

# 浙江大学

## 程序设计专题

### 大程序报告



- |    |       |             |     |                   |
|----|-------|-------------|-----|-------------------|
| 1. | 学生姓名： | <u>沈吕可晟</u> | 学号： | <u>3180101044</u> |
| 2. | 学生姓名： | <u>周宇轩</u>  | 学号： | <u>3180105058</u> |
| 3. | 学生姓名： | <u>陆子仪</u>  | 学号： | <u>3180101939</u> |





2018~2019 春夏学期    2019 年 6 月 14 日

## 1 题目描述和题目要求

### 大程序题目：俄罗斯方块

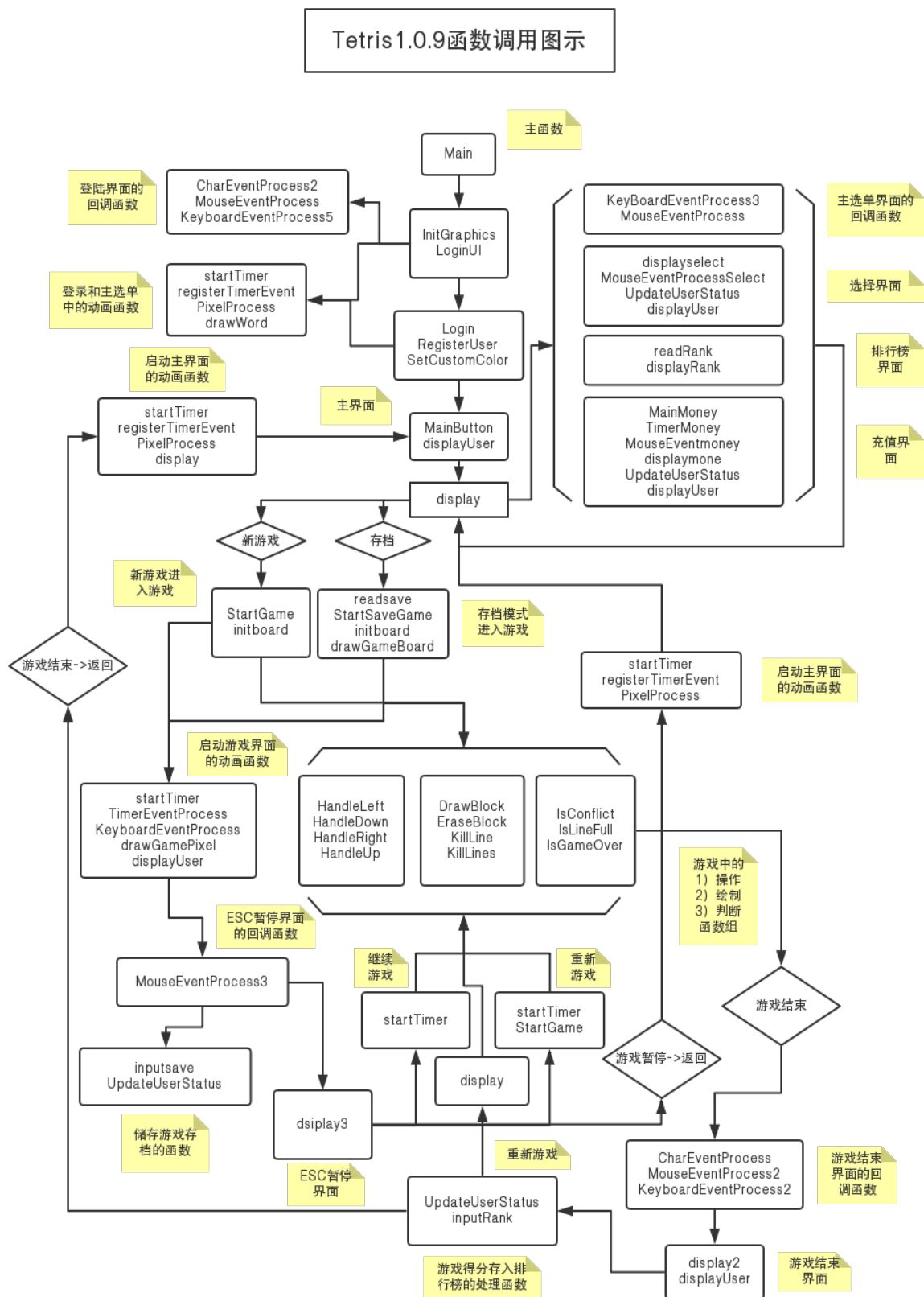
**题目要求：**基于 libgraphics，设计和实现俄罗斯方块游戏。俄罗斯方块的基本规则是移动、旋转和摆放游戏自动输出的各种方块，使之排列成完整的一行或多行并且消除得分。

## 2 需求分析

- (1) 进入游戏界面后会有背景音乐响起，提高用户体验
- (2) 游戏初始界面选择登陆或注册用户或以游客身份进入游戏
- (3) 游戏开始时选择新游戏或者读取存档开始游戏
- (4) 用  键旋转方块
- (5) 用  键和  键左右移动方块
- (6) 用  键使方块加速下落
- (7) 用 `ctrl` 键直接消除最后一行
- (8) 用 `space` 键使得方块一键下落至底部
- (9) 用 `Esc` 键暂停游戏，退出游戏
- (10) 游戏结束后可选择是否将成绩计入排行榜
- (11) 在选项界面中选择游戏难度以及界面颜色
- (12) 在排行榜界面中查看成绩排行
- (13) 在模拟充值系统中进行游戏充值，金币可用于购买道具消除一行。金币也可以通过游戏获得，每局游戏结算时每 50 点积分会被自动兑换成 1 金币记入账户中。
- (14) 按退出按钮退出游戏

### 3 总体设计

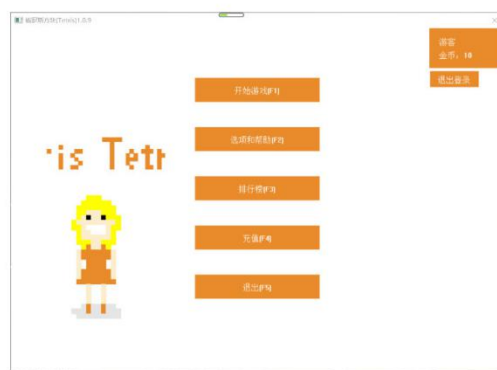
#### 3.1 功能模块设计



## 3.2 游戏界面截图如下：



登录界面



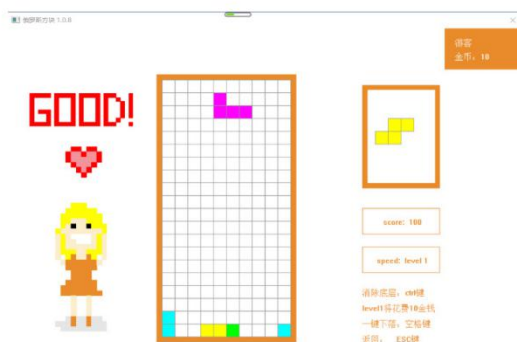
主选单界面



选项及帮助界面



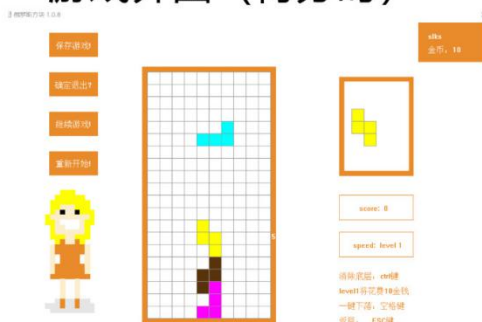
游戏开始模式选择界面



游戏界面 (得分时)



充值界面



游戏界面

### 3.3 数据结构设计

#### 3.3.1 方块数据储存结构:

```

struct SHAPE
{
    int box[8];
    char *color;
    /*每个游戏方块的颜色*/
    int next;
    /*下个游戏方块的编号*/
};

struct SHAPE shapes[MAX_BOX]=
{
    //■■■■  ■■■■  ■■■■  □□□□
    //■■■■  ■□□□  □■■■  □□■□
    //■■■■  □□□□  □■■■  ■■■■
    //□□□□  □□□□  □□□□  □□□□

    {0,1,0,2,0,3,1,1,"CYAN",1},
    {0,2,0,3,1,3,2,3,"CYAN",2},
    {0,3,1,3,1,2,1,1,"CYAN",3},
    {0,1,1,2,1,2,2,"CYAN",0},

    //□■■■  ■□□□  ■■■■  ■■■■
    //□■■■  ■■■■  ■□□□  □□■□
    //■■■■  □□□□  ■□□□  □□□□
    //□□□□  □□□□  □□□□  □□□□

    {0,1,1,1,2,1,3,"MAGENTA",5},
    {0,2,0,3,1,2,2,2,"MAGENTA",6},
    {0,1,0,2,0,3,1,3,"MAGENTA",7},
    {0,3,1,3,2,3,2,2,"MAGENTA",4},

    //■■■■  □■■■
    //■■■■  ■■■■
    //□■■■  □□□□
    //□□□□  □□□□

    {0,3,0,2,1,2,1,1,"YELLOW",9},
    {0,2,1,2,1,3,2,3,"YELLOW",8},

    //□■■■  ■■■■
    //■■■■  □■■■
    //■■■■  □□□□
    //□□□□  □□□□

    {0,1,0,2,1,2,1,3,"BROWN",11},
    {0,3,1,3,1,2,2,2,"BROWN",10},

    //□■■■  ■□□□  ■■■■  □■■■
    //■■■■  ■■■■  □■■■  ■■■■
    //□□□□  ■□□□  □□□□  □■■■
    //□□□□  □□□□  □□□□  □□□□

    {0,2,1,2,1,3,2,2,"GREEN",13},
    {0,1,0,2,0,3,1,2,"GREEN",14},
    {0,3,1,3,1,2,2,3,"GREEN",15},

```

```
{0,2,1,1,1,2,1,3,"GREEN",12},

//■■■■ ■■■■
//■■■■ □□□□
//■■■■ □□□□
//■■■■ □□□□

{0,0,0,1,0,2,0,3,"RED",17},
{0,1,1,1,2,1,3,1,"RED",16},

//■■■■
//■■■■
//□□□□
//□□□□

{0,2,0,3,1,2,1,3,"BLUE",18},
};
```

3.3.2 图像数据储存结构（下述为一个例子，具体见源代码处内容）：

```
int pixel[6][7] = {
    {0,1,1,0,1,1,0},
    {1,2,2,1,2,2,1},
    {1,2,2,2,2,2,1},
    {0,1,2,2,2,1,0},
    {0,0,1,2,1,0,0},
    {0,0,0,1,0,0,0},
};
int i, j;
for(i = 0;i < 7;i++){
    for(j = 0;j < 6;j++){
        if(pixel[j][i]){
            switch(pixel[j][i])
            {
                case 1: SetPenColor("Red"); break;
                case 2: SetPenColor("Pink"); break;
            }
            drawRectangle(x+size*i,y-size*j+num*size,size,size,1);
        }
    }
}
```

3.4 函数功能描述

整体程序共 57 个函数，具体功能如下：

函数原型	功能描述	参数描述	返回值描述	重要局部变量定义	重要局部变量用途描述	函数算法描述
void KeyBoardEventProcess(int key,int event);	处理游戏进行中的键盘输入	int key: 活动的按键 int event: 按键的状态	无	无	无	无
void KeyboardEventProcess2(int key, int event);	处理游戏结束后的键盘输入	int key: 活动的按键 int event: 按键的状态	无	无	无	无
void KeyboardEventProcess3(int key, int event);	处理主界面的快捷键盘输入	int key: 活动的按键 int event: 按键的状态	无	无	无	无
void KeyboardEventProcess4(int key, int event);	处理游戏暂停后的键盘输入	int key: 活动的按键 int event: 按键的状态	无	无	无	无
void	处理是否选择存档处	int key: 活动的按键	无	无	无	无

KeyboardEventProcess5(int key, int event);	的键盘输入	int event: 按键的状态				
void CharEventProcess(char ch);	处理游戏结束后的键盘输入	char ch: 键盘输入的 ASCII 字符	无	无	无	无
void CharEventProcess2(char ch);	处理游戏登录界面的键盘输入 (主要针对输入过程中的使用)	char ch: 键盘输入的 ASCII 字符	无	无	无	无
void MouseEventProcess(int x, int y, int button, int event);	处理游戏开始前的鼠标输入	int x: 鼠标实时横坐标 int y: 鼠标纵实时坐 int button: 鼠标活动按键 int event: 鼠标按键状态	无	无	无	无
void MouseEventProcess2(int x, int y, int button, int event);	处理游戏结束后的鼠标输入	int x: 鼠标实时横坐标 int y: 鼠标纵实时坐 int button: 鼠标活动按键 int event: 鼠标按键状态	无	无	无	无
void MouseEventProcess3(int x, int y, int button, int event);	处理游戏暂停时的鼠标输入	int x: 鼠标实时横坐标 int y: 鼠标纵实时坐 int button: 鼠标活动按键 int event: 鼠标按键状态	无	无	无	无
void MouseEventProcessSelect(int x, int y, int button, int event);	处理进入选项界面之后的鼠标输入	int x: 鼠标实时横坐标 int y: 鼠标纵实时坐 int button: 鼠标活动按键 int event: 鼠标按键状态	无	无	无	无
void MouseEventmoney(int x, int y, int button, int event);	处理进入充值界面之后的鼠标输入	int x: 鼠标实时横坐标 int y: 鼠标纵实时坐 int button: 鼠标活动按键 int event: 鼠标按键状态	无	无	无	无
void display();	处理游戏开始前的显示刷新	无	无	无	无	无
void display2();	处理游戏结束后的显示刷新	无	无	static char result[50] = "";得到的姓名信息 static char name[50] = "Unname";默认的姓名字符串 static int confirmed = 0; 是否确认了姓名的输入 static int rankstatus = 0;排名函数的返回值	无	无
void display3();	处理游戏暂停时的显示刷新	无	无	无	无	无
void displayselect();	处理选择界面的显示刷新	无	无	无	无	无
void displayUser();	处理游戏登录界面的显示刷新	无	无	无	无	无
void displaymoney();	处理游戏模拟充值界面的显示刷新	无	无	无	无	无
void MainButton();	游戏开始前界面的绘制	无	无	无	无	无
void Mainmoney(int addmoney);	游戏充值界面的绘制	无	无	无	无	无
void loginUI();	游戏登陆界面的绘制	无	无	无	无	无
void PixelProcess(int timerID);	游戏开始前界面的动画数据处理	int timerID: 计时器编号	无	无	无	无
int readRank(void);	读取文件中的排名信息	无	-1: 提示打开文件失败 -2: 提示关闭	无	无	无

			文件失败 0: 正常			
int inputRank(char name[], int score);	将姓名和分数经过排名后输入排名文件	char name[]: 姓名 int score: 分数	-1/-3: 提示打开文件失败 -2/-5/-6: 提示关闭文件失败 -4: 提示排行榜已满(此次得分过低) 正整数: 保存分数成功, 返回排行榜总共有多少条记录	无	无	直接插入排序
int displayRank(int page);	处理排行榜的绘制	int page: 排行榜页数	1: 存在下一页 0: 不存在下一页	无		无
void DrawSquare(double x, double y);	绘制俄罗斯方块中的一个标准方块	double x: 起始横坐标 double y: 起始纵坐标	无	无		无
void drawRec(double x, double y, double width, double height, double thickness, char* outercolor, char* innercolor);	绘制一个带边框的长方形	double x: 起始横坐标 double y: 起始纵坐标 double width: 长方形宽度 double height: 长方形高度 double thickness: 边框厚度 char* outercolor: 边框颜色 char* innercolor: 长方形颜色	无	无		无
void TimerEventProcess(int timerID);	处理游戏进程中的计时器响应函数	int timerID: 计时器编号	无	无	无	无
void TimerMoney(int timerID);	处理模拟游戏充值过程中的计时器响应函数	int timerID: 计时器编号	无	无	无	无
void wait();	处理游戏中的暂停函数	无	无	无	无	无
void initboard();	游戏界面的绘制函数	无	无	无	无	无
void StartGame();	游戏数值的初始化	无	无	无	无	无
void drawWord(double x, double y, double size, int num);	在游戏进行中绘制像素对话框	double x: 起始横坐标 double y: 起始纵坐标 double size: 像素点大小 int num: 显示序列	无	无	无	无
void StartSaveGame();	从存档开始的游戏数值初始化	无	无	无	无	无
void drawGameBoard();	用于绘制中心游戏板	无	无	无	无	无
void drawGamePixel();	在游戏进行中绘制像素动画	无	无	无	无	无
void DrawBlock(int BlockIndex, double sx, double sy);	绘制下落的俄罗斯方块	int BlockIndex: 方块编号 double sx: 横坐标 double sy: 纵坐标	无	无	无	无
void drawQR(double x, double y, double size);	绘制支付二维码	double x: 起始横坐标 double y: 起始纵坐标 double size: 像素点大小	无	无	无	无

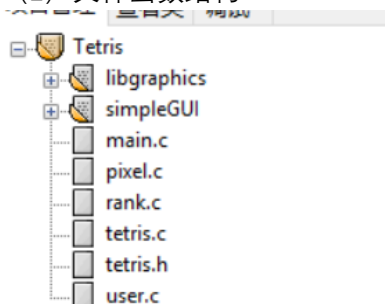


void drawHeart(double x, double y, double size, int num)	绘制小人动画心	double x: 起始横坐标 double y: 起始纵坐标 double size: 像素点大小 int num: 显示序列	无	无	无	无
void EraseBlock(int BlockIndex, double sx, double sy);	擦除下落的俄罗斯方块	int BlockIndex: 方块编号 double sx: 横坐标 double sy: 纵坐标	无	无	无	无
void HandleLeft(int BlockIndex, double *x, double *y);	游戏中对 Left 键的处理函数	int BlockIndex: 方块编号 double *x: 横坐标 double *y: 纵坐标	无	无	无	无
void HandleRight(int BlockIndex, double *x, double *y);	游戏中对 Right 键的处理函数	int BlockIndex: 方块编号 double *x: 横坐标 double *y: 纵坐标	无	无	无	无
void HandleUp(int *BlockIndex, double *x, double *y);	游戏中对 Up 键的处理函数	int BlockIndex: 方块编号 double *x: 横坐标 double *y: 纵坐标	无	无	无	无
int HandleDown(int BlockIndex, double *x, double *y);	游戏中对 Down 键以及对方块下落的的处理函数	int BlockIndex: 方块编号 double *x: 横坐标 double *y: 纵坐标	无	无	无	无
int SpaceDown(int BlockIndex, double *x, double *y);	游戏中对 Space 键的处理函数	int BlockIndex: 方块编号 double *x: 横坐标 double *y: 纵坐标	无	无	无	无
int IsConflict(int BlockIndex, double sx, double sy);	判断方块是否与边界冲突的函数	int BlockIndex: 方块编号 double sx: 横坐标 double sy: 纵坐标	1- 冲突 0- 未冲突	无	无	无
int IsLineFull(int y);	判断某行是否已满的函数	int y: 行坐标	1- 行满 0- 行未满	无	无	无
void KillLine(int y);	游戏中消除一行的函数	int y: 行坐标	无	无	无	无
int KillLines(int y);	游戏中对所有已满行进行消除的函数	int y: 行坐标	返回消除行数	无	无	无
int IsGameOver();	判断游戏是否结束的函数	无	1- 结束 0- 未结束	无	无	无
void Trim(char *src);	给定一个 C-String 的变量(非字面值常量), 去除首尾两端 Blank 的字符	char *src: 字符串地址	无	无	无	无
void SetCustomColor(int num);	设定游戏方块的颜色	int num: 颜色选择	无	无	无	无
int login(char name[], char password[]);	处理用户的登录函数	char name[]: 输入的用户名 char password[]: 输入的用户密码	0: 用户不存在 1: 登陆成功 2: 用户存在但密码错误 -1、-2: 文件打开、关闭问题	无	无	无
int registeruser(char name[], char password[]);	处理用户的注册函数	char name[]: 输入的用户名 char password[]: 输入的用户密码	0: 用户已注册 1: 注册成功 2: 用户名为空 3: 密码为空 -1、-2: 文件的打开、关闭问题	无	无	无
void UpdateUserStatus();	更新用户数据函数	无	无	无	无	第一次以 r

						打开文档并进行遍历，使用了链表进行数据储存，当遍历到需要更新的用户数据时，使用最新数据；第二次以 w+ 打开文档，用链表将文档数据重新写入
int readsave(char name[]);	读取用户的存档函数	char name[]: 需要读取存档的用户名	无	无	无	无
int inputsave();	将当前的用户的游戏状态存入存档文件	无	无	无	无	同 UpdateUserStatus 函数的处理方式

### 3.5 程序文件结构（如图）

#### (1) 文件函数结构



综述：工程文件分为 6 个源程序代码.c 文件和 libgraphics 库及 simpleGUI 库，各个源程序.c 文件所包含的函数如下所示

#### 3.5.1 main.c

Main

#### 3.5.2 pixel.c

五个函数：

drawWord  
drawTitle  
drawPixel  
drawQR  
drawHeart

#### 3.5.3 rank.c

两个函数：

readRank  
inputRank

一个结构：

score

#### 3.5.4 tetris.c

四十二个函数：

```

KeyBoardEventProcess (1, 2, 3, 4, 5)
CharEventProcess(1, 2)
MouseEventProcess (1, 2, 3, Select, money)
display (1, 2, 3, select, money, rank)
MainButton, Mainmoney
PixelProcess
drawSquare, drawRec, drawGameBoard, drawWord,
TimerEventProcess, TimerMoney
Wait
loginUI , Initboard, StartGame, StartSaveGame
drawGamePixel
DrawBlock, EraseBlock
HandleLeft, HandleRight, HandleUp, HandleDown, SpaceDown
IsConflict, IsLineFull
KillLine, KillLines, IsGameOver

```

### 3.5.5 user.c

#### 三个结构:

```

SHAPE
Save
User

```

#### 七个函数:

```

Tris
SetCustomColor
Login
Registeruser
UpdateUserStatus
Readsave、inputsave

```

### 3.5.6 tetris.h

包含了如右图的内容，以及以上的函数原型及注释。

```

//本地头文件包含
#include "imgui.h"

//预处理指令
#define blocksize 0.25
#define boardwidth 2.7
#define boardheight 5.2
#define statuswidth 1.5
#define statusheight 2.0
#define MAX_BOX 19
#define BSIZE 0.25
#define FILLED 1
#define EMPTY 0
#define TRUE 1
#define FALSE 0

#endif

//头文件包含
#ifndef TETRIS_H
#define TETRIS_H
#include "graphics.h"
#include "extgraph.h"
#include "genlib.h"
#include "simpio.h"
#include "conio.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stddef.h>
#include <windows.h>
#include <olecli.h>
#include <mmsystem.h>
#include <wingdi.h>
#include <ole2.h>
#include <ocidl.h>
#include <winuser.h>
#pragma comment(lib, "winmm.lib")

```

### 3.6 多文件构成机制

在 tetris.h 中包含了如下的头文件保护:

```

#ifndef tetris_functions
#define tetris_functions
/*文件内容*/
#endif

```

### 3.6.1 在 main.c 中包含了如下的全局变量：

```
//全局
struct Save{
    char name[50]; //从属的用户
    int Level; //等级
    int Score; //分数
    int GameBoard[20][10]; //游戏版数据
    char GameBoardColor[20][10][15]; //游戏版数据
    int nNext_block_index; //下一个方块类型
    int nCurrent_block_index;
    int time[6];
};

extern struct Save ssave;

struct User{
    char name[50];
    char password[50];
    int money;
    int save_exist;
};

extern struct User suser;
extern int loginstatus;
extern int registerstatus;
/*****
 * 游戏方块存储数组
 *****/
extern double winwidth, winheight; // 窗口尺寸
extern int selected_lable;
extern int MainMenuStatus;
extern int nScore; //分数
extern int nSpeed; //速度等级
extern int nSelectSpeed; //选择速度等级
extern int nSpeedUpScore; //升级速度
extern int bottom; //是否到达底部0/1
extern int bottom2; //一帧落地时是否到达底部0/1
extern int nNext_block_index; //下一个方块类型
extern int nCurrent_block_index; //现在的方块类型编码
extern double nOriginX; //现在的x/y值
extern int GameBoard[20][10]; //游戏版
extern char GameBoardColor[20][10][15];
extern int KeyDown;
extern int PauseNum; //用于暂停函数中的计数
extern int judge; //用于修复加速下落无法清除的bug, 来进行判断的变量
extern char SelectColor[100];
extern char HotKeyColor[100];
extern char HotKeyFont[100];
extern int rankpage;
extern int pixelnum;
extern int titlenum;
extern int animelenth;
extern int addmoney;
extern int user_guest;
struct score {
    int rank;
    char name[50];
    int score;
};
//排行榜数组
```

### 3.6.2 在 tetris.c 中使用外部变量如下：

```
double winwidth, winheight; // 窗口尺寸
int selected_lable;
int MainMenuStatus = -1;
int nScore=0; //分数
int nSpeed=1; //速度等级
int nSelectSpeed=1; //选择速度等级
int nSpeedUpScore; //升级速度分数
int bottom; //是否到达底部0/1
int bottom2;
int nNext_block_index=-1; //下一个方块类型
int nCurrent_block_index=0; //现在的方块类型编码
double nOriginX=4; //现在的x/y值
double nOriginY=5;
int GameBoard[20][10]; //游戏版
char GameBoardColor[20][10][15];
int KeyDown = 0;
int PauseNum;
int judge=0;
char SelectColor[100] = "ORANGE";
char HotKeyColor[100] = "RED";
char HotKeyFont[100] = "YELLOW";
int rankpage = 1;
int pixelnum = 0;
int titlenum = 0;
int animelenth = 0;
int addmoney;
int user_guest = 0;
struct score {
    int rank;
    char name[50];
    int score;
};
//排行榜数组

struct score srank[20] = { 0 }; //排行榜全局变量
```

## 4 部署与运行

### 4.1 编译安装运行说明

编译运行环境：Win 系统，Dev C。

源代码包中包含六个由我们自己编写的头文件以及图形库函数以及一首背景音乐歌曲。

自己编写的：Tetris.c tetris.h rank.c pixel.c main.c user.c

老师提供的：Libgraphics 库 simpleGUI 库

总工程文件为 Tetris.dev，下载解压后，请将 src 文件夹改名为 Tetris 即可进行编译运行。

运行 exe 后的用户使用手册：

本款俄罗斯方块游戏由俄罗斯三轮车团队倾情打造，基本功能与用途如下：

- 1、用户登录系统：进入游戏后可以选择进行登录，获取上次游戏时的金币数以及得分情况，同时可以选择读取以往保存的游戏记录继续游戏。
- 2、正式游戏部分：按键规则及游戏规则在选项界面中有所介绍，可以选择变换界面颜色。

3、分数计算规则：同上，在选项界面中有所介绍

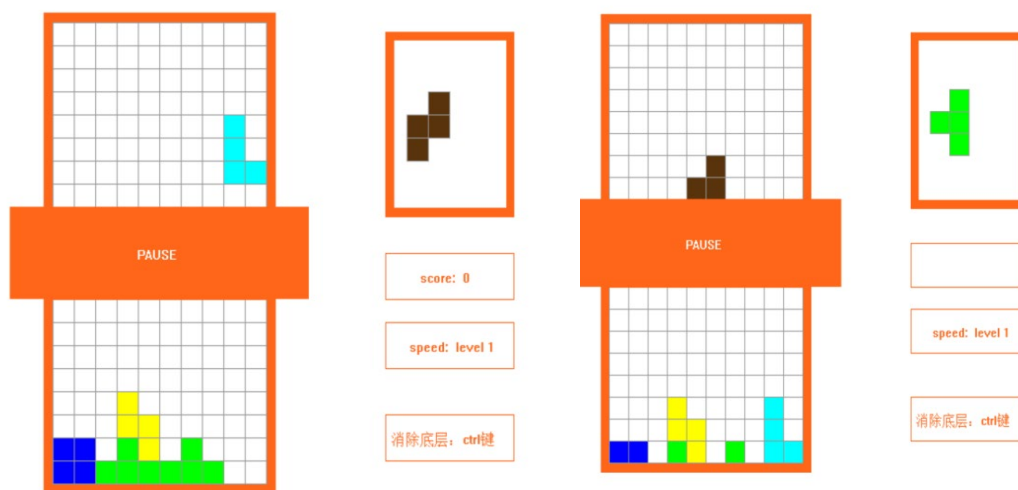
4、用户保存与读取进度部分：在游戏进行至一半时可以选择保存游戏进度，在下次登录该用户时可以读取上次的游戏进度继续游戏。

5、排行榜规则：每次游戏结束后，若分数在排行榜的前 20 位即会上榜。

## 4.2 典型测试情况

### 4.2.1 经典案例 1：1.0.2 版本中修复的加速下落时无法消除的 bug

案例背景：在调试的过程中，我们发现若一直按住下落键使方块一直快速下落会导致其到达底部时无法消除。具体截图如下：



(消除前)

(消除后)

一开始我们认为可能是 IsLineFull 函数以及 KillLines 这一类消除函数以及判断行是否满的函数出现错误，于是这两个函数进行了调试，调试代码如下（人工同步数据库与图形界面）：

```
DrawBlock(17,3,0.25);
DrawBlock(17,3,0.5);
DrawBlock(17,4,0.5);
DrawBlock(17,4,0.25);
DrawBlock(18,5,0.5);
DrawBlock(18,5,1);
for(i=0;i<=9;i++){
    GameBoard[18][i]=1;
    GameBoard[19][i]=1;
    for(i=5;i<=9;i++){
        GameBoard[17][i]=1;
    }
}
i=KillLines(20);
```

经过调试，我们发现这两个函数的判定与消除不存在问题，而后我们才发现若让其自然下落，到达底部时方块是可以正常消除的。这也印证了我们的判断和消除行函数不存在问题。

经过对代码的分析，我们发现一直按住下落键时无法消除的原因是，一直按住下落键使得其无法跳出 KeyboardEventProcess 函数的运行，也就无法对其是否到达底部进行判断。即在此处需要一个判断帮助其退出一直下落加速的函数，进行判定。

```
break;
case VK_DOWN: /* 下落加速键 */
    bottom=HandleDown(nCurrent_block_index,&nOriginX,&nOriginY);
    break;
case VK_ESCAPE: /* 退出游戏 */
```

(修改前)

```
case VK_DOWN: /* 下落加速键 */
    if(bottom==0)
    {
        bottom=HandleDown(nCurrent_block_index,&nOriginX,&nOriginY);
        break;
    }
}

int HandleDown (int BlockIndex, double *x, double *y)
```

(修改后)

#### 4.2.2 经典案例 2: 1.0.3 版本中修改了暂停时可以移动方块的 bug

案例背景：测试 bug 的同学发现，在暂停游戏后，按上下左右键仍然能够使得方块移动并达到底部，然后继续游戏以后游戏会继续正常判定，这个 bug 成为一个游戏作弊的方法。

调试过程：我们找到了有关暂停部分的函数，经过对于代码的分析后认为，这是我们在前面一个键盘响应函数仍然生效时继续注册了另一个键盘响应函数所导致的，所以我们在此取消了前一个键盘响应函数后，debug 完成。

```
void KeyboardEventProcess4(int key, int event)
{
    uiGetKeyboard(key,event);
    switch (event) {
        case KEY_DOWN:
            switch(key)
            {
                case VK_SPACE:/*暂停*/
                    wait();
                    cancelKeyboardEvent();
                    registerKeyboardEvent(KeyBoardEventProcess);
                    break;
            }
        case KEY_UP:
            break;
    }
}
```

#### 4.2.3 经典案例 3: 1.0.7 版本中一键下落的实现导致的同时按下 LEFT (RIGHT) 和 SPACE 键产生的 bug

案例背景：测试发现，当使用了一键下落函数（原 SpaceDown 函数，如今依旧保留在文件中）时，同时按下左右键和一键下落键时会因为同时处理了两个 KeyBoardEventProcess 会导致左右键的处理函数额外在游戏面板上画出左（右）移了的方块图像。

调试过程：一键下落函数是预先进行模拟下落的运算，然后再直接画出下落后的方块，但是，同时按下多个按键时，会到处运行多个按键处理函数。因此一开始 debug 的重点在使得按键处理函数只同时处理一个按键。为此重新调整了键盘处理函数的逻辑结构，以及增添了全局变量用于判断是否有正在处理的键盘函数。然而，各种类似的尝试均不能完全消除这个 bug。于是在多次调试的过程中发现左右键和上下键同时按下并不会产生这样的冲突，于是尝试取消掉一键下落函数，使用 while(1)的方式运行按下 DOWN 键后的函数，并在触底后 break。该尝试获得了成功，并且使得一键下落的视觉效果更佳。至此，debug 完成。



```

int SpaceDown(int BlockIndex, double *x, double *y){
    char ScoreBuffer[100] = {0}, SpeedBuffer[100] = {0};
    char n[100] = {"      score:  "};
    char m[100] = {"      speed:  level  "};
    double j;
    int i;
    int Num = 0;
    for(j=5; j>=0; j=j-BSIZE){
        if(IsConflict(BlockIndex, *x, j)){
            j=j+BSIZE;
            EraseBlock(BlockIndex, *x, *y);
        }
        for(i=0; i<=7; i+=2) GameBoard[(int)((19-((j-1)/BSIZE+shapes[BlockIndex].box[i+1]))[(int)((x-3)/BSIZE+shapes[BlockIndex].box[i]))]=0;
        DrawBlock(BlockIndex, *x, j);
        for(i=0; i<=7; i+=2) GameBoard[(int)((19-((j-1)/BSIZE+shapes[BlockIndex].box[i+1]))[(int)((x-3)/BSIZE+shapes[BlockIndex].box[i]))]=1;

        for(i=20; i>=4; i=i-4) Num=Num+KillLines(i);
        if(Num>0){
            switch(Num){
                case 0:
                    break;
                case 1:
                    nScore+=(100+(nSpeed-1)*50);
                    break;
                case 2:
                    nScore+=(300+(nSpeed-1)*50);
                    break;
                case 3:
                    nScore+=(500+(nSpeed-1)*50);
                    break;
                case 4:
                    nScore+=(800+(nSpeed-1)*50);
                    break;
            }
            itoa(nScore, ScoreBuffer, 10);
            drawRec(6.9, 3, statuswidth, statusheight-1.5, 0, "WHITE", "WHITE");
            SetPenColor("Orange");
            textbox(GenUIID(0), 6.9, 3, statuswidth, statusheight-1.5, strcat(n, ScoreBuffer), 0);
        }
    }
    return 1;
}

switch(event){
    if(KeyDown == 0){
        case KEY_DOWN:
            KeyDown = 1;
            if(key == VK_SPACE){
                while(1){
                    if(bottom==1)
                        break;
                    bottom=HandleDown(nCurrent_block_index, &nOriginX, &nOriginY);
                    //用于修复下落到底时一直按下落键会有未消除的情况(由于按键一直处于DOWN状态, 导致无法跳出KeyboardEvent进行判定)
                    //tm让VK_DOWN的情况不停循环直到触底不就行了, 整这么多余的设的干什么!!! 记得把SpaceDown删了, 还有drawGameBoard也删了。
                    /*bottom2=SpaceDown(nCurrent_block_index, &nOriginX, &nOriginY);
                    for(i=19; i>=0; i--){
                        for(j=0; j<10; j++){
                            if(GameBoard[i][(int)j]){
                                SetPenColor(GameBoardColor[i][(int)j]);
                                drawRectangle(3+BSIZE*j, 1+BSIZE*(19-i), BSIZE, BSIZE, 1);
                            }
                            else {
                                SetPenColor("WHITE");
                                drawRectangle(3+BSIZE*j, 1+BSIZE*(19-i), BSIZE, BSIZE, 1);
                            }
                        }
                    }
                    for(i=19; i>=0; i--){
                        for(j=0; j<10; j++){
                            if(GameBoard[i][(int)j]){
                                SetPenColor("GRAY");
                                MovePen(3+BSIZE*j, 1+BSIZE*(19-i));
                                DrawLine(0, BSIZE);
                                DrawLine(BSIZE, 0);
                                DrawLine(0, -BSIZE);
                                DrawLine(-BSIZE, 0);
                            }
                        }
                    }
                    */
                }
            }
            else if(key == VK_LEFT){
                HandleLeft(nCurrent_block_index, &nOriginX, &nOriginY);
            }
        }
    }
}

```

#### 4.2.4 经典案例 4: 1.0.7 中的从存档开始游戏时绘制了多余的方块的问题

**Bug 解决:** 经过多次反复的将从存档开始游戏中所调用的函数进行注释运行测试, 发现问题在于使用 drawGameBoard 函数时是将 GameBoardColor 中的颜色进行了绘制, 然而游戏保存时下落的方块在 GameBoardColor 中留下了颜色信息, 但是不会在 GameBoard 中留下位置信息, 所以需要用 GameBoard 数组进行辅助判断, 只绘制已经下落完成的方块。

## 5 组内分工

### 5.1 组内分工情况

陆子仪: 用户, 文件, 排行榜系统, 存档系统, 动画制作, 界面美化, 函数优化, 报告书写

沈吕可晟: 正式游戏函数功能, bug 修复, 函数优化, 报告书写, 背景音乐、充值选项部分。

周宇轩: 正式游戏部分图形绘制, bug 测试, 报告书写

## 5.2 个人实践过程中遇到的难点及解决方案

### 5.2.1 陆子仪：

#### 5.2.1.1

难点：对于程序中所用到的各种函数不熟悉。

解决方案：借助网络的帮助以及认真翻阅库中的文件，寻找函数原型及定义。

#### 5.2.1.2

难点：对于 bug 的修复流程的不熟悉，导致对文件进行大量不必要的修改。

解决方案：上网获取相应资料，并进行大量实验，逐渐摸索出对 bug 的排查和修复流程。

### 5.2.2 沈吕可晟：

#### 5.2.2.1

难点：对于工程的建立，以及编译操作的不熟悉。

解决方案：请教他人以获取帮助

#### 5.2.2.2

难点：在修复 bug 过程中，对于 DEV-C++编译器下调试功能的不熟悉

解决方案：上网获取资料，查看 DEV-C++用户使用手册，从简单的调试开始慢慢摸索

#### 5.2.2.3

难点：对于老师所给的部分库函数，（例如键盘、鼠标、计时器响应函数等）由于函数说明不足。无法理解其功能及使用方法。

解决方案：仔细研读库函数的原代码，试图理解其意思，并且询问已经使用过该函数的同学，函数具体使用方法，并在简单的程序中进行试验，熟悉其用法

### 5.2.3 周宇轩：

#### 5.2.3.1

难点：对于工程的建立，以及编译操作的不熟悉。

解决方案：请教他人以获取帮助

#### 5.2.3.2

难点：在寻找 bug 过程中，对于如何分析 bug 的出现原因以及寻找 bug 代码的位置有困难。

解决方案：询问对与函数理解更深的编写函数部分的同学（同时也是修改 bug 的同学），与他讨论函数功能以及整体架构，分析问题根源，寻找错误代码。

## 6 合作纪要与团队总结

### 6.1 合作纪要：

2019 年 3 月 28 日：进行游戏分工

2019 年 5 月 20 日：游戏主题部分完工，开始 bug 修复，开始排行榜系统编写

（V1.0.1 正式版本完工）

2019 年 5 月 23 日：bug 修复基本完成，排行榜系统编写完成，开始函数优化及界面美化。（版本更新至 V1.0.4）

2019 年 5 月 28 日：完成部分界面美化工作，确认后续工作计划，开始用户系统的编写（版本更新至 V1.0.5）

2019 年 6 月 2 日：后续计划工作开展，完成排行榜、用户系统。完善游戏各部分功能，修改 bug

2019 年 6 月 14 日：后续工作完成，版本更新至 1.0.9；

### 6.2 版本更新内容记录如下：

/\*1.0.0 以及 1.0.1 版本为工程初期程序

/\*1.0.2 版本更新内容如下：

- 1、新增选择速度等级功能
- 2、修复加速下落无法消除的 bug



## 3、修复暂停后会加速下落的 bug

\*/

## /\*1.0.3 版本更新内容如下:

- 1、新增换肤功能
- 2、修改了 pause 时可以移动方块的 bug
- 3、增加了游戏说明（在选项中）
- 4、修复一直按住旋转键到底部误消除的 bug

\*/

## /\*1.0.4 版本更新内容如下:

- 1、增加了排行榜模块，排名数据保存在 exe 目录下的 rank.exe 中
- 2、增加了颜色选择，优化了主体颜色选择
- 3、增加了游戏说明（在游戏中）

## /\*1.0.5 版本更新内容如下:

- 1、增加了开始界面的动画效果
- 2、修复了选项更改颜色及时显示的问题
- 3、修复了加速下落无法消掉的 bug

## /\*1.0.6 版本更新内容如下:

- 1、增加登陆界面
- 2、增加用户系统以及相关联的金钱系统
- 3、修改 ESC 界面按键的位置
- 4、解决注册空字符串用户的问题
- 5、保存界面

## /\*1.0.7 版本更新内容如下:

- 1、添加音乐
- 2、修复了用户保存函数以及用户更新函数以及存档保存函数的若干 bug
- 3、将 GameBoardColor 的储存类型由地址类型更改到数组类型，考虑到跨文件以及再次打开程序的地址随机分配问题
- 4、增加了快捷键功能
- 5、排行榜备注游客和用户的区别
- 6、增加了登录界面的返回按键，美化了登录界面

## /\*1.0.8 版本更新内容如下:

- 1、解决再次打开游戏一键下落有很大问题很大问题 !!!（每次退出游戏把各种 Event 取消掉，否则会调出结束画面）
- 2、解决一键下落和 xx 键一起按下的问题 !!!（用 VK\_DOWN 的循环代替 SpaceDown 函数）
- 3、解决读取存档后重开游戏的问题
- 4、丰富了游戏界面以及方块的颜色
- 5、增加了存档的保存时间记录以及显示功能
- 6、修改了 TimerEventProcess 的逻辑结构，用 if-else if 结构代替 switch 结构

## /\*1.0.9 版本更新内容如下:

- 1、美化了保存界面（活动方块不需要消失），读取存档时间也不会不能刷新掉
- 2、解决了从存档开始游戏不显示下一个方块的问题
- 3、用户自动保存选择的颜色方案



4、保存成功将显示保存时间以及保存成功提示

5、用户选择颜色需要花费 200 金



币，并且会实时显示用户信息在选择界面

6、完善了快捷键功能，1) 将快捷键标注在按键上 2) 解决了按下快捷键后需要移动鼠标才能刷新的问题 3) 增加了 MainButton 的二级目录的 ESC 返回功能

### 6.3 qq群交流记录:

Fantast \(\1401241918\) 19:00:27

不要了

Fantast \(\1401241918\) 19:02:56

分数 速度等级 晋级速度 是否到达底部 (0or1) 下一个方块类型编码 现在的方块类型编码 现在下落方块所处x值 y值

timecounter不要

Pausenum是一个计数用的 奇数或偶数

18-广东-科创-陆子仪(1225087727) 19:15:37

pausenum?

18-广东-科创-陆子仪(1225087727) 19:15:43

有没有具体的用途

18-广东-科创-陆子仪(1225087727) 19:20:10

18-广东-科创-陆子仪(1225087727) 18:08:21

在吗

18-广东-科创-陆子仪(1225087727) 18:08:49

沈，你把游戏保存还有读取的整一下呗

18-广东-科创-陆子仪(1225087727) 18:09:23

等等，不用保存的，只用怎么读取游戏后直接开始的

18-广东-科创-陆子仪(1225087727) 18:15:10

我们今晚可以开个会吗

18-广东-科创-陆子仪(1225087727) 18:15:21

现在感觉有点无从下手

Fantast \(\1401241918\) 18:16:16

我现在在寝室

Fantast \(\1401241918\) 18:16:17

的

Fantast \(\1401241918\) 18:16:32

几点?

18-广东-科创-陆子仪(1225087727) 18:16:55

那我来找你

18-眉山-周宇轩(783328347) 19:03:51

新建文本文档.txt  
1153 B



Fantast \(\1401241918\) 11:07:41

emmm我懂了

Fantast \(\1401241918\) 11:07:44

这个报告

Fantast \(\1401241918\) 11:07:49

15号前交

Fantast \(\1401241918\) 11:07:58

你用户写好了么

Fantast \(\1401241918\) 11:08:18

写好了我再写那个金币系统

18-广东-科创-陆子仪(1225087727) 11:08:25

行吧

18-广东-科创-陆子仪(1225087727) 11:08:32

我找时间这两天写一下

Fantast \(\1401241918\) 11:08:37

emm

Fantast \(\1401241918\) 11:09:16

联网的支付借口写不出来...

Fantast \(\1401241918\) 11:09:19

接口

Fantast \(\1401241918\) 11:09:22

Fantast \(\1401241918\) 10:50:50

```
#ifndef tetris_functions
#define tetris_functions
```

Fantast \(\1401241918\) 10:50:56

写保护?

18-广东-科创-陆子仪(1225087727) 10:50:58

中午能开一下会?

18-广东-科创-陆子仪(1225087727) 10:51:00

嗯

18-广东-科创-陆子仪(1225087727) 10:51:01

保护

Fantast \(\1401241918\) 10:51:02

```
} #ifndef TETRIS_H
} #define TETRIS_H
#include "tetris.h"
```

Fantast \(\1401241918\) 10:51:05

最上面了

Fantast \(\1401241918\) 10:51:08

我把你那个删了?

### 6.4 未来展望:

- 1、优化游戏音乐
- 2、考虑删除一些无用的函数 spacedown
- 3、考虑游戏左右边界的方块旋转问题
- 4、考虑增加双人模式?
- 5、考虑增加局域网联机双人模式?
- 6、增加动画效果?

### 6.5 团队总结

#### 6.5.1 优点总结:

- 1、从最初简单的程序成型到最终版本成型，对于每一次我们更新的版本都进行了详细记录，能够清晰

的看到版本的更新内容以及该大程序一步步的逐渐完善过程。

2、最终版本成型后各个函数之间关系清晰，功能明确。所有函数都按照内容分类，并且将同类函数以及功能衔接较紧的函数放在一起，以便更改查找。

3、我们所设计的俄罗斯方块不仅有传统功能，更是在其基础上加入金币、道具、背景音乐、通过二维数组绘画的小人动画等元素，使其整体更为丰满，能够更好的契合用户体验。

4、团队合作愉快，分工明确，效率较高。我们在制作整个俄罗斯方块程序时思路清晰，首先由沈吕可晟同学制作出游戏主体功能部分，周宇轩同学完善其中图形界面，然后再由陆子仪同学完成文件、排行榜等方面内容，在中期时就已经制作出了一个可以进行调试的程序。后续，我们一步步的通过版本更新记录的方式，修改完善源程序，并且加入一些属于我们自己的特色元素，在3中已经有所叙述，所有过程都非常详细的记录在版本的更替记录中。这也让我们大家意识到在团队合作开发程序时，统一书写格式、注释以及更新记录的重要性，为以后的程序编写打下了基础。最终实验报告由大家共同讨论书写完成。

#### 6.5.2 不足之处以及进步之处总结：

1、最开始写大型多程序文件时，我组在整合三人所写内容时，花费了大量时间来统一整体文件内容。然而在后续过程中，我们注意到了这一点，进行了统一，此点为以后的大程序制作留下了宝贵的经验。

2、对于整体文件结构的部分加深了认知，对库函数的理解以各个源文件和工程的联系加深了认知。

3、经过本次大程序实验的编写，小组每一个成员的编写函数能力有了进一步的提升，对于函数之间的引用也有了较深的经验，能够较好的践行自上而下的大型程序设计流程，更加认识到将一个小程序分成很多部分功能函数的重要性，能够很好的梳理解决思路。

## 7 参考文献和资料

C 语言的科学与技术