时钟与闹钟

实验报告

1550431

王甯琪

计科 3班

完成时间：5.2

1. 题目及基本要求

1.1.时钟部分

需要生成数字式和指针式两种时钟，可以通过快捷键进行切换。时钟显示时间需要跟随系统时间变化而变化。

1.2.闹钟部分

同时还需要实现闹钟功能，要能在外部的文件中读取已保存的闹钟，并要能够添加新闹钟。闹钟方式有多种（每日/工作日/单次/倒计时等），到达闹钟规定时间后需要播放音乐以提示。关闭程序再打开后，之前添加的闹钟仍需要生效。

1.3.额外加分部分

和时钟、闹钟相关的功能可以自行添加。添加多时区时钟可以加一分，最后也会代码质量、显示效果、人机交互的友好程度等来给予加分。

2. 整体设计思路

2.1.菜单

和之前大作业类似，最开始展示给用户所有可选菜单项，再接收用户输入，最后运行不同的代码。

因为要实现通过按下快捷键跳转至其他菜单项，所以需要采取两重循环的结构，按下快捷键跳出当前第二重循环并进入其他循环或是退出循环。

2.2.时钟

大致分为获取系统时间，画出时钟图像部分。

获取系统时间可以引用使用系统函数获取。再根据获取到的时间来绘制图像（字体和窗口大小要进行调整，字体调到最小，从而可以通过输出某字符来模拟输出一个点，点连成线就形成了各种图形和文字）。每次输出后要进行清空再进行下一次输出。

2.3.闹钟

大致为读取文件内容到内存，分析闹钟内容，删除过期闹钟，显示当前时间和所有闹钟，循环检测闹钟是否到时间，将新闹钟添加进内存中，结束时将内存中的闹钟信息保存入文件等步骤。

其中闹钟到点的提示播放音乐，可以引入某些头文件，播放wav格式的音乐。

3. 主要功能的实现

3.1. 时钟与闹钟的选择和跳转

使用两重循环，数字式、指针式以及闹钟分别是三个内循环。使用 int 型变量 way 来决定进入的内循环。当要退出整个循环时，way置为0即可。

关于快捷键，可以使用 kbhit() 函数，因为是非阻塞函数，所以不会导致程序运行中断。通过 _getch() 函数得到按下的键返回的键值，从而能够判断用户输入的指令并对way变量进行赋值。

3.2. 指针式时钟

获取系统时间可以使用系统函数，并定义结构体 now_time 保存当前年、月、日、星期、时、分、秒。

对于绘制图形，可以定义 point 结构体，其中包含x坐标，y坐标以及指向下一个 point 结构体的指针。任何形状的图形都可通过生成 point 类型的链表，储存该图形所经历的所有点的坐标，再在字体设置得非常小的情况下输出 ‘.’ 或者 ‘*’，用以画出一个个点，最后点便能连接成完整图形。

如画表盘，只需要定位圆心坐标X,Y和半径R，即可以通过x=X+Rcosa，y=Y+Rsina，0<=a<=2π 来定位每一个点，最后将其画出来即可。

而如果要消除某一轨迹，则只需要在同样位置输出 ‘ ’(空格)即可。这样，便可以通过计算得到某轨迹的坐标链表，从而画出或删除这条轨迹。

由于指针式时钟只需要改变指针位置，而不需要改变表盘和刻度位置，所以每次更新时只需在原指针位置输出空格，并计算新的指针位置，重新画出指针即可，而不用清屏后重新画出表盘和刻度。这样能够节省时间。

需要注意的是，三根指针的位置随时都在变化，分针的位置会随着秒针的移动而变化，时钟的位置也会随着分针的移动而变化，这也是需要考虑在内的。

3.3.数字式时钟

大致思路同指针式时钟类似，不过由于数字式时钟只有0-9，以及 ‘:’ 这个时、分、秒之间的分隔符，所以可以先计算这些字符对应的 point 型链表（采用相对位置）。在得到当前时间后，可以通过组合这些字符来完成储存着最终输出形式的链表，并进行绘制。需要注意的是，由于每个字符保存的是相对位置，所以需要在拼接各个字符时考虑移动问题。比如说最终的链表中已经储存了3的输出信息，若还要继续输出0，需要将现有链表中所有元素的x坐标增加22，以给新添加的字符腾出位置。

3.4.闹钟

在文件save中以 闹钟类型、年、月、日、星期、时、分、秒 的顺序储存一条闹钟信息。并定义 alarm 结构体，储存了闹钟的相应信息。在进入闹钟这个内循环的时候将文件中的内容读入 alarm 结构体中，形成链表。

注意要判断闹钟是否过期，如果闹钟设定时间在当前时间之前，则应该从链表中删除。增加新的闹钟也就是在链表中添加新的项，如果是倒计时的话，则在当前时间基础上加上需要倒计时的时间并加入链表中。

循环检测当前时间是否达到或者超过链表中的每一项对应的时间，如果有符合要求的项，则必须对该闹钟进行操作，并播放音乐提示。

需要注意，不是所有的项都要在到达规定时间后删除。倒计时和单次闹钟需要删除，然而工作日闹钟和每日闹钟需要在判断当前时间是否达到响铃要求后保留，不能删除。

最后退出循环的时候需要将内存中的内容全部覆盖写入 save 文件，以供之后重启闹钟时读取。

4. 调试过程碰到的问题

4.1. 很多变量未初始化就使用

由于使用的指针和引用非常多，所以在程序调试、运行过程中常常出现错误。比如说未初始化就使用、函数传参遇到问题等等。

解决方法是善加利用 Visual Studio 自带的调试功能， Visual Studio 有遇到错误自动中断程序并提示错误原因的功能。使用这个功能发现了很多未初始化的变量以及空指针等等，极大地帮助发现BUG。

4.2. 空间释放顺序错误

由于动态申请空间非常多，又需要及时地释放申请的空间，所以常常出现还需要利用的空间却被删除的错误。

通过画出流程图以及内存分配图，能很好地帮助找出出错的位置，理清整体的思路。同时使用 Visual Studio 自带的调试工具，也能遇到错误自动中断程序，提示空间已被删除。

5. 心得体会

5.1. 在做之前就有大体思路和框架

如果在开始写代码之前已经想好了比较清晰的框架，那么就可以清楚地找出公用函数，那么在写这些公用函数的时候，可以根据调用它的函数的特点来决定这些公用函数的形式。在设计结构体的时候，也要考虑到所有使用这个结构体的函数的特点，从而决定结构体的元素。

比如说在写一个输出图形的函数的时候，考虑到可能不同函数在调用绘图函数的时候需要绘制在不同的坐标出，所以在写这个公用函数的时候，就需要设置两个变量，代表x、y坐标。

一言以蔽之，在开始写代码之前，最好能先想好整个实现过程，想想该怎么编排和分解公用函数、怎样设计结构体会比较好。

5.2. 出错的时候停下来

在发现有bug，程序弹框的时候，我这一次没有像以前一样凭自我感觉来找错误，更多地是通过纸笔来模拟程序运行过程、内存分配过程，分析语句运行顺序，同时使用一些之前没发现过的 Visual Studio 自带的调试功能来进行错误审查。

这样的效率高了很多，有的错误其实自己根本就没有想到。如果是按照之前的办法凭自我感觉找错误，很可能就需要浪费好几倍的时间。

6. 附件：源程序

//代码太多，选取体现思路主体的部分

```
struct now_time
{
	int year;
	int month;
	int day;
	int week;
	int hour;
	int minute;
	int second;
};

struct point
{
	int X;
	int Y;
	point *next = NULL;
};

struct alarm
{
	int type = 0;//1 倒计时 2 单次 3 每日 4 工作日
	now_time *alarm_time=NULL;
	alarm *next=NULL;
	alarm*previous = NULL;
};

void get_now_time(now_time *saver);//获取当前时间
int translate_string_to_num(char *c,int n);//将 char 数组储存的数字转为 int 形数字
void print_now_time(now_time *saver, const int X = 0, const int Y = 0);//默认在窗口左上角输出当前时间
int generate_circle(point *&first, const int X, const int Y,const int R);//以 X,Y 为圆心，生成一个圆的轨迹点 内存申请错误返回-1
void free_points(point *&first);//释放储存轨迹点的空间
void draw_points(point*first, char x);//在所有点处输出字符 x
int  generate_kedu(point *&first, const int X, const int Y, const int R,const int L);//以 X,Y 为圆心,R 为半径生成的表盘的刻度，刻度长度为 L 内存申请错误返回-1
int  generate_zhen(now_time*saver,point *&h_first,point *&m_first,point *&s_first, const int X,
```

```
const int Y);//在以X,Y为圆心,R为半径生成的表盘上,生成时分秒的指针 内存申请错误返回-1
void until_time_change(now_time tm);
int generate_10_nums(point*num[11], const int c, const int k);//生成十个数字的相对坐标 c,k代表长宽 内存申请错误返回-1
int combime_nums(now_time *tm, point *num[11],point *&all_num, const int k,const int X, const int Y);//在 X,Y 处生成记录输出点的链表
void points_link_add_X_Y(point *&first, const int X, const int Y);//整体移动(X,Y)
int copy_points_link(point *&source,point*&destination);//复制
void combine_two_point_links(point *&a,point *&b);//连接两个链表
void clear_point_link(point *&a);//清空链表
int read_alarm_into_memory(fstream &saver,alarm *&head);//将文件中的信息读入内存
void free_alarm(alarm *&head);
void show_all_alarms(alarm *head,const int X,const int Y);//在X,Y处输出所有闹钟
int check_all_alarms(alarm *&head);//检查所有闹钟是否到时间
int two_time_compare(now_time *t1,now_time*t2);//比较两个时间先后
void play_music();
int new_alarm(alarm *&head);
void adjust_time(now_time*&t);//调整时间
void output_to_file(fstream &f,alarm*head);

void which_way_to_go(int branch)
{
     int flag, loop, temp, which = branch,i;
     const int central_X = 180, central_Y = 90, R = 80,num_X=-4,num_Y=15,num_c=20,num_k=16;
     now_time tm;
     point *circle_first, *kedu_first, *shizhen_first, *fenzhen_first, *miaozhen_first, *num[11],*all_num;
     fstream alarm_saver;
     alarm *alarm_head;

     if (generate_10_nums(num, num_c, num_k) == -1)
          return;

     alarm_saver.open(".\\1550431\\alarm.save",ofstream::out|ios::app);
     if(alarm_saver.is_open()==0)
     {
          system("cls");
          cout << "文件打开失败" << endl;
          Sleep(5000);
          return;
     }
     alarm_saver.close();

     while (which)
     {
          loop = 1;
          if(which==1)//指针式
          {
               system("cls");
               setfontsize(hout, L"点阵字体", 1);
               setconsoleborder(hout, 360, 180);
               //setfontsize(hout, L"点阵字体", 1);
               flag = generate_circle(circle_first, central_X, central_Y, R);
               if (flag == -1)//出错
                    return;

               flag = generate_kedu(kedu_first, central_X, central_Y, R, 8);
               if (flag == -1)//出错
               {
                    free_points(circle_first);
                    return;
               }

               get_now_time(&tm);//tm中储存了些许时间元素
               flag = generate_zhen(&tm, shizhen_first, fenzhen_first, miaozhen_first, central_X, central_Y);
               if (flag == -1)//出错
               {
                    free_points(circle_first);
                    free_points(kedu_first);
                    return;
               }

               draw_points(circle_first, '.');
               draw_points(kedu_first, '.');
               draw_points(shizhen_first, '.');
               draw_points(fenzhen_first, '.');
               draw_points(miaozhen_first, '.');

               while (loop)
               {
                    if (_kbhit())
                    {
                         temp = _getch();
                         if (temp == 0)
                         {
                              temp = _getch();
                              if (temp == 67)//F9
                              {
                                   //loop = 0;
                                   which = 0;

     free_points(circle_first);

     free_points(kedu_first);

     free_points(shizhen_first);

     free_points(fenzhen_first);

     free_points(miaozhen_first);
                                   break;
                              }
                              else if (temp == 68)//F10
                              {
                                   //loop = 0;
                                   which = 1;

     free_points(circle_first);
```

```
        free_points(kedu_first);

        free_points(shizhen_first);

        free_points(fenzhen_first);

        free_points(miaozhen_first);
                                    break;
                                    //break;
                               }
                          }
                          else if (temp == 224)
                          {
                               temp = _getch();
                               if (temp == 133)//F11
                               {
                                    //loop = 0;
                                    which = 2;

        free_points(circle_first);

        free_points(kedu_first);

        free_points(shizhen_first);

        free_points(fenzhen_first);

        free_points(miaozhen_first);
                                    break;
                                    //break;
                               }
                               else if (temp ==
134)//F12
                               {
                                    //loop = 0;
                                    which = 3;

        free_points(circle_first);

        free_points(kedu_first);

        free_points(shizhen_first);

        free_points(fenzhen_first);

        free_points(miaozhen_first);
                                    break;
                                    //break;
                               }
                          }
                     }

                     draw_points(shizhen_first, ' ');
                     draw_points(fenzhen_first, ' ');
                     draw_points(miaozhen_first, '
');
                     free_points(shizhen_first);
                     free_points(fenzhen_first);
                     free_points(miaozhen_first);

                     get_now_time(&tm);//tm 中储存了
些许时间元素
                     flag = generate_zhen(&tm,
shizhen_first, fenzhen_first, miaozhen_first,
central_X, central_Y);
                     if (flag == -1)//出错
                     {
                          free_points(circle_first);
                          free_points(kedu_first);
                          return;
                     }

                     draw_points(shizhen_first,
'.');
                     draw_points(fenzhen_first,
'.');
                     draw_points(miaozhen_first,
'.');

                     until_time_change(tm);
                }
           }
           else if(which==2)//数字式
           {
                system("cls");
                setfontsize(hout, L"点阵字体", 1);
                setconsoleborder(hout, 180, 60);
                //setfontsize(hout, L"点阵字体", 1);

                get_now_time(&tm);//tm 中储存了些许时
间元素
                flag = combime_nums(&tm, num, all_num,
num_k, num_X, num_Y);
                if(flag==-1)
                {
                     for (i = 0; i < 11; i++)
                          free_points(num[i]);
                }
                draw_points(all_num, '*');

                while (loop)
                {
                     if (_kbhit())
                     {
                          temp = _getch();
                          if (temp == 0)
                          {
                               temp = _getch();
                               if (temp == 67)//F9
                               {
                                    loop = 0;
                                    which = 0;

     //clear_point_link(all_num);

     free_points(all_num);
                                    break;
                               }
                               else if (temp ==
68)//F10
                               {
                                    loop = 0;
                                    which = 1;

     free_points(all_num);
                                    break;
```

```
                    }
                }
                else if (temp == 224)
                {
                    temp = _getch();
                    if (temp == 133)//F11
                    {
                        loop = 0;
                        which = 2;

    free_points(all_num);
                        break;
                    }
                    else if (temp ==
134)//F12
                    {
                        loop = 0;
                        which = 3;

    free_points(all_num);
                        break;
                    }
                }
            }

            //getchar();
            draw_points(all_num, ' ');
            free_points(all_num);
            get_now_time(&tm);//tm 中储存了
些许时间元素
            flag = combime_nums(&tm, num,
all_num, num_k, num_X, num_Y);
            if (flag == -1)
            {
                for (i = 0; i < 11; i++)
                    free_points(num[i]);
            }
            //getchar();
            draw_points(all_num, '*');

            until_time_change(tm);
        }
    }
    else//闹钟
    {
        //play_music();

    alarm_saver.open(".\\1550431\\alarm.save");
        if (alarm_saver.is_open() == 0)
        {
            system("cls");
            cout << "文件打开失败" << endl;
            Sleep(5000);
            return;
        }
        //play_music();
        system("cls");
        setconsoleborder(hout, 120, 40);
        setfontsize(hout, L"点阵字体", 15);

        if
(read_alarm_into_memory(alarm_saver, alarm_head) ==
-1)
            return;
        check_all_alarms(alarm_head);

        while (loop)
        {
            system("cls");

            if (_kbhit())
            {
                temp = _getch();
                if (temp == 0)
                {
                    temp = _getch();
                    if (temp == 67)//F9
                    {
                        which = 0;

    alarm_saver.close();

    alarm_saver.open(".\\1550431\\alarm.save",
ios::out | ios::binary);

    output_to_file(alarm_saver, alarm_head);

    alarm_saver.close();
                        break;
                    }
                    else if (temp ==
68)//F10
                    {
                        which = 1;

    alarm_saver.close();

    alarm_saver.open(".\\1550431\\alarm.save",
ios::out | ios::binary);

    output_to_file(alarm_saver, alarm_head);

    alarm_saver.close();
                        break;
                    }
                    else if(temp==66)//F8
                    {

    new_alarm(alarm_head);
                        //alarm *rt =
alarm_head;

    //show_all_alarms(alarm_head, 0, 4);
                        //break;
                    }
                }
                else if (temp == 224)
                {
                    temp = _getch();
                    if (temp == 133)//F11
                    {
                        which = 2;

    alarm_saver.close();

    alarm_saver.open(".\\1550431\\alarm.save",
ios::out | ios::binary);
```

```
    output_to_file(alarm_saver, alarm_head);

    alarm_saver.close();
                                break;
                            }
                            else if (temp ==
134)//F12
                            {
                                which = 3;

    alarm_saver.close();

    alarm_saver.open(".\\1550431\\alarm.save",
ios::out | ios::binary);

    output_to_file(alarm_saver, alarm_head);

    alarm_saver.close();
                                break;
                            }
                        }
                    }
```

```
4);
== 1)
些许时间元素
                }
            }
        }
    }
```

```
//Sleep(5000);

show_all_alarms(alarm_head, 0,

if (check_all_alarms(alarm_head)

{
    play_music();
    continue;
}
//Sleep(2000);
//free_alarm(alarm_head);

get_now_time(&tm);//tm 中储存了

print_now_time(&tm);
cout << "\n 按 F8 新建闹钟";
until_time_change(tm);
```