**模拟LED显示屏**

**实验报告**

**班级：计算机1班**

**姓名：贾昊霖**

**学号：1651574**

**完成日期：2018年6月13日凌晨3点13分**

1. **题目要求**

**（1）LED显示的大小、颜色、内容等从同目录下的led.conf中读取。文件中各配置项及其含义如下：**

**行数/列数：汉字数量**

**背景色/前景色：取值范围为0-15**

**特效：Y/N表示显示时是否使用该特效**

**条延时/屏延时：两条/两屏内容的切换延时**

**sentence：显示的内容（程序运行时，按item编号1-n循环依次显示）**

**bg-color：显示内容时的后景色，取值范围为0-15和x，x表示每次显示的时候颜色随机**

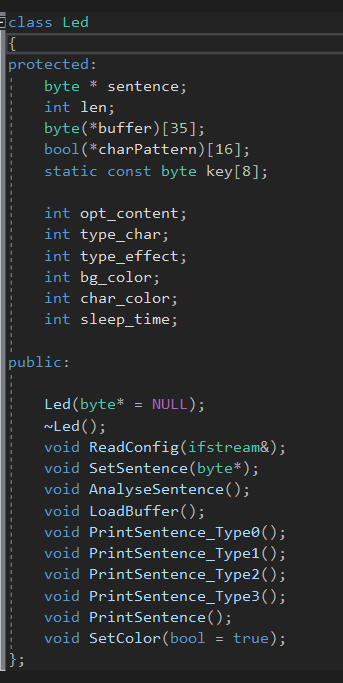
**char-color：显示内容时的前景色，取值范围为0-15和x，x表示每次显示的时候颜色随机**

**（2）LED显示屏的特效：常见的有从右到左的横幅拉动、上下拉动、一次性显示、由内而外、翻书式等等。每个人必须完成三种特效。**

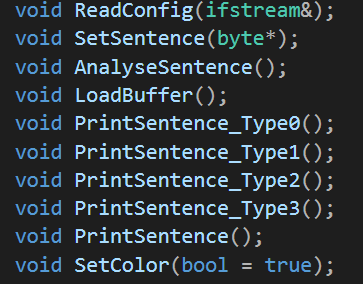
**2.整体设计思路**

**2.1内部数据结构**

1. **Led类: 存储汉字的信息。**
2. **char\* item\_name[20]: 用于储存item的名称，用于对item进行排序以及读取item的相关数据（颜色，内容）**



**2.2主要函数及其功能**



**（1） ReadConfig: 判断读取到的item是否新的item（与之前已经存入到item\_name中的item是否重名）**

**（2）AnalyseSentence: 获取汉字的点阵信息**

**（3）SetColor: 读取item的颜色，创建链表存储item的内容**

**（4）LoadBuffer: 读取已经实现而在配置文件中又要使用的特效编号**

**（5）PrintSentence\_Typex: 实现各种特效（会在show函数中调用）**

**（6）rintSentence: 实现整个模拟LED的函数，调用其他功能函数**

**2.3关于从led.conf读取item的信息**

**（1）按行读取文件的内容存储到临时的字符数组中，当从字符数组中找到“item”这一字符串后，调用函数判断这个是否新的item，如果是则把这个item的名字存入到buffer中**

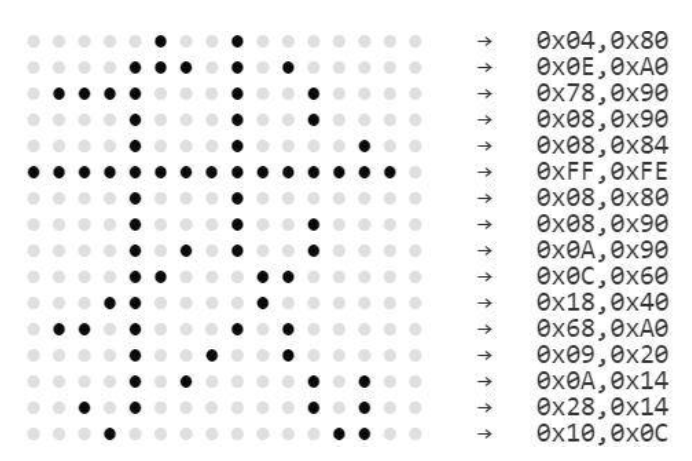
**（2）对item中的所有元素（字符串）进行排序。**

**（3） 读取每个item的内容以及颜色，存储到item，tmp\_buffer对应的如sentence[1]是“item2”，则在item[1]存储item2的内容，item\_color[1]存储item2的颜色。**

1. **主要功能实现**

**3.1汉字的机内码，区位码，偏移量和点阵信息的换算**

**一个汉字包含2个字符，2个字符的机内码分别对应区位码的区号和位号。换算关系为：区位码=机内码-0xA0**

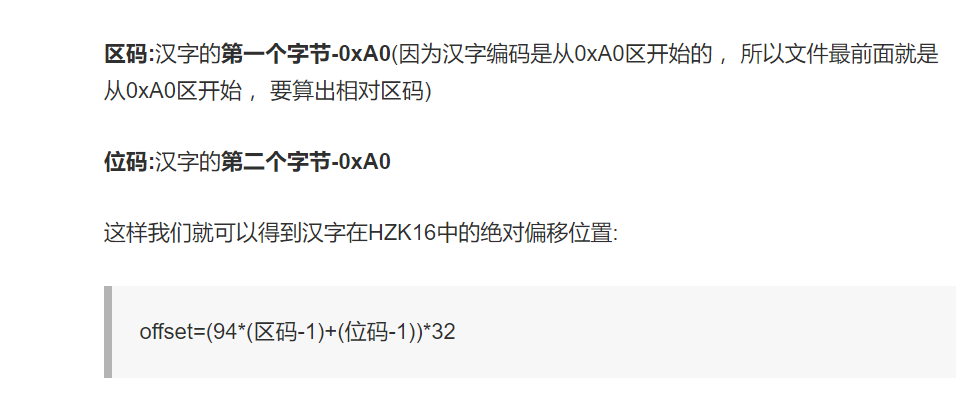


**GB2312编码共94个区，每个区有94个汉字。每个汉字的点阵信息需要（16×16÷8）32个字节。所以汉字在点阵库中的偏移量与**

**该汉字区位码的换算关系为：**

**偏移量=（区号-1）×32×94+（位号-1）×32**

**偏移量代表该汉字的点阵信息在点阵库文件中的位置，所以先让 文件指针从头向后移动偏移量，然后再读取相应大小（32字节）的内容，即可获取汉字的点阵信息。**



**3.2在屏幕上用特效显示item内容（show函数）**

**（1）先把屏幕上要显示的内容对应到一个二维数组buffer与屏幕大小一样。具体做法为：根据从配置文件中读取到的行数和列数确定一屏幕显示多少个汉字，然后把每个汉字的点阵赋值到 screen的对应位置（要打点的位置赋值为1，不需要打点的位置赋0）**

**（2）把screen元素的值设置好后，就调用PrintSentence的内容**

**（3）显示完一屏幕的内容后，判断该条内容是否显示完（每条内容的链表尾部都是NULL，可以从指针是否指向NULL来判断）。如果显示完了就让指针指向item的下一个元素（即下一条内容的链表的表头），否则继续在本链表中读取汉字点阵并显示。**

**3.3切换特效的实现**

**说明：所有特效的实现原理其实都是一样的，也就是在屏幕的不同位置打印内容的先后顺序不同。下面以“从上到下”这种简单的特效为例：**

void Led::PrintSentence\_Type0()

{

int cur\_x, cur\_y;

bool flag;

getxy(cur\_x, cur\_y);

for (int l = 0; l < 7; l++) {

int tmp\_x = cur\_x + 32 \* l;

for (int k = 0; k < 16; k++) {

gotoxy(tmp\_x, cur\_y + k);

for (int j = 0; j < 2; j++) {

for (int i = 0; i < 8; i++) {

flag = buffer[l][k \* 2 + j] & key[i];//

cout << (flag ? "●" : " ");

}

}

}

}

gotoxy(cur\_x, cur\_y + 16);

}

1. **心得体会**

**本大作业花了4个小时左右，因为时间实在挤不开，只能从10开始写，写到现在凌晨3点多，本作业不难，其他特效之类的东西都是换一下数组的输出顺序而已，没有任何难度，因此只简易地制作了3个不知道算不算特效的特效.**

**感觉把之前的大作业合起来看着很有成就感....虽然有些大作业真的没时间做的很完善..**

**5.源代码**

**（省略部分简单函数以及重复代码）**

/\* 1651574 1班 贾昊霖 \*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <iomanip>

#include <cstring>

#include <algorithm>

#include <fstream>

#include <stdint.h>

#include "../common/cmd\_console\_tools.h"

#define BOARD\_COL (32\*6 +3)

#define MAX\_TIME 1000

#define MAX\_LEN (len \* 16)

#define Empty -1

#define Simple 0

#define Traditional 1

using namespace std;

typedef unsigned char byte;

const int sleepTime[] = { 1000,500,300,100,50,20 };

class Led

{

protected:

byte \* sentence;

int len;

byte(\*buffer)[35];

bool(\*charPattern)[16];

static const byte key[8];

int opt\_content;

int type\_char;

int type\_effect;

int bg\_color;

int char\_color;

int sleep\_time;

public:

Led(byte\* = NULL);

~Led();

void ReadConfig(ifstream&);

void SetSentence(byte\*);

void AnalyseSentence();

void LoadBuffer();

void PrintSentence\_Type0();

void PrintSentence\_Type1();

void PrintSentence\_Type2();

void PrintSentence\_Type3();

void PrintSentence();

void SetColor(bool = true);

};

/\* 1651574 1班 贾昊霖\*/

#include "90-b5.h"

const byte Led::key[8] = {

0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01

};

Led::Led(byte\* str)

{

if (!str)

return;

len = strlen((char\*)str) / 2;

sentence = new(nothrow) byte[len \* 2 + 1];

if (!sentence) {

cerr << "内存不足！\n";

exit(ERROR);

}

strcpy((char\*)sentence, (char\*)str);

}

Led::~Led()

{

if (!sentence)

return;

len = 0;

delete sentence;

}

void Led::SetSentence(byte \*str)

{

if (len) {

len = 0;

delete sentence;

}

len = strlen((char\*)str) / 2;

sentence = new(nothrow) byte[len \* 2 + 1];

if (!sentence) {

cerr << "内存不足！\n";

exit(ERROR);

}

strcpy((char\*)sentence, (char\*)str);

}

void Led::PrintSentence\_Type0()

{

int cur\_x, cur\_y;

bool flag;

getxy(cur\_x, cur\_y);

for (int l = 0; l < 7; l++) {

int tmp\_x = cur\_x + 32 \* l;

for (int k = 0; k < 16; k++) {

gotoxy(tmp\_x, cur\_y + k);

for (int j = 0; j < 2; j++) {

for (int i = 0; i < 8; i++) {

flag = buffer[l][k \* 2 + j] & key[i];//

cout << (flag ? "●" : " ");

}

}

}

}

gotoxy(cur\_x, cur\_y + 16);

}

void Led::PrintSentence\_Type1()

{

int cur\_x, cur\_y;

SetColor();

getxy(cur\_x, cur\_y);

int start\_x = 0, times = -1;

while (times++ < MAX\_TIME) {

const int max\_j = 6 \* 16;

start\_x = (start\_x + 1) % MAX\_LEN;

for (int i = 0; i < 16; i++) {

for (int j = 0; j < max\_j; j++) {

cout << (charPattern[(j + start\_x) % (len \* 16)][i] ? "●" : " ");

}

putchar('\n');

}

gotoxy(cur\_x, cur\_y);

Sleep(sleepTime[sleep\_time]);

}

gotoxy(cur\_x, cur\_y + 16);

}

void Led::PrintSentence\_Type2()

{

int cur\_x, cur\_y;

SetColor();

getxy(cur\_x, cur\_y);

int start\_x = 0, times = -1;

while (times++ < MAX\_TIME) {

const int max\_j = 6 \* 16;

start\_x = (start\_x - 1 + MAX\_LEN) % MAX\_LEN;

for (int i = 0; i < 16; i++) {

for (int j = 0; j < max\_j; j++) {

cout << (charPattern[(start\_x + j) % (MAX\_LEN)][i] ? "●" : " ");

}

putchar('\n');

}

gotoxy(cur\_x, cur\_y);

Sleep(sleepTime[sleep\_time]);

}

gotoxy(cur\_x, cur\_y + 16);

}

void Led::PrintSentence\_Type3()

{

int cur\_x, cur\_y;

getxy(cur\_x, cur\_y);

int start\_x = 0, times = -1;

while (times++ < MAX\_TIME) {

const int max\_j = 6 \* 16;

start\_x = (start\_x - 1 + max\_j) % MAX\_LEN;

for (int i = 0; i < 16; i++) {

for (int j = 0; j < max\_j; j++) {

cout << (charPattern[(start\_x + j + len \* 16) % (len \* 16)][i] ? "●" : " ");

}

putchar('\n');

}

gotoxy(cur\_x, cur\_y);

Sleep(sleepTime[sleep\_time]);

SetColor(false);

}

gotoxy(cur\_x, cur\_y + 16);

}

void Led::PrintSentence()

{

LoadBuffer();

switch (type\_effect) {

case(0):

PrintSentence\_Type0();

break;

case(1):

PrintSentence\_Type1();

break;

case(2):

PrintSentence\_Type2();

break;

case(3):

PrintSentence\_Type3();

break;

}

}

void Led::SetColor(bool flag)

{

static int color = 0;

if (!flag) {

setcolor(color, color + 7);

color = (color + 1) % 15;

return;

}

int \_bg, \_char;

if (bg\_color == -1)

\_bg = rand() % 15;

else

\_bg = bg\_color;

if (char\_color == -1)

\_char = rand() % 15;

else

\_char = char\_color;

setcolor(\_bg, \_char);

}

void Led::AnalyseSentence()

{

ifstream charSetBase;

if (type\_char == Simple)

charSetBase.open("HZK16", ios::in | ios::binary);

else

charSetBase.open("HZK16F", ios::in | ios::binary);

if (!charSetBase.is\_open()) {

cerr << "未找到点阵字库，请检查目录\n";

exit(ERROR);

}

buffer = new(nothrow)byte[len][35];

for (int i = 0; i < len; i++) {

int offset = (94 \* (unsigned int)(sentence[i << 1] - 0xA0 - 1) + (sentence[(i << 1) + 1] - 0xA0 - 1)) \* 32;

charSetBase.seekg(offset, ios::beg);

charSetBase.read((char\*)buffer[i], 32);

buffer[i][32] = '\0';

}

charSetBase.close();

}

void Led::LoadBuffer()

{

charPattern = new(nothrow) bool[len \* 16][16];

if (!charPattern) {

cerr << "内存不足！\n";

exit(ERROR);

}

for (int l = 0; l < len; l++) {

int tmp\_add\_l = l \* 16;

for (int j = 0; j < 2; j++) {

for (int i = 0; i < 8; i++) {

int tmp\_add\_ij = tmp\_add\_l + j \* 8 + i;

for (int k = 0; k < 16; k++) {

charPattern[tmp\_add\_ij][k] = buffer[l][k \* 2 + j] & key[i];

}

}

}

}

}

void Led::ReadConfig(ifstream &file)

{

file.open("led（请修改此配置文件）.cfg", ios::in);

if (!file.is\_open()) {

cerr << "未找到配置文件，请检查目录\n";

exit(ERROR);

}

char line[1024], \*p = NULL;

while (!file.eof()) {

file.getline(line, 1024);

if (p = strstr(line, "背景色")) {

p = strchr(line, '=');

if (\*(p + 1) == 'x')

bg\_color = -1;

else

bg\_color = \*(p + 1) - '0';

}

else if (p = strstr(line, "字体颜色")) {

p = strchr(line, '=');

if (\*(p + 1) == 'x')

char\_color = -1;

else

char\_color = \*(p + 1) - '0';

}

else if (p = strstr(line, "特效")) {

p = strchr(line, '=');

type\_effect = \*(p + 1) - '0';

}

else if (p = strstr(line, "屏延时")) {

p = strchr(line, '=');

sleep\_time = \*(p + 1) - '0';

}

else if (p = strstr(line, "字库")) {

if (p = strstr(line, "HZK16F"))

type\_char = Traditional;

else if (p = strstr(line, "HZK16"))

type\_char = Simple;

}

else if (p = strstr(line, "内容")) {

p = strchr(line, '=');

opt\_content = \*(p + 1) - '0';

char findStr[10] = "item";

findStr[4] = opt\_content + '0';

findStr[5] = '\0';

while (!file.eof()) {

file.getline(line, 1024);

if (p = strstr(line, findStr)) {

p = strchr(line, '=');

SetSentence((byte\*)(p + 1));

return;

}

}

}

}

cerr << "config文件有误,请按照程序目录下的config修改！\n";

exit(ERROR);

}

/\* 1651574 1班 贾昊霖\*/

#include "90-b5.h"

int main(void)

{

setfontsize("新宋体",12,6);

setcursor(CURSOR\_INVISIBLE);

setconsoleborder(32 \* 7 + 3, 16 + 3, 32 \* 7 + 3, 16 + 3);

Led led;

ifstream configFile;

led.ReadConfig(configFile);

led.AnalyseSentence();

led.PrintSentence();

return 0;

}