模拟LED显示屏的实现

综合题 实验报告

计2 1752528 朱世轩

2018 .6 .11

装

订

线

1. 题目及基本要求描述

本次综合题题目为实现一个在cmd窗口下模拟LED显示屏的程序
题目要求如下：
• 理解GB2312汉字编码的概念，自行学习HZK16F(繁)和HZK16(简)字库的使用，能够从字库中提取全角字符的点阵信息并打印出来
• 提取点阵信息后，利用转换出的点阵字实现LED显示屏的模拟。
• 要求从配置文件中读取模拟显示屏的配置信息，如行数，列数，屏延时，条延时等等，并对显示屏进行相应的设置。
• 除静止显示外，要完成至少3种显示特效，模拟LED显示屏上的各种特效,如滚动，翻书等等
• 从配置文件中读取到的要显示的内容中可能含有半角字符，要求转为全角后提取相应的点阵信息并显示。
• 程序运行时按item从1-20的顺序循环显示不退出

装

2.LED显示屏模拟实现的基本思路

订

本次综合题我的基本实现思路如下：
• 我先理解了gb2312字库的原理，学习了HZ16字库的使用方法，先写出获取汉字点阵信息的函数(先利用汉字计算出区位码和在字库中的偏移量，然后打开字库文件对点阵信息进行提取)，进行打印测试，确认取得的汉字信息无误；

线

• 然后构建头文件，定义两个结构体，LED结构体用来存放从配置文件中读到的配置信息，Character结构体数组用来存放要显示的item内容的每个字的点阵信息及每个字要显示的位置等信息

• 编写要使用的工具函数，如半角转全角的工具函数，调用90-b3配置文件工具集，从配置文件中读取配置信息的函数，打印函数等

• 在主函数中获取配置信息，循环调用工具函数，将item里的内容逐字转换为点阵信息存进character ch[128]的结构体数组中，再编写实现不同特效的显示函数，调用打印函数进行模拟对应的特效显示

```
#define MAX_LENGTH 128
#define CH_J 0
#define CH_F 1

struct LED
{
    int col;    //列数
    int row;    //行数
    int bgcolor;   //背景色
    char spec[20];   //特效
    int screen_time;  //屏延时
    int sen_time;     //条延时
    char CH_Mode[10];     //字库
    char content[22][MAX_LENGTH];
    int content_color[22];

};

struct Character
{
    unsigned char word[3];
    char graph[16][16];
    int word_x;
    int word_y;
    int chmode;
};
```

头文件及工具函数

源码见最后

```
int GetCharacter(Character &ch) { ... }

int get_info(LED &led) { ... }

int get_ch_info(Character &ch, LED led) { ... }

int GetString(Character ch[128], LED led, int num_of_item) { ... }
```

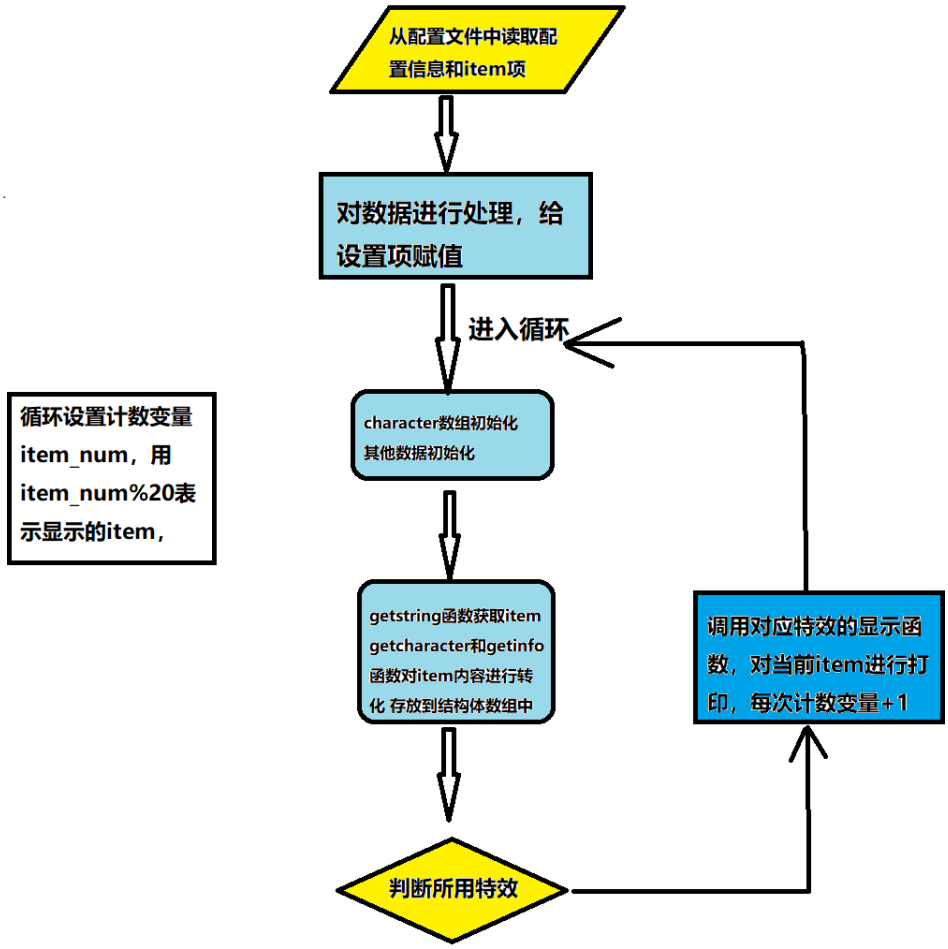
3. 主要功能实现（主要结构的流程框图）

对于模拟LED显示屏主要结构的流程，我的实现流程在主函数中有所体现。
大致就是先获取各项配置信息（行列数，特效，颜色等等），再在循环中
调用工具函数对数据进行处理，打印，main函数如下图所示

```
int main()
{
    Character ch[128];                          //存放单字的结构体数组
    LED led;                                    //存放配置信息的结构体数组
    int num_of_item = 1;                        //内容项
    get_info(led);                              //读取配置信息
    int spec = 1;
    for (int i = 0; i < 20; i++) { ... }
    while (1)
    {
        /*数据初始化*/
        for (int i = 0; i < 128; i++) { ... }
        for (int i = 0; i < 128; i++) { ... }
        setcolor();
        GetString(ch, led, (num_of_item - 1) % 20);
        for (int i = 0; i < 128; i++)
        {
            get_ch_info(ch[i], led);
            GetCharacter(ch[i]);
        }
        if (spec == 1)
            PrintLED(ch, led, (num_of_item - 1) % 20);
        else if (spec == 2)
            PrintLED_spec2(ch, led, (num_of_item - 1) % 20);
        else if (spec == 3)
            PrintLED_spec3(ch, led, (num_of_item - 1) % 20);
        else if (spec == 4)
            PrintLED_spec4(ch, led, (num_of_item - 1) % 20);
        else if (spec == 5)
            PrintLED_spec5(ch, led, (num_of_item - 1) % 20);
        Sleep(led.sen_time * 1000);
        num_of_item++;
    }
    return 0;
}
```

装 订 线

程序主要结构的流程框图如下所示



主要结构流程图

4. 调试过程中遇到的问题及解决过程

对于模拟LED显示屏，我在实现过程中的不同地方都遇到了一些问题；

1. 在打印单个字符时，由于cmd窗口横纵比不是1：1，导致打印出来的图形出现畸变，是扁的；

解决方法：通过一行打印两个空格实现方形字母显示；、

2.在读取配置文件时，发现90-b3配置文件工具集的读取配置文件函数get_value出现错误，同时读取配置文件函数的编写考虑不周全，导致读取配置文件时出现了没有读取到、读取错误等问题，从而打印时无法正常显示；

解决方法：重新编写get_value函数，修改了出现的bug；重新构思综合读取配置文件的函数，

修正读取配置文件的函数，调试使函数能正常读取、配置变量，从而实现正常功能；

3.在实现特效时，尝试使用一个输出函数通过不同变量实现不同特效，但发现不容易实现，并且出现了许多问题；
解决方法：对每个特效编写一个显示函数，显示函数公用基础打印函数。

5.完成本次作业的心得体会

5.1.完成作业的心得体会和经验教训

心得体会：通过这次作业，我有一些心得体会；
• 首先我理解了汉字编码，学会了字库的使用；
• 在模拟LED显示屏的实现中，我沿用了在二维码大作业中的做法，采用结构体和结构体数组来存放配置信息等，方便函数传值的同时使用也很直观。我觉得这个方法可以沿用到很多地方；
经验教训：
• 在写每一个程序的时候都应该考虑日后的使用，尽量考虑周全，防止日后出现问题。这次在使用配置文件工具集时，由于出现问题，我不得不花大量时间修改。

5.2.完成作业的感受（关于分解功能、代码重用等的思考）

经过几次综合题的实现，对于综合题的编写，我已经能比较好地将整个程序划分为较小的功能模块，再逐个函数去编写，而不是漫无目的地想到哪写到哪；
我也已经开始有意识地提取工具函数，先写好工具函数，再进行功能函数的编写，方便调用，提高效率；
关于代码重用，我还是有所欠缺，这次的特效实现很多重复代码，但我并没有想到很好的重用方法，为了稳定，还是每个特效做了单独的显示函数，共同调用单字显示函数。这方面的能力我日后还需加强。

装 订 线

装 订 线

6. 附件：部分源代码

• 头文件部分

```
struct LED
{
    int col;    //列数
    int row;    //行数
    int bgcolor;   //背景色
    char spec[20];   //特效
    int screen_time;  //屏延时
    int sen_time;     //条延时
    char CH_Mode[10];     //字库
    char content[22][MAX_LENGTH];
    int content_color[22];

};

struct Character
{
    unsigned char word[3];
    char graph[16][16];
    int word_x;
    int word_y;
    int chmode;
};
/*取得汉字信息*/
int get_ch_info(Character &ch, LED led);
/*取得汉字点阵信息*/
int GetCharacter(Character &ch);
/*打印汉字点阵*/
int PrintCharacter(Character ch, LED led, int item_num);  //item_num为显示item的序号

int PrintCharacter_spec2(Character ch, LED led, int item_num, int play_num);

int PrintCharacter_spec3(Character ch, LED led, int item_num, int play_num);

int PrintLED_spec4(Character ch[128], LED led, int item_num);

/*取得显示信息*/
int get_info(LED &led);
/*打印模拟led*/
int PrintLED(Character ch[128], LED led, int item_num);  //item_num为显示item的序号

int PrintLED_spec2(Character ch[128], LED led, int item_num);

int PrintLED_spec3(Character ch[128], LED led, int item_num);

int PrintLED_spec5(Character ch[128], LED led, int item_num);
/*讲content内容存进ch结构体数组*/
```

```
int GetString(Character ch[128], LED led, int num_of_item);//item_num为获取item的序号
```

• 工具函数部分

```
int GetCharacter(Character &ch)     //ch为存放汉字信息的结构体数组元素
{
    fstream fp_hzk;
    int i, j, k, offset;
    int flag;
    char buffer[32];                //存放字库中读取出的点阵信息
    unsigned char key[8] = { 0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01 };
    if (ch.chmode)
        fp_hzk.open("hzk16f", ios::in | ios::binary);
    else
        fp_hzk.open("hzk16", ios::in | ios::binary);

    if (!fp_hzk.is_open()) {
        fprintf(stderr, "error hzk16\n");
        return 1;
    }
    offset = (94 * (unsigned int)(ch.word[0] - 0xa0 - 1) + (ch.word[1] - 0xa0 - 1)) * 32;
    fp_hzk.seekg(offset, ios::beg);
    fp_hzk.read(buffer, 32);
    for (k = 0; k < 16; k++)
    {
        for (j = 0;
            j < 2; j++)
        {
            for (i = 0; i < 8; i++)
            {
                flag = buffer[k * 2 + j] & key[i];
                if (flag)
                {
                    ch.graph[k][j * 8 + i] = 1;
                }
                else
                {
                    ch.graph[k][j * 8 + i] = 0;
                }
            }
        }
    }
    fp_hzk.close();
    return 0;
}

int get_info(LED &led)                                          //从配置文件中获取配置信息
{
    fstream fp_led;
    if (open_cfgfile(fp_led, "led.cfg", OPEN_OPT_RDONLY) == 0)
    {
        cout << "无法打开配置文件 " << endl;
        return -1;
```

```
}
item_get_value(fp_led, "setup", "行数", &led.row, TYPE_INT);
item_get_value(fp_led, "setup", "列数", &led.col, TYPE_INT);
item_get_value(fp_led, "setup", "屏延时", &led.screen_time, TYPE_INT);
item_get_value(fp_led, "setup", "条延时", &led.sen_time, TYPE_INT);
item_get_value(fp_led, "setup", "背景色", &led.bgcolor, TYPE_INT);
if (!item_get_value(fp_led, "setup", "字库", led.CH_Mode, TYPE_STRING))
    strcpy(led.CH_Mode, "HZK16");
for (int i = 0; i < 20; i++)
{
    char item_name[64];
    strcpy(item_name, "item");
    if (i < 9)
    {
        item_name[4] = i + 1 + '0';
        item_name[5] = 0;
    }
    else if (i < 19)
    {
        item_name[4] = '1';
        item_name[5] = i - 9 + '0';
        item_name[6] = 0;
    }
    else
    {
        item_name[4] = '2';
        item_name[5] = i - 19 + '0';
        item_name[6] = 0;
    }
    if (!item_get_value(fp_led, "content", item_name, led.content[i], TYPE_STRING))
        led.content[i][0] = 0;
    strcat(item_name, "_color");
    char color;
    if (!item_get_value(fp_led, "content", item_name, &color, TYPE_CHARACTER))
    {
        srand((unsigned)time(NULL));
        led.content_color[i] = rand() % 15;
    }
    if (color == 'x')
    {
        srand((unsigned)time(NULL));
        led.content_color[i] = rand() % 15;
    }
    else
        led.content_color[i] = color - '0';
}
int flag_spec = 0;
for (int i = 0; i < 5; i++)
{
    char spec_name[64];
    strcpy(spec_name, "特效");
    spec_name[4] = i + 1 + '0';
    spec_name[5] = 0;
    if (item_get_value(fp_led, "setup", spec_name, &led.spec[i], TYPE_CHARACTER))
```

装 订 线

```
            flag_spec = 1;
    }
    if (flag_spec == 0)
        led.spec[0] = 'Y';
    return 0;
}

int get_ch_info(Character &ch, LED led)
{
    if (strcmp(led.CH_Mode, "HZK16") == 0)
        ch.chmode = CH_J;
    else
        ch.chmode = CH_F;
    return 0;
}

int GetString(Character ch[128], LED led, int num_of_item)  //将字符串中的全角字符读进ch数组
{
    int i = 0;
    int num = 0;
    char word1 = 0, word2 = 0;
    cout << led.content[num_of_item] << endl;              //test
    while (led.content[num_of_item][num] != '\0'&&i < 128)
    {
        if (led.content[num_of_item][num] < 0x80 && led.content[num_of_item][num]>0x20)//如果是半角字符 先转全角
        {
            ch[i].word[0] = 0xA3;
            ch[i].word[1] = led.content[num_of_item][num] + 0x80;
            ch[i].word[2] = 0;
            i++;
            num++;
        }
        else
        {
            ch[i].word[0] = led.content[num_of_item][num];
            ch[i].word[1] = led.content[num_of_item][num + 1];
            ch[i].word[2] = 0;
            i++;
            num = num + 2;
        }
    }
    return 0;
}
```

• 显示函数部分

```
/*打印单个字 正常 特效1*/
int PrintCharacter(Character ch, LED led, int item_num)              //打印单个字
{
    for (int i = 0; i < 16; i++)
    {
```

```
        gotoxy(ch.word_x, ch.word_y + i);
        for (int j = 0; j < 16; j++)
        {
            if (ch.graph[i][j])
                setcolor(led.content_color[item_num], led.content_color[item_num]);//有字部分
            else
                setcolor(led.bgcolor, led.bgcolor);  //背景部分
            putchar(' ');
            putchar(' ');
        }
        putchar('\n');
    }
    return 0;
}

/*打印单个字 特效2*/
int PrintCharacter_spec2(Character ch, LED led, int item_num, int play_num)       //play_num表示移动了几格
{
    for (int i = 0; i < 16; i++)
    {
        gotoxy(ch.word_x, ch.word_y + i);
        for (int j = (play_num / 2) % 8; j < 16; j++)
        {
            if (ch.graph[i][j])
                setcolor(led.content_color[item_num], led.content_color[item_num]);//有字部分
            else
                setcolor(led.bgcolor, led.bgcolor);  //背景部分
            putchar(' ');
            putchar(' ');
        }
        putchar('\n');
    }
    return 0;
}

/*打印单个字 特效3*/
int PrintCharacter_spec3(Character ch, LED led, int item_num, int play_num)       //play_num表示移动了几格
{
    for (int i = 0; i < 16; i++)
    {
        gotoxy(ch.word_x, ch.word_y + i);
        for (int j = (play_num / 2) % 16; j < 16; j++)
        {
            if (ch.graph[i][j])
                setcolor(led.content_color[item_num], led.content_color[item_num]);//有字部分
            else
                setcolor(led.bgcolor, led.bgcolor);  //背景部分
            putchar(' ');
            putchar(' ');
```

```
        }
        //cout << endl;
        putchar('\n');
    }
    return 0;
}

/*特效1 正常静止显示*/
int PrintLED(Character ch[128], LED led, int item_num)            //正常显示  特效1
{
    setfontsize("点阵", 6);
    setconsoleborder(32 * led.col + 1, 16 * led.row + 1, 32 * led.col + 1, 16 * led.row + 1);
    int count = 0; //计数 已打印的字符数
    while (1)
    {
        for (int i = 0; i < led.row; i++)
        {
            for (int j = 0; j < led.col; j++)
            {
                if (ch[i*led.col + j + count].word[0])
                {
                    ch[i*led.col + j + count].word_x = 32 * j;
                    ch[i*led.col + j + count].word_y = 16 * i;
                    PrintCharacter(ch[i*led.col + j + count], led, item_num);
                }
                else
                    return 0;
            }
        }
        if (count == 0)
        {
            Sleep(led.screen_time * 1000);
            count = (led.col)*(led.row);
        }
        else
        {
            Sleep(led.screen_time * 1000);
            break;
        }
        cls();
    }
    return 0;
}
/*特效2 翻书效果*/
int PrintLED_spec2(Character ch[128], LED led, int item_num)            //打印led界面 特效2
row为行 col为列 item_num为显示第几条
{
    setfontsize("点阵", 6);
    setconsoleborder(32 * led.col + 1, 16 * led.row + 1, 32 * led.col + 1, 16 * led.row + 1);
    int count = 0;
    for (int num = 0; num < led.col * 32; num = num + 16)                    //num记录移动次
数
    {
        for (int j = num / 32; j < led.col; j++)
```

```
        {
            for (int i = 0; i < led.row; i++)
            {
                if (ch[i*led.col + j].word[0])
                {

                    if (j - num / 32 == 0)
                    {
                        ch[i*led.col + j].word_x = 0;
                        ch[i*led.col + j].word_y = 16 * (i);
                        PrintCharacter_spec2(ch[i*led.col + j], led, item_num, num);
                    }
                    else
                    {
                        ch[i*led.col + j].word_x = 32 * (j)-num;
                        ch[i*led.col + j].word_y = 16 * (i);
                        PrintCharacter(ch[i*led.col + j], led, item_num);
                    }
                }
                else
                    break;
            }
        }
        if (count == 0)
        {
            Sleep(led.screen_time * 1000);
            count++;
        }
    }
    return 0;
}
/*特效3 左滑（其实更像毛毛虫= =）*/
int PrintLED_spec3(Character ch[128], LED led, int item_num)
{
    setfontsize("点阵", 6);
    setconsoleborder(32 * led.col + 1, 16 * led.row + 1, 32 * led.col + 1, 16 * led.row + 1);
    for (int num = 0; num < led.col * 32; num++)                    //num记录移动次数
    {
        for (int j = num / 32; j < led.col; j++)
        {
            for (int i = 0; i < led.row; i++)
            {
                if (ch[i*led.col + j].word[0])
                {

                    if (j - num / 32 == 0)
                    {
                        ch[i*led.col + j].word_x = 0;
                        ch[i*led.col + j].word_y = 16 * (i);
                        PrintCharacter_spec3(ch[i*led.col + j], led, item_num, num);
                    }
                    else
                    {
                        ch[i*led.col + j].word_x = 32 * (j)-num;
```

```
                        ch[i*led.col + j].word_y = 16 * (i);
                        PrintCharacter(ch[i*led.col + j], led, item_num);
                    }
                }
                else
                    break;
            }
        }
    }
    return 0;
}
/*特效4 变色 跑马灯*/
int PrintLED_spec4(Character ch[128], LED led, int item_num)
{
    setfontsize("点阵", 6);
    setconsoleborder(32 * led.col + 1, 16 * led.row + 1, 32 * led.col + 1, 16 * led.row + 1);
    int count = 0; //计数 已打印的字符数

    while (1)
    {
        for (int i = 0; i < 10; i++)
        {
            for (int i = 0; i < led.row; i++)
            {
                for (int j = 0; j < led.col; j++)
                {
                    if (ch[i*led.col + j + count].word[0])
                    {
                        ch[i*led.col + j + count].word_x = 32 * j;
                        ch[i*led.col + j + count].word_y = 16 * i;
                        srand((unsigned)time(NULL));
                        led.content_color[item_num] = rand() % 15;
                        PrintCharacter(ch[i*led.col + j + count], led, item_num);
                    }
                    else
                        return 0;
                }
            }
        }
        if (count == 0)
        {
            Sleep(led.screen_time * 1000);
            count = (led.col)*(led.row);
        }
        else
        {
            Sleep(led.screen_time * 1000);
            break;
        }
        //Sleep(led.screen_time * 1000);
        cls();

    }
    return 0;
```

```
}
/*特效5 逐字打印 慢速*/
int PrintLED_spec5(Character ch[128], LED led, int item_num)              //正常显示  特效1
{
    setfontsize("点阵", 6);
    setconsoleborder(32 * led.col + 1, 16 * led.row + 1, 32 * led.col + 1, 16 * led.row + 1);
    int count = 0; //计数 已打印的字符数
    while (1)
    {
        for (int i = 0; i < led.row; i++)
        {
            for (int j = 0; j < led.col; j++)
            {
                if (ch[i*led.col + j + count].word[0])
                {
                    ch[i*led.col + j + count].word_x = 32 * j;
                    ch[i*led.col + j + count].word_y = 16 * i;
                    PrintCharacter(ch[i*led.col + j + count], led, item_num);
                    Sleep(100);
                }
                else
                    return 0;
            }
        }
        if (count == 0)
        {
            Sleep(led.screen_time * 1000);
            count = (led.col)*(led.row);
        }
        else
        {
            Sleep(led.screen_time * 1000);
            break;
        }
        cls();
    }
    return 0;
}
```

- **主函数部分上有贴图 省略**