

《数据结构》课程设计总结

学号: _____

姓名: _____

专业: _____

目 录

| | |
|-----------------------|---|
| 第一部分 算法实现设计说明..... | 1 |
| 1.1 题目 | |
| 1.2 软件功能 | |
| 1.3 设计思想 | |
| 1.4 逻辑结构与物理结构 | |
| 1.5 开发平台 | |
| 1.6 系统的运行结果分析说明 | |
| 1.7 操作说明 | |
| 第二部分 综合应用设计说明..... | |
| 2.1 题目 | |
| 2.2 软件功能 | |
| 2.3 设计思想 | |
| 2.4 逻辑结构与物理结构 | |
| 2.5 开发平台 | |
| 2.6 系统的运行结果分析说明 | |
| 2.7 操作说明 | |
| 第三部分 实践总结..... | |
| 3.1. 所做的工作..... | |
| 3.2. 总结与收获..... | |
| 第四部分 参考文献..... | |

第一部分 算法实现设计说明

1.1 题目

给定一个有向图，完成：

- (1) 建立并显示出它的邻接链表；
- (2) 对该图进行拓扑排序，显示拓扑排序的结果，并随时显示入度域的变化情况；
- (3) 给出它的关键路径（要求：显示出 V_e , V_l , E , L , $L-E$ 的结果）。

1.2 软件功能

该软件的功能为：

- (1) 能够输入顶点个数、弧的个数和顶点之间的关系。
- (2) 能够根据输入的顶点个数、弧的个数和顶点间关系，显示出图的邻接链表。
- (3) 能够根据输入的顶点个数、弧的个数和顶点间关系，对图进行拓扑排序，显示拓扑排序的结果，并显示入度域的变化情况
- (4) 能够给出该图的关键路径。

1.3 设计思想

实现思路：

根据题意，将程序分为三个部分：

- (1) 输入基本信息

通过一个文本框接收顶点的个数，一个文本框接收弧的个数，一个文本框接收各个顶点之间的关系。将这些内容保存于后端当中。

```
ALGraph G;
int vexnum=ui->lineEdit_vexnum->text().toInt();
G.vexnum=vexnum;
int arcnum=ui->lineEdit_arcnum->text().toInt();
G.arcnum=arcnum;
QString Qvertices=ui->lineEdit_vertices->text();
string vertices=Qvertices.toStdString();

int i, j, k, w;
char v1, v2;
ArcNode *p;
for (i = 0; i < G.vexnum; i++)
{
    G.vertices[i].data = vertices[i];
    //scanf("%c", &G.vertices[i].data);
    G.vertices[i].firstarc = NULL;
}
/*qDebug()<<"vexnum:"<<vexnum
<<"arcnum:"<<arcnum
<<"vertices:"<<Qvertices;*/
QString Qrelations=ui->plainTextEdit_relation->toPlainText();
string relations=Qrelations.toStdString();
```

- (2) 对图进行操作

该部分分为三个子问题：显示邻接链表、拓扑排序和显示关键路径。

①邻接链表通过(1)中接收的数据，将邻接链表显示出来

对输入的字符串进行处理，将字符串中的顶点提取出来。

通过顶点之间的关系，将每个顶点指向的下一顶点作为链表的一个结点插入到该顶点的后方。

最后遍历各个顶点，并打印各个顶点所指向的下一个顶点。

```
for (k = 0; k < G.arcnum; k++)
{
    v1=relations[k*6];
    v2=relations[k*6+2];
    w=int(relations[k*6+4]-'0');
    qDebug()<<v1<<v2<<w;
    i = LocateVex(G, v1);
    j = LocateVex(G, v2);
    p = (ArcNode*)malloc(sizeof(ArcNode));
    p->adjvex = j;
    p->info = w;
    p->nextarc = G.vertices[i].firstarc;
    G.vertices[i].firstarc = p;
}
QString output;
for (int i = 0; i < G.vexnum; i++)
{
    if (G.vertices[i].firstarc != NULL)
    {
        output.append(G.vertices[i].data);
        ArcNode *p=G.vertices[i].firstarc;
        output.append("-->");
        output.append(QString::number(G.vertices[i].firstarc->adjvex,10));
        while (p->nextarc != NULL)
        {
            p = p->nextarc;
            output.append("-->");
            output.append(QString::number(p->adjvex,10));
        }
        output.append("\n");
    }
}
```

②拓扑排序通过(1)中接收的数据，对图进行拓扑排序

具体思路见下图程序中注释。

```
int indegree[100]; //创建记录各顶点入度的数组
FindInDegree(G, indegree); //统计各顶点的入度
//建立栈结构，程序中使用的是链表
stack* S;
initStack(&S);
//查找度为 0 的顶点，作为起始点
for (int i = 0; i < G.vexnum; i++)
{
    if (!indegree[i])
    {
        push(S, i);
    }
}
int count = 0;
QString output;
QString sortResult;
//当栈为空，说明排序完成
while (!StackEmpty(*S))
{
    int index;
    //弹栈，并记录栈中保存的顶点所在邻接表数组中的位置
    pop(S, &index);
    //printf("选择顶点%c\n", G.vertices[index].data);
    output.append("选择顶点");
    output.append(G.vertices[index].data);
    sortResult.append(G.vertices[index].data);
```

```

output.append("\n");
++count;
//依次查找跟该顶点相链接的顶点，如果初始入度为 1，当删除前一个顶点后，该顶点入度为 0
for (ArcNode* p = G.vertices[index].firstarc; p; p = p->nextarc)
{
    VertexType k = p->adjvex;
    if (!(--indegree[k]))
    {
        //顶点入度为 0，入栈
        push(S, k);
    }
}
//printf("入度: \n");
output.append("入度: ");
output.append("\n");
for (int i = 0; i < G.vexnum; i++)
{
    //printf("%c:%d ", G.vertices[i].data, indegree[i]);
    output.append(G.vertices[i].data);
    output.append(":");
    output.append(QString::number(indegree[i],10));
    output.append(" ");
}
//printf("\n");
output.append("\n");
}
//如果 count 值小于顶点数量，表明该有向图有环
if (count < G.vexnum)
{
    //printf("该有向图有环\n");
    output.append("该有向图有环");

    return;
}
else
{
    output.append("拓扑排序完成\n");
    output.append("排序结果: "+sortResult);
    //printf("拓扑排序完成\n");
}

ui->textEdit_output->setText(output);
}

```

③关键路径通过(1)中接收的数据，找出关键路径并将其显示出来

```

int e,l;
int *Ve, *VL;
//ArcNode *p;
Ve = new int[G.vexnum];
VL = new int[G.vexnum];
for (i = 0; i < G.vexnum; i++)
    Ve[i] = 0;
for (i = 0; i < G.vexnum; i++)
{
    ArcNode *p = G.vertices[i].firstarc;
    while (p!=NULL)
    {
        k = p->adjvex;
        if (Ve[i] + p->info > Ve[k])
            Ve[k] = Ve[i] + p->info;
        p = p->nextarc;
    }
}
for (i = 0; i < G.vexnum; i++)
    VL[i] = Ve[G.vexnum - 1];
for (i = G.vexnum - 2; i >= 0; i--)
{
    p = G.vertices[i].firstarc;
    while (p!=NULL)
    {
        k = p->adjvex;
        if (VL[k] - p->info < VL[i])
            VL[i] = VL[k] - p->info;
        p = p->nextarc;
    }
}
}

```

```

for (i = 0; i < G.vexnum; i++)
{
    p = G.vertices[i].firstarc;
    while (p != NULL)
    {
        k = p->adjvex;
        e = Ve[i];
        l = Vl[k] - p->info;
        char tag = (e == l) ? '*' : ' ';
        output.append("(");
        output.append(G.vertices[i].data);
        output.append(",");
        output.append(G.vertices[k].data);
        output.append("),e=");
        output.append(QString::number(e,10));
        output.append(",l=");
        output.append(QString::number(l,10)+tag);
        output.append("\n");
        p = p->nextarc;
    }
}
ui->textEdit_output->setText(output);

```

1.4 逻辑结构与物理结构

逻辑结构：链表、数组（线性表）。

物理结构：边结点和单个顶点信息由结构体来表示，多个定点信息通过数组将单个顶点信息相互联系起来，邻接表通过结构体来封装定点信息和图的基本信息来实现。

具体表示如下图所示：

```

typedef int VertexType;

typedef struct ArcNode
{
    int adjvex; //该弧指向的顶点的位置
    struct ArcNode *nextarc; //指向下一条弧的指针
    int info; //结点信息，此处指权重
}ArcNode; //边结点类型

typedef struct VNode
{
    VertexType data; //顶点信息
    ArcNode *firstarc; //指向第一条依附该顶点的指针
}VNode, AdjList[MAX_VERTEX_NUM];

typedef struct
{
    AdjList vertices; //邻接表
    int vexnum, arcnum;
}ALGraph;

int LocateVex(ALGraph G, char u);
void CreateALGraph_adjlist(ALGraph &G);
void PrintGraph_adjlist(ALGraph G);

```

1.5 开发平台

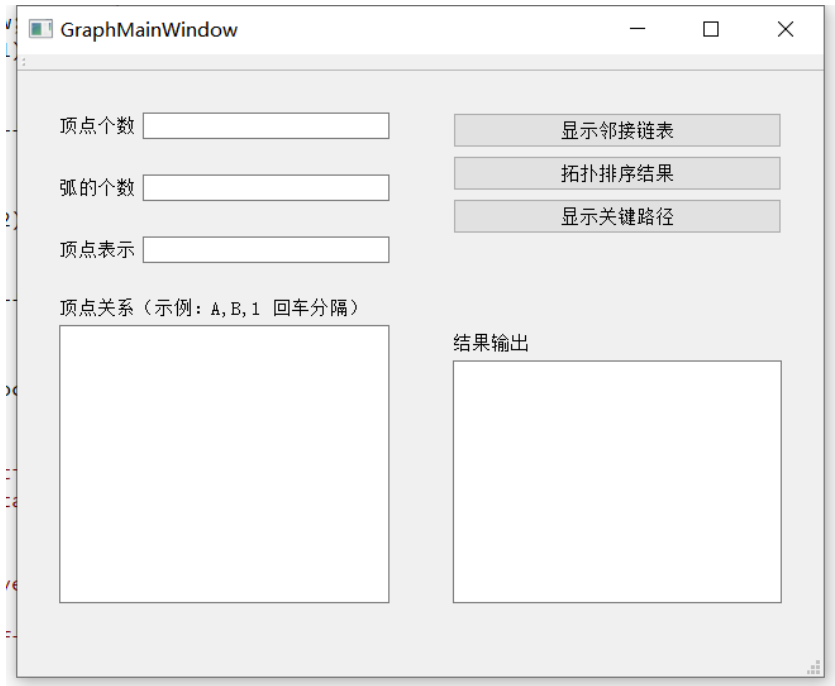
开发平台：Qt5

运行环境：Windows 计算机

1.6 系统的运行结果分析说明

1.6.1 系统运行结果如下：


如下图所示为初始页面。需要输入顶点个数、弧的个数、顶点表示和顶点关系。



The initial window titled "GraphMainWindow" contains the following elements:

- Input fields for "顶点个数" (Number of vertices), "弧的个数" (Number of edges), and "顶点表示" (Vertex representation).
- A text label "顶点关系 (示例: A, B, 1 回车分隔)" (Vertex relationship (example: A, B, 1, separated by Enter)).
- A large empty text area for inputting vertex relationships.
- Three buttons on the right: "显示邻接链表" (Show adjacency list), "拓扑排序结果" (Topological sort result), and "显示关键路径" (Show critical path).
- A "结果输出" (Result output) text label above a large empty text area for displaying results.

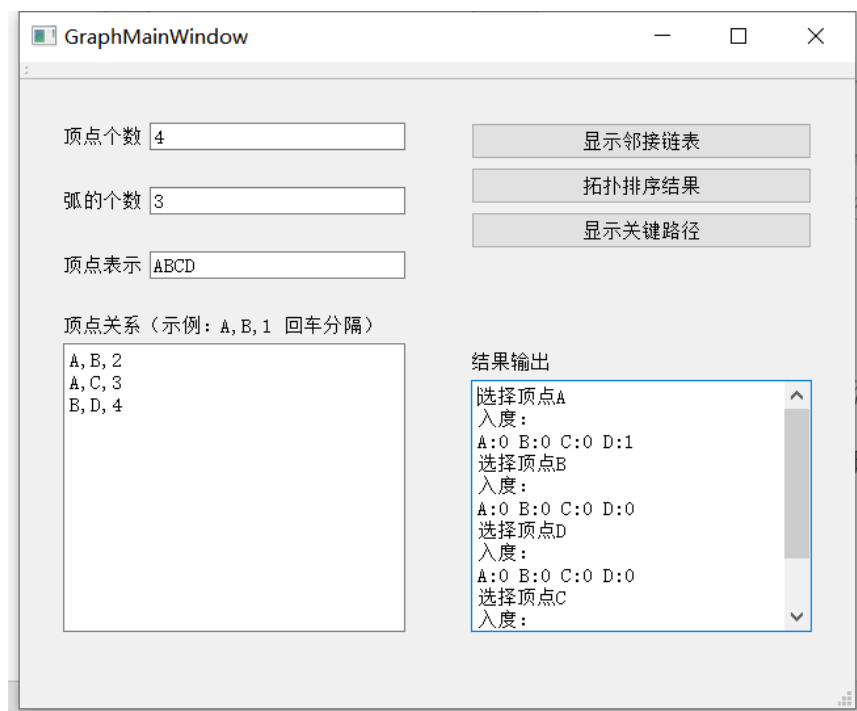
输入图的基本信息后，点击“显示邻接链表”，结果输出框输出邻接链表的表示。



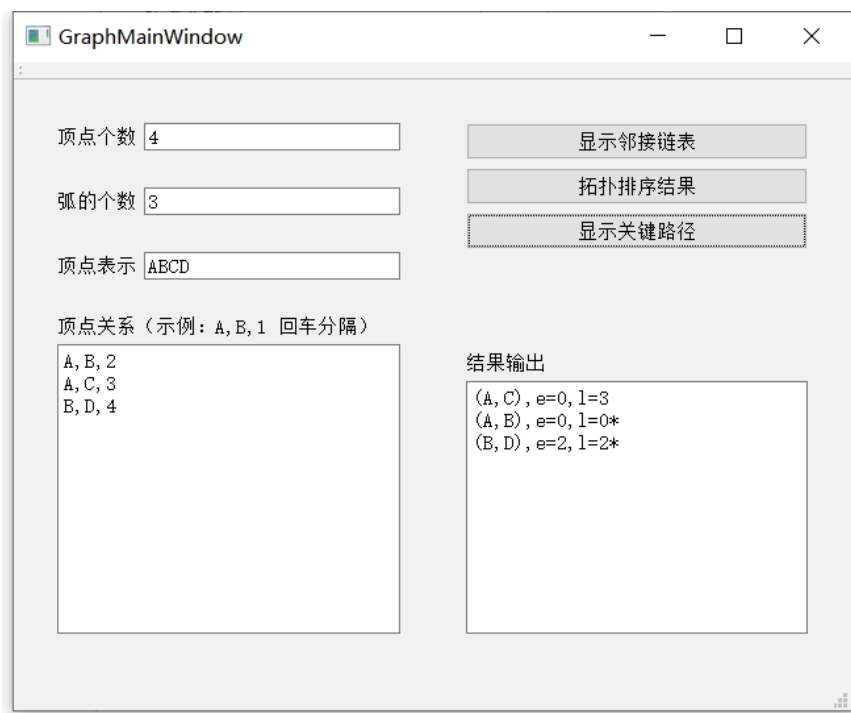
After clicking "显示邻接链表", the interface is updated as follows:

- The "顶点个数" field now contains the value "4".
- The "弧的个数" field now contains the value "3".
- The "顶点表示" field now contains the text "ABCD".
- The "顶点关系" input area now contains the text:
A, B, 2
A, C, 3
B, D, 4
- The "显示邻接链表" button is highlighted with a dashed border.
- The "结果输出" area now displays the following adjacency list representation:
A-->2-->1
B-->3

点击“拓扑排序结果”，结果输出框输出拓扑排序结果和入度域变化。



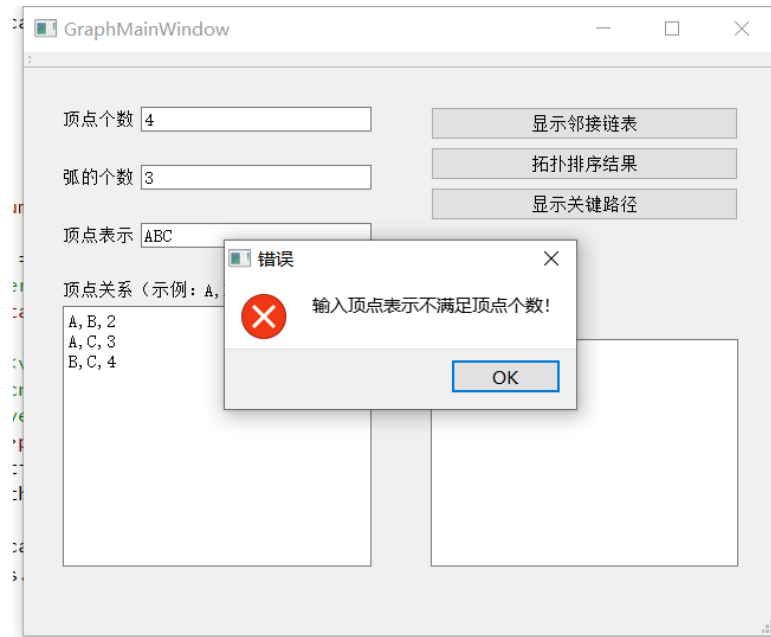
点击“显示关键路径”，结果输出框将输出关键路径的选取过程。



1.6.2 软件对输入错误的处理:

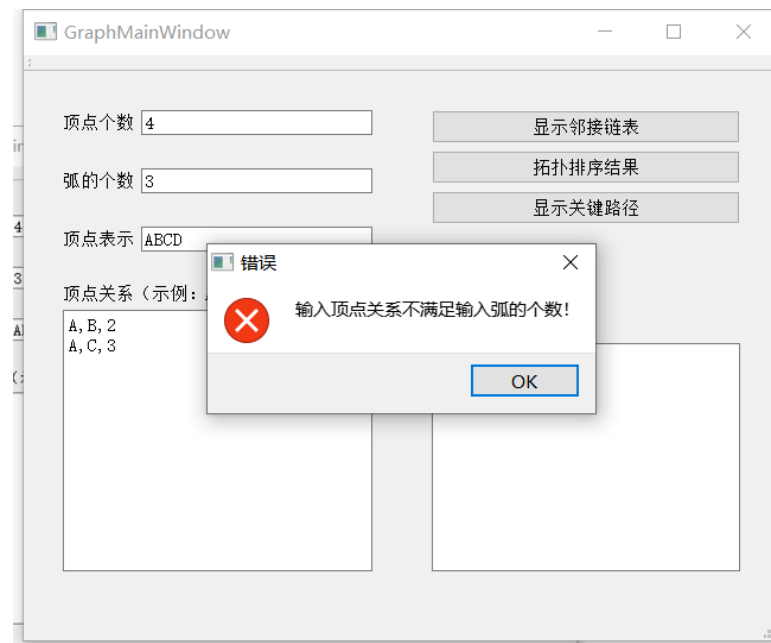
①输入的顶点表示与顶点个数不匹配

如下图所示，输入的顶点个数为4，但是顶点表示只输入了3个顶点，两者数量不匹配，点击右上角任意一个按钮，弹出错误提示框。



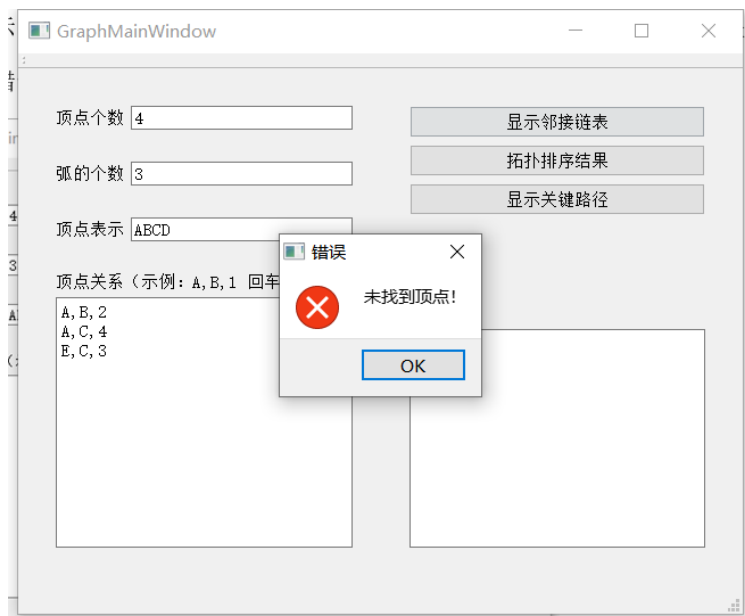
②输入的顶点关系不满足输入弧的个数

如下图所示，输入弧的个数为 3，但是只表示了 2 个弧的顶点关系，点击右上角任意一个按钮，弹出错误提示框。



③没有找到顶点

如下图所示，顶点表示为 ABCD，但是在输入顶点关系的时候出现了 E 顶点，属于没有输入的顶点，点击右上角任意一个按钮，弹出错误提示框。



1.7 操作说明



第二部分 综合应用设计说明

2.1 题目

某高校，教职工 9000 多人，其组织机构较为复杂，顶层分为学院、党群组织、行政机构、直属单位、附属单位等（可参考同济大学主页中介绍）。每一组织部门又分为多个层次，学校各教职工依据职位和角色可以隶属于多个组织部门。现需设计一个组织机构管理系统，系统需管理各级部门及各部门教职工，各部门需设立主管职位一名、主管副职多名、其他人员多名。

（1）动态建立组织结构，即可插入、删除部门等。

(2) 在部门中添加各类型人员，可定义职位。注意一个人可以在多个部门工作，因此人员需用线性表单独存储。

(3) 根据输入的人员名，查找其所在的部门、职位等信息。

2.2 软件功能

该软件的功能为：

- (1) 插入、删除部门。
- (2) 在部门中添加人员并定义该人员的职位。
- (3) 查询人员的所在部门、职位等信息。

2.3 设计思想

根据题意，将程序分为三个部分：

①对部门的操作

对部门的操作选择的数据结构为链表（线性表），这样插入、删除部门的时间复杂度较小，操作更加迅速。插入、删除的操作为链表的插入、删除的操作。

由于链表的插入涉及链表本来为空和链表已经有结点的情况，因此分类讨论。若链表为空，即需要创建第一个结点，则用创建链表的过程，加入一个结点；若链表不为空，则直接使用插入链表结点的方法，插入结点。

添加过程：

```
void MainWindow::on_pushButton_insertDepartment_clicked()
{
    QString output;
    QString Qdepartment=ui->lineEdit_inputDepartment->text();
    string Sdepartment=Qdepartment.toStdString();
    const char* department=Sdepartment.c_str();
    if(_list==NULL)
    {
        CreateList_L(_list,1,department);
    }
    else
    {
        ListInsert_L(_list,1,department);
    }
    //输出
    LinkList p;
    p = _list->next;
    output.append("当前部门: \n");
    //qDebug()<<"当前部门: \n";
    while (p)
    {
        //printf("%s\n", p->data);
        //qDebug()<<p->data;
        output.append(p->data);
        output.append("\n");
        p = p->next;
    }
    ui->lineEdit_inputDepartment->clear();
    ui->textEdit_output->setText(output);
}
```

删除过程：

```

void MainWindow::on_pushButton_deleteDepartment_clicked()
{
    QString output;
    QString Qdepartment=ui->lineEdit_inputDepartment->text();
    string Sdepartment=Qdepartment.toStdString();
    const char* department=Sdepartment.c_str();
    ListDelete_L(_list,department);
    //输出
    LinkList p;
    p = _list->next;
    //qDebug()<<"当前部门: ";
    output.append("当前部门: \n");
    while (p)
    {
        //printf("%s\n", p->data);
        //qDebug()<<p->data;
        output.append(p->data);
        p = p->next;
    }
    ui->lineEdit_inputDepartment->clear();
    ui->textEdit_output->setText(output);
}

```

②对人员的添加操作

对人员的添加操作使用的数据结构为数组（线性表），由于每个部门会有专门对人员个数的记录，则可以直接通过该个数对人员进行添加操作，此方式通过牺牲一个整数的空间，达到时间复杂度最小的结果。同时建立人员的线性表，对人员的姓名、部门和职位进行记录。

```

void MainWindow::on_pushButton_insertStaff_clicked()
{
    //字符串转换
    QString output;
    QString Qdepartment=ui->lineEdit_inputDepartment_staff->text();
    string Sdepartment=Qdepartment.toStdString();
    const char* department=Sdepartment.c_str();

    QString QstaffName=ui->lineEdit_inputStaff_insert->text();
    string SstaffName=QstaffName.toStdString();
    const char* staffName=SstaffName.c_str();

    QString Qjob=ui->lineEdit_inputJob->text();
    string Sjob=Qjob.toStdString();
    const char* job=Sjob.c_str();

    //函数调用
    Link_Insert(_list, department, staffName,job);
    ListInsert_Staff(S,staffName, department,job);
    //qDebug()<<S[0].staffName;

    //输出
    LinkList p;
    p = _list->next;
    //qDebug()<<"当前部门: ";
    output.append("当前部门: \n");
    while (p)
    {
        //printf("%s\n", p->data);
        //qDebug()<<p->data;
        output.append(p->data);
        output.append("\n");
        p = p->next;
    }
    output.append(QstaffName+"信息插入成功! ");
    ui->lineEdit_inputDepartment_staff->clear();
    ui->lineEdit_inputStaff_insert->clear();
    ui->lineEdit_inputJob->clear();
    ui->textEdit_output->setText(output);
}

```

③对人员的查询操作。

通过遍历人员的线性表，对人员进行查询。将人员的姓名进行匹配，若匹配到，则输出该人员的信息。

2.4 逻辑结构与物理结构

逻辑结构：链表、数组（线性表）

物理结构：表示部门的链表通过结构体表示，部门内部的员工信息以及单独的员工名单由数组表示。

2.5 开发平台

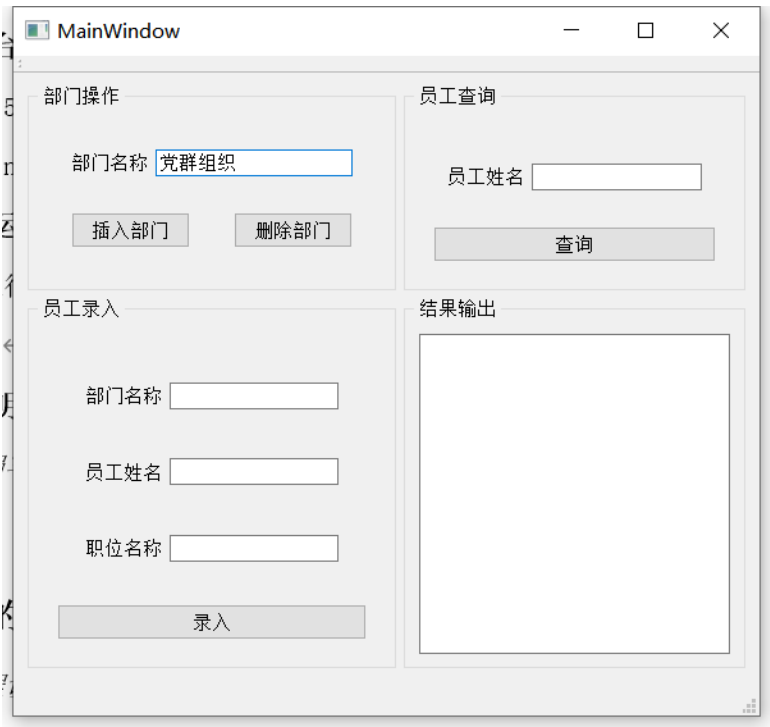
开发平台：Qt5

运行环境：Windows 计算机

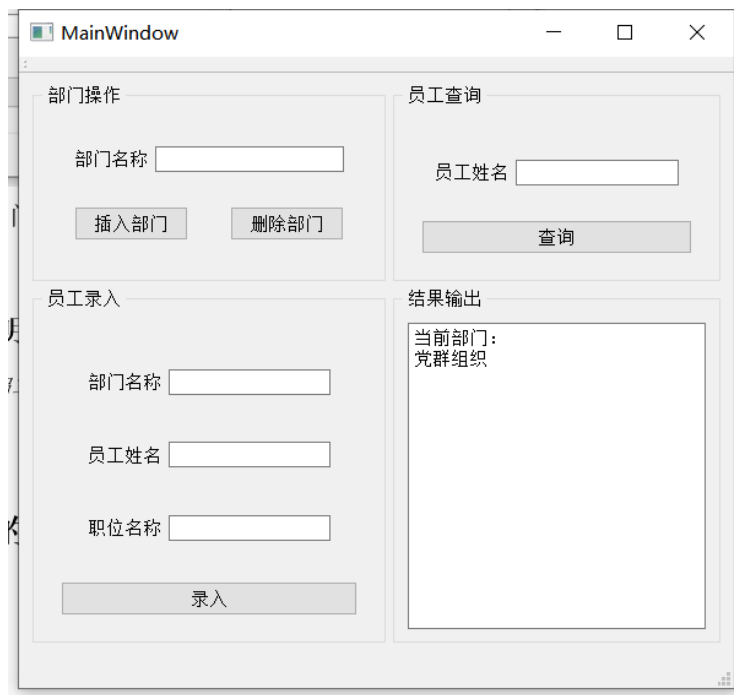
2.6 系统的运行结果分析说明

2.6.1 系统运行结果如下：

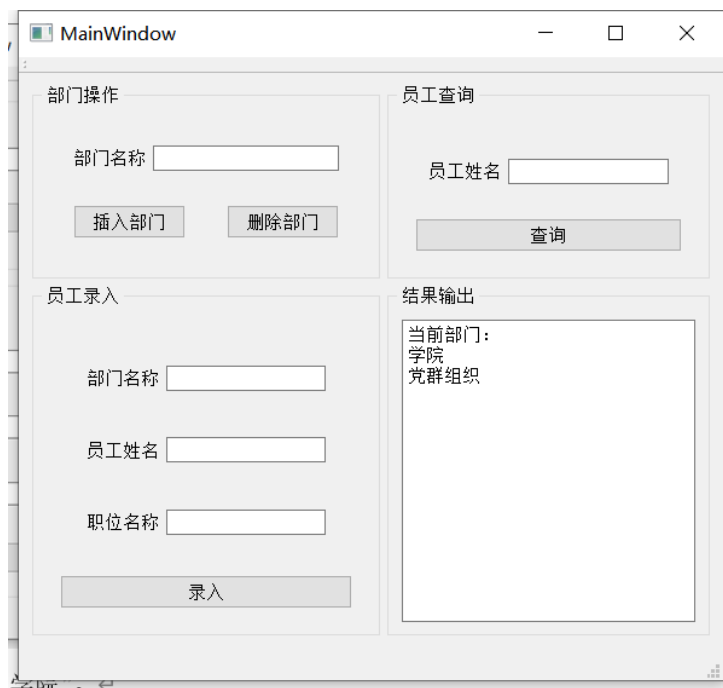
输入部门名称



点击“插入部门”，结果输出框则显示当前部门的结果

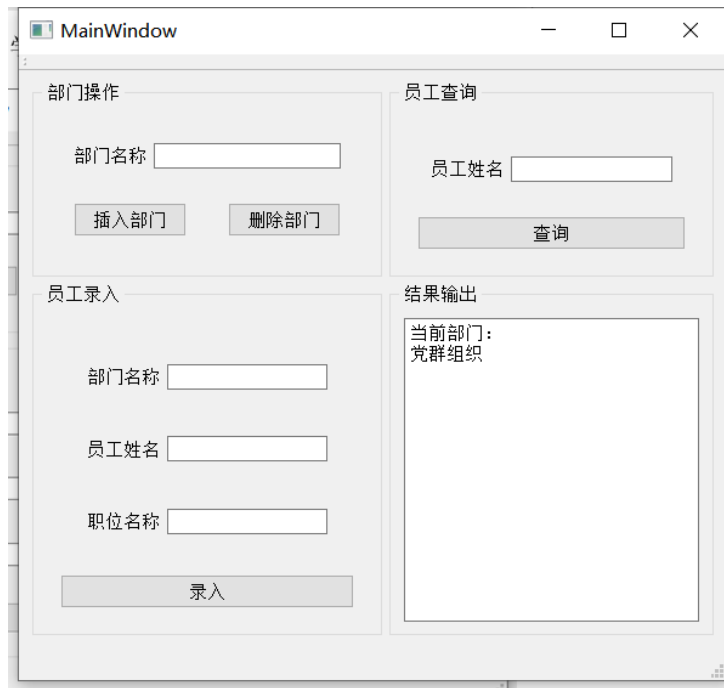


再添加部门“学院”，输出当前部门为学院和党群组织。

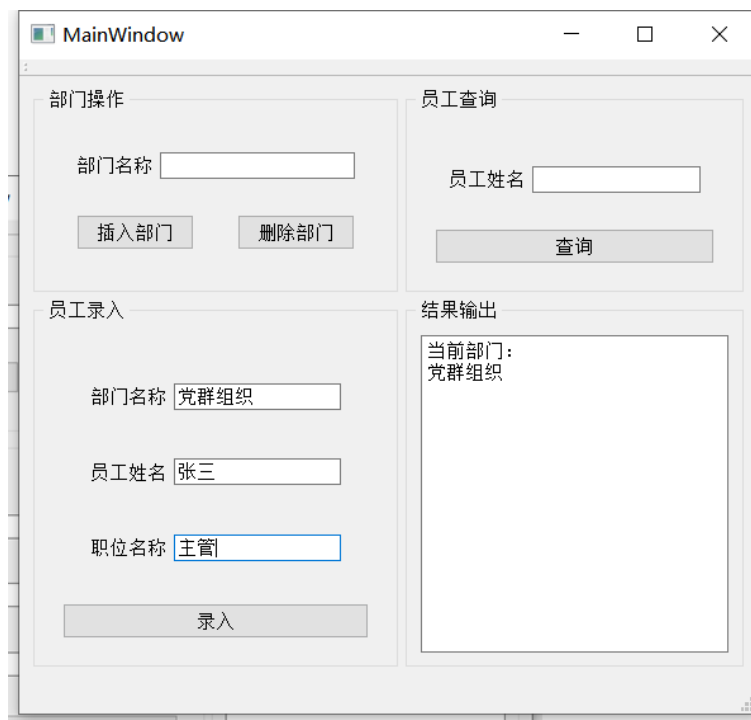


输入部门名称“学院”后，点击“删除部门”。

当前部门中删除学院，并输出当前部门为党群组织。



在员工信息一栏中依次输入部门名称、员工姓名和职位名称。



点击“录入”，将成功把员工信息录入系统当中，且结果输出框会提醒录入员工成功。

The screenshot shows a window titled "MainWindow" with a standard Windows-style title bar (minimize, maximize, close buttons). The window is divided into four main sections:

- 部门操作 (Department Operation):** Contains a text input field for "部门名称" (Department Name) and two buttons: "插入部门" (Insert Department) and "删除部门" (Delete Department).
- 员工查询 (Employee Query):** Contains a text input field for "员工姓名" (Employee Name) and a "查询" (Query) button.
- 员工录入 (Employee Entry):** Contains three text input fields for "部门名称" (Department Name), "员工姓名" (Employee Name), and "职位名称" (Job Title), followed by a "录入" (Enter) button.
- 结果输出 (Result Output):** A large text area displaying the following text:
当前部门:
党群组织
张三信息插入成功!

录入成功以后，查询员工的信息。

This screenshot shows the same "MainWindow" application after the "查询" (Query) button has been clicked. The changes are as follows:

- 员工查询 (Employee Query):** The "员工姓名" (Employee Name) input field now contains the text "张三" (Zhang San).
- 结果输出 (Result Output):** The text area now displays the following text:
当前部门:
党群组织
张三信息插入成功!

点击“查询”，结果输出框中会输出当前查询员工的姓名、部门和职位信息。

MainWindow

部门操作

部门名称

员工查询

员工姓名

员工录入

部门名称

员工姓名

职位名称

结果输出

姓名: 张三
部门: 党群组织
职业: 主管

再加入部门“学院”，并给这个部门录入员工“张三”，此时张三有两个部门的职位。

MainWindow

部门操作

部门名称

员工查询

员工姓名

员工录入

部门名称

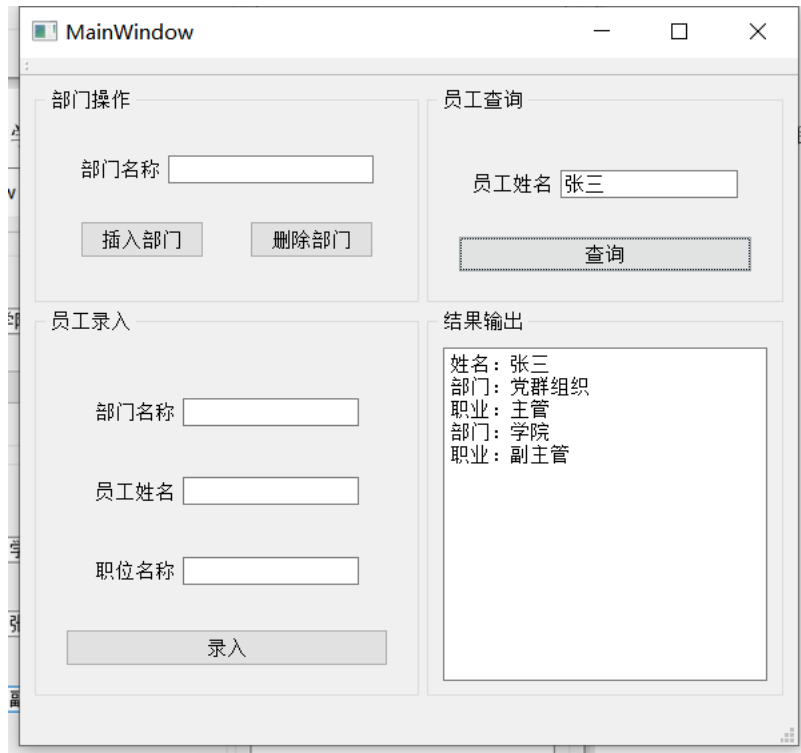
员工姓名

职位名称

结果输出

姓名: 张三
部门: 党群组织
职业: 主管

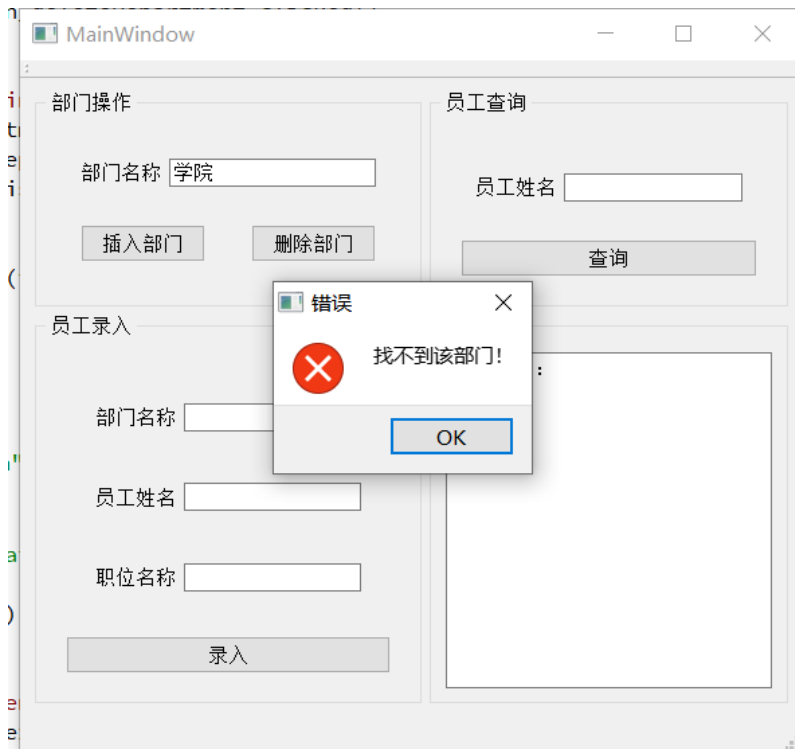
再次点击“查询”，将输出张三的所有信息。



2.6.2 软件对输入错误的处理:

①未找到需要删除的部门

在只输入了“党群组织”这一部门的前提下，输入部门学院，点击删除部门，则弹出错误提示框。

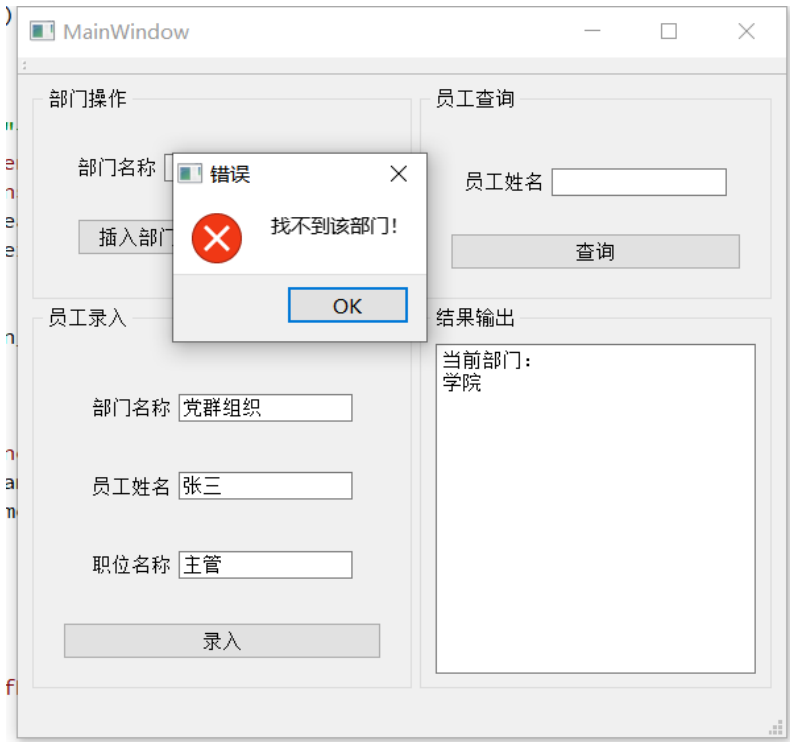


此操作需要在原有代码基础上修改，如下图所示。

```
int judge=ListDelete_L(_list,department);
if(judge==0)
{
    QMessageBox::critical(this,"错误","找不到该部门! ");
}
```

②未找到录入员工对应的部门

在当前的部门只有学院的情况下，若录入党群组织的员工姓名，则弹出错误提示框。

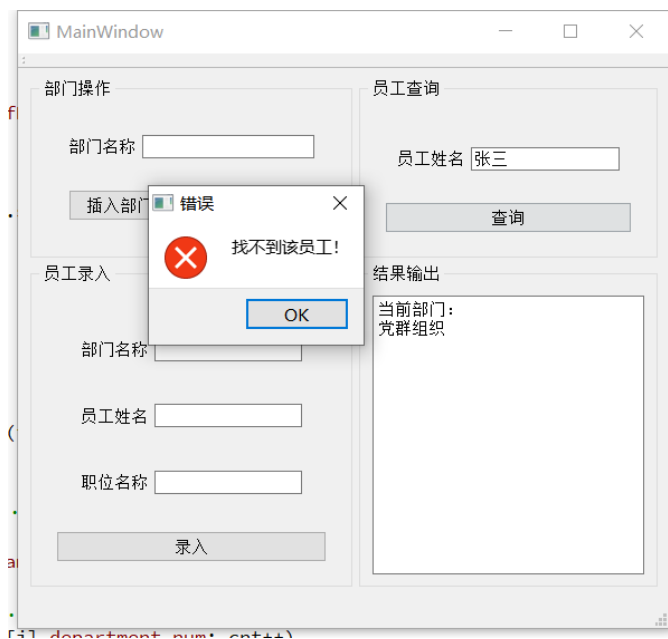


此操作需要在原有代码基础上修改，如下图所示。

```
int judge=Link_Insert(_list, department, staffName,job);
if(judge==0)
{
    QMessageBox::critical(this,"错误","找不到该部门! ");
    return;
}
```

③未找到查询的员工

当没有录入员工张三的情况下，查询“张三”，将弹出错误提示框。



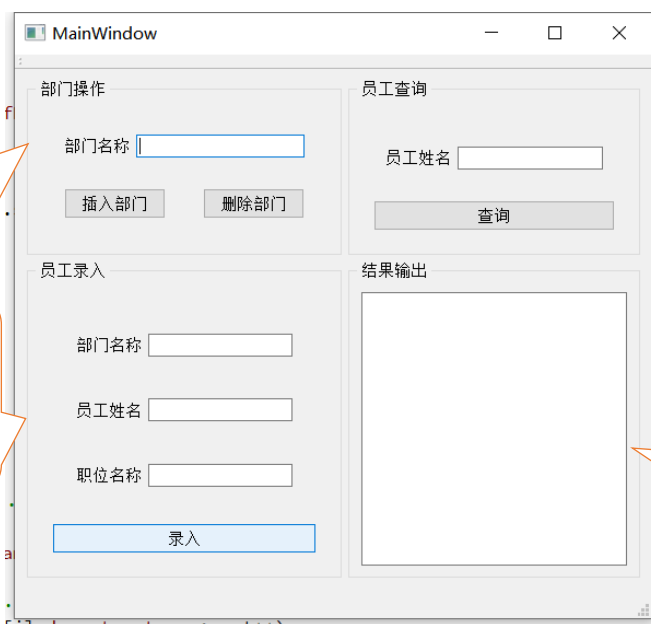
此操作需要在原有代码基础上修改，如下图所示。

```
bool flag=false;
while (true)
{
    if (!strcmp(S[i].staffName, "\0"))
    {
        break;
    }
    else if (!strcmp(S[i].staffName, Name))
    {
        flag=true;
        break;
    }
    i++;
}
if(flag==false)
{
    QMessageBox::critical(this,"错误","找不到该员工!");
    return;
}
```

2.7 操作说明

对部门的操作，在部门名称输入框中输入部门，可对部门进行插入和删除

员工录入部分，依次输入部门名称、员工姓名和职位名称



员工查询部分，输入员工姓名，即可查询员工的部门和职位信息

结果输出框，前三个操作的结果都可在此处显示

第三部分 实践总结

3.1. 所做的工作

3.1.1 后端

- (1) 准备阶段：理解题意，决定使用的物理结构和逻辑结构。
- (2) 代码编写阶段：根据决定的数据结构，按照所给题目顺序，依次完成每一道题的要求。在完成初步的代码以后，通过控制台进行数据的测试，确保结果正确。
- (3) 后期阶段：对代码进行完善，使程序更加简洁易懂。

3.1.2 前端

- (1) 准备阶段：选择开发平台，最终选择 Qt5 作为开发的主要工具。决定开发工具以后，对 Qt5 的主要功能进行了解和学习，掌握 Qt5 的基本用法。
- (2) 代码编写阶段：通过 Qt5 的 ui 设计功能，设计出用户交互界面，并与后端的代码联系起来，此时对后端代码进行修改，使前后端相匹配。
- (3) 后期阶段：最后对程序进行错误处理，对代码进行最后的修改，让程序能在输入错误的情况下作出相应的反应。

3.2. 总结与收获

在大二上学期学习计算机科学导论的时候有承担过前端 gui 的制作，但是效果并不是很理想。数据结构课程设计这门课程让我再次接触了前端制作，并有了更多的感悟，并总结了更多的经验：

- (1) 在制作此类程序的时候，首先搭好框架，捋清需要完成的功能和功能之间的逻辑关系是怎样的，这样后期修改的时候会更加方便。
- (2) 在设计前端的时候，为了适配不同分辨率的设备和不同大小的窗口，应该先将程序的界面分为几个小块，每个小块中又细分小块，最后将不同小块打包成固定的窗口，这样无论窗口如何改变，都不会出现组件错位的现象。

在编写的过程中，我对数据结构的编写有了更加深刻的概念，并可以将数据结构应用到实际场景当中，这对我来说是个非常大的进步，希望以后可以更熟练地掌握数据结构和前端的开发。

第四部分 参考文献

- [1] (美) Jeffrey Richter. Windows 核心编程 (第 5 版). 机械工业出版社

[2] （美）George Shepherd, David Kruglinski. Visual C++.NET 技术内幕（第 6 版）.清华大学出版社

[3] QT5 编程入门教程. <https://blog.csdn.net/p1279030826/article/details/106546752>