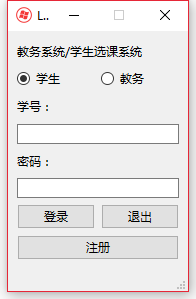
大作业报告：选课系统

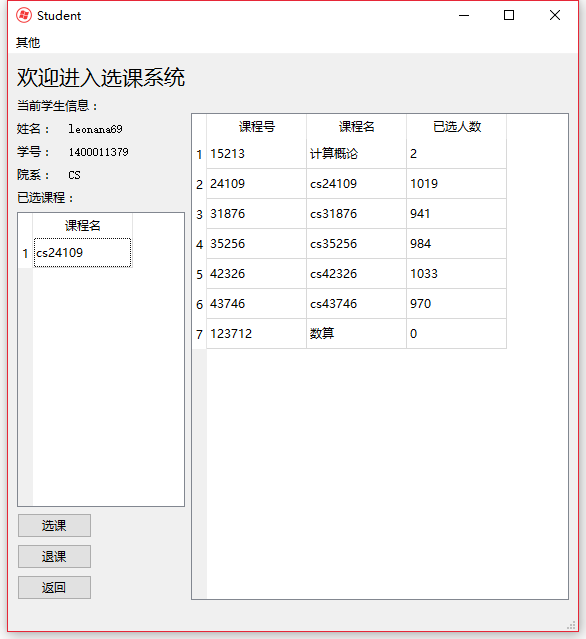
陈国俊 – 1400011379

1. 界面介绍

首先是登录界面，可以选择学生登录或者是教务登录，学生需要输入学号和密码，教务未设置账号，直接输入教务密码即可。新学生可以点击下方注册来创建账号。若初始没有数据文件“my\_data.mdt”，则会自动创建一个数据文件，并且教务的初始密码为“123456”。（文件夹中提供的数据教务密码为“12345678”，密码可在教务界面修改）

选择教务然后输入密码登录后会弹出一个教务界面，界面主要显示了所有学生（按学号排序）的信息以及他们的选课状况，院系之后的是各门课程。显示“Yes”的表示该学生选了该门课。



然后是学生界面（所有学生的密码初始为123456，学号可从教务界面查看），进入后会显示当前学生信息、已选课程和右侧的全部课程信息。

1. 功能介绍

教务：

1. 添加课程

在上方输入课程名称，在下方输入课程号，点击确认即可添加新的课程，若课程名或课程号与已有课程相同，则会提示出错，并要求重新输入。

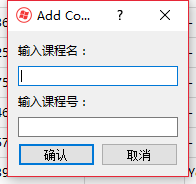
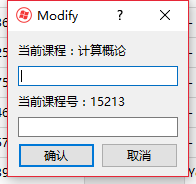


Figure 2修改课程

Figure 1添加课程

1. 删除课程

选中教务界面表格中的课程所在列的任意单列区域（点击一个课程名称），然后点击“删除课程”即可，若选区大于一列或者选区不在课程区域而在学生信息区域，均会有提示。

1. 修改课程

与删除课程选择的区域相同，选中之后点击“修改课程”便会弹出输入框，会显示当前课程名和课程号，可以输入一项或两项，然后点击确认，若与当前已有课程冲突或者两项均无输入则会出现提示。

1. 查看课程

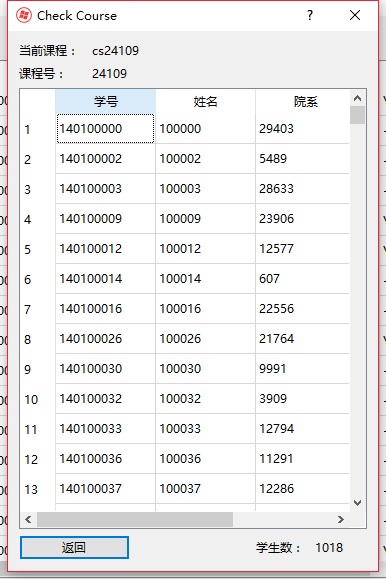
选中一门课程，点击“查看课程”即可显示所有选择了该门课程的人的信息，并显示选课总人数。

Figure 3查看课程

1. 修改密码

点击“修改密码”会弹出输入框，输入两次相同的密码，即可设置新密码。

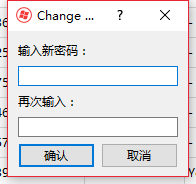


Figure 4修改密码

1. 代码解释

#main.cpp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48 | #include "enterWidget.h"  #include "my\_course.h"  #include <QtWidgets/QApplication>  /\*  \* Global variable  \*/  extern char pw[PW\_N];  extern vector<Course>courses; //全部课程  extern vector<Student>students; //全部学生  extern FILE \*fd; //文件  /\*  \* Main  \*/  int main(int argc, char \*argv[])  {  QApplication my\_course\_system(argc, argv);  /\*  \* 初始化  \*/  courses.clear();  students.clear();  if (!(fd = fopen("./my\_data.mdt", "rb+")))  //若不存在就创建一个  {  qDebug("No file now, creat a file......\n");  fd = fopen("./my\_data.mdt", "wb+");  strcpy(pw, "123456"); //默认初始密码  save();  }    else  /\* 存在就读入数据 \*/  read\_information();  /\* 自动生成数据 \*/  //generate\_data();  My\_Course instance;  instance.show();  return my\_course\_system.exec();  } |

该文件为主函数文件，main函数先初始化读入数据，然后新建一个My\_Course类（主要窗口），然后显示。

#add\_course.h

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | #pragma once  /\*  \* "Add course" window  \*/  #include "ui\_add\_course.h"  #include <QtWidgets/QMainWindow>  class Add\_course : public QDialog  {  Q\_OBJECT  public:  Add\_course(QWidget \*parent = Q\_NULLPTR);  QWidget \*parentWindow;  private slots: //槽函数  void pushButton\_enter\_clicked();  void pushButton\_return\_clicked();  signals:  void add\_course\_event();  private:  int equal\_flag; //判断是否与已有课程重复  Ui::add\_course\_Dialog uia;  }; |

#add\_course.cpp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59 | #include "add\_course.h"  #include "enterWidget.h"  #include "my\_course.h"  #include "my\_message.h"  #include <qdebug.h>  /\* Extern variables \*/  extern vector<Course>courses;  /\* Add\_course init function \*/  Add\_course::Add\_course(QWidget \*parent) : QDialog(parent)  {  parentWindow = parent;  setWindowIcon(QIcon(":/Resources/windows.png"));  setFixedSize(180, 145);  uia.setupUi(this);  /\* Signals \*/  connect(uia.pushButton\_enter, SIGNAL(clicked()), this, SLOT(pushButton\_enter\_clicked()));  connect(uia.pushButton\_return, SIGNAL(clicked()), this, SLOT(pushButton\_return\_clicked()));  }  void Add\_course::pushButton\_enter\_clicked()  {  if (uia.lineEdit\_name->text().length() == 0 || uia.lineEdit\_num->text().length() == 0)  myMessageBox(u8"信息不准为空");  else if (get\_course(uia.lineEdit\_name->text().toStdString()) != courses.end()  || get\_course(uia.lineEdit\_num->text().toStdString()) != courses.end())  myMessageBox(u8"课程号或课程名重复");  else if (!isNum(uia.lineEdit\_num->text().toStdString()))  myMessageBox(u8"课程号必须为数字");  else  {  Course \*ctmp = new Course;  ctmp->set\_name(uia.lineEdit\_name->text().toStdString().c\_str());  ctmp->set\_num(atoi(uia.lineEdit\_num->text().toStdString().c\_str()));  ctmp->students.clear();  courses.push\_back(\*ctmp);  //qDebug("Name: %s, Number: %d\n", ctmp->course\_name, ctmp->course\_num);  /\* Re-sort and save \*/  sort(courses.begin(), courses.end(), cmp\_course\_num);  save();  /\* Inform provost window of the change \*/  emit add\_course\_event();  this->close();  myMessageBox(u8"添加成功");  }  }  void Add\_course::pushButton\_return\_clicked()  {  this->close();  } |

add\_ course类主要是用于添加课程，在选中课程并点击“添加课程”之后，便会创建一个add\_course类并显示窗口，主要函数为两个：

|  |  |
| --- | --- |
| 1  2  3 | private slots:  void pushButton\_enter\_clicked();  void pushButton\_return\_clicked(); |

这两个函数为qt的槽函数，对应“确认”和“返回”按钮的响应函数，通过构造函数中的：

|  |  |
| --- | --- |
| 1  2  3 | /\* Signals \*/  connect(uia.pushButton\_enter, SIGNAL(clicked()), this, SLOT(pushButton\_enter\_clicked()));  connect(uia.pushButton\_return, SIGNAL(clicked()), this, SLOT(pushButton\_return\_clicked())); |

这两个connect将按钮的点击动作和上面两个槽函数连接起来，使得点击按钮就执行相应的函数。

对于add\_course的具体执行，主要是先判断输入框的状态，然后读取信息并新建一个Course类，插入到课程列表当中，并且在成功执行后会发出一个信号通知父窗口新建成功，让父窗口更新数据。

|  |  |
| --- | --- |
| 1  2  3  4 | save();  /\* Inform provost window of the change \*/  emit add\_course\_event(); //发出信号  this->close(); |

对于其他一些窗口类：change\_pw/check\_course/modify\_course/register，这些与add\_course逻辑类似，均为各项功能的模块化，不再一一介绍。

#my\_course.h

my\_course.h中主要定义了学生和课程结构体，宏为各个信息的长度。

$ 课程结构主要有：课程号、课程名、一个vector数组用于记录选了该门课的学生学号以及一些修改函数。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | #define NAME\_N 20  #define PW\_N 24  #define COLLEGE\_N 24  #define COURSE\_NAME\_N 28  /\*  \* 学生和课程结构  \*/  struct Course  {  unsigned int course\_num; //课程号  char course\_name[COURSE\_NAME\_N]; //课程名字  vector<unsigned int>students; //选了该课程的学生学号  void set\_name(const char \*tname) { strncpy(course\_name, tname, COURSE\_NAME\_N); }  void set\_num(const unsigned tnum) { course\_num = tnum; }  };  struct Student  {  unsigned int student\_num; //学号  char student\_name[NAME\_N]; //名字  char college[COLLEGE\_N]; //学院  char pw[PW\_N]; //密码  vector<unsigned int>select; //已选课程的课程号  void set\_num(unsigned tnum) { student\_num = tnum; }  void set\_name(const char \*tname) { strncpy(student\_name, tname, NAME\_N); }  void set\_college(const char \*tcollege) { strncpy(college, tcollege, COLLEGE\_N); }  void set\_pw(const char \*tpw) { strncpy(pw, tpw, PW\_N); }  }; |

$ 学生结构主要有：学生学号、学生姓名、学院、密码、一个vector数组用于记录该学生选了哪些课程，以及一些修改函数。

$ 用于查找的结构体

|  |  |
| --- | --- |
| 1  2 | typedef struct find\_name{ … }  typedef struct find\_number{ … } |

由于要使用vector的迭代器以及find函数来对结构体进行查找，所以要定义这么两个类分别用于查找名字（学生姓名或课程名）和数字（学号或课程号），结构体中重载了()符号，用于定义比较方式。

$ 数据操作

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | /\*  \* Functions of IO  \*/  void save(); //保存信息  void Quit(); //退出  void read\_information(); //读取信息 |

save函数用于将数据保存到my\_data.mdt中，Quit函数为退出时执行的函数，read\_information函数在程序开始阶段从my\_data.mdt中读取数据。具体实现使用了unix的标准IO函数read和write以16进制的形式写入/读取。对于vector这类变长数据的保存，可以通过先保存vector.size()然后再将数据存入，读取的时候先读入size然后根据size读取数据。

$ 比较函数

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | /\*  \* 各种比较函数用于排序  \*/  bool cmp\_course\_num(const Course& ca, const Course& cb);  bool cmp\_course\_name(const Course& ca, const Course& cb);  bool cmp\_student\_num(const Student& sa, const Student& sb);  bool cmp\_student\_name(const Student& sa, const Student& sb); |

因为使用c++自带的sort函数进行排序，所以需要自定义比较函数。

$ 搜索函数

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | /\*  \* 各种搜索函数  \*/  /\* Get a student by std\_num or name \*/  vector<Student>::iterator get\_student(string inf);  vector<Student>::iterator get\_student(unsigned int inf);  /\* Judge a string is num or have illegal characters \*/  bool isNum(string);  /\* Get a course by course\_num or name \*/  unsigned int get\_course\_num(string tname);  string get\_course\_name(unsigned int tnum);  vector<Course>::iterator get\_course(string inf);  vector<Course>::iterator get\_course(unsigned int cnum); |

各类搜索函数用于以string/int形式的学号/课程号获取int/iterator形式的学生/课程。

$ 修改课程函数

|  |  |
| --- | --- |
| 1  2 | /\* Change course informations \*/  void change\_s\_course(Course \*c, int flag, unsigned course\_num = 0); |

以flag区别删除课程或是修改课程，c为需要修改/删除的课程，在每个学生的选课vector中搜索该门课，进行修改或删除。若为修改操作，第三个参数course\_num便为新的课程号。

$ 生成数据

|  |  |
| --- | --- |
| 1  2 | /\* Auto generate plenty of data \*/  void generate\_data(); |

该函数用于自动生成大量数据（每次5门课程和5000名学生）。

#my\_course.cpp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | /\*  \* Global variable  \*/  char pw[PW\_N];  vector<Course>courses; //全部课程  vector<Student>students; //全部学生  Student \*current\_student; //当前登录的学生  Course \*current\_course; //当前操作的课程  char buf[100]; //读取输入的缓冲区  FILE \*fd; //文件描述符 |

该文件主要定义了这些数据实例，其余为对头文件中函数的实现。

pw为教务密码，courses为课程数组，students为学生数组，current\_student为当前选中的学生，current\_course为当前选中的课程，buf为共用的缓冲区，fd为输入/输出的文件描述符。这些实例在其他cpp中以extern变量的形式被引用。

#my\_message.h

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | #pragma once  /\*  \* For emitting a message box  \*/  #include <QMessageBox>  #include <cstring>  #include <QApplication>  static void myMessageBox(QString s)  {  QApplication::setQuitOnLastWindowClosed(false);  QMessageBox::warning(NULL, "Message", s, QMessageBox::Ok, QMessageBox::Ok);  QApplication::setQuitOnLastWindowClosed(true);  } |

这个头文件的定义是为了解决qt的messagebox的一个bug，若不在messagebox前后设置上述选项，那么在关闭弹出的messagebox后整个程序会随之关闭，这不是我们想要的。

#mainProvost.h

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | #pragma once  #include <QtWidgets/QMainWindow>  #include <QMessageBox>  #include "ui\_mainProvost.h"  #include "change\_pw.h"  #include "add\_course.h"  #include "modify\_course.h"  #include "check\_course.h"  /\*  \* Provost functions  \*/  class Provost : public QMainWindow  {  Q\_OBJECT  public:  Provost(QWidget \*parent = Q\_NULLPTR);  QWidget \*parentWindow;  void print\_all\_courses();  int get\_column();  private slots:  void pushButton\_add\_clicked();  void pushButton\_delete\_clicked();  void pushButton\_modify\_clicked();  void pushButton\_return0\_clicked();  void pushButton\_pw\_clicked();  void add\_course\_complete();  void modify\_course\_complete();  void pushButton\_check\_clicked();  private:  Change\_pw \*cp;  Add\_course \*ac;  Modify \*md;  Check\_course \*ck;  Ui::Provost\_MainWindow uip;  }; |

这个类是教务界面窗口类，除了槽函数以外，有两个功能函数：

|  |  |
| --- | --- |
| 22  23 | void print\_all\_courses();  int get\_column(); |

第一个用于打印数据表格，第二个用于获取表格上的选区所在的列号用于对课程进行操作。

除此之外，还有四个窗口类的指针，用于在执行对应功能的时候创建窗口并在最后回收。

|  |  |
| --- | --- |
| 36  37  38  39 | Change\_pw \*cp;  Add\_course \*ac;  Modify \*md;  Check\_course \*ck; |

#mainStudent.h

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | #pragma once  /\*  \* Student functions  \*/  #include <QtWidgets/QMainWindow>  #include "ui\_mainStudent.h"  class Students : public QMainWindow  {  Q\_OBJECT  public:  Students(QWidget \*parent = Q\_NULLPTR);  QWidget \*parentWindow;  void set\_inf();  void print\_selected\_courses();  void print\_all\_courses();  int get\_row(int);  private slots:  void pushButton\_return\_clicked();  void pushButton\_select\_clicked();  void pushButton\_withdraw\_clicked();  private:  Ui::Students\_MainWindow uis;  }; |

学生界面类与教务界面类构造类似，主要由槽函数和功能函数构成，不再具体介绍。

1. 其他

Release版本的exe和用到的动态链接库都在压缩包中，可以直接运行，然后还有一个“演示.gif”为一些功能的演示动画