第61组_对对队作业报告

组员：毛川、毛思雨、何郅瑾

一、程序功能介绍

许多现有的手机 APP 和相关自律和计时软件功能集成度较低，用户需要同时下载多个软件，使用不便。我们开发了一个集成多功能的习惯养成软件，以提高用户的使用便捷性。

日历功能

1. 记录数天后将要发生的事件，或过去某一天的日记、心得体会。
2. 附带倒数日功能，用户可以查看未发生事项距离当日的天数。可以更好地应对考试和 DDL 。
3. 点击日历中的某一天，可以查看当天的事项或编辑当天事项和日记。

日程管理功能：

1. 可以轻松向数据库中加入或删除元素，并设置事件的重要和紧急程度
2. 任务管理：采用“时间管理四象限”方法，对任务进行分类和管理。
3. 打卡功能：用户完成任务后可以进行打卡，系统记录完成情况。

时间管理功能：

1. 倒计时：实现倒计时功能，用户可以设置时间并启动倒计时。
2. 秒表：提供启动、暂停、重置功能。
3. 闹钟：用户可以设置多个闹钟，支持重复和提醒功能。

课程表功能：

1. 在一周的每一天中规划特定的时间段，支持加入和删除课程。
2. 借鉴树洞中的课程表，这里使用不同的颜色以鲜明的背景体现出来，外表更加美观和醒目

主题更换功能：

除了北京大学背景主题，我们还设计了快乐小狗主题，并且实现了一键更换背景和按钮主题的功能，提供了多样和选择。使用户有更好的体验。

二、项目各模块与类设计细节

日历模块

这里有四个类 `calendarMC` 、 `win_cal_viewMC`、 `ModEvent` 和 `CountDownDaysMC`

在 `calendarMC` 中有一个 `QSqlDatabase sqldb` 数据库，这四个类共用一个数据库。 `calendarMC` 在主窗口显示日历，根据数据库中事件的日期筛选并将它们用不同的颜色标注出来。

而 `ModEvent` 可以向数据库中添加和删除事件，在添加事件的时候可以添加事件的详细备注和心情，这些信息在其他特定的窗口、特定的位置显示出来。

```
//单击信号
void calendarMC::clickedSlot(const QDate date)
{
    //为了能把这一天的日期传给新的窗口，需要实现单例化
    win_cal_viewMC* small_win=win_cal_viewMC::getinstance();
    QList<AEventInfo> sssList;
    QSqlQuery sql(sqldb);
    QString strsql = QString("SELECT * FROM event WHERE date = :cdate");//直接排好序
    sql.prepare(strsql);
    sql.bindValue(":cdate",date.toString("yyyy/MM/dd"));
    qDebug()<<"input"<< date;
    // 遍历查询结果
    if(sql.exec()){
        while (sql.next()) {
            AEventInfo info;
            //qDebug()<<"date_val"<<sql.value(2).toString();
            info.name = sql.value(1).toString();
            info.date=sql.value(2).toString();
            info.atimes = sql.value(3).toString();
            info.mood = sql.value(4).toString();
            info.details = sql.value(5).toString();
            sssList.push_back(info);
            qDebug()<<"if execute";
```

```
        }
    }
    else{
        QMessageBox::critical(0,"抱歉","查询失败",QMessageBox::Ok);
    }
    if(sssList.size()==0){
        QMessageBox::information(0,"这一天空空如也","加入日程或写点日记吧！",QMessageBox::Ok);
    }
    else{
        qDebug()<<sssList.size();
        small_win->show();//仍然是单例
        small_win->FindAndPrint(sssList);
    }
}
```

`CountDownDaysMC` 用来显示倒数日窗口，将日历数据库（sqldb）中的数据筛选出来，计算距离今天的天数，排序后显示出来。

`win_cal_viewMC` 在单击 `calendarMC` 窗口中 `QCalendarWIdget` 中的某一天后显示，从数据库中找到这一天的事件。

```
void win_cal_viewMC::FindAndPrint(QList<AEventInfo> speeve){
    //从数据库中找到属于这一天的事件，然后通过“有趣的语句组合起来”
    int cnt = speeve.size();
    ui->tableWidget->clearContents();
    ui->tableWidget->setColumnCount(1);
    ui->tableWidget->setRowCount(cnt);
    //qDebug()<<"cnt:"<<cnt;
    for(int i=0;i<cnt;i++){
        QString s0="这一天，怀着";
        QString s1=speeve[i].mood;
        QString s2="的心情,";
        QString s3="在 "+speeve[i].atimes+" 去迎接（面对）";
        QString s4=speeve[i].name;
        QString s5=" (";
        QString s6=speeve[i].details;
        QString s7=") ";
        qDebug()<<s1+s2+s3+s4+s5+s6+s7;
        QTableWidgetItem *item = new QTableWidgetItem(s1+s2+s3+s4+s5+s6+s7);
        ui->tableWidget->setItem(i,0,item);
    }
    changetheme();
}
```

将SQL 数据库中的信息读取到一个 `AEVEntInfo` 结构体中。这个函数将一部分信息对应的 `AEventInfo` 结构体列表（QList 类型）通过 `setItem` 的方法在表中显示出来。 设置数据表的格式（行和列），并向数据表中加入对应的信息。

```
void CountDownDaysMC::PrintForView(){
    calendarMC* m_ptrcalendar=calendarMC::getinstance();
    auto cnt = m_ptrcalendar->countNum();
    qDebug()<<"here2?";
    QList<AEventInfo> listeve=m_ptrcalendar->selectPage(0,cnt);//仅仅跟踪到它指向的Qlist里面
    cnt = listeve.size();
    ui->tableWidgetCD->clearContents();
    //qDebug()<<"Emepty:"<<listeve.size();
    ui->tableWidgetCD->setRowCount(cnt);
    //qDebug()<<"cnt:"<<cnt;
    for(int i=0;i<listeve.size();i++){
        QTableWidgetItem *item = new QTableWidgetItem(QString::number(i));
        item->setTextAlignment(Qt::AlignCenter); // 设置水平和垂直居中对齐
        ui->tableWidgetCD->setItem(i,0,item);
        ui->tableWidgetCD->setItem(i,1,new QTableWidgetItem(listeve[i].name));
        ui->tableWidgetCD->setItem(i,2,new QTableWidgetItem(listeve[i].mood));
```

```
        QDate date = QDate::fromString(listeve[i].date,"yyyy/MM/dd");
        int diff=m_ptrcalendar->TToday.daysTo(date);
        item=new QTableWidgetItem(QString::number(diff));
        item->setTextAlignment(Qt::AlignCenter);
        ui->tableWidgetCD->setItem(i,3,item);
    }
}
```

日程模块

这里有一个主窗口，用`dailymsy`类实现，按照“时间管理四象限法则”分别设置四个象限`dailymsy1-4`以及一个主要控制窗口`dailymsy_allin`。在`dailymsy`中设置有数据库和数据表。与其余的类共享。

```
void dailyMSY::CreatTableFunc(){//创建sqlite数据表
    QSqlQuery sql(sqldb);
    QString strsql=QString("create table event("
                           "id int primary key not null,"
                           "thingsname text not null,"
                           "im int not null,"
                           "em int not null)");
    //执行SQL语句
    if(sql.exec(strsql)==false){
        //QMessageBox::critical(0,"错误","数据表创建失败",QMessageBox::Ok);
    }
    else{
        //QMessageBox::information(0,"正确","恭喜你，数据表创建成功",QMessageBox::Ok);
    }
}

void dailyMSY::CreatDataFunc(){//创建SQLite数据库
    //1.添加数据库驱动
    sqldb=QSqlDatabase::addDatabase("QSQLITE","myConnection");
    //2.设置数据库名称
    sqldb.setDatabaseName("ThingsDemo.db");
    //3.打开数据库是否成功
    if(sqldb.open()==true){
        //QMessageBox::information(0,"正确","恭喜你，数据库打开成功",QMessageBox::Ok);
    }
    else{
        QMessageBox::critical(0,"错误","数据库打开失败",QMessageBox::Ok);
    }
}
```

在`dailymsy_allin`中设置了事项的添加方式（向数据表中插入数据），以及将数据表中的内容显示在表格中，以及可以选中某一行，相应的行号会显示在表格下方，也可以转到下一行。点击相应的行可以进行整个事项的删除。

```
void dailymsy_allin::on_missionAdd_clicked()//添加任务并且显示
{
    dailyMSY* m_ptrdailymsy_allin=dailyMSY::getinstance();
    auto cnt = m_ptrdailymsy_allin->CountNum();
    BEventInfo info;
    // dailyMSY* m_ptrdailymsy_allin=dailyMSY::getinstance();
    // auto cnt = m_ptrdailymsy_allin->CountNum();
    info.thingsname=ui->thingsname->text();
    info.id=cnt+1;
    //qDebug()<<info.id;
    info.im=ui->imNum->text().toInt();
    info.em=ui->emNum->text().toUInt();
    dailyMSY::getinstance()->addOne(info);//将数据加入到数据库中，并加入在相应的List中
```

```
    PrintP();//将数据显示在TableWidget上；
}
```

四个象限中的内容可以进行浏览，我们根据重要性和紧急程度对存储的数据进行分类。

时间管理模块

主时钟界面（`clockHZJ`）

- **技术实现**：
  - 利用`QLabel`控件展示时间，通过`QTimer`每秒触发更新事件，用`QDateTime`确保时间的实时性。
  - 通过`QPushButton`创建三个功能按钮，分别指向秒表、倒计时和闹钟设置。
- **代码分析**：

```
connect(&timer_cur, SIGNAL(timeout()), this, SLOT(showcurtime()));
timer_cur.start(1000);
\\
void clockHZJ::showcurtime(){
    curDTime = QDateTime::currentDateTime();
    ui->curtime->setText(curDTime.toString("yyyy-MM-dd HH:mm:ss"));
}
```

上述代码展示了如何创建一个每秒触发一次的定时器，并将其与showcurtime()槽函数关联，实现时间的实时更新。

闹钟设置（`alarm`）

- **技术实现**：
  - 使用`QTimeEdit`控件让用户设定具体时间，`QLineEdit`用于输入提醒消息。
  - 通过`QPushButton`提供重复模式选项，如“仅一次”、“每天”。
  - 利用`QTimer`检查当前时间与设定的闹钟时间是否匹配，匹配时通过`QMessageBox`弹出提醒。
- **代码分析**：

```
timerunner.start(1000);
connect(&timerunner,SIGNAL(timeout()), this, SLOT(checktime()));
void alarm::checktime(){
    QDateTime present_time = QDateTime::currentDateTime();
    QString timeText = present_time.toString("hh:mm:ss");
    if(cs1){
        QString settime1=ui->time1->text();
        if(timeText==settime1){
            QMessageBox::information(this,"tip",ui->line1->text());
            if(con1){
                ui->time1->setReadOnly(false);
                ui->check1->setText("启用");
            }
        }
    }
    \\省略部分重复代码
}
```

这段代码展示了如何设置一个每分钟触发一次的定时器，用于检查当前时间是否与设定的闹钟时间匹配。

秒表（`sec_clock`）

- **技术实现**：
  - 使用 `QTimer` 控制秒表的计时，`QLabel` 显示经过的时间。
  - 通过 `QPushButton` 控制秒表的启动、暂停和重置操作。
  - 提供一个 `QListWidget` 用于记录和展示分段时间点。

倒计时（`r_clock`）

- **技术实现**：
  - 使用 `QSpinBox` 控件让用户设定倒计时的小时、分钟和秒数。
  - 通过 `QTimer` 控制倒计时的递减，每次递减1秒。
  - 提供“开始”、“暂停”、“继续”和“结束”按钮，控制倒计时的流程。
- **代码分析**：
  - `r_clock::update()` 实现每秒更新
  - `r_clock::display_number()` 显示剩余时间
  - `r_clock::on_pushButton_clicked()` 开始计时
  - `r_clock::hourChanged()` 等同步显示时间与设置时间
  - `r_clock::on_end_clicked()` 实现立即结束
  - `r_clock::on_pause_clicked()` 实现暂停与继续

课程表模块

这里有两个类：`coursemsy` 和 `schedulemsy`，分别负责课程的插入删除以及课程表的显示。对于课程表，也有相应共享的数据库和数据表，进行课程的存放与删除。

```
void coursemsy::CreatDataFunc(){//创建SQLite数据库
    //1.添加数据库驱动
    sqldb=QSqlDatabase::addDatabase("QSQLITE","msyconnection");
    //2.设置数据库名称
    sqldb.setDatabaseName("CourseShow.db");
    //3.打开数据库是否成功
    if(sqldb.open()==true){
        //QMessageBox::information(0,"正确","恭喜你，数据库打开成功",QMessageBox::Ok);
    }
    else{
        //QMessageBox::critical(0,"错误","数据库打开失败",QMessageBox::Ok);
    }
}
```

在课程表的显示上，我们将不同名称的课程显示为不同颜色，便于大家进行查阅。

```
void scheduleMSY::colored(){
    // 遍历表格并设置背景颜色
    QMap<int, QColor> courseColors = {
        {0, QColor(255, 182, 193)}, // Light Pink
        {1, QColor(173, 216, 230)}, // Light Blue
        //省略了部分代码
    };
    QMap<QString,int> searchcourses={};
    int current_course_number=0;
    for (int row = 0; row < ui->courseTable->rowCount(); ++row) {
        for (int col = 0; col < ui->courseTable->columnCount(); ++col) {
            QTableWidgetItem *item = ui->courseTable->item(row, col);
            if(item){
                item->setTextAlignment(Qt::AlignCenter);
                if (searchcourses.contains(item->text())) {//这是一个第一次出现的课
                    item->setBackground(courseColors[searchcourses[item->text()]]);
```

```
                }
                else {//这个课已经出现过
                    searchcourses[item->text()]=current_course_number;
                    current_course_number++;
                    item->setBackground(courseColors[searchcourses[item->text()]]);
                }
            }
        }
    }
    qDebug()<<current_course_number;
}
```

主题模块

在 `choosetheme` 类中使用了全局变量 `ThemeStyle`，在点击更换按钮之后， `ThemeStyle`更改，所有的窗口接受到信号，调用 `ChangeTheme` 函数，通过样式表更改窗口背景和按钮。

```
\\
void ChooseTheme::changebegin()
{
    ThemeStyle=1;
    emit themeChanged();

}
\\
else if(ThemeStyle==1){
        QPixmap pixmain3(":happydog/time.jpg");
        pixmain3 = pixmain3.scaled(ui->l1->size(), Qt::KeepAspectRatio, Qt::SmoothTransformation);
        ui->l1->setPixmap(pixmain3); // 显示 QLabel
        ui->l1->show()
        this->setStyleSheet(
            "QWidget#alarm{"
            "    background-image: url(:happydog/time.jpg);" // 设置背景图片
            "    background-position: center;" // 将图片放置在中心
            "    background-repeat: no-repeat;" // 禁止图片重复
            "    background-size: 100% 100%;" // 使图片拉伸以适应窗口大小
            "}"
            );

    }
```

以上分别为全局变量的更改和读取全局变量更改后通过样式表一个窗口的按钮。为了使众多窗口同步更换主题，我们用信号机制将各窗口链接起来，即父窗口更换完主题，子窗口接收信号，随后更换主题，以此类推。

可变按钮

将 `QButton` 提升为 `hoverbutton` 类，使一些按钮具有其特性的同时，方便使用 ui 界面对窗口和按钮进行编辑。使用 `animation`类，实现形态和颜色的平滑变化。

```
void HoverButton::animateSize(QSize size)
{
    //std::cout << "Change!" << std::endl;  // 调试输出
    QSize newSize = size;
    // 获取当前几何位置
    QRect startRect =geometry();
    QPoint center = startRect.center();
    QRect endRect(center.x() - newSize.width() / 2,
                  center.y() - newSize.height() / 2,
```

```
                    newSize.width(),
                    newSize.height());
    // 创建动画对象
    QPropertyAnimation *animation = new QPropertyAnimation(this, "geometry");
    // 设置动画持续时间
    animation->setDuration(200);
    // 设置起始值
    animation->setStartValue(startRect);
    // 设置结束值
    animation->setEndValue(endRect);
    // 启动动画，并在结束后自动删除
    animation->start(QAbstractAnimation::DeleteWhenStopped);
}
```

三、小组成员分工

1. 在功能实现上：
   - 毛川负责完成主程序与日历模块
   - 毛思雨完成课程表与日程模块
   - 何郅瑾完成时间管理模块
2. 美工方面：
   - 何郅瑾主要负责图片的搜集与加工
   - 毛思宇负责图片的搜集
   - 毛川负责图片插入
   - 各模块的界面布局在各自初步设计后由毛思雨调整
3. 时间分配：

任务	成员	时长
学习github管理	all	6h*3
设计	all	3h*3
日历	队长	30h
日程和课程表	队员1	30h
时间管理	队员2	20h
界面美化	all	6h*3

四、项目总结与反思

整个项目人均耗时超过40小时，总体来说是一个本科期间比较大的项目。然而，项目完成的过程中也遇到了一些困难，比如某个库无法使用，样式设计后却无法显示，数据库无法进行有效地操作。我们通过查询资料、相互交流解决了绝大多数的问题。在解决问题的同时学习到了新知识，提高了代码能力。

我们使用 **github** 和 **gitkraken** 管理项目，增进了组织能力和团队协作的能力，为以后得工作奠定了基础。同时，大家或多或少对 SQL 数据库语言有所了解，有助于之后在数据科学方面的学习。

经过小组三位成员的共同努力，我们在6月1日前完成了项目的主要部分，并荣幸入选参加课堂上的路演。

然而，在路演过程中我们也了解到其他小组的作品，对我们的设计进行反思后，发现了两处需要改进的方面：
1. 窗口调用较多，如果能将所有功能集成在一个窗口上，作品将更加简约。

2. 日历和日程规划使用了两个数据库，所以两项功能没有很好地关联和同步，这也是以后得项目需要注意的方面。