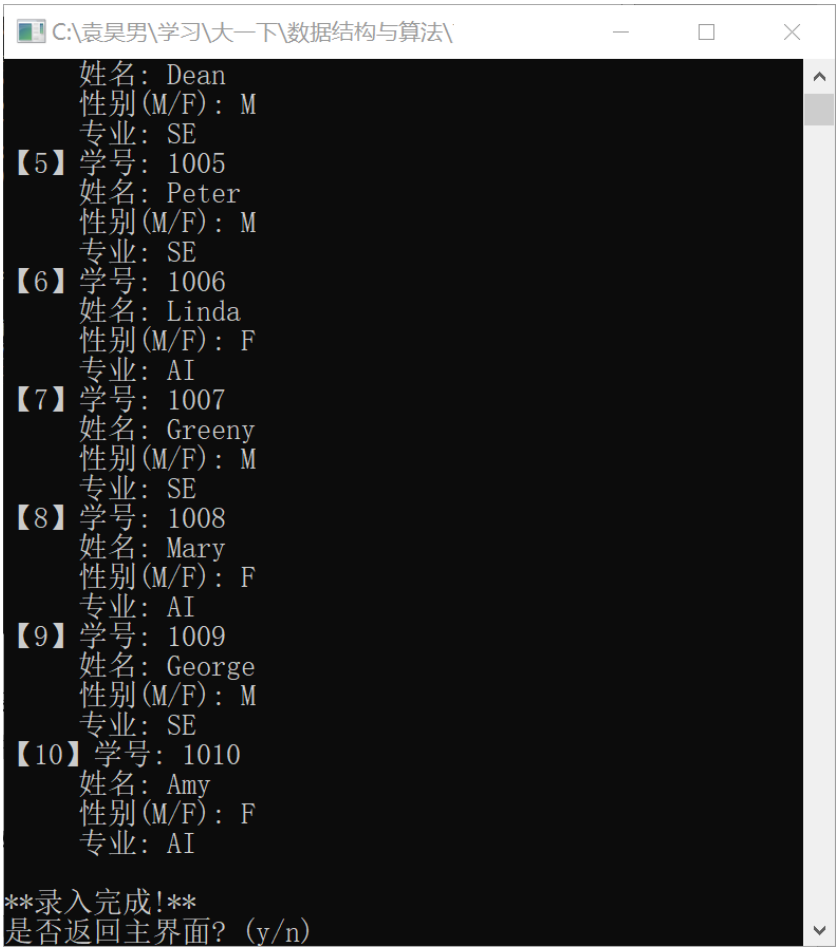

图 5.2.1 录入学生信息

图 5.2.2 生成的 student.dat 文件

3. 输入 3 门课程(数据库、数据结构、程序设计)信息记录，并存入文件 course.dat 中；

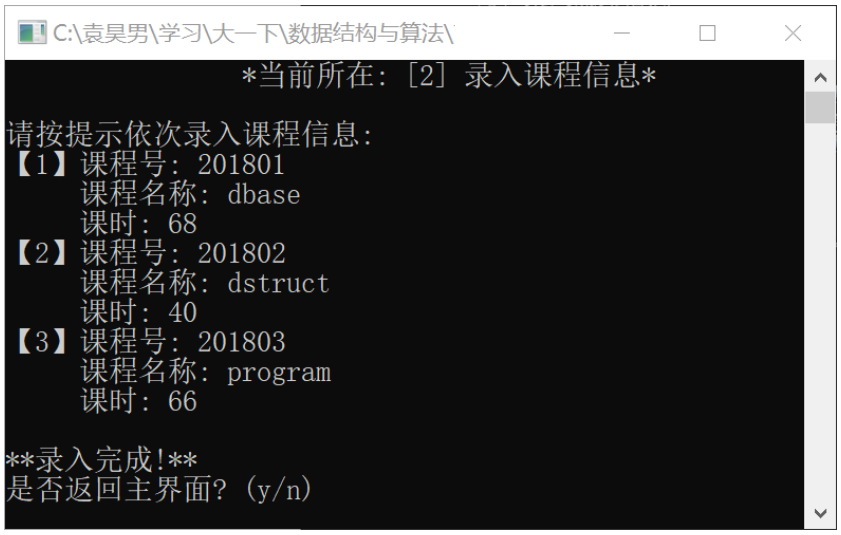

图 5.3.1 录入课程信息

图 5.3.2 生成的 course.dat 文件

4. 输入上述 10 位同学分别选修上述三门课程的考试成绩到文件 courseGrade.dat 中；

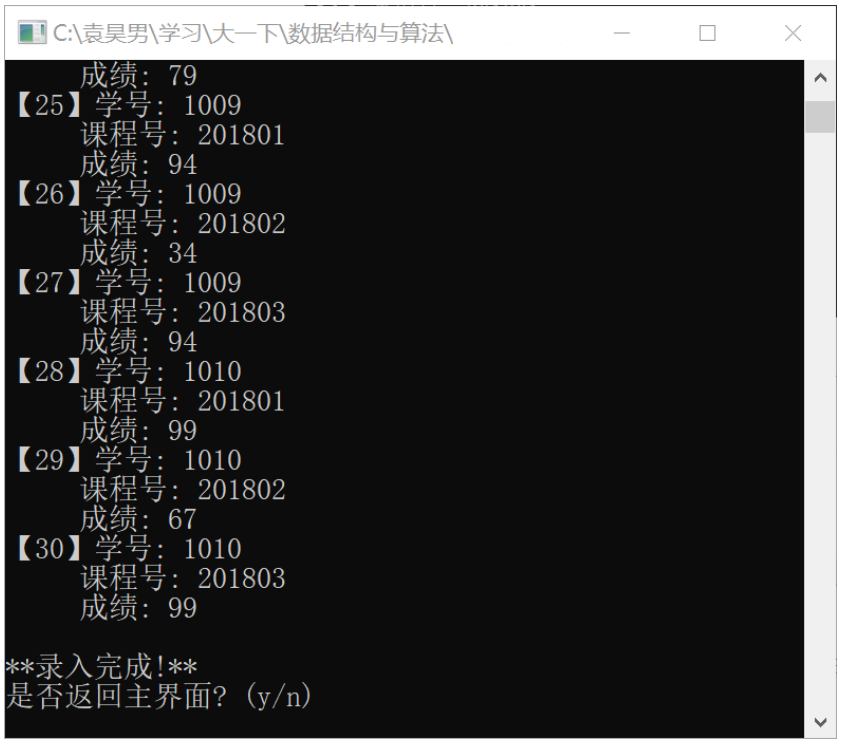

图 5.4.1 录入成绩信息

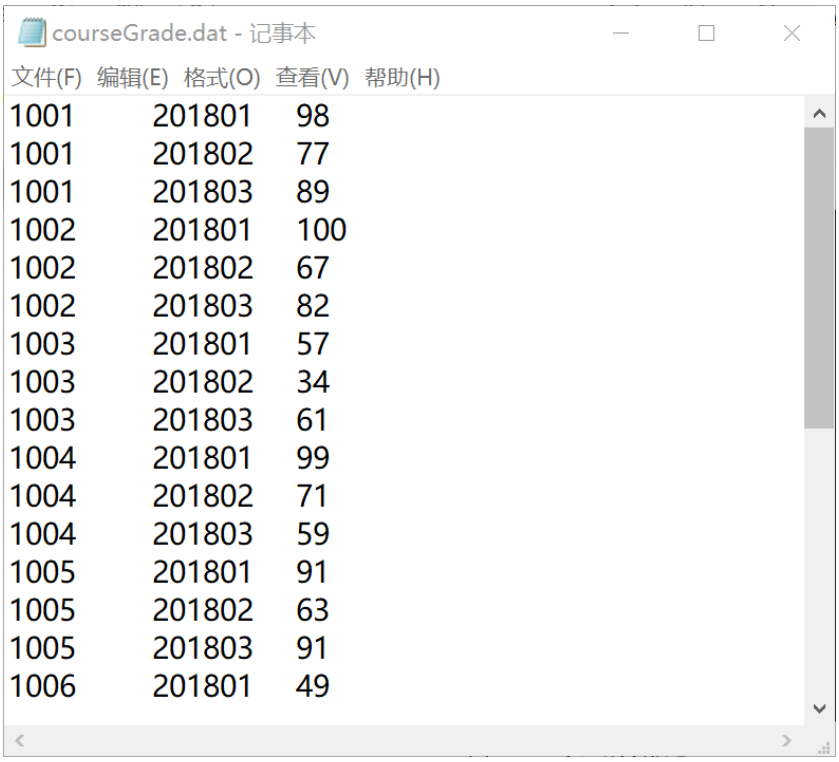

图 5.4.2 生成的 courseGrade.dat 文件

5. 从文件 student.dat 中读出学生信息，生成按照学号升序排列的单向链表，并在屏幕上显示输出；

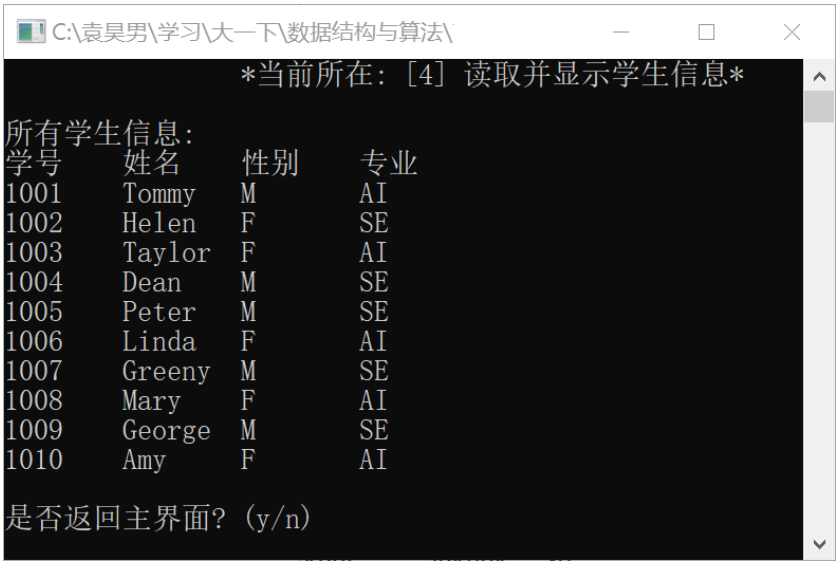

图 5.5.1 从文件读取学生信息并按学号升序输出

6. 从文件 course.dat 中读出课程信息，生成按照课程号升序排列的单向链表，并在屏幕上显示输出；

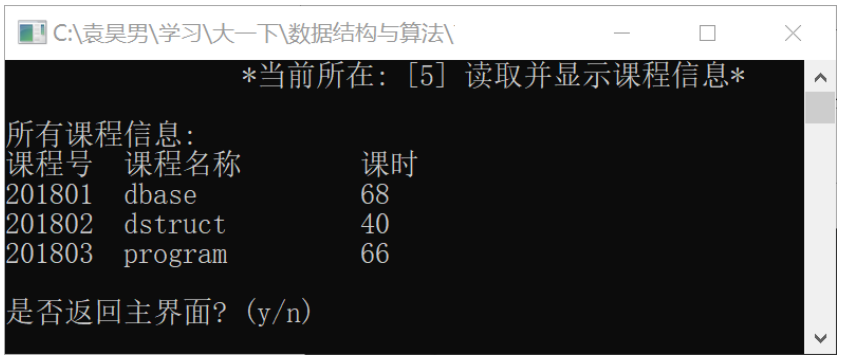

图 5.6.1 从文件读取课程信息并按课程号升序输出

7. 从文件 courseGrade.dat 中读出成绩信息，生成按照学号和课程号升序排列的单链表，并在屏幕上显示输出；

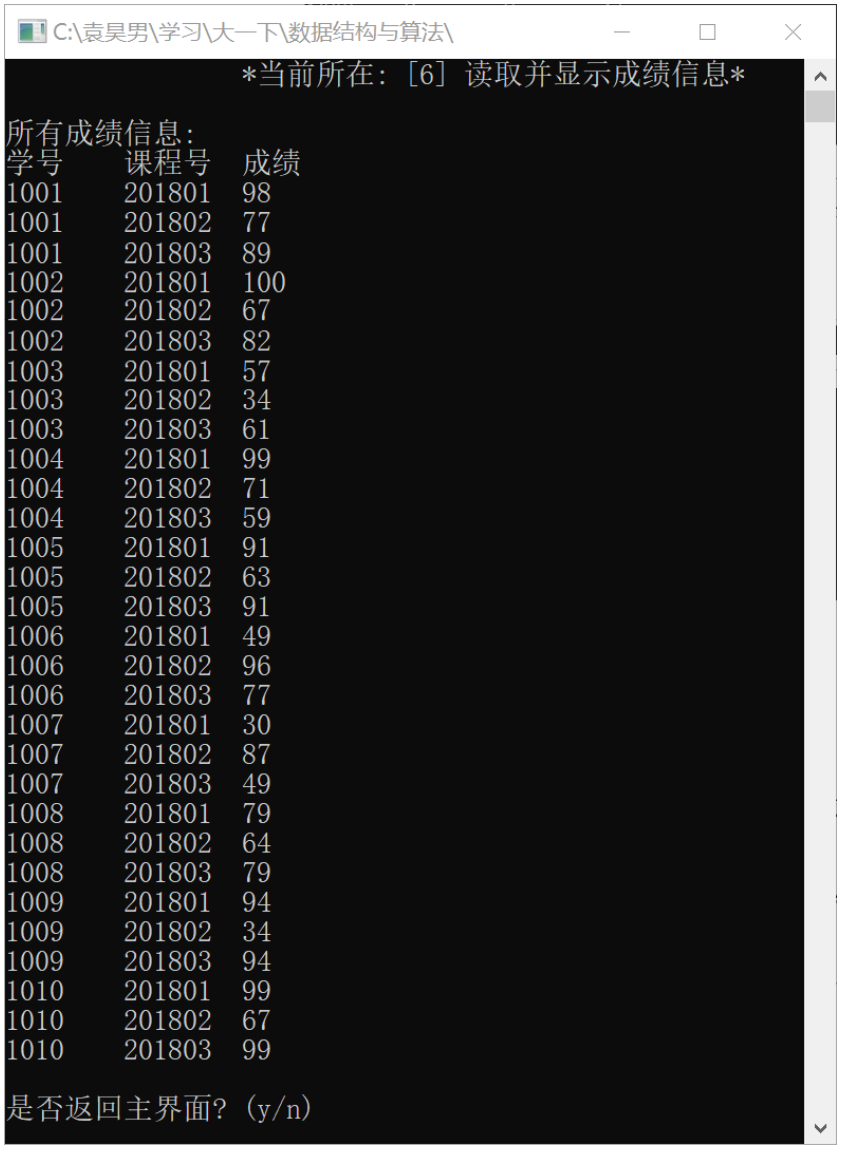

图 5.7.1 从文件读取成绩信息排序后输出

8. 查询所有学生所有课程的考试成绩，生成该课程的成绩单链表，要求包括学号、学生姓名、专业、课程名、考试成绩等信息，按照考试成绩降序排

列，并将学生的该成绩信息输出到文件 studentGrade.dat 中，同时在屏幕上显示输出；

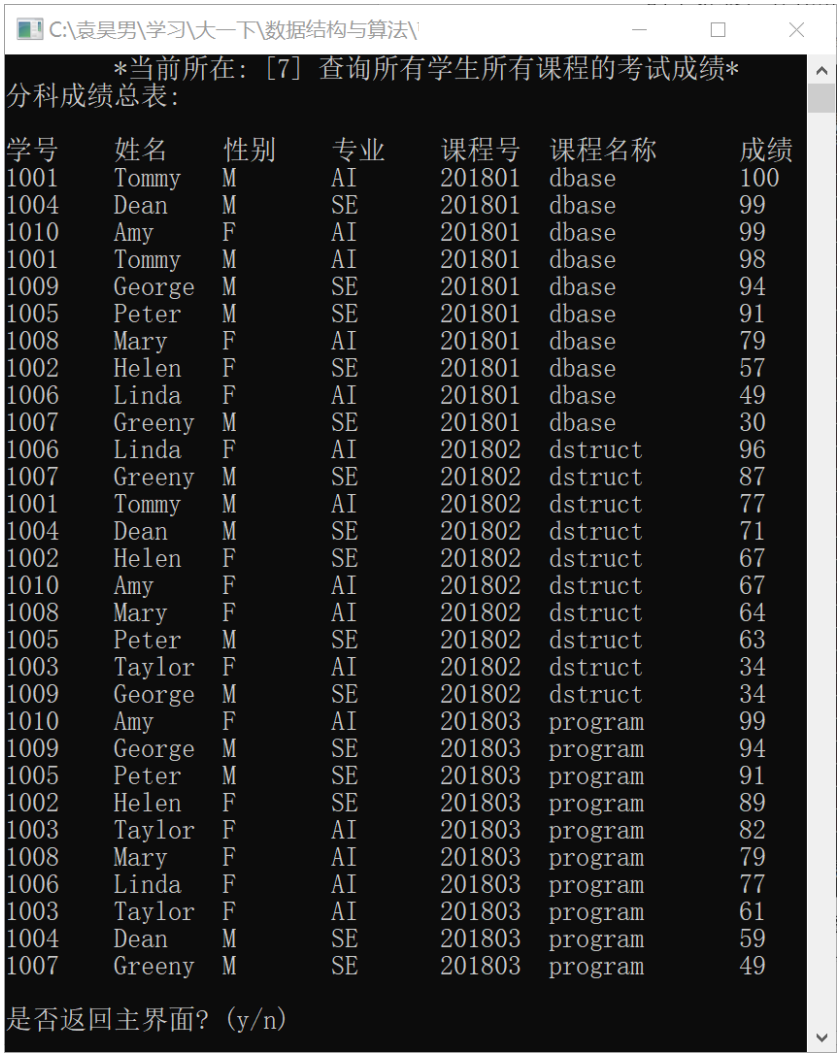

图 5.8.1 查询所有学生所有课程成绩排序后输出

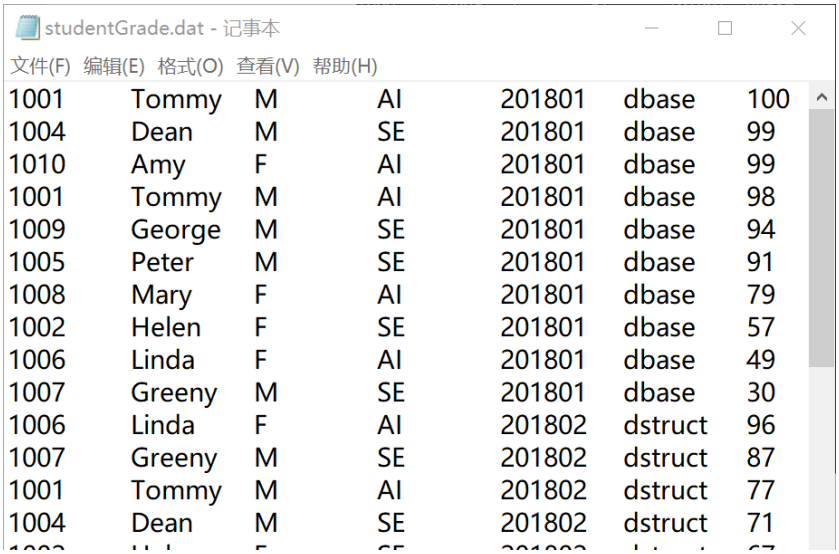

图 5.8.2 生成的 studentGrade.dat 文件

9. 在(7)的链表中，查询指定课程号的所有学生的考试成绩，筛选其中考试成绩小于 60 分的学生信息，生成该课程的成绩单链表，要求包括学号、学生姓名、专业、课程名、考试成绩等信息，按照考试成绩降序排列输出到屏幕上显示；

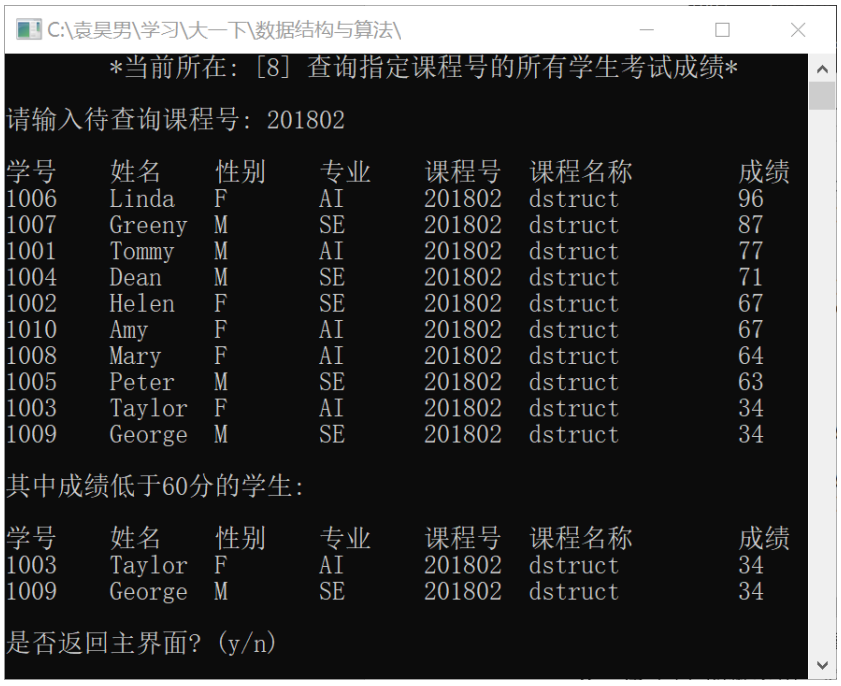

图 5.9.1 查询指定课程号的所有学生成绩并筛选不及格学生信息

10. 在(7)的链表中，查询指定学号的所有课程的考试成绩，筛选其中考试成绩小于 60 分的学生信息，生成该学生的成绩单链表，要求包括学号、学生姓名、专业、课程名、考试成绩等信息，按照考试成绩降序排列输出到屏幕上显示；

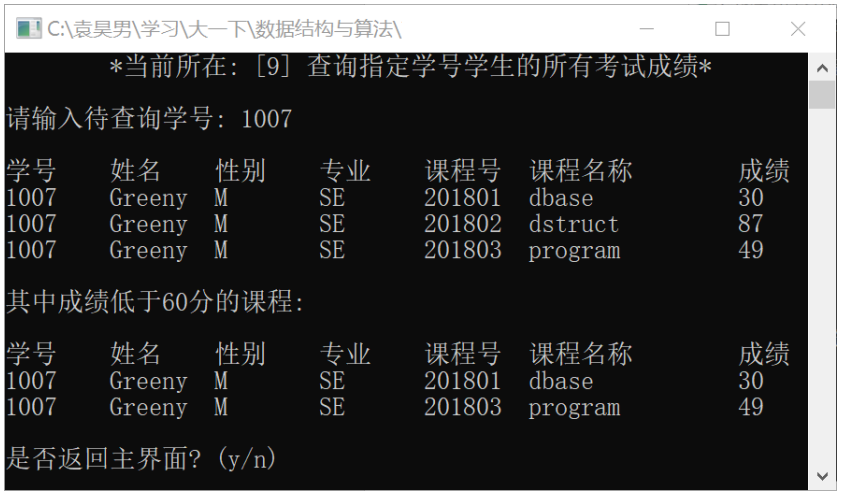

图 5.10.1 查询指定学号的所有课程成绩并筛选不及格学生信息

11. 使用栈实现将(4)的单链表中的学生信息逆序生存新的链表；

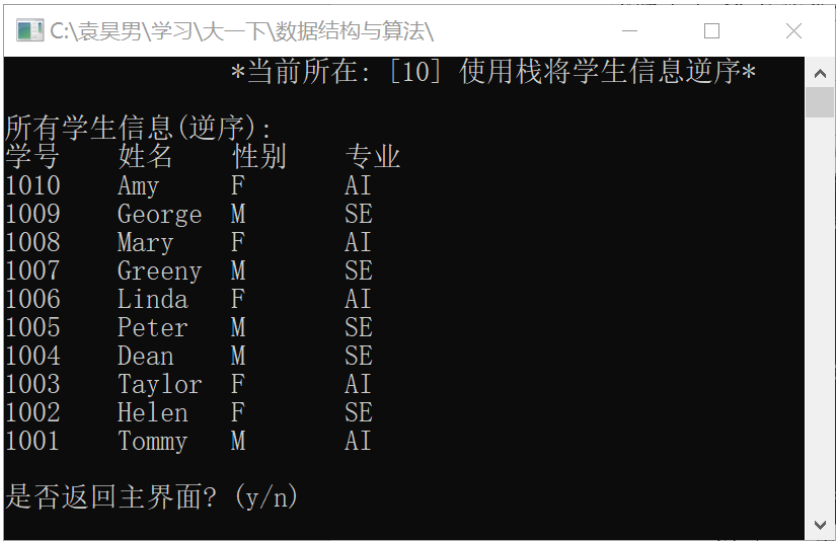

图 5.11.1 采用栈将学生信息逆序

12. 设计使用链式队列完成问题(7)的要求。

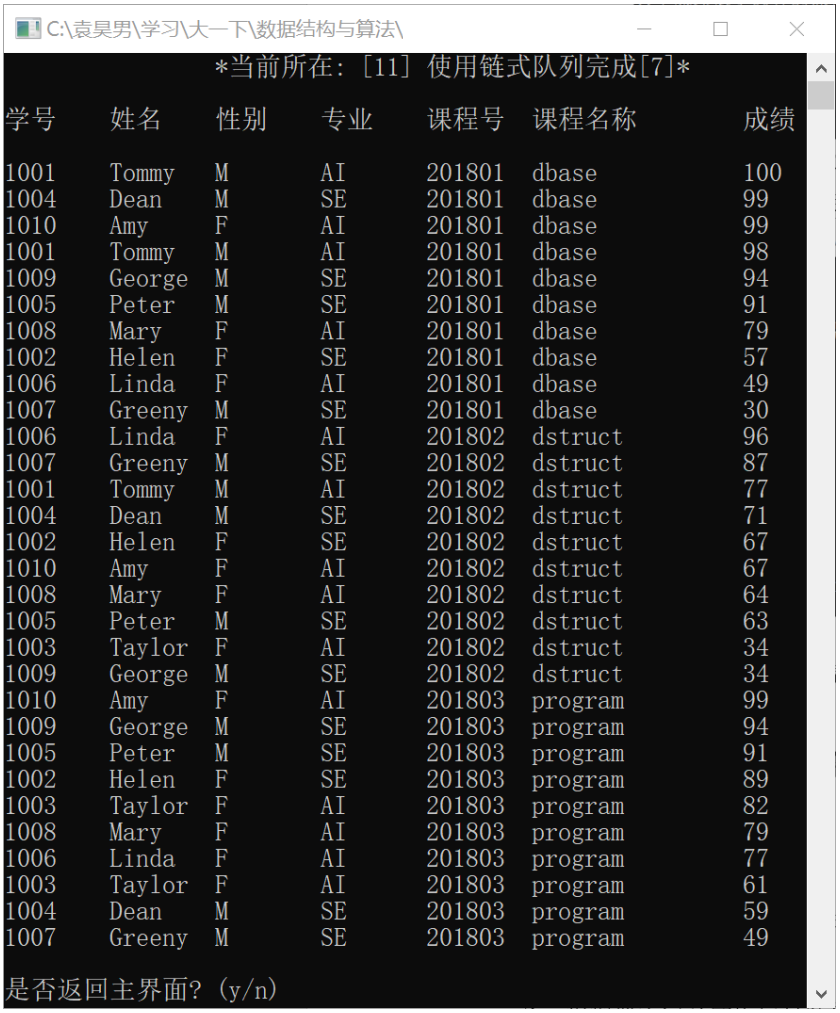

图 5.12.1 使用链式队列完成问题(7)的要求

六、 实验器材(设备、元器件):

个人电脑一台。

七、 实验步骤：

1. 分析所要编写的学生课程成绩查询程序的各功能需求，确定其基本框架和编写思路；
2. 根据要求定义学生信息结构体 Student、课程信息结构体 Course、成绩信息结构体 Grade、联合信息结构体 Connect；定义学生信息链表 StudentList、课程信息链表 CourseList、成绩信息链表 GrandeList、联合信息链表 ConnectList；定义顺序栈 Stack；定义链式队列 Queue；
3. 编写创建文件函数 Create_File；读入信息至文件函数 Input_Student、Input_Course、Input_Grade、Input_All，从文件读入信息函数 Output_Student、Output_Course、Output_Grade，结构体冒泡排序函数 SortS、SortC、SortG、Sort_All；链表创建函数 SList_Create、CList_Create、GList_Create、AList_Create；联合所有信息函数 Connect_All；查询函数 Inquiry_Cno、Inquiry_Sno；栈初始化函数 Stack_Init，压栈函数 Stack_Push，弹栈函数 Stack_Pop；链式队列初始化函数 Queue_Init，入队函数 Queue_Enqueue，队列判空函数 Queue_Empty；以及在各种存储结构中信息输出函数 Print；
4. 调试软件，修改错误，完善功能以达到实验目的要求；
5. 测试软件各功能是否正常；
6. 实验结果分析。

八、 实验结果与分析（含重要数据结果分析或核心代码流程分析）

1. 录入学生、课程、成绩信息至文件【Input 系列函数】

```c
void Input_Student(Student s[], int n, FILE *fp)
{
    int i;
    printf("请按提示依次录入学生信息：\n");
    for (i = 0; i < n; i++)
    {
        printf("【%d】学号：", i + 1);
        scanf("%d", &s[i].sno);
        printf("    姓名：");
        scanf("%s", &s[i].sname);
        printf("    性别(M/F)：");
        scanf("%s", &s[i].sex);
        printf("    专业：");
        scanf("%s", &s[i].major);
        fflush(stdin);
    }
    printf("\n");
```

```
    printf("**录入完成!**\n");
    fp = fopen("student.dat", "w");
    if (fp == NULL)
    {
        printf("Fail to open file!");
        exit(0);
    }
    for (i = 0; i < n; i++)
    {
        fprintf(fp, "%d\t", s[i].sno);
        fprintf(fp, "%s\t", s[i].sname);
        fprintf(fp, "%s\t", s[i].sex);
        fprintf(fp, "%s\t", s[i].major);
        if(i != n-1)
            fprintf(fp, "\n");
    }
    fclose(fp);
}
```

代码分析：以录入学生信息函数为例进行分析，录入课程及成绩信息函数此处省略。本段代码编写的 Input_Student 函数实现的功能是录入学生信息至文件 student.dat。首先使用 for 语句循环从键盘读入信息至结构体数组 Student s[]中，使用 for 语句从键盘循环读入信息时要注意在每一次循环读入结束后加上 fflush(stdin)语句清空输入缓冲区，以确保不影响后面的数据读取。完成读取后采用 C 语言的文件读写操作新建空白文件 student.dat 并将结构体数组里暂存的学生信息以 fprintf 函数格式化输出至文件，注意最后加上 fclose(fp)关闭文件指针。

2. 结构体元素排序【Sort 系列函数】

```
void SortS(Student s[], int n)
{
    Student t[10];
    int i, j;
    for (i = 0; i < n - 1; i++)
        for (j = 0; j < n - 1 - i; j++)
        {
            if (s[j].sno > s[j + 1].sno)
            {
             memcpy(&t[0], &s[j], sizeof(Student));
             memcpy(&s[j], &s[j + 1], sizeof(Student));
             memcpy(&s[j + 1], &t[0], sizeof(Student));
            }
        }
}
```

代码分析：以对学生学号为关键字冒泡排序为例进行分析，其他关键字排

序此处省略。本段代码编写的 SortS 函数实现的功能是对结构体数组 Student s[]的成员 sno 进行冒泡升序排列。冒泡排序采用两个嵌套的 for 循环进行数据之间的两两比较，并借助另一结构体数组 Student t[10]对结构体成员 sno 升序排序。在关键字进行两两比较时注意不能直接套用交换两个 int 型变量的“三行式”算法，因为在这里 sno 是结构体的成员，对结构体的成员两两交换需要使用 memcpy 函数实现。

3. 从文件读入学生、课程、成绩信息【Output 系列函数】

```
1. void Output_Student(Student s[], FILE *fp, int n)
2. {
3.     int i;
4.     if ((fp = fopen("student.dat", "r")) == NULL)
5.     {
6.         printf("Fail to open file!");
7.         exit(0);
8.     }
9.     for (i = 0; i < n; i++)
10.    {
11.        fscanf(fp, "%d", &s[i].sno);
12.        fscanf(fp, "%s", s[i].sname);
13.        fscanf(fp, "%s", s[i].sex);
14.        fscanf(fp, "%s", &s[i].major);
15.    }
16.    SortS(s, n);
17.}
```

代码分析：以读入学生信息函数为例进行分析，读入课程及成绩信息函数此处省略。本段代码编写的 Output_Student 函数实现的功能是从文件 student.dat 读入学生信息至结构体数组 Student s[]，并调用 SortS 函数对其关键字 sno 进行升序排序。采用 C 语言的文件读写操作 fscanf 格式化读入信息。

4. 从结构体数组建立链表【List_Create 系列函数】

```
1. void SList_Create(StudentListPtr L, Student s[], int n)
2. {
3.     SPtr p;
4.     if (*L == NULL)
5.     {
6.         p = (SPtr)malloc(sizeof(student_node));
7.         if (p == NULL)
8.             return;
9.         *L = p;
10.        (*L)->next = NULL;
11.    }
```

```
    int i;
    for (i = n - 1; i >= 0; i--)
    {
        SPtr q;
        q = (SPtr)malloc(sizeof(student_node));
        q->next = NULL;
        q->sno = s[i].sno;
        strcpy(q->sname, s[i].sname);
        strcpy(q->sex, s[i].sex);
        strcpy(q->major, s[i].major);
        if ((*L)->next == NULL)
            (*L)->next = q;
        else
        {
            SPtr t;
            t = (SPtr)malloc(sizeof(student_node));
            t = (*L)->next;
            (*L)->next = q;
            q->next = t;
        }
    }
}
```

代码分析：以建立学生信息链表函数为例进行分析，课程、成绩及联合信息链表建立函数此处省略。本段代码编写的 SList_Create 函数实现的功能是从结构体数组 Student s[]中以头插法建立学生信息链表。当链表头结点指针所指内容不为空时，初始化链表头结点。因为采用头插法插入元素，故建立的链表中的元素顺序与原序列中的元素顺序相反。因此从结构体数组 s 的最后一个元素开始插入到链表中即可实现链表中的元素顺序与原序列中的元素顺序一致。另外，在插入第一个元素结点时需要单独处理。

5. 查询所有学生所有课程考试成绩【Connect_All 函数】

```
void Connect_All(Student s[], Course c[], Grade g[], Connect A[], int n1, int n2, int n3)
{
    int i = 0, j = 0, k = 0;
    for (i = 0; i < n3; i = i + n2)
    {
        for (j = 0; j < n2; j++)
        {
            A[i + j].sno = s[k].sno;
            strcpy(A[i + j].sname, s[k].sname);
            strcpy(A[i + j].sex, s[k].sex);
            strcpy(A[i + j].major, s[k].major);
        }
        k++;
```

```
    }
    for (i = 0; i < n1; i++)
    {
        for (j = 0; j < n3; j++)
        {
            if (s[i].sno == g[j].sno)
            {
                k = 0;
                A[j].cno = g[j].cno;
                A[j].score = g[j].score;
                ++k;
                if (k == n2)
                    break;
            }
        }
    }
    for (i = 0; i < n3; i++)
    {
        for (j = 0; j < n2; j++)
        {
            if (A[i].cno == c[j].cno)
                strcpy(A[i].cname, c[j].cname);
        }
    }
}
```

代码分析：本段代码编写的 Connect_All 函数实现的功能是从结构体数组 Student s[]、Course c[]和 Grade g[]中将每一个学生每一门课程的信息串并起来保存至联合信息结构体 Connect A[]中。形式参数 n1 指学生人数 N，n2 指课程数 course，n3 指串并信息条数，即 n3=N*course。因为每一个学生有 n2 门课程的成绩信息，因此前两个嵌套的 for 循环从 s[]中调用 strcpy 函数将每一个学生的信息复制至 A[]中并重复 n2 次；其次的两个嵌套的 for 循环根据 sno 为关键字在 g[]中遍历查询对应的课程号及该课程对应的成绩，并将其复制到 A[]中对应的位置；最后的两个嵌套的 for 循环根据 cno 为关键字在 c[]中查询对应课程号的 cname，并调用 strcpy 函数将课程名称复制到 A[]中对应的位置。至此，结构体 A[]中就保存了每一个学生的学号、姓名、性别、专业、课程号、课程名称及课程成绩，实现了查询所有学生的所有考试成绩的功能。

6. 查询不及格学生信息【Inquiry 系列函数】

```
int Inquiry_Cno(Connect A[], Connect ICno[], int n1, int
cno, int n2, Connect ICno1[])
{
    int i, j = 0, k = 0;
```

```
    for (i = 0; i < n1; i++)
    {
        if (A[i].cno == cno)
        {
            memcpy(&ICno[j], &A[i], sizeof(Connect));
            j++;
        }
    }
    for (i = 0; i < n2; i++)
    {
        if (ICno[i].score < 60)
        {
            memcpy(&ICno1[k], &ICno[i], sizeof(Connect));

            k++;
        }
    }
    return k;
}
```

代码分析：以查询指定课程号不及格学生为例进行分析，查询指定学生不及格课程号此处省略。本段代码编写的 Inquiry_Cno 函数实现的功能是查询指定课程号 cno 不及格学生，并返回不及格人数 k。第一个 for 循环遍历联合信息结构体 A[]，将指定课程号的所有学生信息复制到新的联合信息结构体 ICno[]中；第二个 for 循环遍历 ICno[]，若结构体成员 score 的值小于 60，则将该条学生联合信息复制到新的联合信息结构体 ICno1[]中，并令 k 自增 1 计数未及格学生人数。

7. 栈的相关操作【Stack 系列函数】

```
void Stack_Init(Stack *s)
{
    s->top = -1;
}

void Stack_Push(StackPtr s, StudentListPtr L, int n)
{
    SPtr p = NULL;
    p = p = (SPtr)malloc(sizeof(student_node));
    p = (*L)->next;
    int i = 0;
    for (i = 0; i < n; i++)
    {
        s->top++;
        s->sreverse[s->top].sno = p->sno;
        strcpy(s->sreverse[s->top].sname, p->sname);
        strcpy(s->sreverse[s->top].sex, p->sex);
```

```
        strcpy(s->sreverse[s->top].major, p->major);
        p = p->next;
    }
    ++s->top;
}

void Stack_Pop(StackPtr s, Student rs[], int n)
{
    int i = 0;
    for (i = 0; i < n; i++)
    {
        --s->top;
        rs[i].sno = s->sreverse[s->top].sno;
        strcpy(rs[i].sname, s->sreverse[s->top].sname);
        strcpy(rs[i].sex, s->sreverse[s->top].sex);
        strcpy(rs[i].major, s->sreverse[s->top].major);
    }
}
```

代码分析：本段代码编写的三个链栈的函数功能是使用栈将学生信息逆序。Stack_Init 函数对链栈 s 进行初始化：将 top 值置为-1 指示栈底；Stack_Push 函数将 StudentList 中的元素压入栈中，首先定义的 p 指针指向链表中的第一个学生信息，使用 for 循环遍历链表，依次将元素存入链栈的每一个结构体数组单元 sreverse[]中，并通过成员 top 值的增加来指示结构体数组的下标；Stack_Pop 函数将链栈 s 中的元素弹栈至结构体数组 Student rs[]中，使用 for 循环遍历链栈中的元素，通过成员 top 值的减少来指示结构体数组的下标。

8. 链式队列的相关操作【Queue 系列函数】

```
void Queue_Init(QueuePtr *q)
{
    *q = (QueuePtr)malloc(sizeof(queue));
    QueueNodePtr ptr = (QueueNodePtr)malloc(sizeof(queue_node));
    if (ptr)
    {
        (*q)->front = (*q)->rear = ptr;
        ptr->next = NULL;
    }
}

bool Queue_Empty(QueuePtr q)
    return(q->front->next == NULL && q->rear->next == NULL);

void Queue_EnQueue(QueuePtr q, Connect A[], int n)
```

```
{
    int i;
    for (i = n - 1; i >= 0; i--)
    {
        queue_node *p = (queue_node*)malloc(sizeof(queue_node));
        memcpy(p->Queue, &A[i], sizeof(Connect));
        p->next = NULL;
        if (Queue_Empty(q))
        {
            q->front->next = p;
            q->rear = p;
        }
        else
        {
            q->front->next = p;
            p->next = q->rear;
            q->rear = p;
        }
    }
}
```

代码分析：本段代码编写的三个链式队列的函数功能是使用链式队列查询所有学生所有课程的考试成绩。Queue_Init 函数对链式队列初始化，头指针 front 和尾指针 rear 指向头结点 ptr，ptr 指针置空；Queue_Empty 函数判断队列是否为空，若空则返回 1，不空则返回 0；Queue_EnQueue 函数将联合信息结构体 A[]中的元素入队，因为采用头插法插入元素，故队列中的元素顺序与原序列中的元素顺序相反。因此从结构体数组 A 的最后一个元素开始插入到新增结点中即可实现队列中的元素顺序与原序列中的元素顺序一致。另外，在插入第一个元素结点时需要单独处理。

9. 循环菜单界面

```
int opt = 1;
char yn;
while (opt != 12)
    {system("cls");
        printf("欢迎使用学生成绩查询系统!\n");
                //菜单内容省略
        printf("请输入您的选择: ");
        scanf("%d", &opt);
        switch (opt)
        {
        case 1://省略 break;
        case 2://省略 break;
        ……
        }
```

```
15.     }
```

代码分析：本段代码功能是实现学生课程成绩查询程序的循环菜单界面，采用 while 循环，根据用户输入的 opt 不同从而进入不同的功能，并在每个功能结束后回到菜单界面。

九、 总结及心得体会：

本次实验所编写的学生课程查询程序考察了对数据结构中数组、结构体、链表、栈、队列的理解及应用；考察了对简单排序算法的应用及 C 语言文件读写的一般方法，既巩固了 C 语言的基本语法及操作，又对本学期数据结构的相关知识进行实际应用与操作。通过本次实验，我掌握了有关知识及其应用，熟悉了在 Visual Studio 环境下编写代码、测试代码的方法及编写程序的一般步骤，掌握了对代码进行理论分析的能力。

十、 对本实验过程及方法、手段的改进建议：

本次实验内容基本覆盖到了数据结构线性存储结构中的大部分内容，能较好地考察学生对于该部分内容的掌握情况。美中不足的是，部分要求过于重复累赘，以及对于栈和队列的要求显得有些生硬和多余。

改进建议：根据实际情况更合理考虑每种数据结构需完成的功能要求，例如采用结构体数组或链表读写学生、课程、成绩信息，采用链栈查询所有学生所有课程的考试成绩并排序，采用队列查询指定课程号的不及格学生信息等，既全面地考察对相关知识的掌握情况，又减少了重复操作，提高了实验效率。

报告评分：

指导教师签字：