面向对象编程技术

实验三实验报告

班 级：通信一班

学 号：201900800413

姓 名： 左长盛

时 间：2021-10-20

**目 录**

1 实验要求 1

2 实验过程及结果 1

2.1 创建工程 1

2.2 程序流程和数据模型分析 1

2.3 问题及解决方案 1

3 实验总结 1

附录 关键代码 1

# 实验要求

1. 改进ChildWin示例程序，在子窗口内显示出窗口特定信息（窗口ID号）；尝试在每个子窗口中绘制特定颜色的矩形
   * 在子窗口的窗口过程函数的刷新处理消息中增添处理代码；
   * 用GetWindowLong函数取出子窗口ID；
   * 根据子窗口ID设定子窗口内需要显示的信息（使用wsprintf函数组织待显示的字符串）
   * 尝试在子窗口中绘制特定颜色的矩形（使用CreateSolidBrush、Rectangle函数）
2. 跟踪消息传递过程
   * 在Childwin的主窗口过程函数的WM\_CREATE、WM\_PAINT、WM\_COMMAND、WM\_LBUTTONDOWN处设置断点
   * 在子窗口的WM\_CREATE、WM\_PAINT、WM\_CHILDWIN、WM\_LBUTTONDOWN处设置断点
   * 运行程序，查看各断点的响应**过程**，以此理解消息传递的次序
3. **设计、实现一个软键盘**

用子窗口实现一个软键盘（在窗口的客户区用子窗口绘制一个仿真的键盘），鼠标点击每个子窗口，可输入不同的字符。将输入的字符显示到窗口中。

1. 在自己的工作目录中建立新工程用于控件测试〔可以将Easywin工程的easywin.c拷贝到新工程的目录中，修改文件名为ctltest.cpp（后缀名改为.cpp），编译、链接并使程序正常工作（包括修正编译错误）〕。
2. \* 用Button完成4\*所要求的软键盘实现。比较与原有实现的不同之处。
3. 创建各种类型的控件子窗口，包括static、edit、listbox、combobox、button、scrollbar等；在主窗口的窗口过程函数中增加相应的消息处理实现主窗口与子窗口的通信（可尝试设置控件的标题/内容，获取控件的标题/内容以及在控件有点击等操作时在主窗口给出提示）。
4. 同步骤5，新建用于资源测试的工程。
5. 增加新的资源：光标、图标，并正确显示
   * 用资源编辑器编辑一个光标（彩色），注意**删除**资源中**不需要**的image类型，否则显示光标可能不正确。
   * 为光标设定热点（hot spot）
   * 用资源编辑器编辑一个图标
   * 修改注册窗口类的代码，使程序能正常使用自定义的光标和图标
   * 用DrawIcon函数，在WM\_PAINT消息中增加处理，在窗口客户区中显示自定义的光标及图标。
   * 创建一个另外光标供后续实验步骤使用
6. 增加菜单
   * 菜单中建立一个菜单项命令以修改窗口使用的光标，可命名为“ChangeCursor”，对应的ID假设为ID\_CHANGECURSOR
   * 修改**注册窗口类**的代码，使程序运行时能正确载入菜单
   * 换一种方式，通过修改**创建窗口（CreateWindow）**的代码，使菜单能正确载入
7. 增加菜单消息响应代码
   * 增加WM\_COMMAND的消息处理
   * 参照第五章菜单示例代码中的处理方式，响应菜单命令
   * 对ID\_CHANGECURSOR命令的响应为：改变窗口使用的光标，可以使用SetClassLong (hwnd, GCL\_HCURSOR,  
      (LONG)LoadCursor((HINSTANCE)GetWindowLong(hwnd, GWL\_HINSTANCE),   
      MAKEINTRESOURCE(IDC\_CURSOR2) ) );
8. 处理鼠标右键，增加快捷菜单处理
   * 右键发出的菜单命令同样是通过WM\_COMMAND消息响应的，增加快捷菜单之后，程序应该能通过主菜单或快捷菜单两种方式发出同样的命令
9. 将本次实验做好的工程的编译、链接增加到程序构建使用的批处理文件中，用批处理文件**一次性自动**构建实验一、二、三的所有工程。

# 实验过程及结果

## 创建工程

本次实验创建了三个项目，分别实现了 ：

ChildWin 包含使用子窗口的软键盘实现、子窗口ID号显示与颜色设置，

Lab3\_EasyWin 中使用push botton 的控件进行软键盘的实现，创造push button 、static 的矩形框、scollbar 的控件；

ResourcesTest 工程中，自定义鼠标、icon、菜单，位于资源文件中，在顶部的菜单，以及右键的弹出菜单中，均可进行改变窗口鼠标类型的操作；

创建了 ListBox 与edit 、comboBox 控件

## 程序流程和数据模型分析

实现子窗口的软键盘：



实现控件的软键盘：



实现菜单：



## 问题及解决方案

1、子窗口的软键盘实现：（要求1-3）

在子窗口内显示出窗口特定信息（窗口ID号）；设置子窗口标识值，x\*8+y，y应该小于8，共设置3列8行24个子窗口，每个子窗口的键值为 ID 号作为相对 ‘a’的偏移量；

在子窗口显示该子窗口的键值与ID号；使用GetWindowLong(hWnd, GWL\_ID) 得到当前键值，通过GetClientRect接收当前子窗口所在的矩形区域，setText产生相关区域的坐标，drawText绘制矩形框内部的信息，通过ID号，可以计算出当前所在子窗口的键值；

尝试在每个子窗口中绘制特定颜色的矩形：

定义0-255的随机数的生成函数Random\_0\_255；CreateRectRgnIndirect返回即将进行刷色的区域，CreateSolidBrush返回一个已定义颜色的刷子，FillRgn将使整个区域刷满刷子的颜色，刷子的颜色将随机生成；

显示点击的子窗口的键值：

子窗口被点击后，将向父窗口发出 command消息，在这里将更新Child\_ID的取值，同时，出现打印错误。



将字符串的存储向量改为全局变量时，解决此问题。

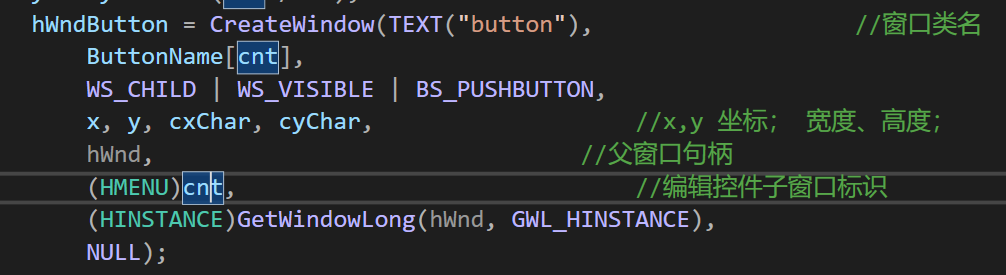
程序演示：





2、控件测试：（要求4-7）

使用 push button 实现软键盘：



共创建26个push botton ，分2行13列进行显示，在创建button 之初，可以设定按键包含的文本信息，储存在二维数组 Button Name 中；

LOWORD(GetDialogBaseUnits()) 可以获得当前系统中的 字符长度，HIWORD(GetDialogBaseUnits())得到系统字符的宽度；根据这两项设置按键的长和宽；

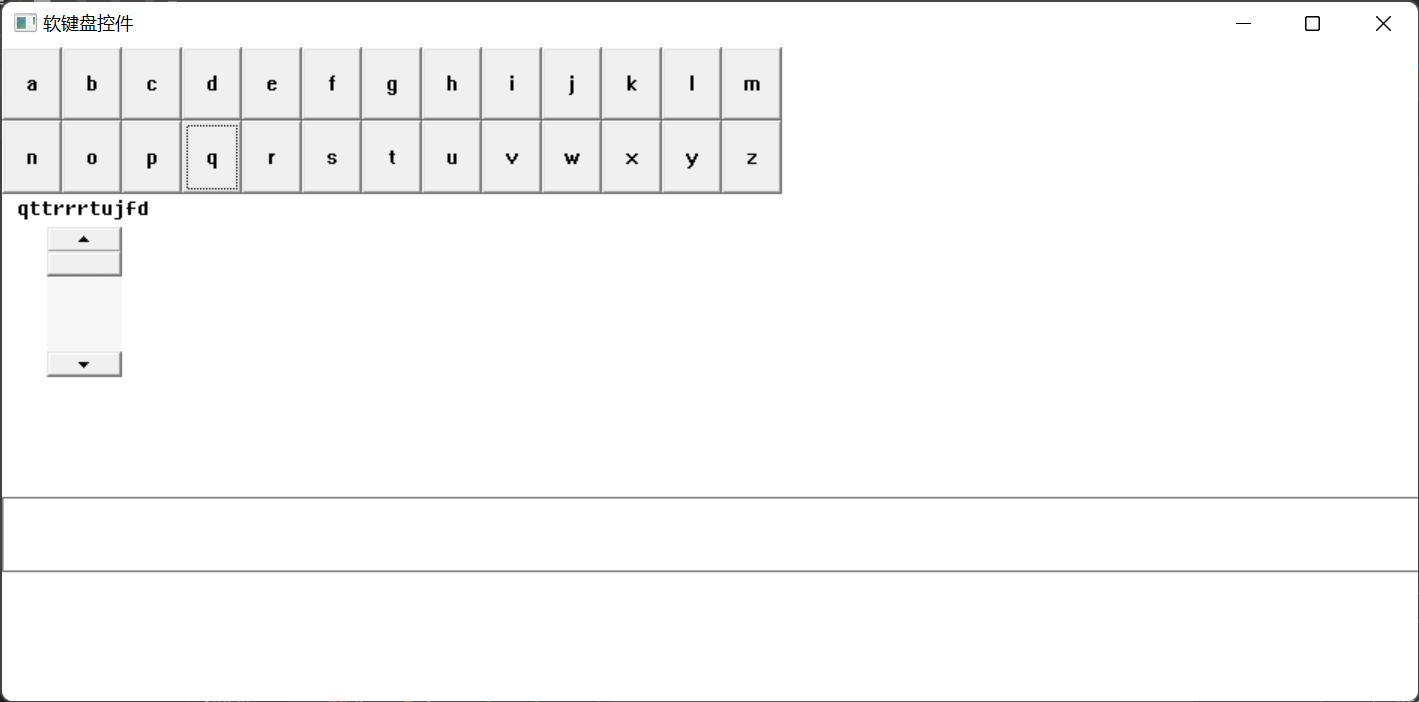
子窗口ID 号为 相对于 ‘a’ 的偏移量；

在WM\_SIZE 中，定义 一个矩形，设置其大小，用来显示按键输入字符串；

按键相应，位于WM\_Command 中， 按键的 ID 号 保存于 wParam 的低字节中，取出后计算得到·键值key ,将key 加入到 szBuffer1 中，使用 TextOut 函数显示；

需要 利用GetDC 与ReleaseDC 获得与释放 当前系统的 hdc ;

实验结果为：



同时设置了其它类型的控件：点击滚动条时，发出WM\_VCSROLL 的信息，将打印滚动条的ID:

case WM\_VSCROLL:

{

int i = GetWindowLong((HWND)lParam, GWL\_ID);

wsprintf(szBuffer2, L"滚动条的ID为%d", i);

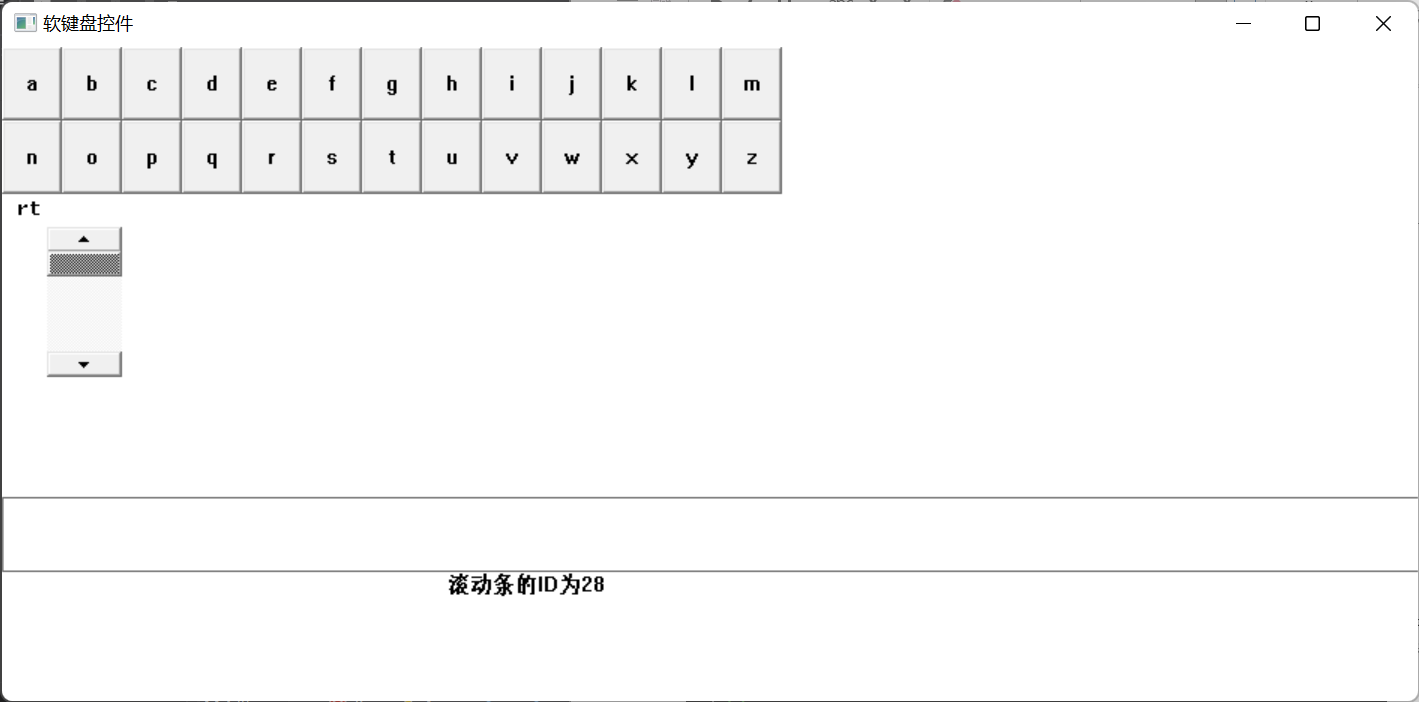
hdc = GetDC(hWnd);

DrawText(hdc, szBuffer2, -1, &rect2, DT\_CENTER | DT\_NOCLIP);

//SetWindowText(hWndScroll, szBuffer2);

//InvalidateRect(hWnd, &rect2, true);

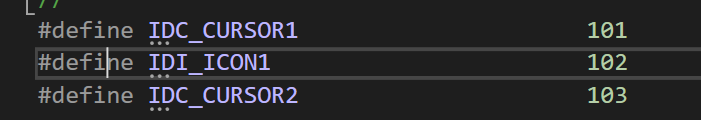
return 0;

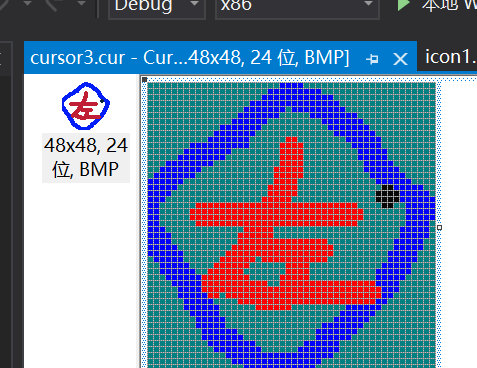


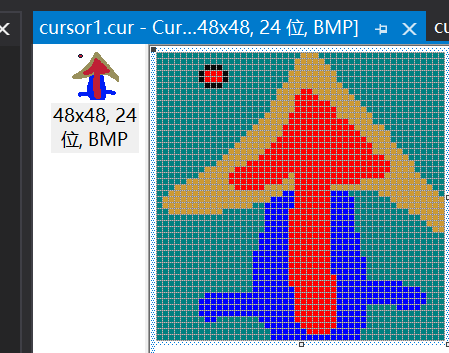
3、资源测试：(8、9)

新建工程 Resource Test ，粘贴建立Edit 的代码，并将创建edit 的代码注释；

资源文件右键新建鼠标、Icon，修改自定生成的位图、图标，仅保留自创的 鼠标、图标，在resource.h 找到：





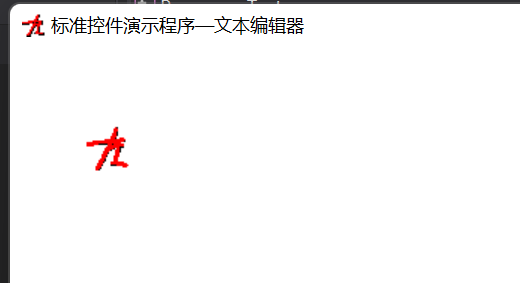


在主窗口中，使用 MAKE INT RESOURCE 的宏，显示自定义的鼠标；右上角选择

设置作用点工具，设置 hot spot;

获得hdc 后，定义 ICON 的实列句柄hIcon,加载待显示的图标：

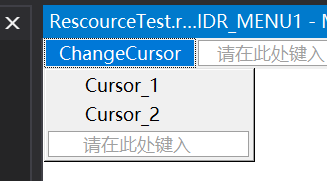
hIcon = LoadIcon(hInstance,MAKEINTRESOURCE( IDI\_ICON1));可以使用 DrawIcon在指定位置显示出自定义的图标;



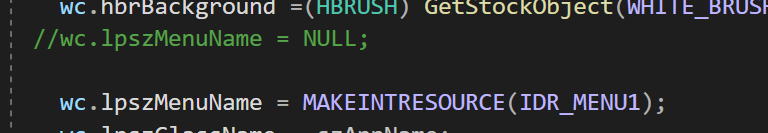
4、添加菜单(10-12):

以同样的方法建立新的鼠标，定义有IDC\_CURSOR2 ；

通过资源的图形化界面 ，建立可视化的停靠窗口顶端的 主菜单；

在右侧属性处，可以修改点击菜单选项时，发出的命令的宏定义名称

通过修改注册类的 方法载入菜单：找到 windows class 中的MeanName 修改：



通过创建窗口 加载菜单:

修改CreateWindow 中的第九个参数，填入Menu的句柄；

窗口的响应代码：

hInstance =(HINSTANCE) GetWindowLong(hWnd, GWL\_HINSTANCE);

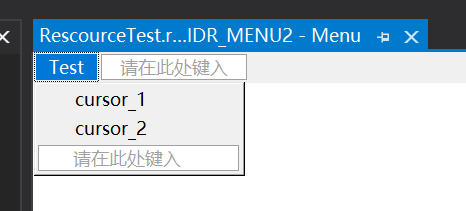
获取主窗口的实例句柄；菜单元素被点击后，将产生 WM\_COMMAND的消息，在wParam 得低字节，将选项的标志宏定义 取出，在 resource.h 中 可以查到：

hCursor = LoadCursor(hInstance, MAKEINTRESOURCE(IDC\_CURSOR1));

SetClassLong(hWnd, GCL\_HCURSOR, (long)hCursor);

加载自定义光标，使用 SetClassLong 改变系统使用的 光标；以后的修改方法完全相同；

产生右键的弹出菜单：

新建另外一个菜单，

在点击鼠标右键时显示；

case WM\_RBUTTONDOWN:

hMenu = LoadMenu(hInstance, MAKEINTRESOURCE(IDR\_MENU2));

point.x = LOWORD(lParam);

point.y = HIWORD(lParam);

ClientToScreen(hWnd, &point);

TrackPopupMenu(hMenu, TPM\_RIGHTBUTTON, point.x, point.y, 0, hWnd, NULL);

lParam 中取出 当前鼠标相对 全屏幕的 位置， ClientToScreen(hWnd, &point) 将其转换为相对当前窗口的 坐标值；TrackPopupMenu 显示 右键菜单；

创建ListBox 控件：

hWndList = CreateWindow(TEXT("listbox"),

NULL,

WS\_CHILD | WS\_VISIBLE | LBS\_STANDARD,

500, 50, 200, 200,

hWnd, (HMENU)2, hInstance, NULL);

//ShowWindow(hListBox,SW\_SHOW);

SendMessage(hWndList, LB\_ADDSTRING, 0,(LPARAM)L"201900800413");

SendMessage(hWndList, LB\_ADDSTRING, 0, (LPARAM)L"通信一班");

SendMessage(hWndList, LB\_ADDSTRING, 0, (LPARAM)L"宇宙无敌超级大帅哥正是在下");

SendMessage 向控件中添加字符串；

通过 课上的讲解后，进一步对菜单的功能进行拓展：

case WM\_INITMENUPOPUP:

CheckMenuItem((HMENU)wParam, ID\_WIN\_EDIT, bEditShow ? MF\_CHECKED : MF\_UNCHECKED);

CheckMenuItem((HMENU)wParam, ID\_WIN\_LIST, bListShow ? MF\_CHECKED : MF\_UNCHECKED);

在菜单的选择状态前打上一个对号，实现隐藏与显示某一项窗口的功能；

Init Menu Popup 在下拉菜单 活跃前，允许程序在菜单显示之前修改菜单；

CheckMenuItem 转换Menu 的 是否 有被选择（在前面打一个对号表示被选择）状态；

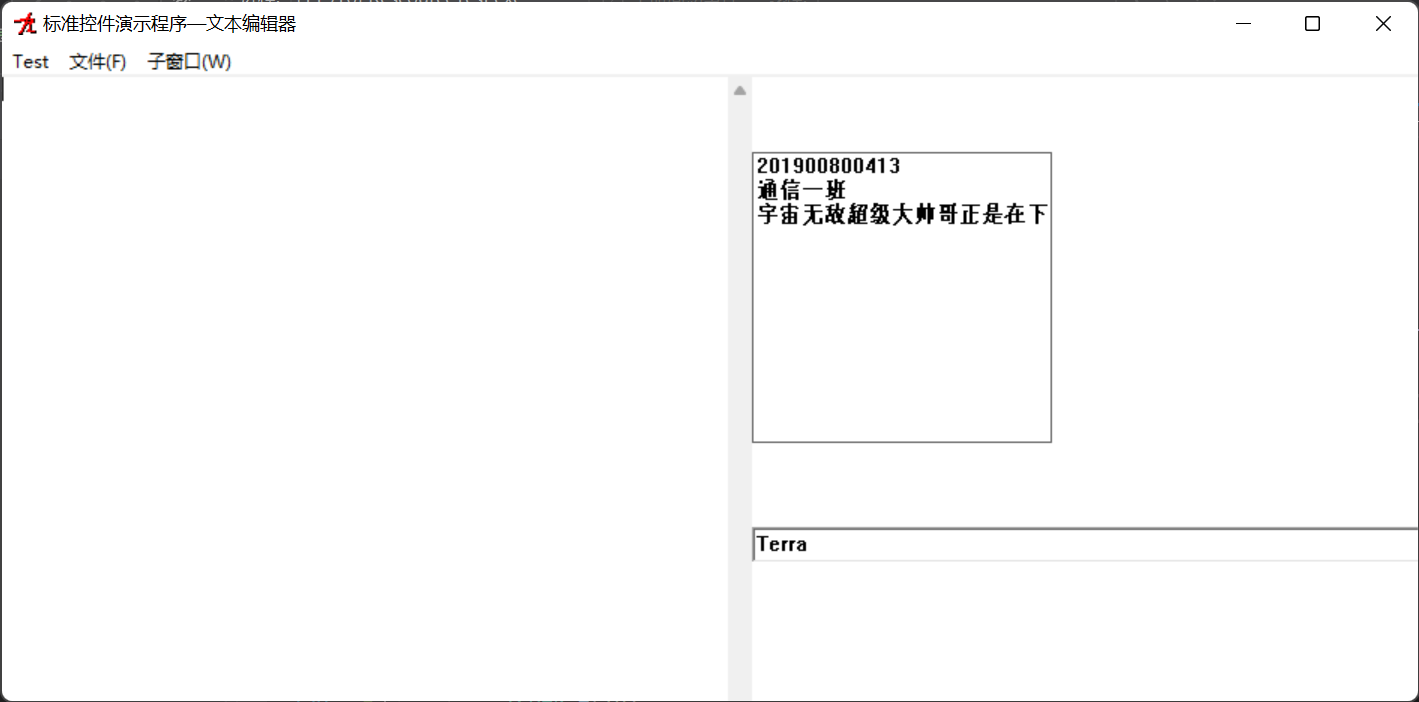
创建ComboBox 的控件：

hWndComboBox = CreateWindow(TEXT("COMBOBOX"), TEXT(""),

CBS\_DROPDOWN | CBS\_HASSTRINGS | WS\_CHILD | WS\_OVERLAPPED | WS\_VISIBLE,

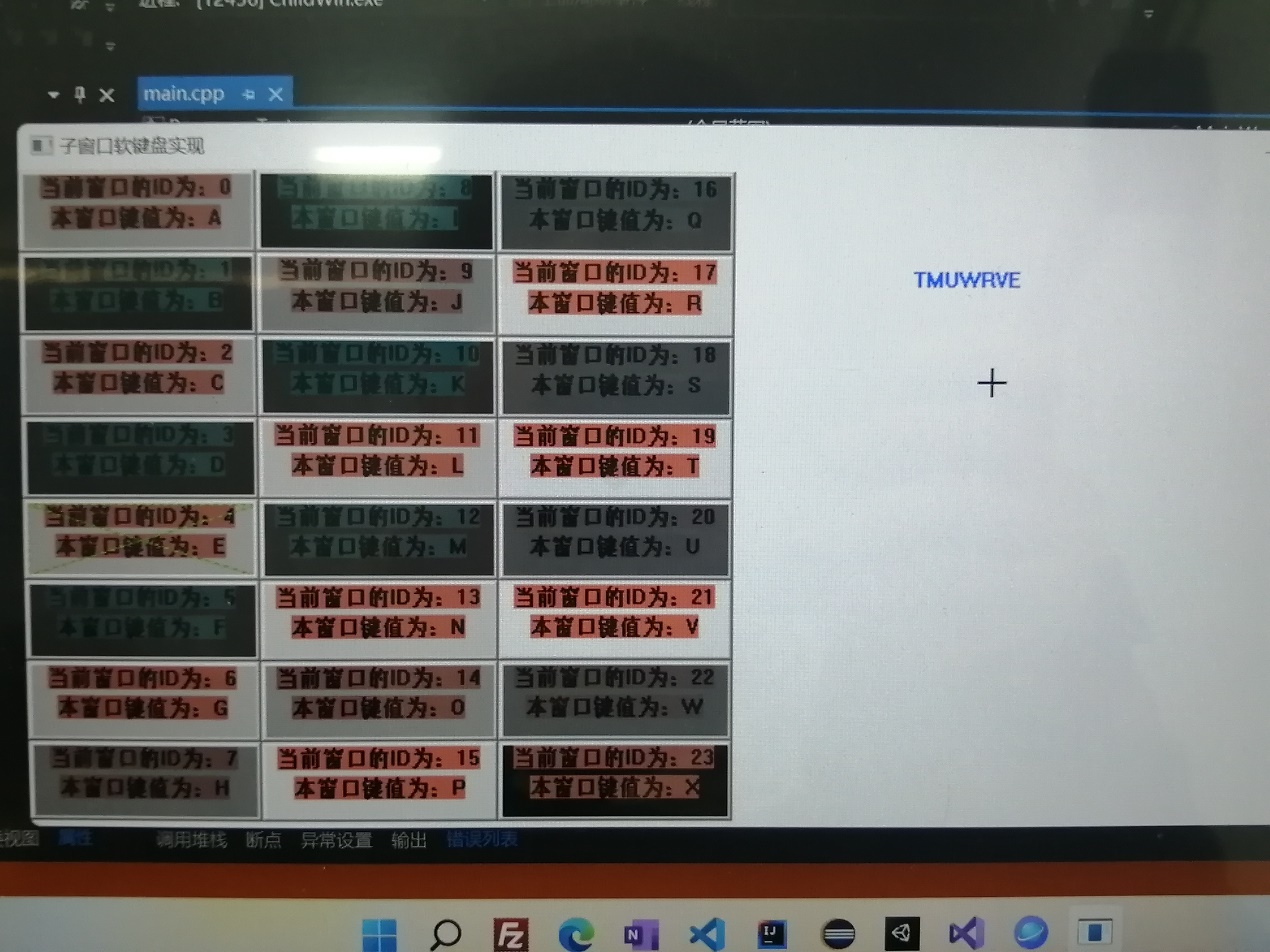
500, 300, 700, 800, hWnd, (HMENU) 3, (HINSTANCE)GetWindowLong(hWnd, GWL\_HINSTANCE),NULL);

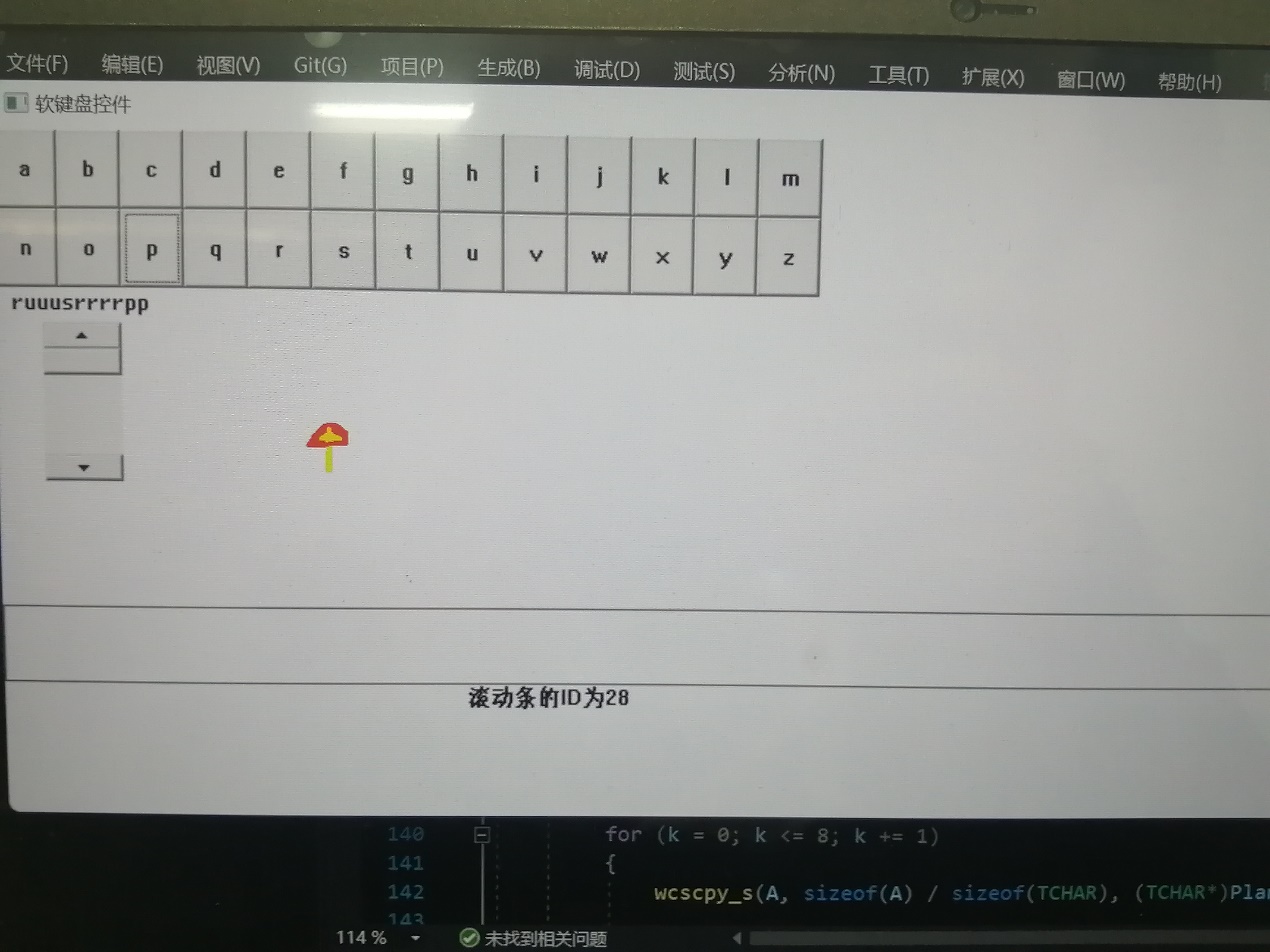
运行后：

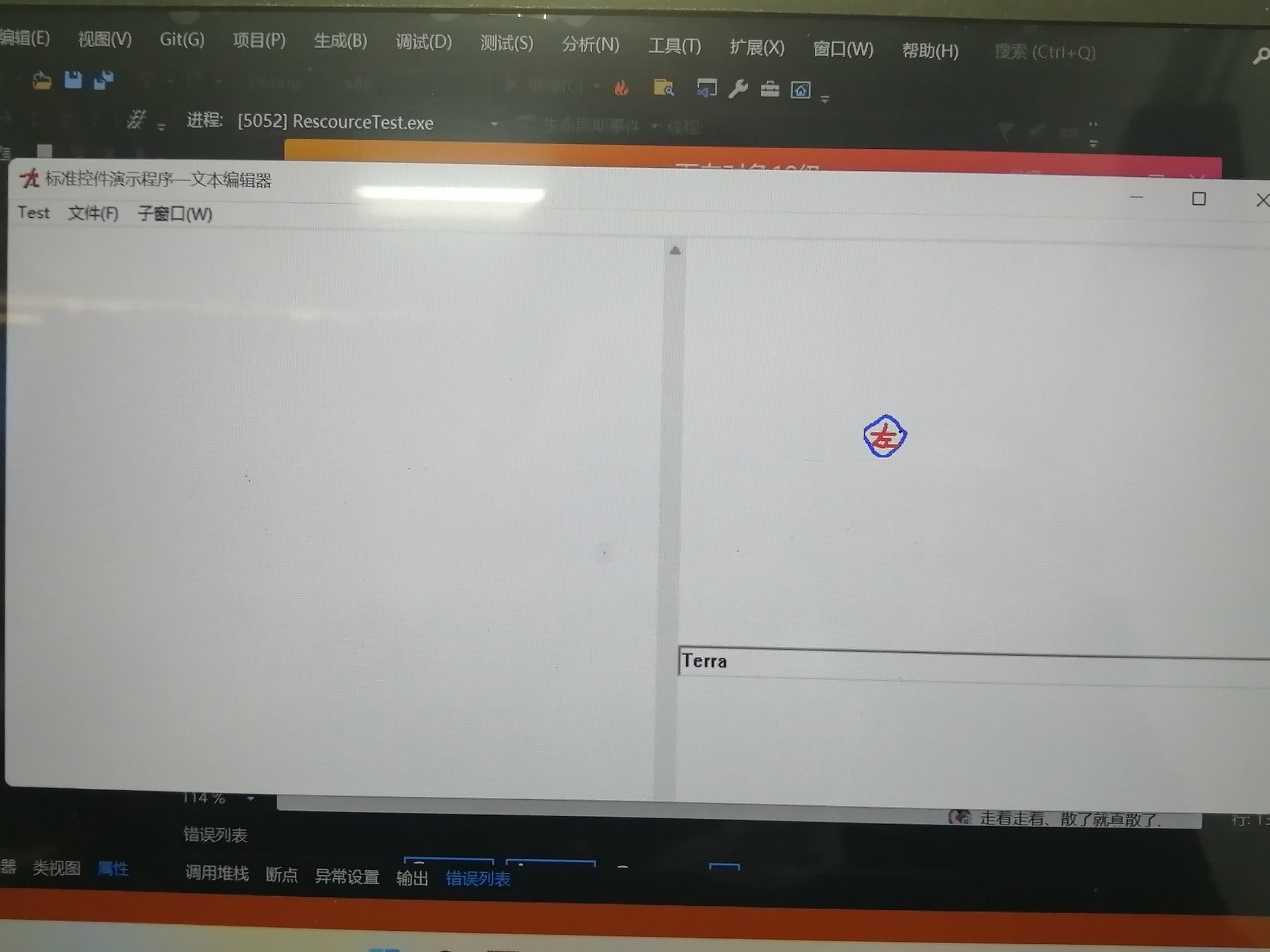


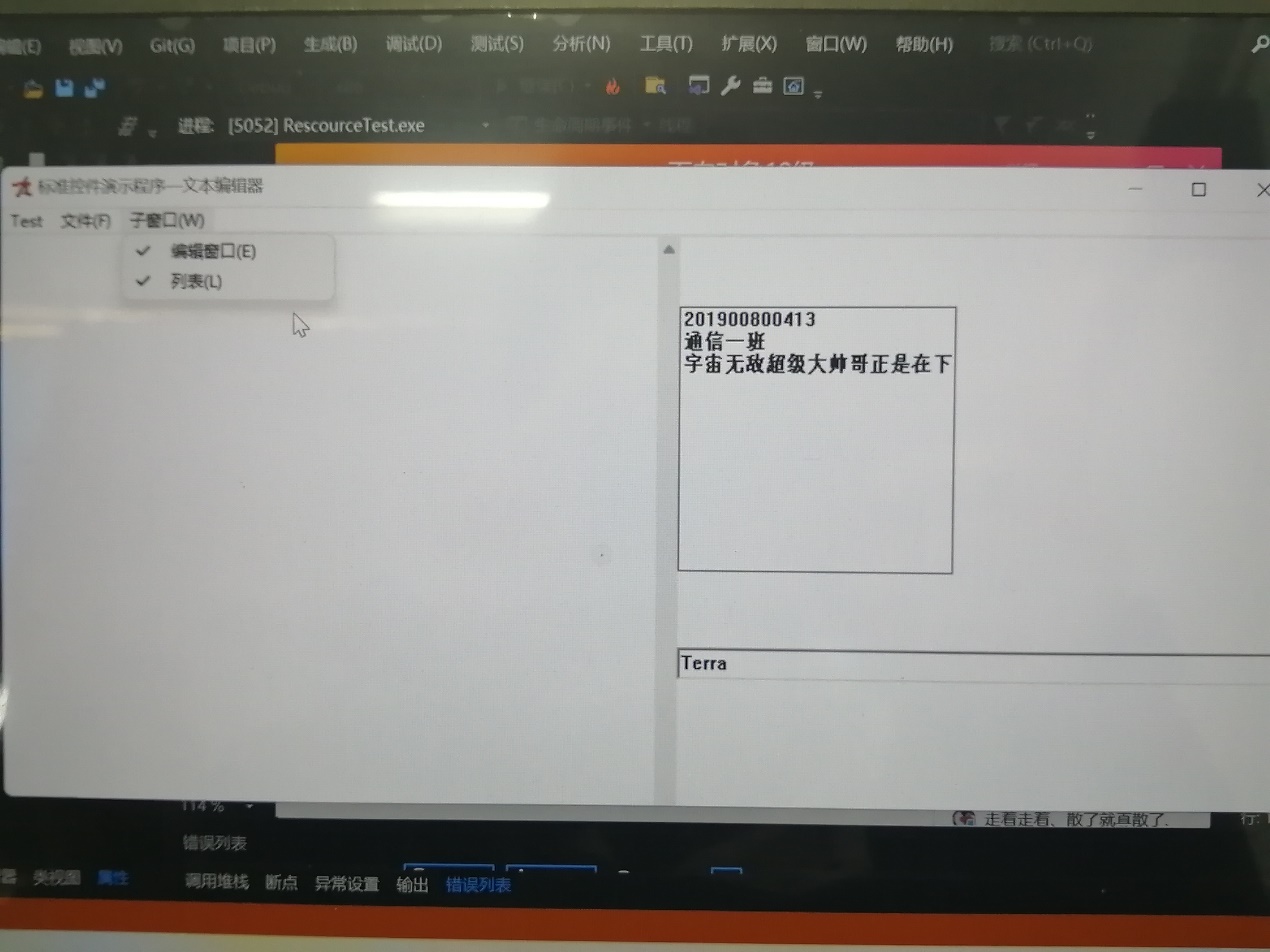
至此，所有功能均实现 完成；

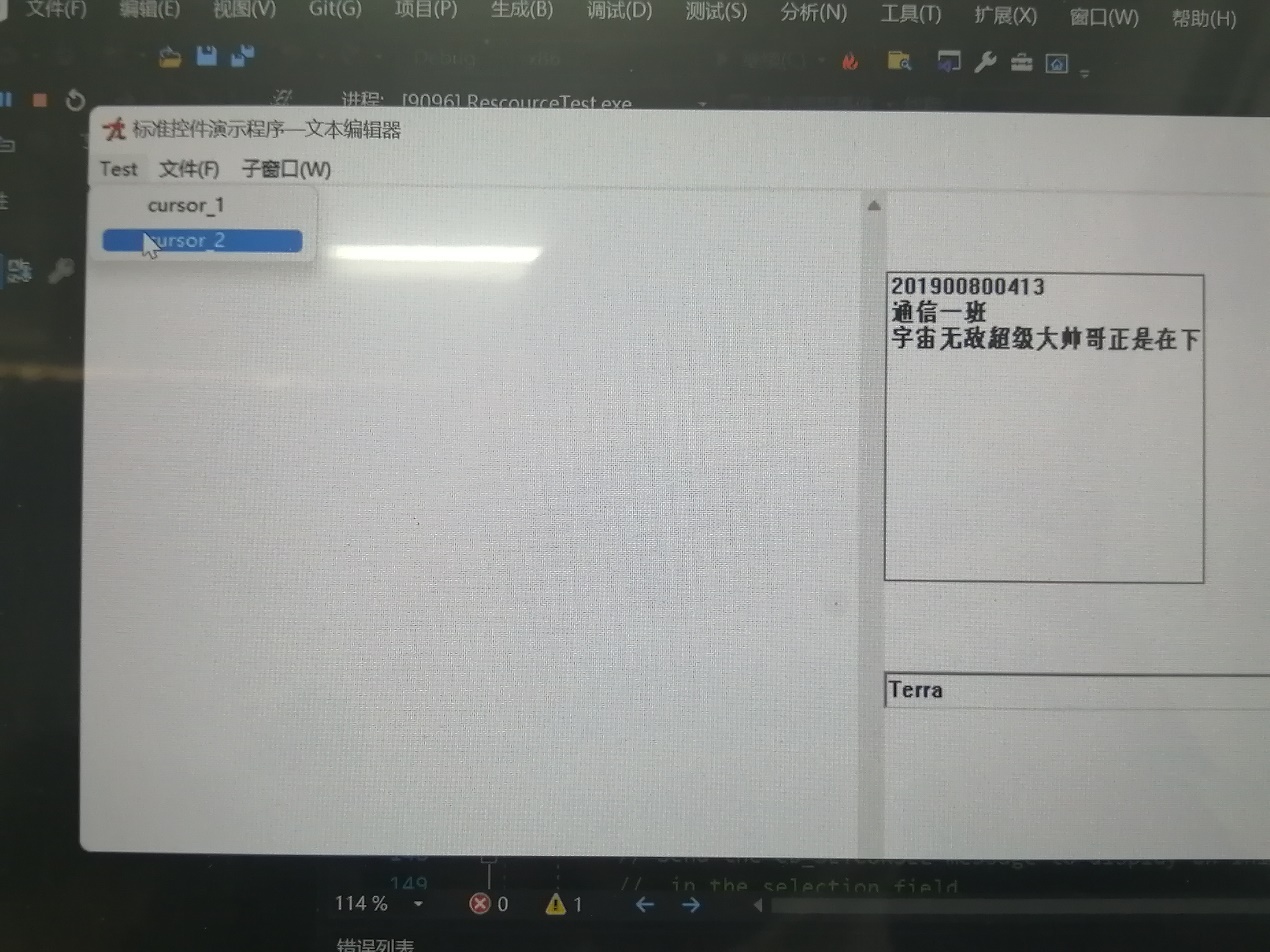
对所有系统整体演示：









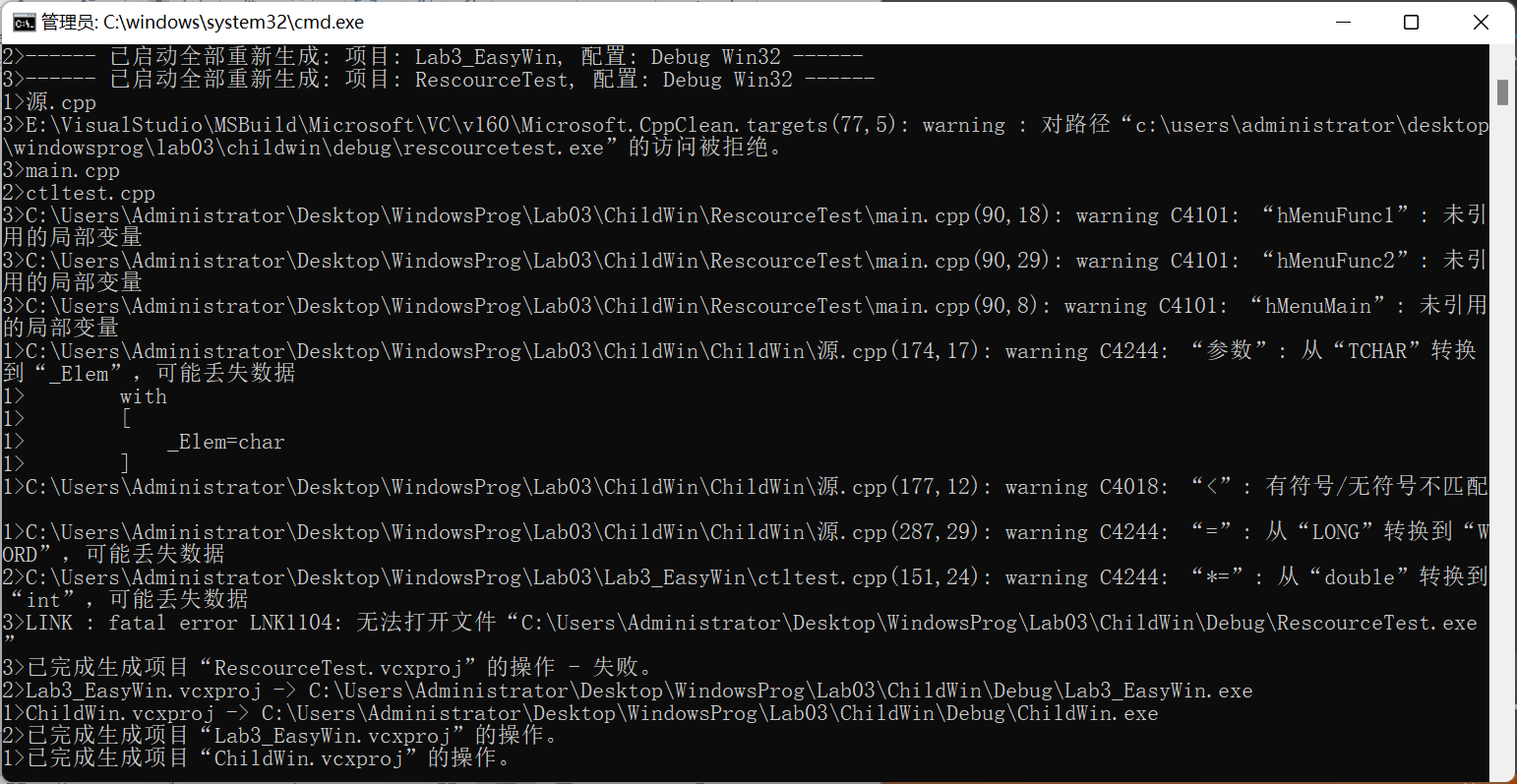


编写批处理文件：

cd \

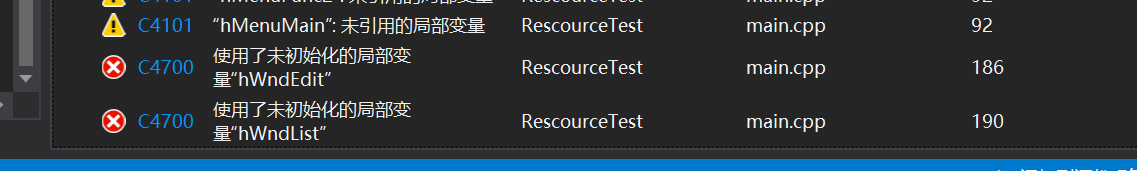
C:\>cd C:\Users\stu\Desktop\Lab\_01

C:\Users\stu\Desktop\Lab\_01>"C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\Common7\IDE\devenv.com" lab01.sln /rebuild



由于本实验使用的 三个工程均位于同一个文件目录下，将其中的一个工程进行编译，成功完成对三个工程的总编译；

# 实验总结

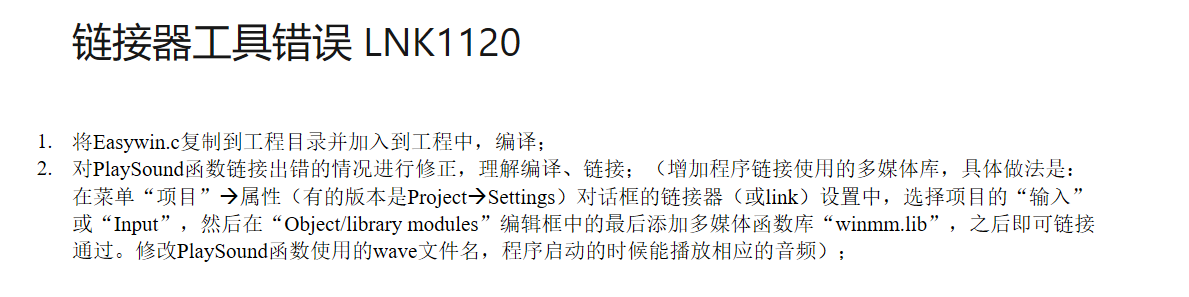
1、

此时，在WM\_CREATE 中均进行了初始化；

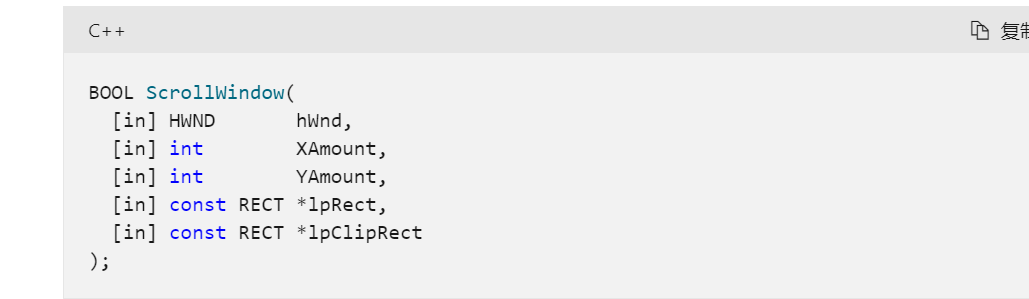
解决方法为：

定义 静态的HWND变量，

2、复制代码时，产生的链接器错误：



3、



设置某一区域，产生水平、垂直的滚动条，同时指定所在的矩形区域；

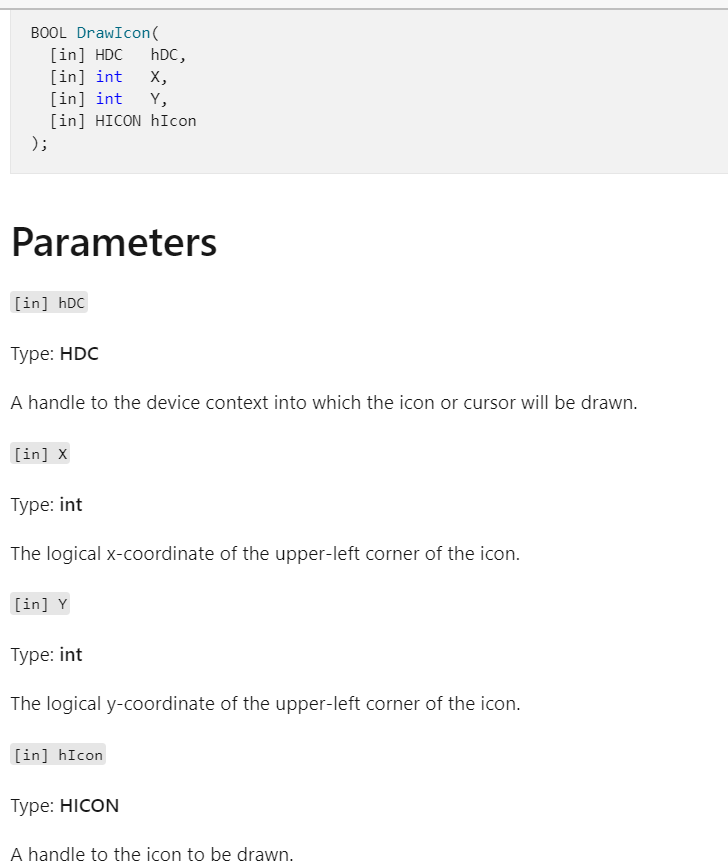
4、HDC 的理解



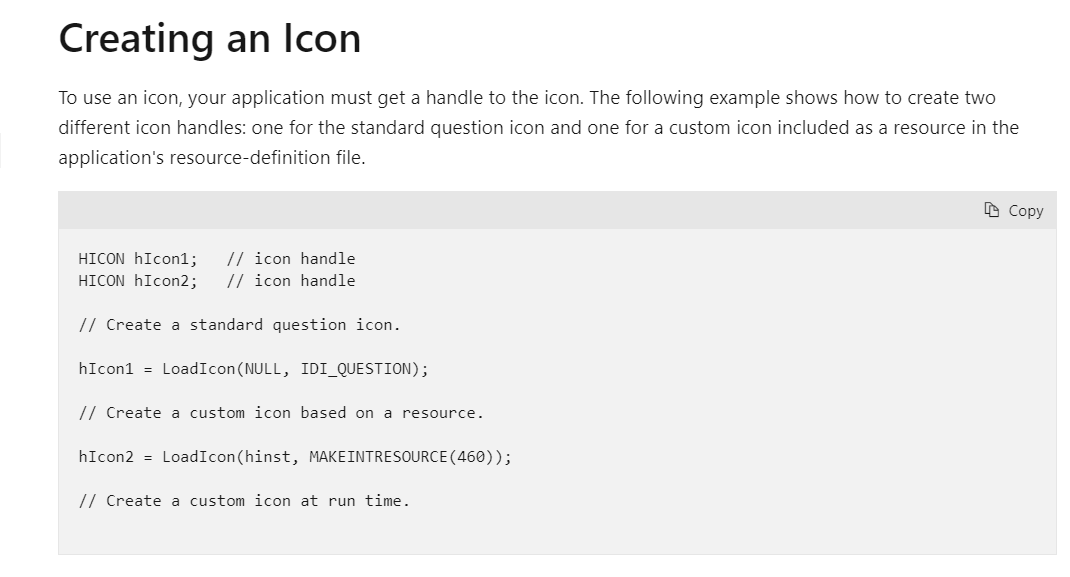
5、TextOut 函数

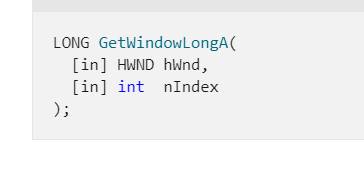


Draw Icon函数

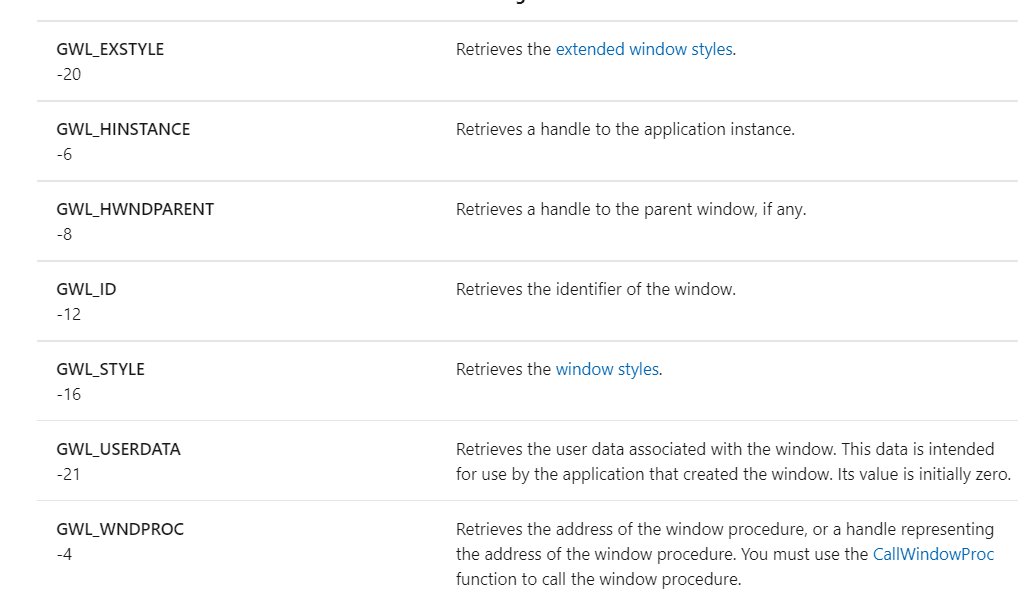


创建Icon





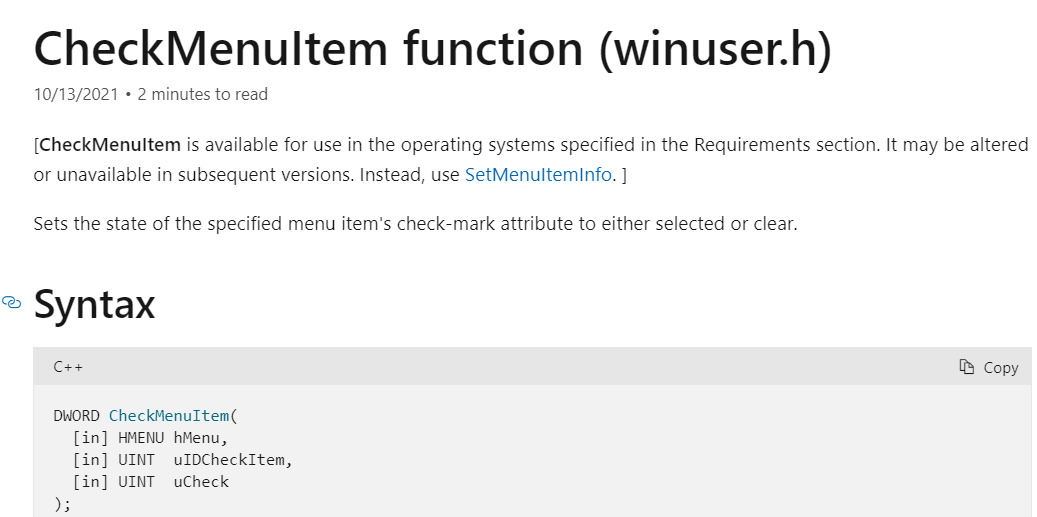
nIndex 的相关命令：



在菜单显示之前修改相关选项的状态：



ChechItem 是否选择为 选择状态（在选项前 打一个对号）





# 附录 关键代码

1、子窗口的标识值 ，

for (x = 0; x < COL\_NUM; x++)

for (y = 0; y < ROW\_NUM; y++)

{

nChildWin\_ID = x << 3 | y; //子窗口标识值

hChildWnd[x][y] = CreateWindow(szChildName, NULL,

WS\_CHILDWINDOW | WS\_DLGFRAME | WS\_VISIBLE,

0, 0, 0, 0,

hWnd,

(HMENU)(nChildWin\_ID),//创建子窗口ID号

//0,

hInst, NULL);

}

生成随机颜色：

int Random\_0\_255(void)

{

SYSTEMTIME systime;

GetSystemTime(&systime);

//为rand()得到随机数作准备;

srand(systime.wMilliseconds);

return rand() \*2021 % 255;

}

子窗口的颜色显示：

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

HBRUSH hBrush;

RECT clientRect;

RECT textRect;

HRGN bgRgn;

HPEN hPen;

TCHAR szBuffer[30];

Current\_ID = GetWindowLong(hWnd, GWL\_ID);

GetClientRect(hWnd, &clientRect);

bgRgn = CreateRectRgnIndirect(&clientRect);

hBrush = CreateSolidBrush(RGB(Random\_0\_255(), Random\_0\_255(), Random\_0\_255()));

SelectObject(hdc, hBrush);

//选用

FillRgn(hdc, bgRgn, hBrush);

hPen = CreatePen(PS\_DOT, 1, RGB(0, 255, 0));

SelectObject(hdc, hPen);

SetBkColor(hdc, RGB(Random\_0\_255(),100 ,100));

//Rectangle(hdc, 0, 0, 150, 200);

SetRect(&textRect, 0, 0, 150,10);

wsprintf(szBuffer,L"当前窗口的ID为：%d",Current\_ID);

DrawText(hdc, szBuffer, -1, &textRect, DT\_CENTER | DT\_NOCLIP);

SetRect(&textRect, 0, 20, 150, 30);

wsprintf(szBuffer, L"本窗口键值为：%c", Current\_ID+'A');

DrawText(hdc, szBuffer, -1, &textRect, DT\_CENTER | DT\_NOCLIP);

//反相绘制被命中的子窗口

if (GetWindowLong(hWnd, 0))

{

GetClientRect(hWnd, &rect);

//用黑色画矩形rect的左边线和上边线

MoveToEx(hdc, 0, rect.bottom - 1, NULL);

LineTo(hdc, 0, 0);

LineTo(hdc, rect.right - 1, 0);

MoveToEx(hdc, 1, 1, NULL);

LineTo(hdc, rect.right - 1, rect.bottom - 1);

MoveToEx(hdc, 1, rect.bottom - 1, NULL);

LineTo(hdc, rect.right - 1, 1);

//用白色画矩形rect的右边线和边线

SelectObject(hdc, GetStockObject(WHITE\_PEN));

MoveToEx(hdc, rect.right - 1, 0, NULL);

LineTo(hdc, rect.right - 1, rect.bottom - 1);

LineTo(hdc, 0, rect.bottom - 1);

SelectObject(hdc, GetStockObject(BLACK\_PEN));

}

EndPaint(hWnd, &ps);

return 0;

父窗口显示 来自 子窗口点击的键值：

case WM\_COMMAND:

{

//子窗口被命中的通知消息，菜单、子窗口的消息接收

//当子窗口被命中时发送此消息通知主窗口，以便由主窗口确定是否命中另一个子窗口，

//若是，则发送消息通知子窗口函数改变前一次被命中子窗口的状态，

if (LOWORD(wParam) != nChildWin\_ID)

{

SendMessage((HWND)hChildWnd[nCol][nRow], WM\_CHILDWIN, 0, 0L);

//更新的被命中的子窗口

nChildWin\_ID = LOWORD(wParam);

nCol = nChildWin\_ID >> 3;

nRow = (nChildWin\_ID > 8) ? (nChildWin\_ID % 8) : nChildWin\_ID;

}

key = nChildWin\_ID + 'A';

gStrMinWIN += key;

int i = 0;

for (; i < gStrMinWIN.size(); i++)

{

szBuffer[i] = gStrMinWIN[i];

}

szBuffer[i] = '\0';

InvalidateRect(hWnd, NULL, TRUE);

return 0;

}

2、创建26个push botton：

cxChar = LOWORD(GetDialogBaseUnits());

cyChar = HIWORD(GetDialogBaseUnits());

cxChar \*= 5;

cyChar \*= (5 \* 0.618);

for (int cnt = 0; cnt < 26; cnt++)

{

int x, y;

x = cxChar \* (cnt % 13);

y = cyChar \* (cnt / 13);

hWndButton = CreateWindow(TEXT("button"), //窗口类名

ButtonName[cnt],

WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

x, y, cxChar, cyChar, //x,y 坐标； 宽度、高度；

hWnd, //父窗口句柄

(HMENU)cnt, //编辑控件子窗口标识

(HINSTANCE)GetWindowLong(hWnd, GWL\_HINSTANCE),

NULL);

}

按键响应：

WORD key;

key = LOWORD(wParam) + 'a';

int index = lstrlen(szBuffer);

szBuffer[index] = key;

szBuffer[index + 1] = '\0';

ScrollWindow(hWnd, 0, -cyChar, &rect1, &rect1);

hdc = GetDC(hWnd);

SelectObject(hdc, GetStockObject(SYSTEM\_FIXED\_FONT));

TextOutW(hdc, 10, 100, szBuffer, lstrlen(szBuffer));

ReleaseDC(hWnd, hdc);

ValidateRect(hWnd, &rect1);

4、

通过注册窗口类加载菜单：

wc.lpszMenuName = MAKEINTRESOURCE(IDR\_MENU1);

通过创建窗口 加载菜单:

hwnd = CreateWindow(szAppName,

TEXT("标准控件演示程序—文本编辑器"),

WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

NULL,

// NULL,

LoadMenu(hInstance,MAKEINTRESOURCE(IDR\_MENU1)), //创建窗口时，载入菜单；

hInstance,

NULL);

窗口的响应代码与右键浮动菜单的实现：

hInstance =(HINSTANCE) GetWindowLong(hWnd, GWL\_HINSTANCE);

hIcon = LoadIcon(hInstance,MAKEINTRESOURCE( IDI\_ICON1));

hdc = GetDC(hWnd);

switch (message)

{

case WM\_CREATE:

//hMenuMain = LoadMenu(hInstance, L"Main\_Menu");

//hMenuFunc1 = LoadMenu(hInstance, L"Corsor1");

//hMenuFunc2 = LoadMenu(hInstance, L"Corsor2");

//SetMenu(hWnd, hMenuMain);

return 0;

case WM\_RBUTTONDOWN:

hMenu = LoadMenu(hInstance, MAKEINTRESOURCE(IDR\_MENU2));

point.x = LOWORD(lParam);

point.y = HIWORD(lParam);

ClientToScreen(hWnd, &point);

TrackPopupMenu(hMenu, TPM\_RIGHTBUTTON, point.x, point.y, 0, hWnd, NULL);

//MessageBox(hWnd, L"Hello", L"test", MB\_OKCANCEL);

return 0;

//case WM\_SETFOCUS:

// SetFocus(hWndEdit);

// return 0;

//case WM\_PAINT:

// DrawIcon(hdc, 50, 50, hIcon);

//case WM\_SIZE:

// MoveWindow(hWndEdit, 0, 0, LOWORD(lParam), HIWORD(lParam), TRUE);

// return 0;

case WM\_COMMAND: // 响应编辑控件通知消息

switch (LOWORD(wParam))

{

case ID\_MYCURSOR1:

hCursor = LoadCursor(hInstance, MAKEINTRESOURCE(IDC\_CURSOR1));

SetClassLong(hWnd, GCL\_HCURSOR, (long)hCursor);

break;

case ID\_MYCURSOR2:

hCursor = LoadCursor(hInstance, MAKEINTRESOURCE(IDC\_CURSOR3));

SetClassLong(hWnd, GCL\_HCURSOR, (long)hCursor);

break;

case ID\_cursor\_1:

hCursor = LoadCursor(hInstance, MAKEINTRESOURCE(IDC\_CURSOR1));

SetClassLong(hWnd, GCL\_HCURSOR, (long)hCursor);

break;

case ID\_cursor\_2:

hCursor = LoadCursor(hInstance, MAKEINTRESOURCE(IDC\_CURSOR3));

SetClassLong(hWnd, GCL\_HCURSOR, (long)hCursor);

break;

//case IDM\_MAIN:

default:

break;

}

return 0;

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

}

5、

创建ListBox

hWndList = CreateWindow(TEXT("listbox"),

NULL,

WS\_CHILD | WS\_VISIBLE | LBS\_STANDARD,

500, 50, 200, 200,

hWnd, (HMENU)2, hInstance, NULL);

//ShowWindow(hListBox,SW\_SHOW);

SendMessage(hWndList, LB\_ADDSTRING, 0,(LPARAM)L"201900800413");

SendMessage(hWndList, LB\_ADDSTRING, 0, (LPARAM)L"通信一班");

SendMessage(hWndList, LB\_ADDSTRING, 0, (LPARAM)L"宇宙无敌超级大帅哥正是在下");

选择状态转换：

case WM\_INITMENUPOPUP:

CheckMenuItem((HMENU)wParam, ID\_WIN\_EDIT, bEditShow ? MF\_CHECKED : MF\_UNCHECKED);

CheckMenuItem((HMENU)wParam, ID\_WIN\_LIST, bListShow ? MF\_CHECKED : MF\_UNCHECKED);

return 0;

显示与隐藏窗口：

case ID\_WIN\_EDIT:

ShowWindow(hWndEdit, bEditShow ? SW\_HIDE :SW\_SHOW);

bEditShow = !bEditShow;

break;

case ID\_WIN\_LIST:

ShowWindow(hWndList, bListShow ? SW\_HIDE : SW\_SHOW);

bListShow = !bListShow;

break;

Resource.h

#define IDC\_CURSOR1 101

#define IDI\_ICON1 102

#define IDC\_CURSOR2 103

#define IDR\_MENU1 104

#define IDC\_CURSOR3 105

#define IDR\_MENU2 107

#define ID\_CHANGECURSOR\_FUNC1 40001

#define ID\_CHANGECURSOR\_FUNC2 40002

#define ID\_CHANGECURSOR\_COUSOR 40003

#define ID\_CHANGECURSOR\_CURSOR 40004

#define ID\_MYCURSOR1 40005

#define ID\_MYCURSOR2 40006

#define ID\_cursor\_1 40009

#define ID\_cursor\_2 40010