**SHANGHAI  UNIVERSITY**

**<数据结构>研讨报告**

|  |  |  |  |
| --- | --- | --- | --- |
| **总分100** | | | **得分** |
| 报告 | 逻辑结构（40分） | |  |
| 语言表达（30分） | |  |
| 格式规范（20分） | |  |
| 个人体会（10分） | |  |
| **合 计** | | |  |
| **学    院** | **计算机工程与科学学院** | | |
| **组    名** | **陈宇鑫小组** | | |
| **题    目** | **学生选课系统** | | |
| **日    期** | **2023-1-11** | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **学号** | **姓名** | **专业** | **分工** |
| 22121402 | 陈宇鑫 | 人工智能 | 功能函数编写，主函数编写和测试 |
| 22121497 | 郑凯风 | 智能科学与技术 | 头文件中各功能函数的编写和测试 |
| 22120051 | 党昊天 | 智能科学与技术 | 功能函数编写，程序结构设计，bug修复 |
| 22121496 | 古彬玮 | 人工智能 | 代码测试，实验报告撰写 |

1. **目的与要求**

1. 掌握串、数组和广义表的逻辑结构定义和各种存储结构的实现。

2. 熟练运用串、数组和广义表的的各种存储结构以及各种基本操作。

3. 根据实际问题的需要，选择串、数组和广义表适合的存储结构解决问题。

**二、实验环境**

Windows操作系统

Visual Studio 2022

**三、实验内容**

**[问题描述]**假设一所学校有n（0≤n≤1000）个学生，m（0≤m≤200）门课程，一门课程的学生容量为sn（0≤sn≤45）人，一个学生最多可选5门课。

**[基本要求]**基于十字链表结构设计一个选课系统，主要功能要求如下：

    1) 系统初始运行时自动加载课程信息、学生信息及选课信息的数据文件；

    2) 查询某个学生的选课情况（输出所选课程及学分）；

    3) 修改某个学生的选课情况，包括选课和退选；

    4) 增加和删除某名学生；

    5) 查询某门课程的被选情况（包括剩余名额及选课学生信息）；

    6) 修改某门课程的学生容量；

    7) 增加和删除某门课程；

    8) 保存当前课程信息、学生信息及选课信息到指定文件。

**[测试数据]**

课程信息：course.txt（包含m=100门课程信息，有四列数据：课程号  课程名 学分 课程容量；初始数据中每门课程的学生容量均为45人）

学生信息：stu.txt（包含n=900名学生，有两列数据：学号姓名）

选课信息：selection.txt（包含一个n\*m的矩阵，每行按顺序对应stu.txt中的一名学生，每列按顺序对应course.txt中的一门课程；矩阵元素为1表示选课，0表示未选）

**四、实验内容的设计与实现**

在本次基于栈的区域填充实验中，我们按照要求使用十字链表实现学生和课程的存储。

下面介绍一下我们组**十字链表**是如何存储数据的。  
1.CrossNode 类：

CrossNode 类用于表示十字链表的节点。这个节点包含两个指针，down 和 right，分别指向下一个学生节点和下一个课程节点。

2.CrossList 类：

CrossList 类是主要的数据结构，表示了整个十字链表。该类包含了学生和课程的头节点，以及一个二维数组用于表示学生和课程的选课关系。学生头节点的 down 指针指向学生节点链表的开头，课程头节点的 right 指针指向课程节点链表的开头。

二维数组中的每个元素表示一个学生选修一个课程的情况，通过该数组可以高效地检索学生和课程的选课关系。

3.学生和课程节点的添加：

使用 add\_student 和 add\_courses 方法，可以向十字链表中添加新的学生和课程节点。这样就在链表中建立了学生和课程的基本信息。

4.选课关系的表示：

使用 SetElem 方法可以设置稀疏矩阵中学生选课的关系。该方法允许在矩阵中标记学生和课程之间的选课关系，用于表示学生的选课情况。

5.选课系统的操作：

提供了学生登录、选课、退课、查询课程信息、修改课程学生容量等一系列操作。这些操作通过调用 CrossList 类的方法，利用十字链表结构来进行高效的数据检索和修改。

通过十字链表的实现，该代码在存储和管理学生和课程的选课关系上具有高效性，避免了存储冗余信息的空间浪费。这种数据结构的选择使得在实际操作中能够迅速地找到所需的信息，实现了一个简单而有效的选课系统。

下面是十字链表构建的关键代码段：

|  |
| --- |
| **1.CrossNode定义**  template<class ElemType>  class CrossNode  {  public:      ElemType data;            // 数据域，通常表示学生或课程信息  CrossNode<ElemType>\* right; // 右指针，指向下一个相同类型的节点（学生或课程）  CrossNode<ElemType>\* down;  // 下指针，指向下一个节点（学生节点指向下一个学生，课程节点指向下一个课程）  // 构造函数等相关成员函数  };  **2.CrossList定义**  template<class ElemType>  class CrossList  {  private:  CrossNode<ElemType>\* student\_head; // 学生头节点  CrossNode<ElemType>\* course\_head;  // 课程头节点      int matrix[MaxStuNum][MaxCouNum]; // 二维数组，表示选课关系  public:  // 构造函数等相关成员函数      void add\_student(const Student<ElemType>& s);      void add\_courses(const Course<ElemType>& c);      void SetElem(int i, int j, int value);  // 其他操作函数...  };  **3.学生和课程节点的添加**  template<class ElemType>  void CrossList<ElemType>::add\_student(const Student<ElemType>& s)  {  // 创建新的学生节点，将其插入学生链表中  }  template<class ElemType>  void CrossList<ElemType>::add\_courses(const Course<ElemType>& c)  {  // 创建新的课程节点，将其插入课程链表中  }   1. **选课关系的表示**   template<class ElemType>  void CrossList<ElemType>::SetElem(int i, int j, int value)  {   // 将 matrix[i][j] 设置为 value，表示学生 i 选修了课程 j  } |

下面介绍我们的读文件过程。

1.学生信息文件 (stu.txt) 的读取过程：

通过 ifstream 打开名为 "stu.txt" 的文件流。

如果文件无法打开，输出错误信息并返回。

使用 while 循环从文件中读取学生学号和姓名信息。

在循环中，创建 Student 对象，将读取到的学号和姓名信息赋给该对象。

调用 C.add\_student(s) 将学生对象添加到十字链表中。

关闭文件流。

2.课程信息文件 (course.txt) 的读取过程：

通过 ifstream 打开名为 "course.txt" 的文件流。

如果文件无法打开，输出错误信息并返回。

使用 while 循环从文件中读取课程信息。

在循环中，创建 Course 对象，将读取到的课程信息赋给该对象。

调用 C.add\_courses(c) 将课程对象添加到十字链表中。

关闭文件流。

3.稀疏矩阵文件 (selection.txt) 的读取过程：

通过 ifstream 打开名为 "selection.txt" 的文件流。

如果文件无法打开，输出错误信息并返回。

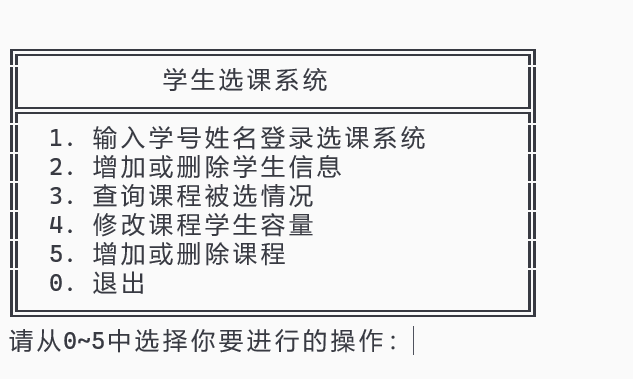
使用嵌套的 for 循环遍历稀疏矩阵的每个元素。

在循环中，读取一个整数，并调用 C.SetElem(i, j, value) 将该整数设置为稀疏矩阵中对应位置的值。

关闭文件流。

这些过程将文件中的学生信息、课程信息和稀疏矩阵的数据读取到相应的数据结构中，用于构建和初始化十字链表。

|  |
| --- |
| // 读学生  string filePath = "stu.txt";  ifstream inputFile(filePath);  if (!inputFile.is\_open()) {  cerr << "无法打开文件: " << filePath << endl;  return 1;  }  // 读取学生学号和姓名信息  int studentID;  string studentName;  while (inputFile >> studentID >> studentName) {  // 在这里可以处理读取到的学生学号和姓名，例如输出到控制台  Student<int> s(studentID, studentName);  C.add\_student(s);  }  inputFile.close();  //读课程  string filePath\_2 = "course.txt";  ifstream inputFile\_2(filePath\_2);  if (!inputFile\_2.is\_open()) {  cerr << "无法打开文件: " << filePath\_2 << endl;  return 1;  }  // 读取课程信息  string num;  string name;  int credit;  int capacity;  while (inputFile\_2 >> num >> name>>credit>>capacity) {  // 在这里可以处理读取到的课程信息，例如输出到控制台  Course<int> c(num, name,credit,capacity);  C.add\_courses(c);  }  inputFile\_2.close();  //读稀疏矩阵  int i = 0, j = 0;  int value;  string filePath\_3 = "selection.txt";  ifstream inputFile\_3(filePath\_3);  if (!inputFile\_3.is\_open()) {  cerr << "无法打开文件: " << filePath\_3 << endl;  return 1;  }  for (i = 0;i < 900;i++) {  for (j = 0;j < 100;j++) {  inputFile\_3 >> value;  C.SetElem(i, j, value);  }  }  inputFile\_3.close(); |



选课界面如上，系统界面设计清晰简洁，以控制台形式呈现。

用户可以通过输入数字选择不同的操作，如1.登录选课系统、2.增加或删除学生信息、3.查询课程被选情况、4.修改课程学生容量、5.增加或删除课程，以及6.退出系统。整体布局采用了简约的表格形式，使用户能够轻松理解和操作。这种用户界面设计使得选课过程直观明了，提升了用户体验。

**五、收获与体会**

陈宇鑫：

首先，构思程序结构是整个实验的基础。我们需要设计一个灵活而又高效的系统，使得学生能够方便地选课，同时保证选课系统的稳定性和可扩展性。在这个阶段，我们考虑了如何利用十字链表来表示课程和学生之间的关系，以及如何设计用户界面使其用户友好。

小组合作解决各种程序中的报错是实验过程中的一个重要环节。在编写代码的过程中，难免会遇到各种bug和错误。通过团队协作，我们能够更快速地定位和解决这些问题。每个人都带来了独特的视角和经验，促使我们更全面地审视代码，确保系统的稳定性和可靠性。

文件读取是整个选课系统的一个关键功能。通过读取文件，我们能够实现对学生信息、课程信息等数据的导入和导出，使得系统更具实用性。在这一过程中，我们学习了如何处理文件IO，确保数据的正确性和完整性。

总的来说，这个选课系统实验不仅帮助我们理解了数据结构中的十字链表的应用，还培养了我们在团队协作中解决问题的能力。通过每个阶段的努力，我们成功地完成了一个功能完善的学生选课系统，这为我们将理论知识应用到实际中提供了宝贵的经验。这次实验让我深刻体会到，在编程的过程中，不仅仅是解决问题，更是一个不断学习和提升的过程。

郑凯风：

通过这次研讨，我深刻体会到十字链表类对于处理学生与课程这种相互映射的数据的高效性，学生可以选多门课，课程下又可以有多名学生，这种结构正对应了十字链表的结构。同时，在编写过程中，我深刻体会到初期对于程序底层结构的设计的重要性，不同与其他小组，我们将行列数组单独设计成了区别于十字链表节点类的学生和课程类数组，这为之后的逻辑实现提供了便利也制造了困难，也让我积累了宝贵的经验。总之，这次研讨让我受益匪浅，在今后的研讨中我会更加积极不断提升自己的能力。

古彬玮：

在编写这段代码的过程中，我们深刻体会到数据结构在实际问题中的应用。通过实现十字链表，我们成功地建立了一个学生选课系统，有效地存储和管理学生、课程以及选课关系。这次经历让我更熟悉了文件读取、链表操作等基础编程技能，并提高了对数据结构设计的理解。同时，通过实现选课系统，我感受到了编程对问题建模和解决的能力，对我今后的学习和工作具有积极的影响。

党昊天： 通过完成十字链表的实际应用，我深刻感受到合理结构对于数据管理的高效，同时时间与空间之间相对不可完全相互兼顾的矛盾。编写过程中，我深刻体会到对于复杂程序顶层设计的重要性，高级功能应在基础函数的基础上进行编写，而不是全重新编写。顶层设计与合理分工对于程序开发具有基础性的影响。这一系列的经验教训，对我今后的开发路程有着积极的贡献。

# **参考文献：**