

《软件设计与开发实践 A》 课程报告

任务名称： 日程管理清单

班 级： 5 班 组号： 4

学 号： 190110509

姓 名： 王铭

学年学期： 2020 年秋季学期

任课教师： 房敏、谢佳

哈尔滨工业大学（深圳）计算机科学与技术学院

前 言

《软件设计与开发实践 A》是基于自选项目的实践训练，学生将综合利用《集合论与图论》、《数据结构》、《算法设计与分析》、《高级语言程序设计 I 及 II》等方面的基本概念、原理、技术和方法，开展实际应用问题设计求解和对应系统软件开发两大方面的实践。

通过本课程的学习、训练和实践，引导学生熟练掌握问题设计求解和软件编程开发的一般过程、相关技术、方法和途径；训练综合运用所学的理论知识和方法独立分析和解决问题，提高问题分析、问题求解和软件开发能力；培养学生能够针对实际问题，选择适当的数据结构、设计有效算法，提高程序设计的能力和编码质量；训练和学会用系统的观点和软件开发一般规范进行软件设计开发，培养软件工作者所应具备的科学工作方法和作风，提高工程素质；并通过采用团队协作、构建项目组的形式，来培养学生的团队合作与交流能力。

本课程要求学生分组进行（每组 1~3 人），通过一定的调研来自行结合实际应用需求来选题，并由任课教师来对学生选题做筛选评定。要求所设计开发的软件具有一定的实用性和系统完整性，要有较友好的图形交互操作界面，并对输入数据有较强的完整性约束，要以用户需求作为出发点来设计软件界面和功能模块。本课程主要教学环节包括：学生自选任务、软件开发、软件验收、任务报告撰写提交和任务资料整理归集等。

教师评语：

报告成绩：

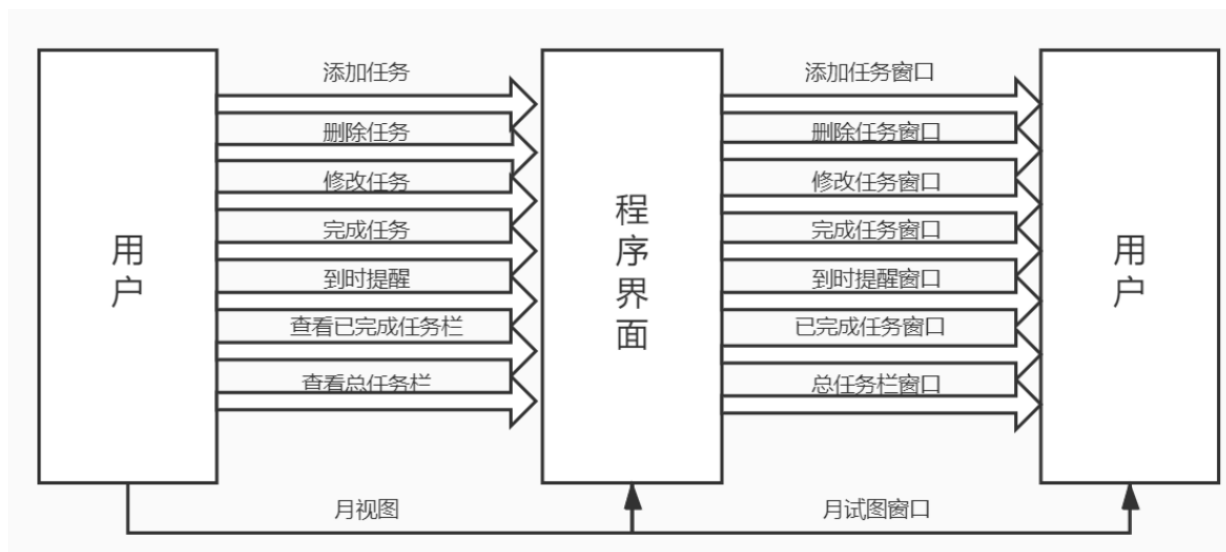
1. 选题背景与应用意义

大学生活不同于之前的中学生活，我们有更多的自由时间，但当代绝大多数大学生没有办法能够合理地规划自己的空闲时间，养成了拖延症的陋习，导致学习和娱乐时间不平衡，作息不规律，效率低下等，从而使得作业经常没办法按时完成或踩点完成，无法合理复习一门课程以致于考试成绩一般，且经常性的晚睡晚起，对身体造成了许多损害。

基于上述现象，我们希望能够开发出一款日程管理类的软件，帮助大学生记录下自己的时间分配安排，同时设置任务标签帮助他们将任务分类，形成不同的模块，还设置了任务等级，以帮助他们确认不同任务的重要性顺序。同时，我们希望添加按时提醒的功能，催促他们在任务结束时间之前完成任务，同时利于同学的习惯养成。

总而言之，我们希望这款日程管理类软件能够帮助当代大学生对自己的时间进行规划，且按照自己的计划正确实施，摆脱由于时间安排混乱造成的学习效率低下的现象。

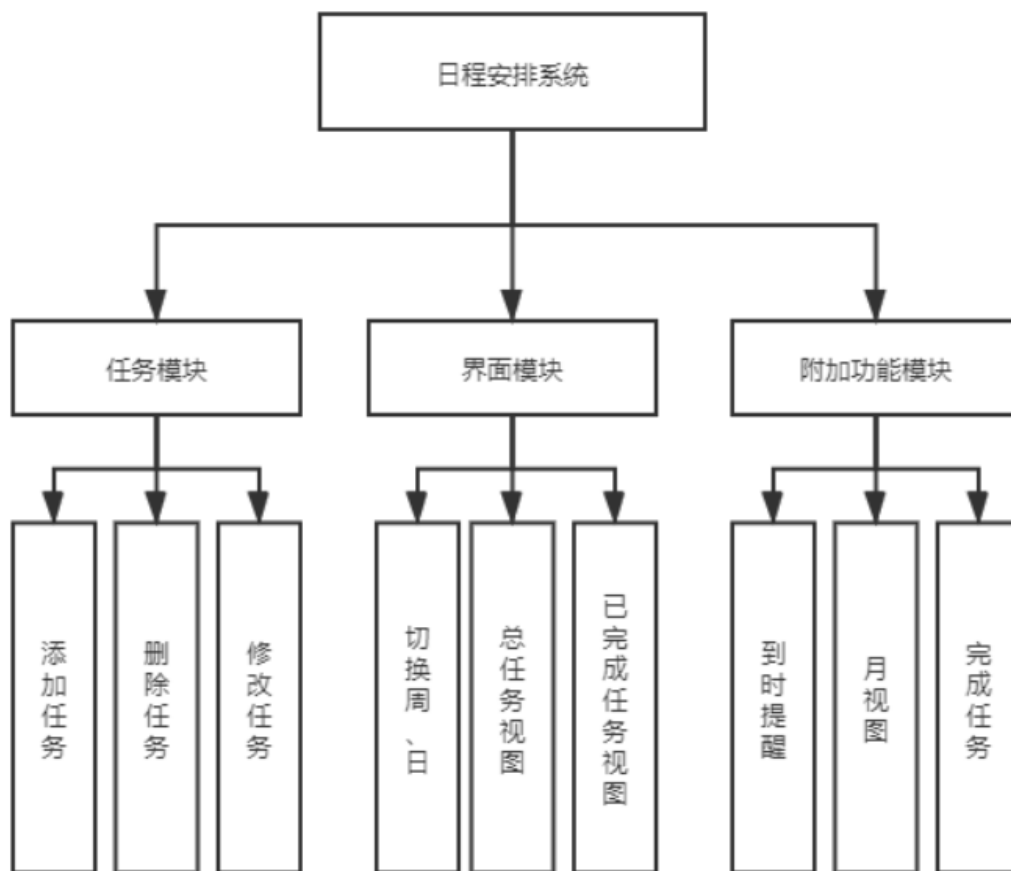
2. 需求分析



用户基本要求：添加、删除、修改、完成任务，设置到时提醒，查看已完成任务栏及总任务栏。提供给用户包含上述操作的按钮的基本界面，在任务栏中添加查看各种视图的工具栏。

设计完成后，应提供给用户一个打包好的可执行的 exe 程序。

3. 系统主要功能设计



任务模块：设置相应的任务类，提供向类中添加任务、删除任务及修改任务的函数。

界面模块：默认提供周视图，提供切换周、日的按钮，在菜单栏中添加总任务视图及已完成任务视图的工具栏，便于用户查看。

附加功能模块：已完成任务后，为用户提供将任务设置为已完成任务的操作。在菜单栏中提供月视图选项，便于查看。给任务添加到时提醒功能，在用户选择的时间点提醒用户完成该任务。

4. 核心算法设计与分析

（根据软件功能设计，概述所用到的主要数据结构；用伪代码或程序流程图的形式来详细描述核心算法的功能及过程，并分析其时间、空间复杂度）

本项目主要用到的数据结构有双向循环链表，键值对，数组，字符串，枚举体。

其中，双向循环列表用于在总任务类和日任务类中的任务存储，采用双向循环列表便于对任务进行添加、删除操作。键值对用于保存日期，打印到周视图中以及建立任务日期与日任务类的联系，用键值对便于查询。字符串用于打印等基本操作。枚举体用于表示任务等级。

核心算法：归并排序。

归并排序的伪代码如下：

```
MERGE(A,p,q,r)
    n1=q-p+1;
    n2=r-q;
    create new arrays L[n1+1] and R[n2+1]
    for i=0 to n1-1
        L[i]=A[p+i]
    for j=0 to n2-1
        R[j]=A[q+1+j]
    L[n1]=1000//假设 1000 是无穷大
    R[n2]=1000
    i=j=0
    for k=p to r
        if(L[i]<=R[j])
            A[k]=L[i]
            i=i+1
        else
            A[k]=R[j]
            j=j+1
MERGE_SORT(A,p,r)
    if p<r
        q=floor((p+r)/2)
        MERGE_SORT(A,p,q)
        MERGE_SORT(A,q+1,r)
        MERGE(A,p,q,r)
```

归并排序的空间复杂度为 $O(n)$ ，时间复杂度为 $O(n\log n)$

5. 系统核心模块实现

（给出编程语言、开发环境及支撑软件、软件系统架构；给出主要数据结构的定义代码以及核心算法对应的关键函数定义代码，包括输入参数的含义、函数输出结果等；核心函数的实现代码段及注解等；主要功能界面截图及对应文字概述等）

本项目采用 C++、Qt 编写，主要使用软件为 Clion, Visual Studio, Qt Creator。

数据结构定义代码：

双向链表：

```
list<Task> allTask_list;
list<Task> finishedTask_list;
QList<QTableWidget*> list;
QList<QLabel*> label_list;
QList<QString> week_list;
QList<QWidget*> tab_list;
QList<QString> item_list;
```

键值对：

```
map<string,DayTask> DayTask_map;
QMap<int,QList<QString>>> weekContainer;
```

数组：

```
int row[54][7];
```

枚举体：

```
enum TaskLevel {
    level_1, level_2, level_3
};
```

关键函数定义代码：

//重载运算符>,方便每一个 Task 对象进行排序

```
bool Task::operator>(const Task &rhs) const {
    if(this->getStartTime() != rhs.getStartTime())
        return this->getStartTime() > rhs.getStartTime();
    else
        return this->getTaskLevel() < rhs.getTaskLevel();
}
```

//添加任务并排序 addTask()

//传入参数任务类对象 Task 并添加至相应日任务集合中，最后更新存储每一天对应日任务类的 map 集合

```
void DayTask::addTask(Task task) {
    task_label_list.push_back(task.getTaskLabel());
    task_list.push_back(task);
    task_list.sort();
    allTask.addAllTask(task);
    allTask.setDayTaskMap(this->time_of_Today, *this);
}
```

```
}
//删除任务 deleteTask()
//传入参数任务类对象 Task 从相应日任务集合中删除，最后更新存储每一天对应日任务
//类的 map 集合
void DayTask::deleteTask(Task task) {
    task_label_list.remove(task.getTaskLabel());
    task_list.remove(task);
    allTask.deleteFromAllTask(task);
    allTask.setDayTaskMap(time_of_Today, *this);
}
//完成任务 finishTask()
//传入参数任务类对象 Task 从相应日任务集合中删除，存储到已完成任务集合中，最后
//更新存储每一天对应日任务类的 map 集合
void DayTask::finishTask(Task task) {
    task_label_list.remove(task.getTaskLabel());
    task_list.remove(task);
    allTask.addFinishedTask(task);
    allTask.setDayTaskMap(time_of_Today, *this);
}
//添加到总任务集合并进行排序 addAllTask(Task task)
//参数 Task 为任务类对象
void AllTask::addAllTask(Task task) {
    allTask_list.push_back(task);
    allTask_list.sort();
}
//添加到已完成任务集合并进行排序 addFinishedTask(Task task)
//参数 Task 为任务类对象
void AllTask::addFinishedTask(Task task) {
    finishedTask_list.push_back(task);
    finishedTask_list.sort();
}

//删除 Task 自总任务集合 deleteFromAllTask(Task task)
//参数 Task 为任务类对象
void AllTask::deleteFromAllTask(Task task) {
    allTask_list.remove(task);
}
//查询时间（某天）是否有对应任务集合
//参数 time_of_Today 为日期，返回布尔值，当天有日任务类则返回 1，否则返回 0
```

```

bool AllTask::ifExistDayTaskList(string time_of_Today) {
    return DayTask_map.count(time_of_Today);
}
//依据时间(某天)查找对应任务集合 getDayTaskMap(string time_of_Today)
//参数 time_of_Today 为日期，返回当天的日任务类。
DayTask &AllTask::getDayTaskMap(string time_of_Today) {    //"10-13"
    return DayTask_map[time_of_Today];
}

//添加每一天及对应任务 map 集合 setDayTaskMap(string time_of_Today,list<Task>
//task_listOfDay)
//参数 time_of_Today 为日期，以及相应日任务类 dayTask，将其存入键值对中
void AllTask::setDayTaskMap(string time_of_Today, DayTask dayTask) {
    DayTask_map[time_of_Today] = dayTask;
}

Task MainWindow::getData(int currentRow) {
    int nowIndex = ui->tabWidget->currentIndex();
    QAbstractItemModel *model = list.at(nowIndex)->model();
    QModelIndex index_Name = model->index(currentRow, 0);
    QModelIndex index_Level = model->index(currentRow, 1);
    QModelIndex index_StartTime = model->index(currentRow, 2);
    QModelIndex index_EndTime = model->index(currentRow, 3);
    QModelIndex index_Label = model->index(currentRow, 4);
    QModelIndex index_today = model->index(currentRow, 5);
    string taskName = model->data(index_Name).toString().toString();
    string taskLevel = model->data(index_Level).toString().toString();
    string time_Of_Today =
model->data(index_today).toString().toString();
    TaskLevel level;
    if(taskLevel == "一")    level = TaskLevel::level_1;
    else if(taskLevel == "二")    level = TaskLevel::level_2;
    else level = TaskLevel::level_3;
    string taskStartTime =
model->data(index_StartTime).toString().toString();
    string taskEndTime =
model->data(index_EndTime).toString().toString();
    string taskLabel = model->data(index_Label).toString().toString();
    Task task(taskStartTime, taskEndTime, taskName, level, taskLabel);
    return task;
}

```

参数：currentRow 表示用户当前所选任务在界面上的行号，返回用户所选行的任务具体实现类。


```

void CalendarWidget::paintCell(QPainter *painter, const QRect &rect, const QDate &date) const
{
    painter->save();
    painter->setRenderHint(QPainter::Antialiasing);
    /* 绘制背景 */
    QString time_Of_Task = date.toString("MM-dd");
    int n = allTask.getDayTaskMap(time_Of_Task.toString()).getTaskList().size();

    painter->setPen(Qt::NoPen);
    switch(n){
        case 0: painter->setBrush(QColor(240, 255, 255));
            break;
        case 1: painter->setBrush(QColor(210, 210, 255));
            break;
        case 2: painter->setBrush(QColor(0, 206, 250));
            break;
        case 3: painter->setBrush(QColor(0, 134, 139));
            break;
        case 4: painter->setBrush(QColor(34, 149, 34));
            break;
        case 5: painter->setBrush(QColor(255, 215, 0));
            break;
        case 6: painter->setBrush(QColor(205, 92, 0));
            break;
        case 7: painter->setBrush(QColor(255, 106, 106));
            break;
        default: painter->setBrush(QColor(139, 0, 0));
            break;
    }
    painter->drawRoundedRect(rect.x()+1, rect.y()+1, rect.width()-2, rect.height()-2, 3, 3);

    /* 绘制前景 */
    QFont dateFont = painter->font();
    dateFont.setPixelSize(rect.height()-10);
    painter->setFont(dateFont);
    painter->setPen(QColor(0, 0, 0)); // 字体颜色
    QRect dateRect = QRect(rect.x()+3, rect.y()+3, rect.width()-6, rect.height()-6);
    painter->drawText(dateRect, Qt::AlignCenter, QString::number(date.day()));

    painter->restore();
}

```

主要功能界面：

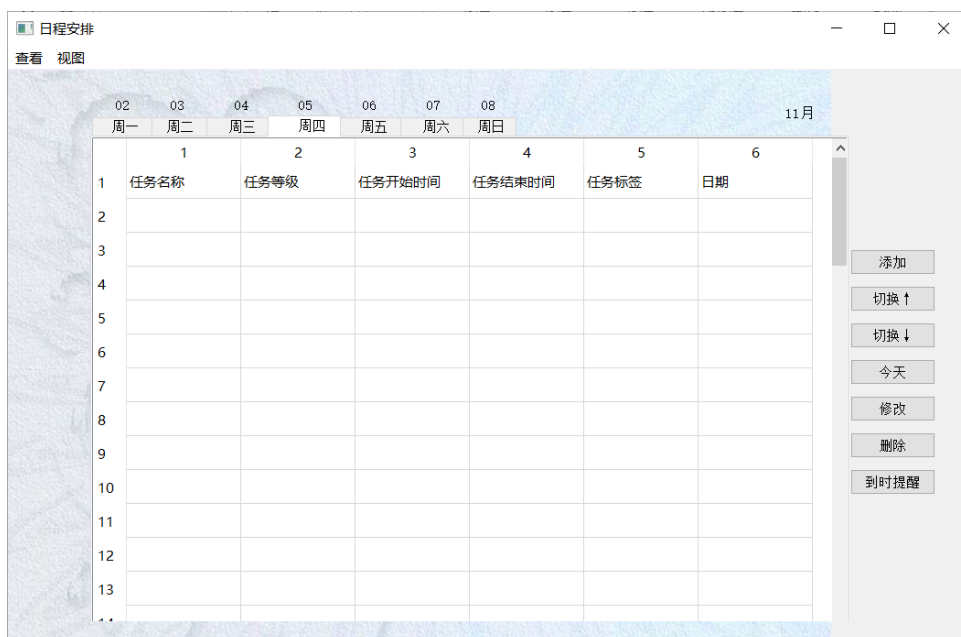


图 1 主视图

Dialog

?

×

请输入任务名称：

请输入开始时间：

周四 11-05 22:47

请输入结束时间：

周四 11-05 22:47

请选择任务等级：

—

请输入任务标签：

确认

取消

图 2 添加任务界面

日程安排

查看 视图

0203040506070811月

周一周二周三周四周五周六周日

	1	2	3	4	5	6
1	任务名称	任务等级	任务开始时间	任务结束时间	任务标签	日期
2	学习	—	22:49	22:49	学习	11-05
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						

modify

?

×

请输入任务名称：

OK

Cancel

添加

切换↑

切换↓

今天

修改

删除

到时提醒

图 3 修改任务界面

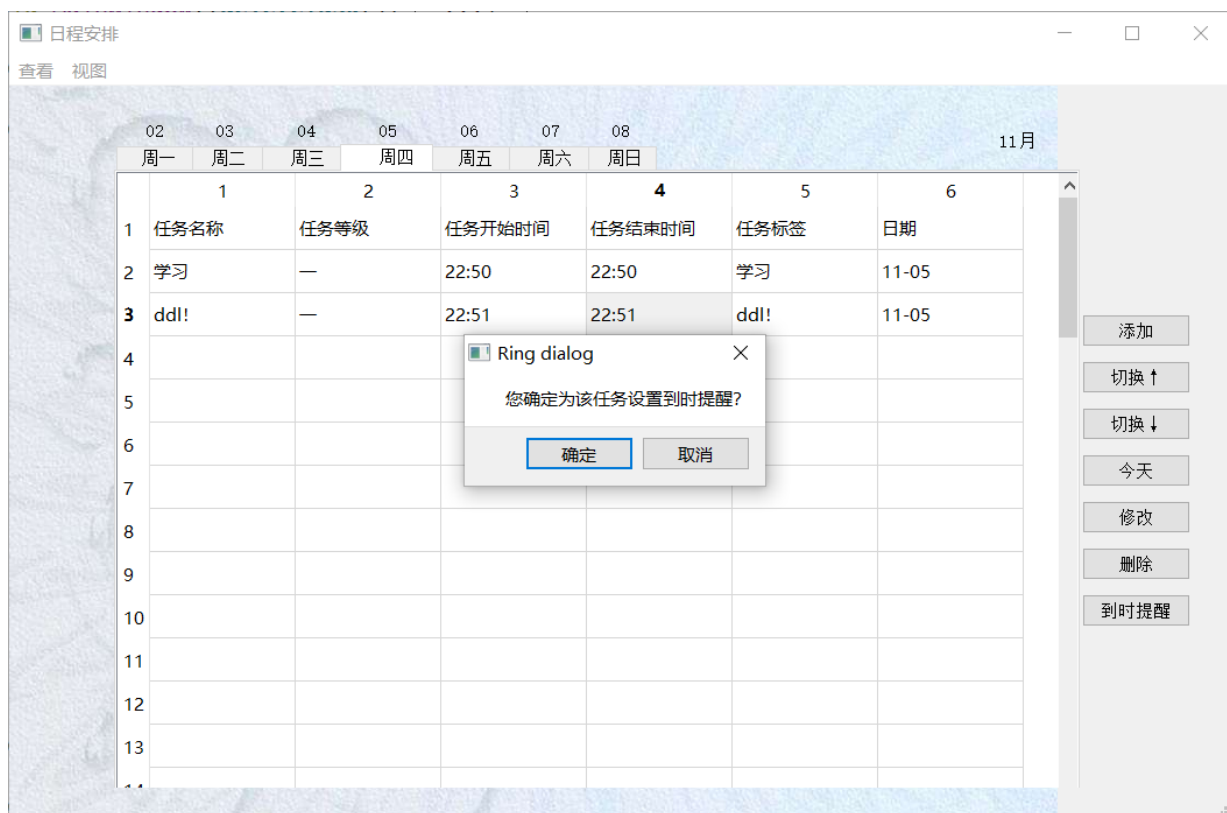


图 4 到时提醒对话框

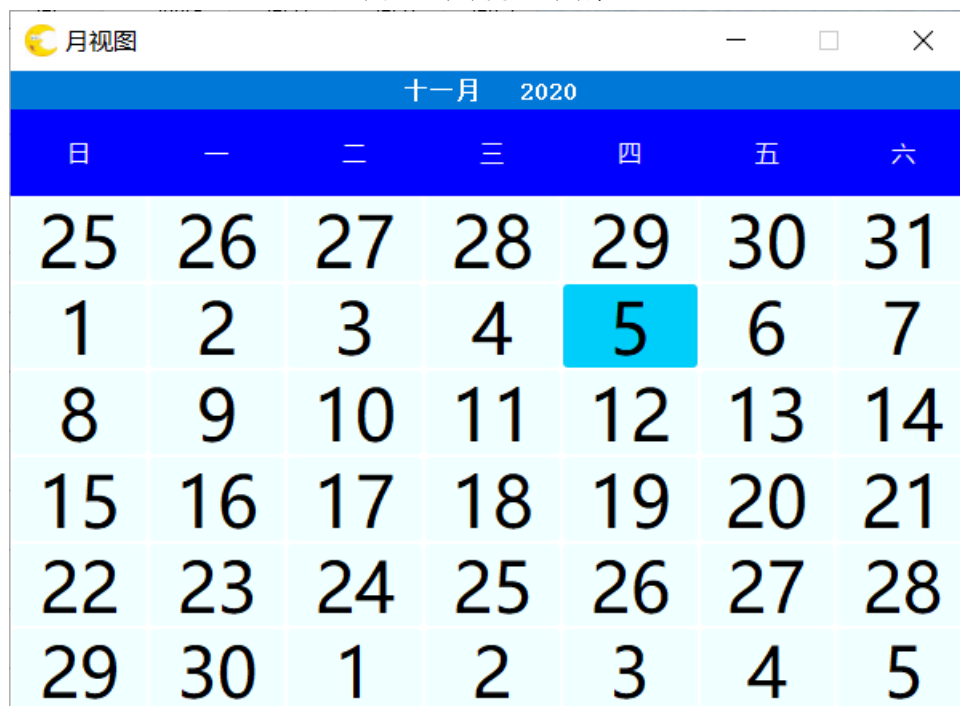
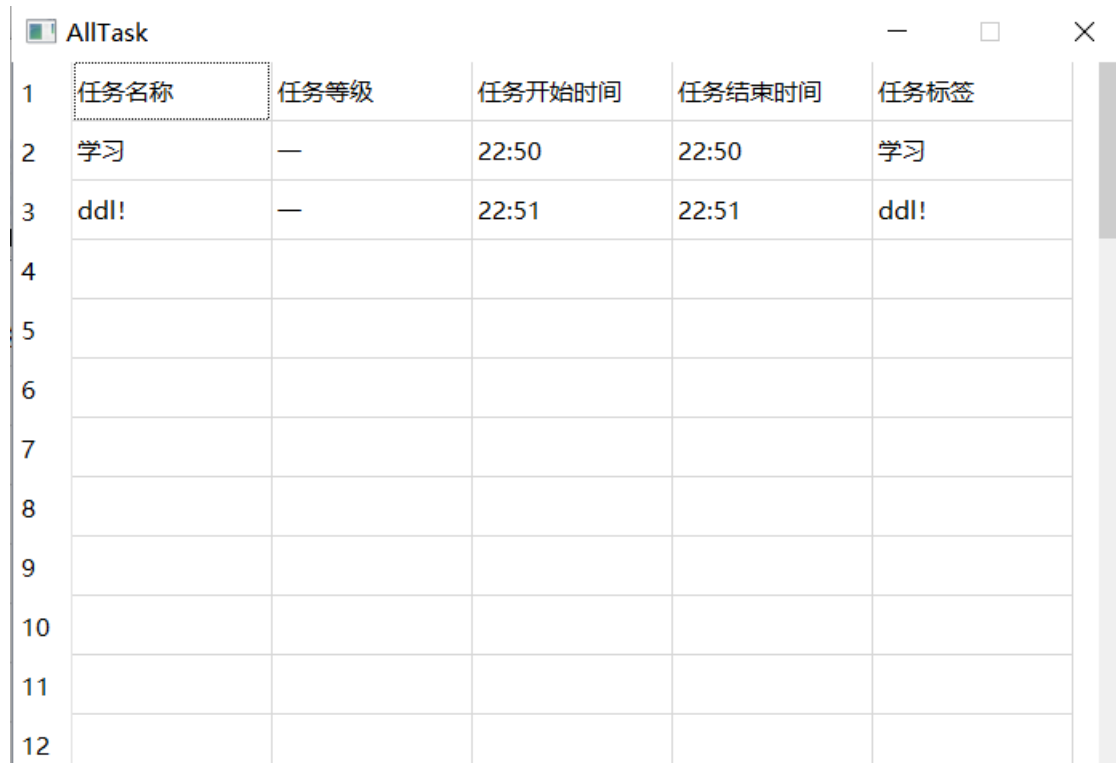


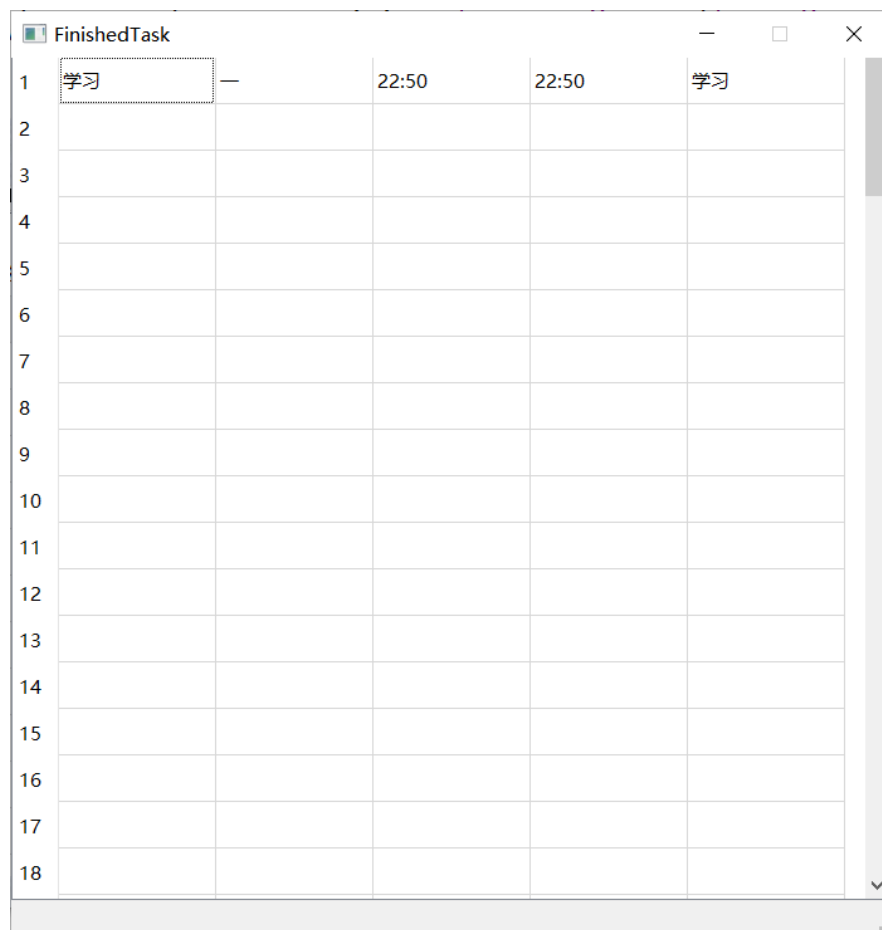
图 5 月视图界面



The image shows a window titled "AllTask" with a table of tasks. The table has 6 columns: "任务名称" (Task Name), "任务等级" (Task Level), "任务开始时间" (Task Start Time), "任务结束时间" (Task End Time), and "任务标签" (Task Tag). The first two rows contain data: "学习" (Study) and "ddl!". The rest of the rows are empty. The window has a standard macOS-style title bar with a close button (X) and a scroll bar on the right.

	任务名称	任务等级	任务开始时间	任务结束时间	任务标签
1	学习	—	22:50	22:50	学习
2	ddl!	—	22:51	22:51	ddl!
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					

图 6 总任务栏



The image shows a window titled "FinishedTask" with a table of completed tasks. The table has 6 columns: "任务名称" (Task Name), "任务等级" (Task Level), "任务开始时间" (Task Start Time), "任务结束时间" (Task End Time), and "任务标签" (Task Tag). The first row contains data: "学习" (Study), "—", "22:50", "22:50", and "学习". The rest of the rows are empty. The window has a standard macOS-style title bar with a close button (X) and a scroll bar on the right.

	任务名称	任务等级	任务开始时间	任务结束时间	任务标签
1	学习	—	22:50	22:50	学习
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					

图 7 已完成任务栏

6. 调试分析记录

遇到的问题：

部分情况未考虑到导致的 bug，如 1. 用户可以删除初始化界面栏，2. 设置任务的开始时间可以小于任务的结束时间的 bug，3. 一个提示对话框会出现两次的 bug，4. 切换周的时候，周视图上部分日所对应的任务与实际任务不对应。

解决过程：

1. 增加判断语句，当用户所选栏为默认栏时，不提供删除操作
2. 增加判断语句，当任务开始时间小于任务的结束时间时，弹出提示信息的对话框。
3. 在每次调用打印函数时，先解除任务与相应对话框的信号槽。
4. 每次切换周的时候，先对周视图做清屏操作，然后用一个 for 循环去查找对应天的任务，并将其打印到程序显示窗口上。

功能改进：

1. 增加全局变量 `currentWeekIndex`, `currentIndex`, 便于记录当前用户所在周及所在星期数。
2. 将相同代码抽象成一个函数，减少相同代码出现次数。
3. 建立一个二维数组，保存每一周的每一天的任务数量，便于控制打印时的 for 循环。
4. 将 `label` 标签显示的日期存入一个键值对数据结构，通过周编号 `currentWeekIndex` 获取该周的日期，再打印到 `label` 上，避免了运用查找算法，降低时间复杂度。
5. 当用户删除的任务属于当天的任务时（增加 `if` 语句进行判断），减少打印该周的任务数，减少 for 循环和查找的次数，降低时间复杂度。
6. 当用户修改的任务属于当天任务时（增加 `if` 语句进行判断），减少打印时 for 循环的遍历次数和查找次数，降低时间复杂度。

7. 运行结果与分析

（要有多组正确的实际测试数据，并给出相应运行结果截图，并对多组结果进行比较分析）
添加多项任务：

	02	03	04	05	06	07	08	11月
	周一	周二	周三	周四	周五	周六	周日	
1	任务名称	任务等级			任务开始时间	任务结束时间	任务标签	日期
2	预习算法	—			14:00	14:16	学习	11-05
3	预习数论	—			15:00	16:00	学习	11-05
4	预习汇编	—			16:00	17:00	学习	11-05
5								
6								
7								
8								
9								
10								
11								
12								
13								

	02	03	04	05	06	07	08	11月
	周一	周二	周三	周四	周五	周六	周日	
1	任务名称	任务等级			任务开始时间	任务结束时间	任务标签	日期
2	复习算法	—			14:16	14:16	学习	11-07
3	开黑	二			20:00	22:00	game!	11-07
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								



输入开始时间小于结束时间及报错窗口：

Dialog

请输入任务名称：跑步

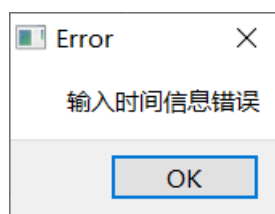
请输入开始时间：周六 11-07 22:00

请输入结束时间：周六 11-07 21:00

请选择任务等级：一

请输入任务标签：运动

确认 取消



未完成任务时总任务栏和已完成任务栏：

AllTask					
	任务名称	任务等级	任务开始时间	任务结束时间	任务标签
1	汇编作业	一	08:00	10:00	学习
2	海贼王	二	12:00	12:20	看视频
3	预习算法	一	14:00	14:16	学习
4	复习算法	一	14:16	14:16	学习
5	预习数逻	一	15:00	16:00	学习
6	预习汇编	一	16:00	17:00	学习
7	开黑	二	20:00	22:00	game!
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					

FinishedTask					
	任务名称	任务等级	任务开始时间	任务结束时间	任务标签
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					

完成”海贼王”任务后总任务栏和已完成任务栏

AllTask					
	任务名称	任务等级	任务开始时间	任务结束时间	任务标签
1	汇编作业	一	08:00	10:00	学习
2	海贼王	二	12:00	12:20	看视频
3	预习算法	一	14:00	14:16	学习
4	复习算法	一	14:16	14:16	学习
5	预习数逻	一	15:00	16:00	学习
6	预习汇编	一	16:00	17:00	学习
7	开黑	二	20:00	22:00	game!
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					

FinishedTask					
1	海贼王	二	12:00	12:20	看视频
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					

修改预习汇编的任务等级及任务开始时间：

4	预习汇编	二	16:20	17:00	学习	11-05
---	------	---	-------	-------	----	-------

相应总任务栏中变化：

7	预习汇编	二	16:20	17:00	学习	
---	------	---	-------	-------	----	--

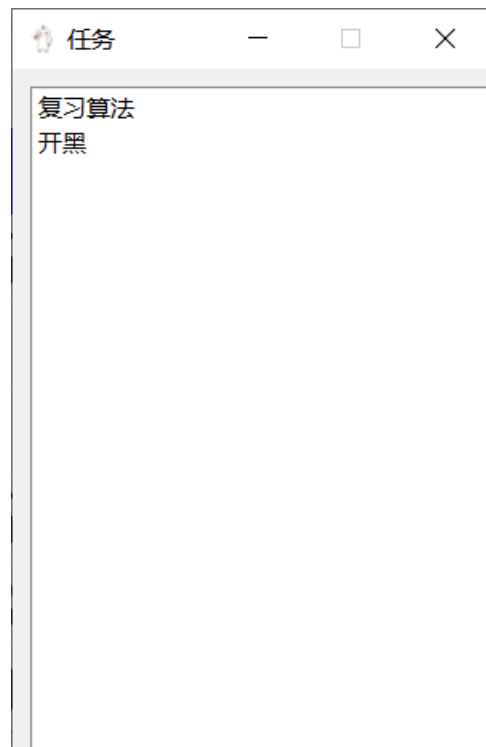
月视图（不同颜色对应不同任务数目）：

十一月 2020						
日	一	二	三	四	五	六
25	26	27	28	29	30	31
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	1	2	3	4	5

11.5 日任务：

任务
预习算法
预习数逻
预习汇编

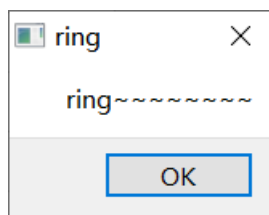
11.7 日任务：



11.8 日任务：



到时提醒窗口：



8. 教师指导建议及解决记录

8.1 开题指导及中期检查

在开题指导时，老师建议我们多做一些创新的功能，如通过文件导入、到时提醒等功能。

在中期检查时，我们任务进度正常，老师未给出太多意见。

针对老师在开题时提出的部分创新功能，在中期实现了我们软件的基本功能之后，我们采用了 C++ 中的 ostream 流，使用二进制的存储及读取操作，并进行了时间长达一周半的调试，但文件的存储及读取部分出现了未知错误，导致没有办法从文件中正确读取或读取信息缺失等问题，查阅多篇博客也未能解决，最后放弃。且由于此项耽误的时间过久，导致在结题时未能添加到时提醒功能。

8.2 软件验收

软件验收时，老师还是建议我们添加到时提醒的功能。

在结题答辩过后，我们又经过了调试，完成了到时提醒功能。

9. 总结

此次软件设计课，是我们第一次用学过的编程语言写出一点东西，这次课程对我而言总体来讲还是很有意义的。本门课程提供了一个开发一个很基础的软件的机会，在这个过程中，我学习了新的知识，如 Qt、C++。同时意识到了开发软件并不意味着开始编码，在软件开发之前，要先做好需求分析，概要设计，模块设计等。然后思考实现大致功能需要用到什么算法以及数据结构。此外，要学会小组分工合作，不同于以往大多都是单干的学习任务，开发一款软件需要多人一起合作，这能提高工作效率，缩短开发时间。分工完后，组内人员仍要保持积极沟通，互相了解项目进度，有利于从整体把握这个项目。

在本次软件开发过程中，仍然遇到了许多问题，如面向对象学习的不够透彻，在相互调用时会出现一些问题。Qt 中部分控件的功能了解不够全面，导致 ui 设计较为简陋，在组内进行 Qt 与 C++ 之间的对接时，也出现部分库问题。还有在对软件的功能进行测试

的过程中出现了许多没有预想到的 BUG 等等问题。解决这些问题花费了大量时间，但很有价值。如对一些特殊情况时出现的特定 BUG，要专门设置对应于此的解决方案，解决边缘问题，这对完善一款软件是非常重要的。同时，对一些未知原因的 BUG，可以采用“COUT” debug 的方法，通过提供多个测试数据，找出出现该 bug 的原因，通过多个这种“痛苦”的过程，提高了我的程序调试能力。

在这个软件开发过程中，我还对自己的代码进行了规范。与人合作要求自己写的代码要让别人一目了然，这样能提高软件开发的效率。同时，由于初期任务较重，时间较为紧张，代码编写不太规范，重复代码较多，在项目中期过后，我对自己写的重复代码进行了抽象成函数的操作，减少了许多冗余代码，这些都有利于我养成良好的编码习惯。