装 订 线

时钟-闹钟实现报告

班级：计算机一班
姓名：王哲源
学号：1652228

完成日期：2017.4.24

1.题目及其描述

题目主要目的在于通过CMD界面下点阵字体实现机械式及电子式的时钟并且同时支持闹钟功能。其实现主要可以分为以下几个小部分

1.1 机械表盘的打印

在 cmd 界面下绘制圆形表盘及指针并模拟机械表表盘的运行

1.2 电子表的实现

利用点阵字体在 cmd 界面下实现电子表显示

1.3 实现不同时区间的时钟切换

通过指定按键使用户能在不同时区之间的时钟随意切换

1.4 实现 1.1 与 1.2 之间的随时切换

通过指定按键使用户能在两者之间随意切换

1.5 实现闹钟功能

在显示时钟的同时支持用户设定闹钟并按时响铃

1.6 在 1.5 的基础上实现闹钟退出后保存

实现当用户退出闹钟应用后将用户当前设定闹钟的保存，并在闹钟再次启动后恢复之前用户的设定

2.整体设计思路

以下对于程序的一些设计思路进行概述

2.1 关于机械表盘的打印

由于作业下发时并未对画图函数进行具体的限定，因此采用的思路是通过将 cmd 界面的字体调至最小然后利用 cmd 界面字体长宽比 3：5 的特性进行圆的拟合绘制，指针思路同表盘，利用折线对直线进行拟合。

2.2 电子表界面的打印

相较于机械表的绘制，电子表的绘制则更加简单。由于电子表只需要显示数字式的界面，因

此利用点阵字体库对时分秒的数字进行顺序打印即可。

同时，与之类似的是除了表盘界面以外的提示性文字等均与数字的打印相类似，均为通过在cmd界面下绘制点阵字实现的

2.3 关于同时支持时钟的实时显示与键位读取

由于时钟的显示与键位的读取存在同步性，而暂时没有掌握双线程实现的方法，因此此处采用的是对于键位读取设定一个响应时间，在响应时间内若未读取到键位操作，则暂时退出键位读取的步骤，对时钟界面进行更新，待更新后再进行读取的等待，重复以上步骤以实现要求。

但本思路的弊端也很明显，由于响应时间的执行无法实现完全精确的执行，因此可能会出现时钟走秒不够均匀。

2.4 关于闹钟的存储

鉴于用户设置的闹钟可能存在多个，因此闹钟的存储采用动态链表的方式进行存储，在用户进行修改时对动态链表进行维护即可

2.5 关于闹钟响铃的实现

在对时钟界面更新前，对链表内闹钟逐个判断，若时间与当前时间相同，则停止界面更新进入响铃界面，利用系统播放音乐函数播放闹铃，同时等待用户进一步操作以退出闹钟

2.6 关于闹钟的本地存储

如1.6所述该程序需要支持用户的闹钟本地存储，因此采用在本地创建配置文件的方式，在程序执行前进行载入，后在用户修改时同步对配置文件进行修改，通过此方法实现闹钟的存储

3.主要功能实现

以下对整个程序的具体实现进行较为详细的介绍

3.1 程序运行初始化

程序运行开始需要进行两步，一是将cmd窗口字体设置为最小，同时调节cmd窗口大小以支持点阵字体的打印；二是读取配置文件将闹钟设置录入链表中。

3.1.1 闹钟链表的存储

此处闹钟链表采用一个空头节点的方式，通过动态分配空间，实现一个双向循环链表。同时，当某次操作执行到头节点时，通过特判将节点查询至头节点的下一个节点，若仍然为头节点，则判定链表为空

3.2 机械表的实现

机械表采用 1.1 中所述的方式进行绘制，而每次绘制后对本次指针坐标进行存储，在进行下一次绘制时只需要对前一次所绘制的坐标进行覆盖即可，无需对区域进行整体覆盖。同时可以采用宏定义的方式对表盘界面及提示界面进行划分打印即可。

打印时采用 2.1 中的方式，通过计算坐标与圆/直线方程取最近似值的方式，利用 cmd 字体设定为极小的特点进行拟合打印即可。

3.2.1 提示界面及日期信息

为了防止界面不断闪频，提示界面与日期信息与表盘界面将分开进行打印。通过判断设定是否更新来判断是否重新打印两者。同时特别注意，日期信息在日期变更时也应当进行重新打印，因此利用一个固定变量存储前一次的日期信息，若发现日期变更则对日期信息见面进行重新打印即可。

3.2.2 关于日期的获取与时区计算与打印

利用系统函数 time(0)可以获取到从 1900 年至今的秒数，而通过函数 local_time 即可将秒数转换入一个结构体中，其成员下可以获取年月日等诸多日期信息。将其进行逐个提取并打印即可。

同时应当注意的是 time(0)获取到的信息为当前系统所设置的时区下的时间信息，因此当手动进行切换时需要注意时差的加减，而日期的打印更要注意国际日期变更线的日期变化。

而时区信息与日期信息等诸多信息的打印只需要在头文件中进行指针数组的常量定义即可。

3.3 电子表界面的打印

电子表打印的方式采用 2.2 中所述的方法即可。此处提及以下本程序中所需要用到而设计的一些打印函数

3.3.1 字符打印

这是所有点阵字体打印的最基础方式，通过从点阵字库中获取点阵字体的信息通过坐标参数获取点阵字的输出坐标。通过二进制的 01 位判断该位置为点/空而实现点阵字的打印。后两种方法的打印均基于该步

3.3.2 数字的打印

这里可以考虑到，本程序中需要打印的数字为年月日及时钟，均为 4 位或 2 位的打印，因此这里可以采用两位两位打印的方式，每次打印两位数字，而年的打印则采用分两段即可。对于没两位数字，将数字分解为十位和个位。且由于日期和时间的特殊性其前置 0 无需被忽略，因此只需要对照数字进行打印即可无需进行特判，分别调用 3.3.1 中的函数即可。

3.3.3 字符串的打印

由于中文的存储方式为两个 ascii 表示一个中文字符，因此对于给定（传入）的字符串每次从中获取其中的两位字符进行 3.3.1 中所述的打印即可。

这里特别应当注意，由于输出字符串中可能存在空格及英文字母，为了与字库中的文件相匹配，此处的空格与英文字符均应当为全角字符，否则将会由于字符串长与所取时的方式不匹配而导致乱码

3.4 闹钟相关

已保存闹钟的获取与存储在 3.1 中已有提及，因此这里不再进行赘述。这里对闹钟配置文件的同步修改进行解释。当用户修改闹钟时配置文件相应的修改。由于配置文件我采用了顺序存储，且对于闹钟在链表内定位较为不易，因此当用户对闹钟修改时这里采用将配置文件清空重新打印的方式以实现配置文件的同步修改。

3.5 音乐播放

闹钟到点时需要支持音乐播放，而该程序是在 cmd 界面下实现的，因此这里播放音乐采用的是系统中较为原始的 PlaySound 函数，通过设定参数 SND_ASYNC 与 SND_LOOP 分别实现音乐的后台播放与循环播放，在用户触发停止操作前音乐会不断播放至结束。

4. 调试过程中遇到的问题

以下对整个程序的具体实现进行较为以下将对调试过程中遇到的几个主要问题进行概述

4.1 配置文件获取时 eof 无法正确判断

最开始测试配置文件读取功能时发现当配置文件初始为空时，程序在打印完时钟界面后会出现死循环导致程序崩溃退出。通过逐条的方式发现是链表判断被执行了，而此时链表本应为空。因此返回配置文件读取部分，发现当配置文件为空时，fstream 中的 eof 函数却不为 1，而在执行完一次读取操作后程序却能正常退出。

后通过查找 eof 的运行机制得知，在文件为空时 eof 不会立即判定为 1，而是在进行一次读取操作之后才会将 eof 标记置为 1——即 eof 标志的判定是基于一次读取操作之后的。因此解决方法就是在每次读取之前进行一次 peek 操作，然后判断 eof 标志是否被置为 1，若置为 1 即退出读取操作。

4.2 绘制的圆形表盘不完整

绘制表盘的操作是利用穷举矩形区域内点(x,y)距离圆心的距离的平方与 R^2 的差值的最小值来判断的。而最开始判断的打印只以 x 轴为标准进行穷举，打印后发现接近 x 轴附近的圆弧存在缺失，经分析认为是接近 x 轴同时满足平方差值最小的点不足，即误差较小（误差大画出的点

多，线更粗）导致的。因此后采用对 x 轴枚举一次，再对 y 轴枚举一次的方法解决了该问题。

4.3 测试闹钟时的问题

测试闹钟时发现，若出现单次闹钟，退出闹钟后又会发生再响，经过检查发现是单次闹钟执行后并未被删除导致的。但修改后又出现非单次闹钟在触发时立即按停则又会再响的情况。与第一次的情况对比发现是在非单次闹钟停止后，若操作后的时间未发生改变，则判定闹钟时间仍会与前一次重合，因此这里采用加入计时机制，若前一次闹钟成功播放，则在闹钟播放后一分钟内禁止判定其他闹钟的播放。

5. 总结部分

5.1 收获

本次程序的主要难度基本集中于界面打印及时间判断，在老师仅提供点阵字库及使用方法的情况下其余资料几乎都为自己查询的，不仅掌握了关于点阵字库打印的有关知识，也学会了许多实用的打印方式和系统时间获取方式，同时也学到了如何在 cmd 界面中插入音效的知识。

与此同时，由于本次作业不再出现分小题的要求，因此初期程序的规划显得尤为重要。本次程序的编写是通过机械式->电子式->闹钟->配置文件的方式来进行的，每一部分相对独立，在分别测试无误之后才将其拼接起来进行统一测试。实际操作也证明了之前几次大作业打下的分步习惯对于本次作业有着极大的帮助，不仅使思路更加清晰而不会无从下手，同时也使得程序的检查和查错更加方便。

除此之外，本次程序编写使我注意到了宏定义及全局常量的重要性。宏定义可以将常量定义为字符常量，通过该形式可以使程序编写更加清晰，也可以使程序阅读更加简易方便。而全局常量可以将一些需要的字符串/参数等事先定义好，也使得修改时更加方便

6. 附件：程序（不包括头文件定义及打印函数）

时钟部分：

```
void clock_work(Alarm *head, int &option, int &mode)
{
    system("cls");

    int time_zone = 8;
    Point last_P[3][N] = { { 0,0 } };
    const Point O = { 100,50 };
    static DWORD pre_time = 0;

    bool fst = 1;
    bool setting_change = 0;
    bool day_change = 0;
    while (1)
    {
        mode == Mechanical ?
            output_mechanical_clock(O, last_P, time_zone, day_change) :
            output_digital_clock(time_zone, day_change);
        if (fst || setting_change)
        {
            mode == Mechanical ?
                output_help(Mechanical_Y) :
                output_help(Digital_Y);
            int Y = mode == Mechanical ? Mechanical_zone_Y : Digital_zone_Y;
            output_time_zone(Y, time_zone);
            int X = mode == Mechanical ? 200 : 10;
            Y = mode == Mechanical ? 10 : 3;
```

```
                output_day_info(mode,
time_zone, X, Y);
                fst = setting_change =
day_change = 0;
            }
            else if (day_change)
            {
                int X = mode == Mechanical ?
200 : 10;
                int Y = mode == Mechanical ?
10 : 3;
                output_day_info(mode,
time_zone, X, Y);
                day_change = 0;
            }

            if
(GetTickCount()-pre_time>60000          &&
check_alarm(head, time_zone))
            {
                pre_time = GetTickCount();
                setting_change = 1;
                continue;
            }

            int key = get_key(hin);
            if (key == TLE || key ==
MOUSE_NO_ACTION)
                continue;
            if (key == VK_F1)
            {
                setting_change = 1;
                mode ^= 1;
                system("cls");
                continue;
            }
            if (key == VK_F2)
            {
                option ^= 1;
                return;
            }
            if (key == VK_UP || key ==
VK_DOWN)
            {
                setting_change = 1;
                time_zone += key == VK_UP ?
-1 : 1;
                if (time_zone < 0)
                    time_zone = 23;
                if (time_zone > 23)
                    time_zone = 0;
                system("cls");
                continue;
            }
        }
    }
```

```
    bool check_alarm(Alarm *head, const int
time_zone)
    {
        int delta = time_zone - base_zone;
        time_t cur_sec = int(time(0)) + delta
* 3600;
        if (time_zone > 12)
            cur_sec -= 24 * 3600;
        tm *cur_time = localtime(&cur_sec);
        int w = cur_time->tm_wday;
        int h = int(cur_time->tm_hour);
        int m = int(cur_time->tm_min);
        if (!w)
            w = 7;

        Alarm *cur = head;
        while ((cur = cur->nxt) != head)
        {
            if (check_time(cur, w, h, m))
            {
                if (!cur->mode)
                {
                    cur->pre->nxt       =
cur->nxt;
                    cur->nxt->pre       =
cur->pre;
                    delete cur;
                    refresh_ini(head);
                }
                Alarm_Clock_Rang();
                system("cls");
                return 1;
            }
        }
        return 0;
    }

    void output_mechanical_clock(const Point
O, Point last[3][N], const int time_zone, bool
&day_change)
    {
        int delta = time_zone - base_zone;
        time_t cur_sec = int(time(0)) + delta
* 3600;
        tm *cur_time = localtime(&cur_sec);
        int h = int(cur_time->tm_hour);
        if (h >= 12)
            h -= 12;
        int m = int(cur_time->tm_min);
        int s = int(cur_time->tm_sec);
        static int pre_day = 0;
        if (!pre_day)
            pre_day = cur_time->tm_yday;
        else      if      (pre_day      !=
(cur_time->tm_yday))
```

```
            day_change = 1, pre_day =
cur_time->tm_yday;

        double sec_ang = 2 * pi*1.0*s / 60;
        double min_ang = 2 * pi*1.0*m / 60;
        double hour_ang = 2 * pi*1.0*h / 12 +
min_ang / 12;
        sec_ang = pi / 2 - sec_ang;
        min_ang = pi / 2 - min_ang;
        hour_ang = pi / 2 - hour_ang;
        if (sec_ang <= -pi)
            sec_ang += 2 * pi;
        if (min_ang <= -pi)
            min_ang += 2 * pi;
        if (hour_ang <= -pi)
            hour_ang += 2 * pi;

        Draw_Circle(0);
        Draw_Hand(0, last, sec_ang, 2);
        Draw_Hand(0, last, min_ang, 1);
        Draw_Hand(0, last, hour_ang, 0);
    }
    void output_digital_clock(const int
time_zone, bool &day_change)
    {
        int delta = time_zone - base_zone;
        time_t cur_sec = int(time(0)) + delta
* 3600;
        tm *cur_time = localtime(&cur_sec);
        int h = int(cur_time->tm_hour);
        int m = int(cur_time->tm_min);
        int s = int(cur_time->tm_sec);
        static int pre_day = 0;
        if (!pre_day)
            pre_day = cur_time->tm_yday;
        else if (pre_day !=
(cur_time->tm_yday))
            day_change = 1, pre_day =
cur_time->tm_yday;

        const int X = 0;
        const int Y = 65;
        gotoxy(hout, X, Y);

        output_dig(h, Normal);
        output_ch((unsigned char *)" : ",
Normal);
        output_dig(m, Normal);
        output_ch((unsigned char *)" : ",
Normal);
        output_dig(s, Normal);
    }
    void output_help(const int Y)
    {
        gotoxy(hout, 0, Y + 1);
        output_str((unsigned char *)"按↑↓
键改变时区", LED);
        gotoxy(hout, 0, Y + 17);
        output_str((unsigned char *)"按F 1
键切换至数字式", LED);
        gotoxy(hout, 0, Y + 33);
        output_str((unsigned char *)"按F 2
切换至闹钟设定", LED);
    }
    void output_time_zone(const int Y, const
int zone)
    {
        const int X = 3;
        setcolor(hout, 0, 0);
        For(j, 0, 16)
        {
            gotoxy(hout, X, Y + j);
            For(i, 1, 16 * 12)
                printf(" ");
        }
        gotoxy(hout, X, Y);

        output_str((unsigned char
*)Zone[zone], Normal);
        setcolor(hout, 0, 7);
    }
    void output_day_info(const int mode,
const int time_zone, const int X, const int Y)
    {
        int delta = time_zone - base_zone;
        time_t cur_sec = int(time(0)) + delta
* 3600;
        if (time_zone > 12)
            cur_sec -= 24 * 3600;
        tm *cur_time = localtime(&cur_sec);
        int y = 1900 + cur_time->tm_year;
        int m = 1 + cur_time->tm_mon;
        int d = cur_time->tm_mday;
        int w = cur_time->tm_wday;

        gotoxy(hout, X, Y);
        output_dig(y / 100, Normal),
output_dig(y % 100, Normal),
output_str((unsigned char*)"年", Normal);
        gotoxy(hout, X, Y + 20);
        output_dig(m, Normal),
output_str((unsigned char*)"月", Normal),
output_dig(d, Normal), output_str((unsigned
char *)"日", Normal);
        gotoxy(hout, X, Y + 40);
        output_str((unsigned char*)Week[w],
Normal);
    }
    闹钟部分：
    void alarm_work(Alarm *head, int &option)
    {
        Alarm *cur;
```

```
        cur = head->nxt;
        output_alarm_menu();
        output_alarm(cur, head);

        while (1)
        {
            refresh_ini(head);
            int op = get_key(hin, 0);
            while (op != VK_BACK && op != VK_DELETE && op != VK_F1 && op != VK_F2
                && op != VK_UP && op != VK_DOWN)
                op = get_key(hin, 0);
            if (op == VK_BACK)
            {
                option ^= 1;
                return;
            }
            if (op == VK_DELETE)
            {
                if (cur != head)
                {
                    Alarm *tmp = cur->nxt;
                    cur->pre->nxt       = cur->nxt;
                    cur->nxt->pre       = cur->pre;
                    delete cur;
                    cur = tmp;
                    if (cur == head)
                        cur = head->nxt;
                }
                output_alarm(cur, head);
                continue;
            }
            if (op == VK_UP || op == VK_DOWN)
            {
                cur = op == VK_UP ? cur->pre : cur->nxt;
                if (cur == head)
                    cur = op == VK_DOWN ? head->nxt : head->pre;
                output_alarm(cur, head);
                continue;
            }
            if (op == VK_F1 && cur == head)
                continue;
            Alarm *tmp = op == VK_F1 ? tmp = cur : tmp = head;
            bool Cancle = alarm_set(head, tmp);
            if (!Cancle)
                cur = tmp;
            output_alarm_menu();
            output_alarm(cur, head);
        }
    }
    bool alarm_set(Alarm *const head, Alarm *&cur)
    {
        if (cur == head)
        {
            Alarm *New = new(nothrow) Alarm;
            if (New == NULL)
            {
                printf("Alarm setting Error\n");
                exit(-1);
            }
            New->h = New->m = New->mode = 0;
            New->nxt = New->pre = New;
            cur = New;
        }
        int choose = 1;
        output_alarm_setting();
        while (1)
        {
            output_alarm(cur, choose);
            int op = get_key(hin, 0);
            while (op != VK_BACK && op != VK_F1 && op != VK_UP && op != VK_DOWN && op != VK_RETURN)
                op = get_key(hin, 0);
            if (op == VK_BACK)
                return 1;
            if (op == VK_RETURN)
            {
                if (cur->nxt == cur)
                {
                    cur->nxt = head->nxt;
                    head->nxt->pre = cur;
                    head->nxt = cur;
                    cur->pre = head;
                }
                return 0;
            }
            if (op == VK_UP || op == VK_DOWN)
            {
                int delta = op == VK_UP ? 1 : -1;
                if (choose == 1)
                {
                    cur->h += delta;
                    if (cur->h > 23)
                        cur->h -= 24;
                    if (cur->h < 0)
                        cur->h += 24;
                }
                else if (choose == 2)
                {
                    cur->m += delta;
```

```
                    if (cur->m > 59)
                        cur->m -= 60;
                    if (cur->m < 0)
                        cur->m += 60;
                }
                else
                {
                    cur->mode += delta;
                    if (cur->mode > 10)
                        cur->mode -= 11;
                    if (cur->mode < 0)
                        cur->mode += 11;
                }
                continue;
            }
            if (op == VK_F1)
            {
                ++choose;
                if (choose > 3)
                    choose -= 3;
                continue;
            }
        }
    }

    void output_alarm(Alarm *cur, const Alarm
*head)
    {
        const int X = 0;
        const int Y = 3;
        gotoxy(hout, X, Y);

        if (cur == head)
            output_str((unsigned char *)"当
前无闹钟", Abnormal);
        else
        {
            output_dig(cur->h, Abnormal);
            output_ch((unsigned char *)":
", Abnormal);
            output_dig(cur->m, Abnormal);
        }
        gotoxy(hout, X, Y + 18);
        output_str(cur == head ? (unsigned
char*)"       无       "    :    (unsigned
char*)Mode[cur->mode], Abnormal);
    }
    void output_alarm(Alarm *cur, const int
choose)
    {
        const int X = 0;
        const int Y = 3;

        gotoxy(hout, X, Y);
        output_dig(cur->h,  choose  ==  1  ?
Abnormal : Normal);
        output_ch((unsigned  char  *)" : ",
Normal);
        output_dig(cur->m,  choose  ==  2  ?
Abnormal : Normal);
        gotoxy(hout, X, Y + 18);
        output_str((unsigned          char
*)Mode[cur->mode], choose == 3 ? Abnormal :
Normal);

    }
    void output_alarm_menu()
    {
        system("cls");
        const int X = 0;
        const int Y = 3;

        gotoxy(hout, 0, Y + 38);
        output_str((unsigned char *)"按↑↓
查看已有闹钟", LED);
        gotoxy(hout, 0, Y + 56);
        output_str((unsigned char *)"按F1
键修改当前闹钟", LED);
        gotoxy(hout, 0, Y + 74);
        output_str((unsigned char *)"按F2
新建闹钟", LED);
        gotoxy(hout, 0, Y + 92);
        output_str((unsigned char *)"按Delete删除", LED);
        gotoxy(hout, 0, Y + 110);
        output_str((unsigned char *)"按Backspace退回", LED);
    }
    void output_alarm_setting()
    {
        system("cls");
        const int X = 0;
        const int Y = 3;

        gotoxy(hout, 0, Y + 38);
        output_str((unsigned char *)"按↑↓
调节当前所选数值", LED);
        gotoxy(hout, 0, Y + 56);
        output_str((unsigned char *)"按F1
键切换修改项", LED);
        gotoxy(hout, 0, Y + 74);
        output_str((unsigned char *)"按回车
键确定修改", LED);
        gotoxy(hout, 0, Y + 92);
        output_str((unsigned char *)"按Backspace取消", LED);
    }
    工具函数：
    void Draw_Circle(const Point O)
    {
        const int R = 40;
        const int R2 = R*R;
```

```
        const int low = 0.x - 2 * R, high = 0.x
+ 2 * R;
        const int _low = 0.y - R, _high = 0.y
+ R;

        setcolor(hout, 7, 0);

        For(x, low, high)
        {
            int min_dis = R2, dis;
            For(y, _low - 1, _high + 1)
                if ((dis = abs((x - 0.x)*(x
- 0.x) / 4 + (y - 0.y)*(y - 0.y) - R2)) < min_dis)
                    min_dis = dis;
            For(y, _low - 1, _high + 1)
                if (abs((x - 0.x)*(x - 0.x)
/ 4 + (y - 0.y)*(y - 0.y) - R2) == min_dis)
                {
                    gotoxy(hout, x, y);
                    printf(" ");
                }
        }
        For(y, _low, _high)
        {
            int min_dis = R2, dis;
            For(x, low - 1, high + 1)
                if ((dis = abs((x - 0.x)*(x
- 0.x) / 4 + (y - 0.y)*(y - 0.y) - R2)) < min_dis)
                    min_dis = dis;
            For(x, low - 1, high + 1)
                if (abs((x - 0.x)*(x - 0.x)
/ 4 + (y - 0.y)*(y - 0.y) - R2) == min_dis)
                {
                    gotoxy(hout, x, y);
                    printf(" ");
                }
        }

        setcolor(hout, 0, 7);
        gotoxy(hout, 0.x, 0.y);
    }
    void Draw_Hand(const Point 0, Point
last[3][N], double ang, int op)
    {
        int len = op ? (op == 2 ? Second :
Minute) : Hour;
        setcolor(hout, 0, 7);
        sFor(i, 0, N)
        {
            if              (!last[op][i].x
&& !last[op][i].y)
                break;
            gotoxy(hout,     last[op][i].x,
last[op][i].y);
            printf(" ");
            last[op][i].x = last[op][i].y =
0;
        }

        int dx = (ang >= -pi / 2 && ang <= pi
/ 2) ? 1 : -1;
        int dy = ang > 0 ? -1 : 1;
        int cnt = 0;
        setcolor(hout, 7, 0);
        if (fabs(ang - pi / 2) < eps ||
fabs(ang + pi / 2) < eps)
        {
            for (int y = 0.y; abs(y - 0.y) <=
len; y += dy)
            {
                gotoxy(hout, 0.x, y);
                printf(" ");
                last[op][cnt++]          =
{ 0.x,y };
            }
        }
        else
        {
            double k = tan(ang);
            int preY = 0.y; // avoid draw up
and down
            double  limit_x  =  fabs(2  *
len*cos(ang))+1;
            for (int x = 0.x; 1.0*abs(x - 0.x)
< limit_x; x += dx)
            {
                int minY;
                double min_delta = len;
                double limit_y = fabs(k*(x
- 0.x)) + 4;
                for (int y = 0.y; 1.0*abs(y
- 0.y) < limit_y; y += dy)
                {
                    double delta;
                    if      ((delta      =
fabs(fabs(k*(x - 0.x) / 2) - fabs(y - 0.y))) <
min_delta)
                        min_delta = delta,
minY = y;
                }

                bool overflow = 0;
                if ((x - 0.x)*(x - 0.x) / 4
+ (minY - 0.y)*(minY - 0.y) >= len*len)
                    minY    =    0.y    +
(int)sqrt(len*len - (x - 0.x)*(x - 0.x) / 4)*dy,
overflow = 1;
                if ((minY - preY)*dy < 0)
                    break;

                gotoxy(hout, x, minY);
                printf(" ");
```

```
                    last[op][cnt++]          =
{ x,minY };
                    if(abs(minY-preY)>1)
                        for  (int  y  =  preY;
abs(y - preY) <= abs(minY - preY); y += dy)
                        {
                            gotoxy(hout,   x,
y);
                            printf(" ");
                            last[op][cnt++] =
{ x,y };
                        }

                    if (overflow)
                        break;
                    preY = minY;
                }
            }
            setcolor(hout, 0, 7);
            gotoxy(hout, 0.x, 0.y);
        }
```

装 订 线