

PA1 实验报告

——运用线性表实现学生选课系统

姓名：刘博洋 学号：2153538 日期：2022 年 10 月 19 日

1. 涉及数据结构和相关背景

本次实验主要涉及的数据结构为线性表的两种类型：顺序表和链表。这两种数据结构是相对最好理解和使用最为广泛的，是数据结构的基础知识。顺序表和链表二者各有优劣，本次实验通过实现简易的学生选课系统对两种数据类型进行分析和比较。

线性表的抽象数据类型定义

ADT List{

数据对象： $D=\{a_i | a_i \in D, i=1, 2, \dots, n, n \geq 0\}$

数据关系： $R=\{\langle a_{i-1}, a_i \rangle | a_{i-1}, a_i \in D, i=2, 3, \dots, n\}$

基本操作：

- 1) InitList(&L) : 初始化, 构造一个空的线性表 L。
- 2) ListLength(L) : 求长度, 返回线性表中元素个数。
- 3) GetElem(L, i, &e) : 取 L 中第 i 个数据元素值赋给 e。
- 4) LocateElem(L, e, compare()) : 根据数据关系 compare() 查找, 返回 L 中第 1 个与 e 满足关系 compare() 的数据元素的位序, 若不存在, 返回值为 0。
- 5) ListInsert(&L, i, e) 在 L 中第 i 个位置前插入新的数据元素 e, L 的长度加 1。
- 6) ListDelete(&L, i, &e) 删除 L 的第 i 个数据元素, 用 e 返回其值, 表长减 1。
- 7) DestroyList(&L) 销毁线性表 L。
- 8) ClearList(&L) 将 L 重置为空表。
- 9) ListEmpty(L) 判断 L 是否为空表, 是返回 true, 否则返回 false。
- 10) PriorElem(L, cur_e, &pre_e) 若 cur_e 是 L 的数据元素, 且不是第 1 个, 用 pre_e 返回 cur_e 的前驱, 否则操作失败, pre_e 无定义。
- 11) NextElem(L, cur_e, &next_e) 若 cur_e 是 L 的数据元素, 且不是最后一个, 用 next_e 返回 cur_e 的后继, 否则操作失败, next_e 无定义。
- 12) ListTraverse(L, visit()) 遍历, 依次对 L 的每一个元素调用函数 visit()。

}ADT List

2. 实验内容

2.1 问题描述

用线性表知识为一所拥有接近 40000 名学生和 3000 门课程的大学, 生成一个选课系统,

2.2 基本要求

- (1) 能够输入所有学生信息（学号，姓名，性别,...）；
- (2) 能够输入所有课程信息（课号，课名，学分,...）；
- (3) 能够查找、插入、删除学生记录；
- (4) 能够查找、插入、删除课程记录；
- (5) 能够输入学生选课信息，例如给定（学号 a，课号 b），就表示学生 a 注册了课程 b；
- (6) 能够输出某门课程的所有选课学生的名单，包含学生所有信息（学号、姓名、性别。。。）；
- (7) 能够输出某位学生的所有选课课程清单，包含课程的所有信息（课号、课名、学分。。。）；

2.3 数据结构设计

2.3.1 顺序表方法

分别建立两个顺序表 `Stulist` 和 `Courselist`，可以实现对学生和课程信息的删除和添加。每一个学生类型中包含一个课程类型的数据域，表示学生所有所选的课，在每一个课程类型中包含一个学生类型的数据域，表示选课学生列表。

先通过 `input` 函数初始化选课系统，即批量导入学生和课程列表，由于学生选课的顺序对课程本身不造成影响，可以直接简化为将插入的数据放到顺序表尾，这样使插入操作的时间复杂度由 $O(n)$ 变为 $O(1)$ 。然后设计 `search`, `insert`, `del` 等函数，实现删除、插入、查找等功能。在插入操作中，由于学生选课的顺序对课程本身不造成影响，可以直接简化为将插入的数据放到顺序表尾，这样使插入操作的时间复杂度由 $O(n)$ 变为 $O(1)$ 。在删除过程中添加确认删除的功能，防止误删，查找时设计分字段查找功能，提高查找的效率和便捷性。设计函数 `register` 表示输入学生信息和课程信息表示学生注册课程。值得注意的是，最后添加 `menu` 函数，每一次完成功能后按任意键返回并清屏，实现有可视化界面的用户友好型程序。

2.3.2 链表方法

建立两个线性链表 `Stulist` 和 `Courselist`，每个节点包含一个学生类型的数据域或一个课程类型的数据域。每一个学生类型中包含一个课程类型的数据域，表示学生所有所选的课，在每一个课程类型中包含一个学生类型的数据域，表示选课学生列表。

先通过 `input` 函数初始化选课系统，即批量导入学生和课程列表，由于学生选课的顺序对课程本身不造成影响，可以直接简化为将插入的数据放到顺序表尾，这样使插入操作的时间复杂度由 $O(n)$ 变为 $O(1)$ 。然后设计 `search`, `insert`, `del` 等函数，实现删除、插入、查找等功能。在插入操作中，由于学生选课的顺序对课程本身不造成影响，可以直接简化为将插入的数据放到顺序表尾，这样使插入操作的时间复杂度由 $O(n)$ 变为 $O(1)$ 。在删除过程中添加确认删除的功能，防止误删，查找时设计分字段查找功能，提高查找的效率和便捷性。设计函数 `register` 表示输入学生信息和课程信息表示学生注册课程。值得注意的是，最后添加 `menu` 函数，每一次完成功能后按任意键返回并清屏，实现有可视化界面的用户友好型程序。

2.4 功能说明（函数、类）

顺序表方法：

```
struct course;
struct student {
    char name[20];
    long int no;
    char gender;
```

```

    course *cour;
    int cournum=0;//cournum 表示每个学生课程的数量
};
struct course {
    char classname[20];
    long int classno;
    Float credit;
    student classstu[200];
    int classstunum=0;//classstunum 表示每节课程选课的学生数量
};//两个节点类型 student 和 course
typedef struct {
    student* stu;
    int listsize;
    int length;
}Stulist;
typedef struct {
    course* cou;
    int listsize;
    int length;
}Courselist;
Stulist L;
Courselist Q;//定义两个顺序表

Status Initliststu(Stulist &L)
{
    L.stu = (student*)malloc(STU_INIT_SIZE
    * sizeof(student));
    L.stu->cour = (course*)malloc(COURSE_INIT_SIZE * sizeof(course));
    if (!L.stu) exit(OVERFLOW);
    L.length = 0;
    L.listsize = STU_INIT_SIZE;
    L.stu->cournum = 0;
    return OK;
}
Status Initlistcourse(Courselist& Q)
{
    Q.cou = (course*)malloc(COURSE_INIT_SIZE
    * sizeof(course));
    if (!Q.cou) exit(OVERFLOW);
    L.length = 0;
    L.listsize = STU_INIT_SIZE;
    Q.cou->classstunum = 0;
    return OK;
}
};//对两个顺序表进行初始化操作，注意这里要对数据成员 cou 和 stu 的数量 cournum

```

和 classstunum 进行初始化，为 0。

Status Insertstu(Stulist&L)//由于选课系统无需在意顺序，插入操作直接执行到最后一个就行

```
{
    student a;
    while (1) {
        cout <<"请输入分别输入插入学生的学号、姓名、性别" << endl;
        cin >> a.no >> a.name >> a.gender;
        if (cin.good() == 1 && (a.gender == 'm' || a.gender == 'M' || a.gender == 'f' || a.gender == 'F'))
            break;
        cin.clear();
        cin.ignore();//处理输入错误
    }
```

```
        if (L.length + 1 >= L.listsize)
```

L.stu = (student*)realloc(L.stu, 100*sizeof(student));//如果空间不够需要重新分配空间

```
        if (!L.stu) exit(OVERFLOW);
        L.length++;
        L.stu[L.length-1] = a;
        L.stu[L.length - 1].cournum = 0;
        L.stu[L.length-1].cour = (course*)malloc(COURSE_INIT_SIZE *
sizeof(course));//注意每一次插入后，要为新的节点初始化数据域 stu，以及 stu 类型中的 cournum 为 0，以及为表示选课列表的内置顺序表分配空间。
        cout << "插入成功，按任意键继续" << endl;
        _getch();//用 getch 来读入一个字符，实现按任意键建继续的功能
    }
```

Status Insertcourse(Courselist&Q)//由于选课系统无需在意顺序，插入操作直接执行到最后一个就行

```
{
    course c;
    while (1) {
        cout <<"请输入分别输入插入课程的课号、课程名、学分" << endl;
        cin >> c.classno >> c.classname
            >> c.credit;
        if (cin.good() == 1)
            break;
        cin.clear();
        cin.ignore();//处理输入错误
    }
```

```

        if (Q.length + 1 >= Q.listsize)
            Q.cou = (course*)realloc(Q.cou, 100 * sizeof(course)); //如果空间不够
需要重新分配空间
        if (!Q.cou) exit(OVERFLOW);
        Q.length++;
        Q.cou[Q.length-1] = c;
        Q.cou[Q.length-1].classstunum = 0;
        cout << "插入成功, 按任意键继续" << endl;
        _getch();
} //执行学生信息和课程信息的插入功能

```

```

void searchcou(Courselist& Q) //查找功能
{
    cout << "请选择查找字段" << endl;
    cout << "1. 课号" << endl;
    cout << "2. 课程名" << endl; //选择查找字段
    int i;
    while (1) {
        cin >> i;
        if ((i == 1 || i == 2) && cin.good() == 1)
            break;
        cout << "输入错误, 请重新输入" << endl;
        cin.clear();
        cin.ignore(); //处理输入错误
    }
    int j;
    switch (i)
    {
        case 1:
            long int num;
            cout << "请输入查询目标课号: " << endl;
            while (1)
            {
                cin >> num;
                if (cin.good() == 1)
                    break;
                cout << "输入错误, 请重新输入" << endl; //处理输入错误
            }
            for (j = 0; j < Q.length; j++)
            {
                if (Q.cou[j].classno == num)
                    break;
            }

```

```

if(i==Q.length)
    cout << "查询失败！没有找到相关课程信息" << endl;//处理输入错误
else {
    cout << "查询成功！课程信息如下：" << endl;
    cout << "课号：" << Q.cou[j].classno << endl;
    cout << "课程名：" << Q.cou[j].classname << endl;
    cout << "学分" << Q.cou[j].credit << endl;
    cout << "选课学生："<<endl;
    int m = 0;
    while (m<Q.cou->classstunum)
    {
        cout << "姓名：" << Q.cou->classstu[m].name << " 学号：" <<
Q.cou->classstu[m].no<< " 性别" << Q.cou->classstu[m].gender << endl;
        m++;
    }//输出查找的结果
    }
    break;
case 2:
    char cname[20];
    cout << "请输入查询目标课程名：" << endl;
    while (1)
    {
        cin >> cname;
        if (cin.good() == 1)
            break;
        cout << "输入错误，请重新输入" << endl;//处理输入错误
    }
    for (j = 0; j < Q.length; j++)
    {
        if (strcmp(Q.cou[j].classname,cname)==0)
            break;
    }
    if(j==Q.length)
        cout << "查询失败！没有找到相关课程信息" << endl;
    else {
        cout << "查询成功！课程信息如下：" << endl;
        cout << "课号：" << Q.cou[j].classno << endl;
        cout << "课程名：" << Q.cou[j].classname << endl;
        cout << "学分" << Q.cou[j].credit << endl;
        cout << "选课学生："<<endl;
        int m = 0;//输出查找的结果
        while (m < Q.cou->classstunum)
        {
            cout << "姓名：" << Q.cou->classstu[m].name << " 学号：" <<

```

```

Q.cou->classstu[m].no << " 性别" << Q.cou->classstu[m].gender << endl;
        m++;
    }
}
break;

}
_getch();
}
void searchstu(Stulist & L)
{
    cout << "请选择查找字段" << endl;
    cout << "1. 学号" << endl;
    cout << "2. 姓名" << endl;
    int i;//选择查找字段
    while (1) {
        cin >> i;
        if ((i == 1 || i == 2) && cin.good() == 1)
            break;
        cout << "输入错误，请重新输入" << endl;
        cin.clear();
        cin.ignore();
    }
    int j;
    switch (i)
    {
    case 1:
        long int num;
        cout << "请输入查询目标学号：" << endl;
        cin >> num;

        for (j = 0; j < L.length; j++)
        {
            if (L.stu[j].no==num)
                break;
        }
        if (j == L.length)
            cout << "查询失败！没有找到相关学生信息" << endl;
        else {
            cout << "查询成功！学生信息如下：" << endl;
            cout << "学号：" << L.stu[j].no << endl;
            cout << "姓名：" << L.stu[j].name << endl;
            cout << "性别" << L.stu[j].gender << endl;
            cout << "选课列表：" << endl;

```

```

        int m = 0;
        while (m < L.stu->cournum)
        {
            cout << "课程名: " << L.stu->cour[m].classname << " 课号:
" << L.stu->cour[m].classno << " 课程学分" << L.stu->cour[m].credit << endl;
            m++;
        }
    }
    break;
case 2:
    char sname[20];
    cout << "请输入目标学生姓名" << endl;
    cin >> sname;
    for (j = 0; j < L.length; j++)
    {
        if (strcmp(L.stu[j].name, sname) == 0)
            break;
    }
    if (j == L.length)
        cout << "查询失败! 没有找到相关学生信息" << endl;
    else {
        cout << "查询成功! 学生信息如下:" << endl;
        cout << "学号:" << L.stu[j].no << endl;
        cout << "姓名:" << L.stu[j].name << endl;
        cout << "性别" << L.stu[j].gender << endl;
        cout << "选课列表:" << endl;
        int m = 0;
        while (m < L.stu->cournum)
        {
            cout << "课程名:" << L.stu->cour[m].classname << " 课号:
" << L.stu->cour[m].classno << " 课程学分" << L.stu->cour[m].credit << endl;
            m++;
        }
    }
    break;
}
_getch();
}

```

```

void delstu(Stulist& L)//删除操作
{

```

```

    cout << "请输入想要删除学生的学号" << endl;
    long int delnum;
    char confirm;

```



```

while (1)
{
    cin >> delnum;
    student* p = L.stu;
    while (p->no != delnum && p != NULL)
        p++; //删除先要查询到信息位置
    if (p == NULL)
    {
        cout << "查询不到此学生信息，请重新输入" << endl;
        continue;
    }
    else {
        cout << "您要删除的学生为" << endl;
        cout << "姓名: " << p->name << " 学号: " << p->no << " 性别: " <<
p->gender << endl;
        cout << "您是否确认删除? (Y/N)"; //确认删除功能的实现

        cin >> confirm;
        if (confirm == 'Y' || confirm == 'y')
        {
            student* q=L.stu + L.length - 1;
            for (p;p <= q;p++)
                *p = *(p + 1);
            L.length--;
            cout << " 删除成功，按任意键继续" << endl;
        }
        else if (confirm == 'N' || confirm == 'n' && cin.good() == 1)
        {
            cout << "按任意键退出" << endl;
        }
        _getch();
        break;
    }
}

}

void delcou(Courselist& Q)
{
    cout << "请输入想要删除课程的课号" << endl;
    long int delclassnum;
    char confirm;
    while (1)
    {

```

```

        cout << "请输入想要删除学生的学号" << endl;
        cin >> delclassnum;
        course* p = Q.cou;
        while (p->classno != delclassnum && p != NULL)
            p++;
        if (p == NULL)
        {
            cout << "查询不到此课程信息，请重新输入" << endl;
            continue;
        }
        else {
            cout << "您要删除的课程为" << endl;
            cout << "课程名：" << p->classname << " 课号：" << p->classno <<
" 学分" << p->credit << endl;
            cout << "您是否确认删除？(Y/N)";
            cin >> confirm;
            if (confirm == 'Y' || confirm == 'y')
            {
                course * q = Q.cou + Q.length - 1; //确认删除功能的实现
                for (p; p <= q; p++)
                    *p = *(p + 1);
                Q.length--;
                cout << " 删除成功，按任意键继续" << endl;
            }
            else if (confirm == 'N' || confirm == 'n' && cin.good() == 1)
            {
                cout << "按任意键退出" << endl;
            }
            _getch();
            break;
        }
    }
}

int menu()
{
    cout << "*****"
<< endl;
    cout << "*"                学生选课系统                "*"
<< endl;
    cout << "*"                1. 批量输入学生信息                "*"
<< endl;
    cout << "*"                2. 批量输入课程信息                "*"

```

```

<< endl;
    cout << "*"          3. 插入学生信息          "*"
<< endl;
    cout << "*"          4. 删除学生信息          "*"
<< endl;
    cout << "*"          5. 查找学生信息          "*"
<< endl;
    cout << "*"          6. 插入课程信息          "*"
<< endl;
    cout << "*"          7. 删除课程信息          "*"
<< endl;
    cout << "*"          8. 查找课程信息          "*"
<< endl;
    cout << "*"          9. 学生选课              "*"
<< endl;
    cout << "*"          10. 显示选课名单          "*"
<< endl;
    cout << "*"          11. 显示学生课表          "*"
<< endl;
    cout << "*"          0. 退出                  "*"
<< endl;
    cout << "*"
<< endl;
    cout << "*"
<< endl;
    cout << "*"
<< endl;
    cout << "*"
<< endl;
    cout << "*"
<< endl;
    cout << "*****"
<< endl;
    int i;
    cout << "请输入要执行的操作[0-11]:";
    cin >> i;
    return i;
} //菜单功能, 返回一个选择值

void inputstu(Stulist &L)
{
    int i;
    cout << " 请输入信息录入的学生人数: ";
    cin >> i;

```

```

    cout << " 请分别输入每个学生的学号、姓名和性别" << endl;
    for (int j = 0; j < i; j++)
    {
        while (1)
        {
            cin >> L.stu[j].no >> L.stu[j].name >> L.stu[j].gender;
            L.stu[j].cournum = 0;
            L.stu[j].cour = (course*)malloc(COURSE_INIT_SIZE *
sizeof(course));
//每一次输入要为 cour 分配一个储存空间（因为类型定义中没有预先定义数组大小）
            if (cin.good() == 1 && (L.stu[j].gender == 'm' || L.stu[j].gender
== 'M' || L.stu[j].gender == 'f' || L.stu[j].gender == 'F'))
                break;
            cout << "输入错误，请重新输入该学生信息" << endl;
            cin.clear();
            cin.ignore();//处理输入错误
        }

    }
    L.length += i;
    cout << "录入成功！按任意键继续...";
    _getch();
}

void inputcourse(Courselist& Q)
{
    int i;
    cout << " 请输入信息录入的课程门数：";
    cin >> i;
    cout << " 请分别输入每节课的课号、课程名和学分" << endl;
    for (int j = 0; j < i; j++)
    {
        while (1)
        {
            cin >> Q.cou[j].classno >> Q.cou[j].classname >> Q.cou[j].credit;
            Q.cou[j].classtunum = 0;
            if (cin.good() == 1)
                break;
            cin.clear();
            cin.ignore();//处理输入错误
        }
    }

    Q.length += i;

```

```

        cout << "录入成功！按任意键继续...";
        _getch();
    } //初始化选课系统，批量输入学生和课程信息

void registercourse(Stulist &L, CourseList &Q)
{
    int i = 0;
    int j = 0;
    while (1) {
        j = 0;
        i = 0;
        cout << "请输入需要选课的学生学号以及课程课号：";
        long int stunum, coursenum;
        cin >> stunum >> coursenum;
        while (i < L.length)
        {
            if (L.stu[i].no == stunum)
                break;
            i++;
        }
        if (i == L.length)
            cout << "查询不到此学生信息，请重新输入" << endl;
        while (j < Q.length)
        {
            if (Q.cou[j].classno == coursenum)
                break;
            j++;
        }
        if (j == Q.length)
            cout << "查询不到此课程信息，请重新输入" << endl;
        if (i < L.length && j < Q.length)
            break; //处理输入错误
        cin.clear();
        cin.ignore();

    }
    L.stu[i].cour[L.stu[i].cournum] = Q.cou[j];
    Q.cou[j].classstu[Q.cou[j].classstunum] = L.stu[i];
    L.stu[i].cournum++;
    Q.cou[j].classstunum++;
    cout << "选课成功，按任意键退出" << endl;
    _getch();
} //选课功能，通过输入学号和课号来实现选课

void registercourse(Stulist &L, CourseList &Q)

```

```

{
    int i = 0;
    int j = 0;
    while (1) {
        j = 0;
        i = 0;
        cout << "请输入需要选课的学生学号以及课程课号： ";
        long int stunum, coursenum;
        cin >> stunum >> coursenum;
        while (i<L.length)
        {
            if (L.stu[i].no == stunum)
                break;
            i++;
        }
        if (i==L.length)
            cout << "查询不到此学生信息，请重新输入" << endl;
        while (j <Q.length)
        {
            if (Q.cou[j].classno == coursenum)
                break;
            j++;
        }
        if (j==Q.length)
            cout << "查询不到此课程信息，请重新输入" << endl;
        if (i<L.length&& j<Q.length)
            break;
        cin.clear();
        cin.ignore();

    }
    L.stu[i].cour[L.stu[i].cournum] = Q.cou[j];
    Q.cou[j].classstu[Q.cou[j].classstunum] = L.stu[i];
    L.stu[i].cournum++;
    Q.cou[j].classstunum++;
    cout << "选课成功，按任意键退出" << endl;
    _getch();
} //显示某一某门课程的所有选课学生的名单、输出某位学生的所有选课课程清单

```

链表方法：
 typedef int Status;

```

struct course;
struct student {
    char name[20];
    long int no;
    char gender;
    course* cour;
    int cournum = 0;
};
struct course {
    char classname[20];
    long int classno;
    float credit;
    student classstu[200];
    int classstunum = 0;
}; //建立两个数据域类型
typedef struct StuNode {
    student stu;
    struct StuNode* next;
}StuNode,*Stulist;
typedef struct CourseNode {
    course cou;
    struct CourseNode* next;
}CourseNode,*Courselist; //结点类型
Stulist L;
Courselist Q;
Status Initliststu(Stulist & L)
{
    L=(Stulist)malloc(sizeof(StuNode));
    if (!L)
        exit(OVERFLOW);
    L->next = NULL; /* 指针域为空*/
    L->stu.cour= (course*)malloc(200 * sizeof(course));
    if (!L->stu.cour) exit(OVERFLOW);
    L->stu.cournum = 0; //初始化时要把类中 cournum 一同初始化

    return OK;
}
Status Initlistcourse(Courselist& Q)
{
    Q=(Courselist)malloc(sizeof(CourseNode));
    if (!Q) exit(OVERFLOW);
    Q->next = NULL;
    Q->cou.classstunum = 0; //初始化时要把类中 classstunum 一同初始化
    return OK;
}
Status Insertstu(Stulist& L) //由于选课系统无需在意顺序，插入操作直接执行头插法
{
    Stulist p;
    p= (Stulist)malloc( sizeof(StuNode));
    if (!p) exit(OVERFLOW);
    while (1) {
        cout << "请输入分别输入插入学生的学号、姓名、性别" << endl;
        cin >> p->stu.no >> p->stu.name >> p->stu.gender;
        if (cin.good() == 1 && (p->stu.gender == 'm' || p->stu.gender== 'M' ||
p->stu.gender == 'f' || p->stu.gender == 'F'))
            break;
        cin.clear();
    }
}

```

```

        cin.ignore();
    }
    L->next = p;
    p->next = NULL;//插入
    p->stu.cournum = 0;
    p->stu.cour = (course*)malloc(3000 * sizeof(course));
    cout << "插入成功, 按任意键继续" << endl;
    _getch();
}
Status Insertcourse(Courselist& Q)//由于选课系统无需在意顺序, 插入操作直接执行头插法比较简单
{
    Courselist p;
    p = (Courselist)malloc(sizeof(CourseNode));
    if (!p) exit(OVERFLOW);
    while (1) {
        cout << "请输入分别输入插入课程的课号、课程名、学分" << endl;
        cin >> p->cou.classno >> p->cou.classname >> p->cou.credit;
        if (cin.good() == 1)
            break;
        cin.clear();
        cin.ignore();
    }

    Q->next = p;
    p->cou.classstunum = 0;
    cout << "插入成功, 按任意键继续" << endl;
    _getch();
}

void searchcou(Courselist& Q)//查找操作
{
    cout << "请选择查找字段" << endl;
    cout << "1. 课号" << endl;
    cout << "2. 课程名" << endl;//设置两个字段方便查找
    int i;
    while (1) {
        cin >> i;
        if ((i == 1 || i == 2) && cin.good() == 1)
            break;
        cout << "输入错误, 请重新输入" << endl;
        cin.clear();
        cin.ignore();
    }
    int j;
    Courselist q = Q;
    switch (i)
    {
        case 1:
            long int num;
            cout << "请输入查询目标课号: " << endl;
            while (1)
            {
                cin >> num;
                if (cin.good() == 1)
                    break;
                cout << "输入错误, 请重新输入" << endl;
            }

```



```

while (q != NULL)
{
    if (q->cou.classno == num)
        break;
    q = q->next;
}
if (q==NULL)
    cout << "查询失败! 没有找到相关课程信息" << endl;
else {
    cout << "查询成功! 课程信息如下:" << endl;
    cout << "课号:" << q->cou.classno << endl;
    cout << "课程名:" << q->cou.classname << endl;
    cout << "学分" << q->cou.credit << endl;
    cout << "选课学生:" << endl;
    int m = 0; //输出查找到的所有信息
    while (m < q->cou.classstunum)
    {
        cout << "姓名:" << q->cou.classstu[m].name << " 学号: " <<
q->cou.classstu[m].no << " 性别" << q->cou.classstu[m].gender << endl;
        m++;
    }
    break;
}
case 2:
char cname[20];
cout << "请输入查询目标课程名: " << endl;
while (1)
{
    cin >> cname;
    if (cin.good() == 1)
        break; //处理输入错误
    cout << "输入错误, 请重新输入" << endl;
}
while (q != NULL)
{
    if (strcmp(q->cou.classname, cname) == 0)
        break;
    q = q->next;
}
if (q==NULL)
    cout << "查询失败! 没有找到相关课程信息" << endl;
else {
    cout << "查询成功! 课程信息如下:" << endl;
    cout << "课号:" << q->cou.classno << endl;
    cout << "课程名:" << q->cou.classname << endl;
    cout << "学分" << q->cou.credit << endl;
    cout << "选课学生:" << endl;
    int m = 0;
    while (m < q->cou.classstunum)
    {
        cout << "姓名:" << q->cou.classstu[m].name << " 学号: " <<
q->cou.classstu[m].no << " 性别" << q->cou.classstu[m].gender << endl;
        m++;
    }
    break;
}
}

```

```

    _getch();
}
void searchstu(Stulist& L)//查询学生信息
{
    cout << "请选择查找字段" << endl;
    cout << "1. 学号" << endl;
    cout << "2. 姓名" << endl;
    int i;
    Stulist q=L;
    while (1) {
        cin >> i;
        if ((i == 1 || i == 2) && cin.good() == 1)
            break;
        cout << "输入错误, 请重新输入" << endl;
        cin.clear();
        cin.ignore();
    }
    int j;
    switch (i)
    {
    case 1:
        long int num;
        cout << "请输入查询目标学号: " << endl;
        cin >> num;

        while (q != NULL)
        {
            if (q->stu.no == num)
                break;
            q = q->next;
        }
        if (q==NULL)
            cout << "查询失败! 没有找到相关学生信息" << endl;
        else {
            cout << "查询成功! 学生信息如下:" << endl;
            cout << "学号:" << q->stu.no << endl;
            cout << "姓名:" << q->stu.name << endl;
            cout << "性别" << q->stu.gender << endl;
            cout << "选课列表:" << endl;
            int m = 0;
            while (m < q->stu.cournum)
            {
                cout << "课程名: " << q->stu.cour[m].classname << " 课号: "
                << q->stu.cour[m].classno << " 课程学分" << q->stu.cour[m].credit << endl;
                m++;
            }
        }
        break;
    case 2:
        char sname[20];
        cout << "请输入目标学生姓名" << endl;
        cin >> sname;
        while (q != NULL)
        {
            if (strcmp(q->stu.name, sname)==0)
                break;
            q = q->next;
        }
    }
}

```

```

        if (q==NULL)
            cout << "查询失败！没有找到相关学生信息" << endl;
        else {
            cout << "查询成功！学生信息如下：" << endl;
            cout << "学号：" << q->stu.no << endl;
            cout << "姓名：" << q->stu.name << endl;
            cout << "性别" << q->stu.gender << endl;
            cout << "选课列表：" << endl;
            int m = 0;
            while (m < q->stu.cournum)
            {
                cout << "课程名：" << q->stu.cour[m].classname << "    课号："
                << q->stu.cour[m].classno << "    课程学分" << q->stu.cour[m].credit << endl;
                m++;
            }
            break;
        }
        _getch();
    }
}

```

void delstu(Stulist& L)//删除操作

```

{
    cout << "请输入想要删除学生的学号" << endl;//通过学号删除，防止重名导致
    错删
    long int delnum;
    char confirm;
    while (1)
    {
        cin >> delnum;
        Stulist p = L,q;
        while (p->next->stu.no != delnum && p->next != NULL)
            p++;
        if (p->next== NULL)
        {
            cout << "查询不到此学生信息，请重新输入" << endl;
            continue;
        }
        else {
            q = p->next;
            cout << "您要删除的学生为" << endl;
            cout << "姓名：" << q->stu.name << " 学号：" << q->stu.no << " 性
            别：" << q->stu.gender << endl;
            cout << "您是否确认删除？(Y/N)";//确认删除，防止误删

            cin >> confirm;
            if (confirm == 'Y' || confirm == 'y')
            {
                p->next = q->next;
                free(q);
                cout << " 删除成功，按任意键继续" << endl;
            }
            else if (confirm == 'N' || confirm == 'n' && cin.good() == 1)
            {
                cout << "按任意键退出" << endl;
            }
        }
    }
}

```

```

        _getch();
        break;
    }

}

}

void delcou(Courselist& Q)
{
    cout << "请输入想要删除课程的课号" << endl;
    long int delclassnum;
    char confirm;
    Courselist p = Q, q;
    while (1)
    {
        cout << "请输入想要删除学生的学号" << endl;
        cin >> delclassnum;
        while (p->next->cou.classno != delclassnum && p->next != NULL)
            p++;
        if (p->next == NULL)
        {
            cout << "查询不到此课程信息，请重新输入" << endl;
            continue;
        }
        else {
            q = p->next;
            cout << "您要删除的课程为" << endl;
            cout << "课程名：" << q->cou.classname << " 课号：" << q->cou.classno
            << " 学分" << q->cou.credit << endl;
            cout << "您是否确认删除？(Y/N)";
            cin >> confirm;
            if (confirm == 'Y' || confirm == 'y')
            {
                p->next = q->next;
                free(q);
                cout << " 删除成功，按任意键继续" << endl;
            }
            else if (confirm == 'N' || confirm == 'n' && cin.good() == 1)
            {
                cout << "按任意键退出" << endl;
            }
            _getch();
            break;
        }
    }
}
}
}

```

```

int menu()//打印菜单，返回一个输入值
{
    cout << "*****"
    << endl;
    cout << "                学生选课系统                *"
    << endl;
    cout << "                1. 批量输入学生信息                *"
    << endl;
    cout << "                2. 批量输入课程信息                *"
}

```

```

<< endl;
    cout << "*"          3. 插入学生信息          "*"
<< endl;
    cout << "*"          4. 删除学生信息          "*"
<< endl;
    cout << "*"          5. 查找学生信息          "*"
<< endl;
    cout << "*"          6. 插入课程信息          "*"
<< endl;
    cout << "*"          7. 删除课程信息          "*"
<< endl;
    cout << "*"          8. 查找课程信息          "*"
<< endl;
    cout << "*"          9. 学生选课              "*"
<< endl;
    cout << "*"          10. 显示选课名单          "*"
<< endl;
    cout << "*"          11. 显示学生课表          "*"
<< endl;
    cout << "*"          0. 退出                  "*"
<< endl;
    cout << "*"          "*"
<< endl;
    cout << "*"          "*"
<< endl;
    cout << "*"          "*"
<< endl;
    cout << "*"          "*"
<< endl;
    cout << "*"          "*"
<< endl;
    cout << "*****"
<< endl;
    int i;
    cout << "请输入要执行的操作[0-11]:";
    cin >> i;
    return i;
}

```

```

void inputstu(Stulist& L)//初始化选课系统，开始时批量导入学生和课程信息
{
    int i;
    cout << " 请输入信息录入的学生人数: ";
    cin >> i;
    cout << " 请分别输入每个学生的学号、姓名和性别" << endl;
    for (int j = 0; j < i; j++)
    {
        while (1)
        {
            Stulist p;
            p = (Stulist)malloc(sizeof(StuNode));
            p->next = L->next;
            L->next=p;
            cin >> p->stu.no >> p->stu.name >> p->stu.gender;
            p->stu.cournum = 0;
            p->stu.cour = (course*)malloc(100 * sizeof(course));
            if (cin.good() == 1 && (p->stu.gender == 'm' || p->stu.gender ==

```

```

'M' || p->stu.gender == 'f' || p->stu.gender == 'F'))
    break;
    cout << "输入错误, 请重新输入该学生信息" << endl;
    cin.clear();
    cin.ignore();
}

}
cout << "录入成功! 按任意键继续...";
_getch();
}
void inputcourse(Courselist& Q)
{
    int i;
    cout << " 请输入信息录入的课程门数: ";
    cin >> i;
    cout << " 请分别输入每节课程的课号、课程名和学分" << endl;
    for (int j = 0; j < i; j++)
    {
        while (1)
        {
            Courselist p;
            p = (Courselist)malloc(sizeof(CourseNode)); //每插入一个都要分配
            空间
            p->next = Q->next;
            Q->next = p;
            cin >> p->cou.classno >> p->cou.classname >> p->cou.credit;
            p->cou.classstunum = 0;
            if (cin.good() == 1)
                break;
            cin.clear();
            cin.ignore(); //处理输入错误
        }
    }
    cout << "录入成功! 按任意键继续...";
    _getch();
}

```

```

void registercourse(Stulist& L, Courselist& Q)
{
    Stulist p = L;
    Courselist q = Q;
    while (1) {
        cout << "请输入需要选课的学生学号以及课程课号: ";
        long int stunum, coursenum;
        cin >> stunum >> coursenum;
        while (p!=NULL)
        {
            if (p->stu.no == stunum)
                break;
            p = p->next;
        }
        if (p==NULL)
            cout << "查询不到此学生信息, 请重新输入" << endl;
        while (q!=NULL)
        {
            if (q->cou.classno == coursenum)

```

```

        break;
        q = q->next;
    }
    if (q==NULL)
        cout << "查询不到此课程信息，请重新输入" << endl;
    if (q && p)
        break;
    cin.clear();
    cin.ignore();

}
p->stu.cour[p->stu.cournum] = q->cou;
q->cou.classstu[q->cou.classstunum] = p->stu;
p->stu.cournum++;
q->cou.classstunum++; //将学生信息导入到课程选课名单，将课程信息导入到学
生课表
cout << "选课成功，按任意键退出" << endl;
_getch();
}

```

```

void show_stu_list(Courselist& Q) //通过课号显示所有选课名单
{
    cout << "请输入需要查询的课号" << endl;
    long int num;
    cin >> num;
    Courselist p = Q;
    while(p!=NULL)
    {
        if (p->cou.classno == num)
            break;
        p = p->next;
    }
    if (p==NULL)
    {
        cout << "该课号不存在！按任意键退出" << endl;
        _getch();
        return;
    }
    else {
        for (int j = 0; j < p->cou.classstunum; j++)
        {
            cout << "姓名：" << p->cou.classstu[j].name << " 学号：" <<
p->cou.classstu[j].no << " 性别：" << p->cou.classstu[j].gender << endl;
        }
        cout << "按任意键退出" << endl;
        _getch();
    }
}

void show_course_list(Stulist& L) //通过学号查看某一同学课表
{
    cout << "请输入需要查询的学生学号" << endl;
    long int num;
    Stulist p = L;
    cin >> num;
}

```

```

while (p != NULL)
{
    if (p->stu.no == num)
        break;
    p = p->next;
}
if (p==NULL)
{
    cout << "该课号不存在! 按任意键退出" << endl;
    _getch();
    return;
}
else {
    for (int j = 0; j < p->stu.cournum; j++)
    {
        cout << "课程名: " << p->stu.cour[j].classname << " 课号: " <<
p->stu.cour[j].classno << " 学分: " << p->stu.cour[j].credit << endl;
    }
    cout << "按任意键退出" << endl;
    _getch();
}
}
int main()
{
    Initliststu(L);
    Initlistcourse(Q);
    while (1)
    {
        int ret = menu();
        if (ret == 0)
            break;
        if (ret == 1)
        {
            inputstu(L);
        }
        else if (ret == 2)
        {
            inputcourse(Q);
        }
        else if (ret == 3)
        {
            Insertstu(L);
        }
        else if (ret == 4)
        {
            delstu(L);
        }
        else if (ret == 5)
        {
            searchstu(L);
        }
        else if (ret == 6)
        {
            Insertcourse(Q);
        }
        else if (ret == 7)
        {

```



```

        delcou(Q);
    }
    else if (ret == 8)
    {
        searchcou(Q);
    }
    else if (ret == 9)
    {
        registercourse(L, Q);
    }
    else if (ret == 10)
    {
        show_stu_list(Q);
    }
    else if (ret == 11)
    {
        show_course_list(L);
    }
    system("cls");
}

return 0;
} //显示某一门课程的所有选课学生的名单、输出某位学生的所有选课课程清单
# 总之清晰描述类、函数的功能，参数、输出含义即可

```

2.1.5 功能展示

主菜单界面，实现功能选择，用户使用性好

```

***** 学生选课系统 *****
*                               *
* 1. 批量输入学生信息          *
* 2. 批量输入课程信息          *
* 3. 插入学生信息              *
* 4. 删除学生信息              *
* 5. 查找学生信息              *
* 6. 插入课程信息              *
* 7. 删除课程信息              *
* 8. 查找课程信息              *
* 9. 学生选课                  *
* 10. 显示选课名单             *
* 11. 显示学生课表             *
* 0. 退出                      *
*                               *
*****
请输入要执行的操作[0-11]:

```

输入操作，在选课系统开始前先批量导入学生信息，每一步有文字说明引导，导入成功后按任意键退出输入功能，重新进入菜单选择。

```

***** 学生选课系统 *****
*                               *
* 1. 批量输入学生信息          *
* 2. 批量输入课程信息          *
* 3. 插入学生信息              *
* 4. 删除学生信息              *
* 5. 查找学生信息              *
* 6. 插入课程信息              *
* 7. 删除课程信息              *
* 8. 查找课程信息              *
* 9. 学生选课                  *
* 10. 显示选课名单             *
* 11. 显示学生课表             *
* 0. 退出                      *
*                               *
*****
请输入要执行的操作[0-11]:1
  请输入信息录入的学生人数: 3
  请分别输入每个学生的学号、姓名和性别
211234 Mark m
207643 Juli f
211235 Rox m
录入成功! 按任意键继续...

```

录入课程信息

```
请输入要执行的操作[0-11]:2
  请输入信息录入的课程门数: 2
  请分别输入每节课的课号、课程名和学分
100987 math 3
100546 cs 4
录入成功! 按任意键继续...
```

插入学生信息，由于学生信息的顺序对于选课无影响，直接将其插入最后（链表方式直接采用头插法），提示插入成功，按任意键回到菜单

```

*****
*                               *
*      学生选课系统          *
*                               *
*      1. 批量输入学生信息    *
*      2. 批量输入课程信息    *
*      3. 插入学生信息        *
*      4. 删除学生信息        *
*      5. 查找学生信息        *
*      6. 插入课程信息        *
*      7. 删除课程信息        *
*      8. 查找课程信息        *
*      9. 学生选课            *
*      10. 显示选课名单       *
*      11. 显示学生课表      *
*      0. 退出                *
*                               *
*                               *
*                               *
*                               *
*                               *
*                               *
*****
请输入要执行的操作[0-11]:3
请输入分别输入插入学生的学号、姓名、性别
2153538 刘博洋 m
插入成功，按任意键继续

```

查询操作，设定两种字段查询方式供使用者选择，提升了查询的便利性和效率，提示查询成功，并输入查询目标的所有信息（此处由于没有选课所以选课列表为空）

```

*
*
*****
请输入要执行的操作[0-11]:5
请选择查找字段
1. 学号
2. 姓名
1
请输入查询目标学号:
2153538
查询成功! 学生信息如下:
学号:2153538
姓名:刘博洋
性别:m
选课列表:

```

删除操作，通过学号来删除，避免发生名字相同而误删的情况。在输入学号删除后，执行确认删除步骤，通过键盘输入 y/n 来决定是否确认删除，防止数据的误删，提高程序的健壮性。

| | |
|--|--|
| 请输入要执行的操作[0-11]:4 请输入想要删除学生的学号 2153538 您要删除的学生为 姓名: 刘博洋 学号: 2153538 性别: m 您是否确认删除? (Y/N)N 按任意键退出 | 请输入要执行的操作[0-11]:4 请输入想要删除学生的学号 2153538 您要删除的学生为 姓名: 刘博洋 学号: 2153538 性别: m 您是否确认删除? (Y/N)y 删除成功, 按任意键继续 |
|--|--|

再次查询发现已经找不到相关信息，证明删除成功。

```
请输入要执行的操作[0-11]:5
请选择查找字段
1. 学号
2. 姓名
1
请输入查询目标学号:
2153538
查询失败！没有找到相关学生信息
```

选课功能，通过输入学号和课号，来实现对学生 a 注册课程 b，（防止出现多个课程名相同但任课老师不同导致选课出错的情况）找到了课程和学生提示选课成功，并把学生加入选课列表，把课程加入学生课表。

```
学生选课系统
1. 批量输入学生信息
2. 批量输入课程信息
3. 插入学生信息
4. 删除学生信息
5. 查找学生信息
6. 插入课程信息
7. 删除课程信息
8. 查找课程信息
9. 学生选课
10. 显示选课名单
11. 显示学生课表
0. 退出
*****
请输入要执行的操作[0-11]:9
请输入需要选课的学生学号以及课程课号: 211234 100987
选课成功，按任意键退出
```

如果输入错误导致找不到课程和学生，提示选课错误

```
请输入要执行的操作[0-11]:9
请输入需要选课的学生学号以及课程课号: 211235 100988
查询不到此课程信息，请重新输入
请输入需要选课的学生学号以及课程课号: 211235 100987
选课成功，按任意键退出
```

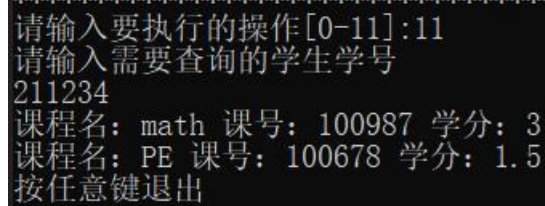
输出某一个课程所有选课的学生名单，按任意键退出。

```
请输入要执行的操作[0-11]:10
请输入需要查询的课号
100987
姓名: Mark 学号: 211234 性别: m
姓名: Rox 学号: 211235 性别: m
按任意键退出
```

插入课程信息 提示插入成功

```
请输入要执行的操作[0-11]:6
请输入分别输入插入课程的课号、课程名、学分
100678 PE 1.5
插入成功，按任意键继续
```

输出某个学生的所有课程列表



```
请输入要执行的操作[0-11]:11
请输入需要查询的学生学号
211234
课程名: math 课号: 100987 学分: 3
课程名: PE 课号: 100678 学分: 1.5
按任意键退出
```

3. 实验总结

3.1 程序分析

3.1.1 健壮性分析

本程序对数据的输入输出做了比较详尽的错误处理，比如基本上在每一个涉及到输出的地方，本程序都采用

```
while (1)
{
    cin>>xxx;
    if(cin.good()==1&&(数据符合常理要求))
        break;
    cin.clear();
    cin.ignore();
}
```

的方式对错误输入进行处理，比如输入学生信息时学号是否误输入了字母，性别是否是除了‘m’ ‘M’ ‘F’ ‘f’之外的其他不正确值。

并且在查询操作时，如果对输入进行查询无结果，输出提示查询失败并重新输入。

在实现删除操作时，加入是否确认删除选项，输入 y/n 判断是否确认删除，防止误删。

在实现选课功能时，如果查询到输入的学号或者课号无匹配，则提示输入错误，找不到匹配的课程或学生，重新输入。

3.1.2 效率分析

运用顺序表实现本问题时，插入由于可以直接在表尾插入，时间复杂度为 $O(1)$ ；删除操作需要先遍历找到要删除的元素，再把 $n-i$ 个元素全部挪动，时间复杂度为 $O(m+n)$ ；查找操作需要遍历整个顺序表，时间复杂度为 $O(n)$ ；学生选课操作是要先查询到学生和课程信息，然后把学生信息录入课程选课列表中，并且把课程信息录入学生课表中，时间复杂度为 $O(m+n)$ 。

运用线性链表实现本问题时，插入直接在表头插入即可，时间复杂度 $O(1)$ ；由于本题是通过学号来删除，实现并不知道要删除的元素位置删除操作需要遍历找到需要删除的元素，再执行删除，实际上时间复杂度也需要 $O(n)$ ；查找操作也是遍历整个链表，复杂度 $O(n)$ ；选课操作是要先遍历链表查询到学生和课程信息，然后把学生信息录入课程选课列表中，并且把课程信息录入学生课表中，时间复杂度为 $O(m+n)$ 。

从空间效率来看，由于有实际经验估计，用顺序表预先设置一个比较大的值，插入时空间不够再用 `realloc` 来分配，但是如果碰到实际情况与经验估计相差较大，即总体数量

较小时，会造成极大的空间浪费。

如果用链表进行操作，那么不用预先分配空间，在每一次插入和录入的过程中进行空间的分配，这样最大限度的节约了空间，并且不会造成溢出，这是链表的优势所在。

但是在实际选课列表中，可能会对课程和学生进行频繁的访问操作，这时候可能会将数据域中加一个序号元素，比如把所有选课学生导入一张 excel 表格后，自动会对所有元素进行编号，再次访问时，若使用顺序表的访问就不必遍历整张表而是可以以 $O(1)$ 的复杂度直接访问目标元素，这是顺序表顺序存储的优势所在。

3.1.3 调试与改进分析

本问题在编写和调试过程中遇到的主要困难是众多函数的交叉和众多类型和指针的混杂，容易导致边界条件出错或者指针指向错误或者数据溢出。由于函数众多，执行到出现问题的那一步可能成本较高，调试时可以构造测试程序对某一部分进行测试，得出正确结果再在源程序上修改。由于程序比较长，在书写过程中尽可能的少增加全局变量和静态变量，将每一部分的功能封装在函数中，这样如果某一个功能出问题不必牵连全文修改，而只需要关注某一个函数。这是未来做大项目以及工程中必不可少的编程习惯和思维。

本程序可读性有着较好的可读性，添加了众多注释，并且在程序界面进行了菜单选择、输入提示、功能完成成功和失败提示、输入错误提示等等。对于用户使用起来比较友好，使用性高，能完全根据界面的提示来进行操作。

如上一部分分析看来两种方式的时间复杂度差别不大。但是可以进行如下优化：在学生插入时按照学号大小顺序进行插入，即同曾经做过的用 $O(1)$ 的复杂度求队列最大值同样的方法，构造一个辅助链表，课程也进行同样操作，使 list 中学生和课程始终以某种顺序进行排列，那么删除时便可以根据学号的顺序知道删除的位置，这样用链表删除的操作时间复杂度可以缩小到 $O(1)$ 。

3.2 总结和收获

线性表是最常用且灵活的一种数据结构，作为基本的线性结构，我们在这一章学习的线性表主要包括顺序表和链表两种，二者各有其优劣。其中顺序表的结构简单、存储效率高，结构紧凑，可以直接存取。

但是涉及插入和删除操作时需要遍历许多数据元素，复杂度较高，对于事先不知道长度的线性表数据，需要预先分配较大空间并不断增加空间，操作麻烦。这时候就需要用到链表，链表虽然储存空间不连续。但是对于需要经常插入和删除的数据或问题，链表的操作更为简单，使用中也可以将链表进行改进，根据问题需要构造双向链表、循环链表等。此题虽然功能上对删除和插入的要求比较高，但是由于插入和删除的内容也需要遍历得到，所以实际上复杂度与顺序存储相差不大，考虑空间的利用效率链表更占优势，但是考虑实际情况，可能会多次访问表内元素，则运用顺序表更为方便。

本次实验是一项比较浩大的工程，实现了很多功能以及可视化的菜单选择，通过完成这次实验，我极大的提高了自己编写长程序的能力和纠错调试能力，也通过线性表和顺序表的对比深刻的理解了二者的基本操作、优劣之处，分析了各自的适用范围。