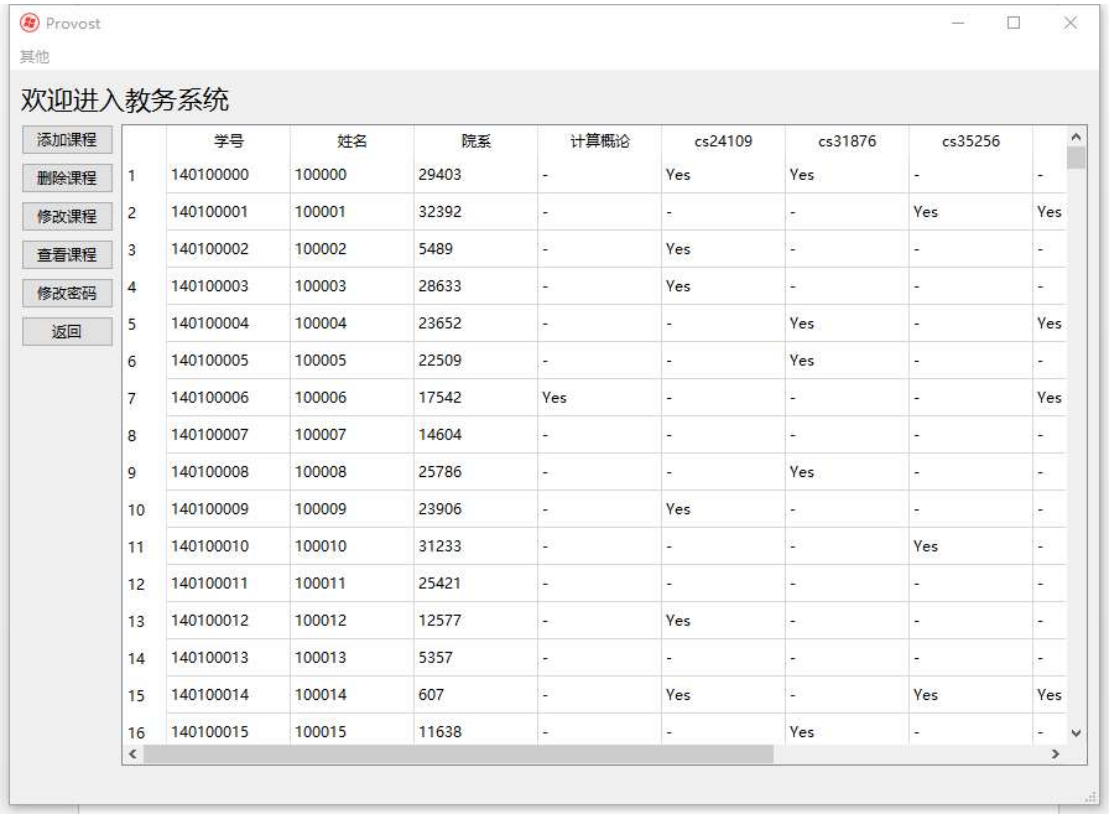
大作业报告：选课系统

陈国俊 - 1400011379

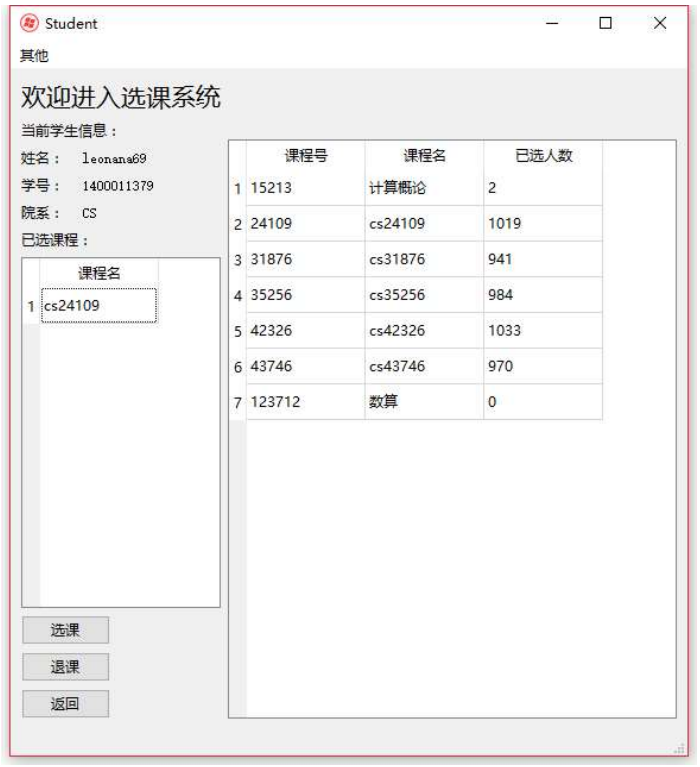
一、界面介绍

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。



	学号	姓名	院系	计算概论	cs24109	cs31876	cs35256	
1	140100000	100000	29403	-	Yes	Yes	-	-
2	140100001	100001	32392	-	-	-	Yes	Yes
3	140100002	100002	5489	-	Yes	-	-	-
4	140100003	100003	28633	-	Yes	-	-	-
5	140100004	100004	23652	-	-	Yes	-	Yes
6	140100005	100005	22509	-	-	Yes	-	-
7	140100006	100006	17542	Yes	-	-	-	Yes
8	140100007	100007	14604	-	-	-	-	-
9	140100008	100008	25786	-	-	Yes	-	-
10	140100009	100009	23906	-	Yes	-	-	-
11	140100010	100010	31233	-	-	-	Yes	-
12	140100011	100011	25421	-	-	-	-	-
13	140100012	100012	12577	-	Yes	-	-	-
14	140100013	100013	5357	-	-	-	-	-
15	140100014	100014	607	-	Yes	-	Yes	Yes
16	140100015	100015	11638	-	-	Yes	-	-



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。

二、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。

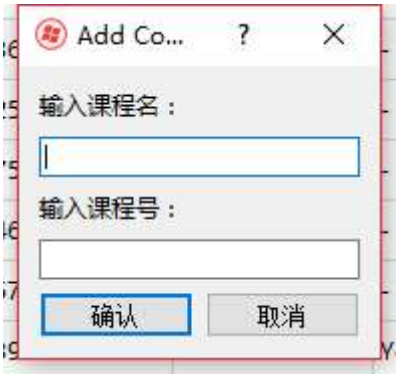


Figure 1 添加课程

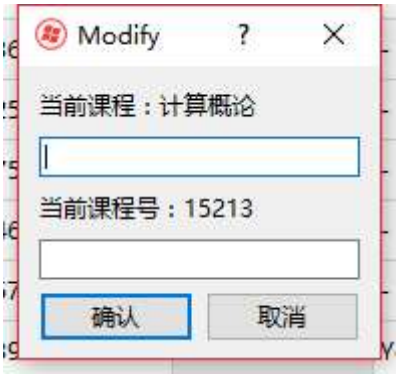


Figure 2 修改课程

2. 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。

3. 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。

4. 查看课程

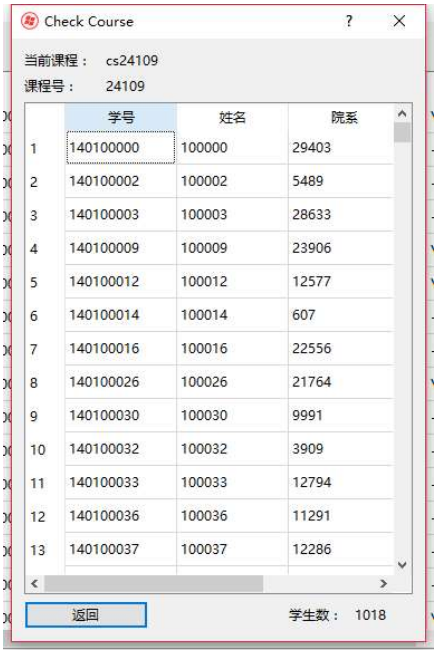


Figure 3 查看课程

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。

5. 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。

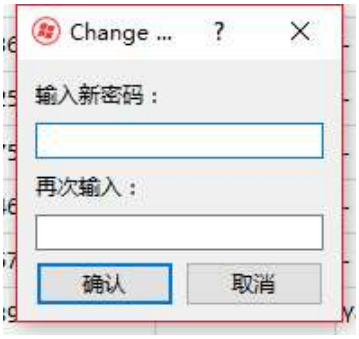


Figure 4 修改密码

三、代码解释

#main.cpp

```
1   #include "enterWidget.h"
2   #include "my_course.h"
3   #include <QtWidgets/QApplication>
4
5   /*
6    * Global variable
7    */
8
9   extern char pw[PW_N];
10  extern vector<Course>courses;        //全部课程
11  extern vector<Student>students;      //全部学生
12  extern FILE *fd;                  //文件
13
14
15  /*
16   * 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

```
#pragma once

/*
 * "Add course" window
 */
```

```
6
7   #include "ui_add_course.h"
8   #include <QtWidgets/QMainWindow>
9
10  class Add_course : public QDialog
11  {
12      Q_OBJECT
13
14  public:
15      Add_course(QWidget *parent = Q_NULLPTR);
16      QWidget *parentWindow;
17
18      private slots: //槽函数
19      void pushButton_enter_clicked();
20      void pushButton_return_clicked();
21
22  signals:
23      void add_course_event();
24
25  private:
26      int equal_flag; //判断是否与已有课程重复
27      Ui::add_course_Dialog uia;
28  };
```

#add_course.cpp

```
1   #include "add_course.h"
2   #include "enterWidget.h"
3   #include "my_course.h"
4   #include "my_message.h"
5   #include <qdebug.h>
6
7   /* Extern variables */
8   extern vector<Course>courses;
9
10  /* Add_course init function */
11  Add_course::Add_course(QWidget *parent) : QDialog(parent)
12  {
13      parentWindow = parent;
14      setWindowIcon(QIcon(":/Resources/windows.png"));
15      setFixedSize(180, 145);
16      uia.setupUi(this);
17      /* Signals */
18      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：

```
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：

```
/* 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。

```
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。

```
#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
```

```
14      vector<unsigned int>students;        //选了该课程的学生学号
15      void set_name(const char *tname) { strncpy(course_name, tname,
16  COURSE_NAME_N); }
17      void set_num(const unsigned tnum) { course_num = tnum; }
18  };
19
20  struct Student
21  {
22      unsigned int student_num;     //学号
23      char student_name[NAME_N];    //名字
24      char college[COLLEGE_N];      //学院
25      char pw[PW_N];                //密码
26      vector<unsigned int>select;       //已选课程的课程号
27      void set_num(unsigned tnum) { student_num = tnum; }
28      void set_name(const char *tname) { strncpy(student_name, tname, NAME_N); }
29      void set_college(const char *tcollege) { strncpy(college, tcollege,
30  COLLEGE_N); }
31      void set_pw(const char *tpw) { strncpy(pw, tpw, PW_N); }
32  };
```

$ 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   typedef struct find_name{ … }
2   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    * Functions of IO
3    */
4
5   void save();              //保存信息
6   void Quit();              //退出
7   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

```
/*
 * 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

```
/*
 * 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

```
/* 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

```
/* 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

```
/*
 * 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

```
#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   #pragma once
2
3   #include <QtWidgets/QMainWindow>
4   #include <QMessageBox>
5   #include "ui_mainProvost.h"
6   #include "change_pw.h"
7   #include "add_course.h"
8   #include "modify_course.h"
9   #include "check_course.h"
10
11  /*
12   * Provost functions
13   */
14
15  class Provost : public QMainWindow
16  {
17      Q_OBJECT
18
19  public:
20      Provost(QWidget *parent = Q_NULLPTR);
21      QWidget *parentWindow;
22      void print_all_courses();
23      int get_column();
24
25      private slots:
26      void pushButton_add_clicked();
27      void pushButton_delete_clicked();
28      void pushButton_modify_clicked();
29      void pushButton_return0_clicked();
30      void pushButton_pw_clicked();
31      void add_course_complete();
32      void modify_course_complete();
33      void pushButton_check_clicked();
34
35  private:
36      Change_pw *cp;
37      Add_course *ac;
38      Modify *md;
```

```
39      Check_course *ck;
40      Ui::Provost_MainWindow uip;
41  };
```

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      void print_all_courses();
23      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      Change_pw *cp;
37      Add_course *ac;
38      Modify *md;
39      Check_course *ck;
```

#mainStudent.h

```
1   #pragma once
2   /*
3    * Student functions
4    */
5   #include <QtWidgets/QMainWindow>
6   #include "ui_mainStudent.h"
7
8   class Students : public QMainWindow
9   {
10      Q_OBJECT
11
12  public:
13      Students(QWidget *parent = Q_NULLPTR);
14      QWidget *parentWindow;
15      void set_inf();
16      void print_selected_courses();
17      void print_all_courses();
18      int get_row(int);
19
20      private slots:
21      void pushButton_return_clicked();
22      void pushButton_select_clicked();
23      void pushButton_withdraw_clicked();
24
25  private:
26      Ui::Students_MainWindow uis;
27  };
```

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。

四、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