Qt 大作业报告

2300013105 阮宗泽

一、功能简介

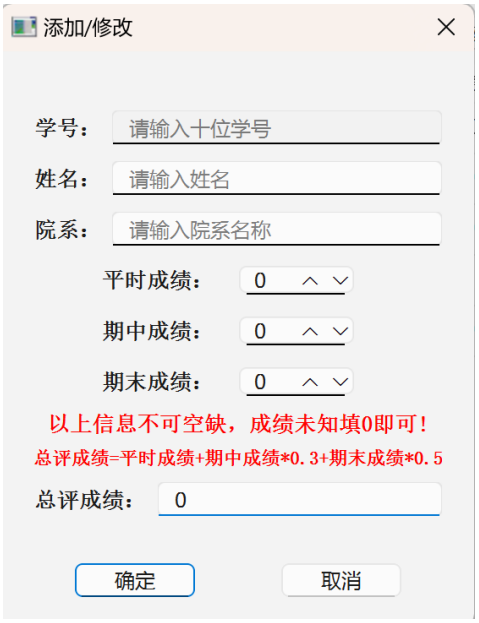
我设计的程序名叫 PKU 课程学生信息管理系统，它可以帮助老师实现课程学生的信息、成绩等数据的集中管理，并且操作简洁便利。主界面的所有学生信息的显示无法直接修改，只能通过上下方总计 6 个按键进行操控。

主界面展示如下（图中为示例数据）：

该程序的具体功能如下。

1.1 添加学生信息

点击“添加信息”按钮后，会弹出信息添加/修改界面，用户可以在这个界面填写需要添加的学生的学号、姓名、院系、成绩等信息。成绩信息限定为整数，可使用滚轮调整。总评成绩将根据公式自动计算，不可被用户手动修改。添加完成后立即在主界面显现，并且表格将自动滚动到添加信息的底部。



1.2 删除学生信息

单击某条学生信息后（未选中则默认为第一条），点击“删除信息”将进行删除操作，并且会再次询问是否删除，删除成功后也会进行提示。

1.3 修改学生信息

修改学生信息及成绩需要权限，单击某条学生信息后（未选中则默认为第一条），点击“修改信息”后会弹出填写密码的窗口，若密码正确则跳转到信息添加/修改界面，与 1.1 类似，区别在于此时会自动填写好原先

的内容，用户直接进行修改即可。修改成功后也会进行提示。

1.4 对全部信息根据总评成绩升序/降序排列

点击上方两个按钮即可对表格中的所有学生及其个人信息按照该生总评成绩进行升序/降序排序。

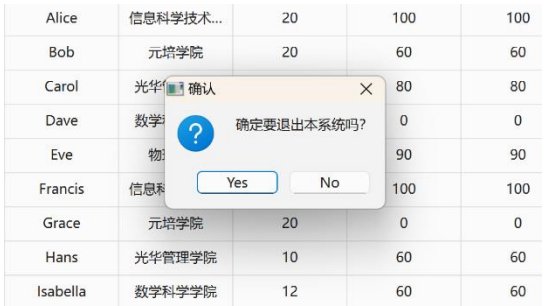
1.5 保存修改

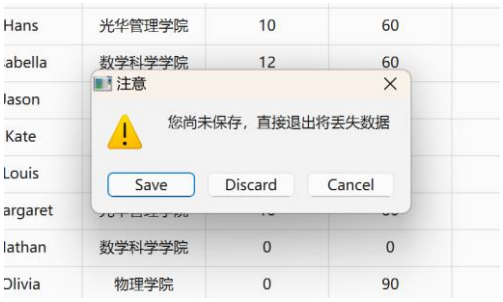
如果对表中信息进行了改动（添加信息、删除信息、修改信息），可以点击“保存修改”按钮对数据进行保存，未保存时强行退出程序会给出保存提示。如果数据没有被修改则无需保存，此时点击会提示“无需保存”。

在点击“保存修改”之前，所有数据将会进行缓存，直到确认保存才会记入数据文件中。

1.6 程序退出提示

关闭程序前会进行二次确认，若未保存还会进行保存提示。





二、项目各模块与类设计细节

2.1 类设计

除了主窗口 Widget 类，还有一个用于存放学生信息的 Student 类。

```
class Student
{
private:
    QStringList info;
public:
    Student(QStringList &list);
    ~Student(){};
    bool operator<(const Student &obj);

    friend class Widget;
    friend bool studentcmp(const Student obj1, const Student obj2);
};

bool studentcmp(const Student obj1, const Student obj2);
```

以及两个界面设计师类 PasswordDialog 类和 InfoEditPage 类，用于实现多窗口。

```
class PasswordDialog : public QDialog
{
    Q_OBJECT

public:
    explicit PasswordDialog(QWidget *parent = nullptr);
    ~PasswordDialog();
    bool checkpass();//用于在其他类中获取Haspassed的值

private slots:
    void on_pushButton_clicked();

private:
    Ui::PasswordDialog *ui;
    bool Haspassed;//标记是否通过验证
};
```

```
class InfoEditPage : public QDialog
{
    Q_OBJECT

public:
    explicit InfoEditPage(QWidget *parent = nullptr);
    ~InfoEditPage();
    void calculateoverallscore();//计算总评成绩
    QStringList& getdata();//在其他类中读取Dataline

    friend class Widget;

private slots:
    void on_CancelButton_clicked();
    void on_OkButton_clicked();
    void on_spinBox1_editingFinished();
    void on_spinBox2_editingFinished();
    void on_spinBox3_editingFinished();

private:
    Ui::InfoEditPage *ui;
    int overallscore;//存储总评成绩
    QStringList Dataline;//存储填写入的全部信息
};
```

2.2 模块设计

这里是所有用于实现功能的函数。

```
Widget::Widget(QWidget *parent)//主窗口构造函数
    : QWidget(parent)
    , ui(new Ui::Widget) {...}

Widget::~Widget()//主窗口析构函数 {...}

void Widget::closeEvent(QCloseEvent *event)//窗口关闭时询问是否退出 {...}

void Widget::ReadinData(const QString &fname)//从文件读取数据 {...}

void Widget::SavetoVector()//缓存数据 {...}

void Widget::SaveData(const QString &fname)//向文件中保存数据 {...}

void Widget::on_AddButton_clicked()//“添加信息”按键 {...}

void Widget::on_SaveButton_clicked()//“保存修改”按键 {...}

void Widget::on_DeleteButton_clicked()//“删除信息”按键 {...}

void Widget::on_EditButton_clicked()//“编辑信息”按键 {...}

void Widget::on_LessButton_clicked()//升序排序按键 {...}

void Widget::on_GreaterButton_clicked()//降序排序按键 {...}
```

主界面设计主要由若干按键 PushButton 以及一个 TableView 实现。TableView 用于数据展示，PushButton 用于制作操控按键。

数据展示的实现是通过 QStandardItemModel 类的 datamodel 对象与 TableView 构建联系（使用 setModel 成员函数），并在后续中操作中直接修改 datamodel 中的内容，TableView 的显示也会随之改变。

从文件中读取数据并显示的功能是使用 QFile 类进行文件操作，使用 QTextStream 类进行数据读取，读取方式为按行读取，并逐次存入一个 QStringList 对象中，同时也创建一个 Student 对象，存入 studentsdata（本质是一个存放 Student 类对象的一维数组）中缓存数据。再构建一个 QStandardItem*类型的列表，对其中的每个对象进行格式初始化（例如居中对齐）并顺次拷入 QStringList 对象中的数据，随后以行的形式整体传入 datamodel 中。（*）

```
//读取一行，存到studentsdata中
QStringList Dataline = stream.readLine().split(" ");
studentsdata.push_back(Student(Dataline));
//在屏幕上显示
QList<QStandardItem*> items;
for(QString tmpdata : Dataline)
{
    QStandardItem *tmpitem = new QStandardItem();
    tmpitem->setText(tmpdata);//设置文字
    tmpitem->setTextAlignment(Qt::AlignCenter);//居中对齐
    items.push_back(tmpitem);
}
datamodel->appendRow(items);
```

添加与修改数据的过程类似于上述。只需将在 InfoEditPage 界面内填写的数据读写入一个 QStringList 中，仿照上述过程将其加入 studentsdata 以及 datamodel 中即可。区别在于添加数据在二者的尾部，修改数据需要在原先位置插入。

删除数据需要调用 TableView 中的 currentIndex()函数读取当前选中的行号信息，再使用 datamodel->removeRow(…)即可。

排序功能直接对缓存的 studentsdata 进行快速排序（std::sort），需事先对 Student 类进行“<”的重载。再逐一读取排好序的 Student 类对象的成员变量 QStringList info，仿照（*）过程写入 datamodel 中即可。注意，写入前需要使用 clear()清空 datamodel。

如何判断文件是否需要保存？在所有会对数据库进行修改的操作后，会把主窗口类的成员变量 Hassaved（bool 类型）的值置为 false，并在触发主窗口关闭事件 closeEvent()与单击“保存修改”按键 on_SaveButton_clicked()时，先判断 Hassaved 的值，再决定下一步的操作。

还有更多的功能实现方式，可以参考源代码内的注释。

三、成员分工

我独自一人完成，没有分工……因为给我分配的组员处于失联状态，联系不到，我只好自己做了。如果 Ta 或 Ta 们没有提交任何 Qt 大作业，请不要把我的作品同他们产生任何联系，也不能共享我的分数；如果 Ta 或 Ta 自己也独立完成了大作业，那就分开算分即可。

四、总结与反思

我认为我的程序功能还不算完善，还有提升空间。比如说界面设计比较单一，只注重了实用性但没有注重美观性。其次，我的程序在功能上也有一些没有实现的。比如说，填写密码界面需要对输入的信息进行隐藏，以及用户可以在第一次使用时可以自行设置密码，并且可以通过原密码修改新密码，这样才更能接近真实的管理系统。