概念题

1. 请简述事件驱动的程序流程控制与传统的流程控制的区别。
   程序的任何一个动作都由某个事件（用户的键盘、鼠标、菜单等操作）激发；
   每个事件都会向应用程序发送一些消息；
   每个应用程序都有一个消息队列，系统把属于应用程序的消息放入消息队列；
   大部分的消息都关联到某个窗口；
   应用程序不断从消息队列中获取消息并调用相应窗口的消息处理函数处理消息。

2. 基于Windows API的事件驱动程序设计中，主函数WinMain的主要功能是什么？
   注册窗口类（窗口的基本信息）；
   创建应用程序的主窗口（其它窗口等到需要时再创建）；
   进入消息循环，直到接收到WM_QUIT消息时，消息循环结束。

3. 请简述微软MFC的作用。
   提供一些类描述应用中对象的基本功能，应用程序继承这些类实现各自的功能；
   提供对基于“文档－视”结构应用框架的支持。

编程题

1. 基于Windows API事件驱动设计一个窗口。（仅展示自己实现的代码）

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
    switch (message)
    {
    case WM_COMMAND:
        {
            int wmId = LOWORD(wParam);
            // 分析菜单选择:
            switch (wmId)
            {
            case ID_Plus:
                DialogBox(hInst, MAKEINTRESOURCE(IDD_PlusBox), hWnd, Plus);
                break;
            case ID_Minus:
                DialogBox(hInst, MAKEINTRESOURCE(IDD_MinusBox), hWnd, Minus);
                break;
            case ID_Multiply:
                DialogBox(hInst, MAKEINTRESOURCE(IDD_Multibox), hWnd, Multiply);
                break;
            case ID_Divide:
                DialogBox(hInst, MAKEINTRESOURCE(IDD_DivideBox), hWnd, Divide);
```

```
                break;
            case ID_COORD:
                DialogBox(hInst, MAKEINTRESOURCE(IDD_CoordBox), hWnd, C_coord);
                break;
            case ID_Letter:
                DialogBox(hInst, MAKEINTRESOURCE(IDD_LetterBox), hWnd, Letter);
            case IDM_ABOUT:
                DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
                break;
            case IDM_EXIT:
                DestroyWindow(hWnd);
                break;
            default:
                return DefWindowProc(hWnd, message, wParam, lParam);
            }
        }
        break;
    case WM_DESTROY:
        PostQuitMessage(0);
        break;
    default:
        return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}

INT_PTR CALLBACK Plus(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    UNREFERENCED_PARAMETER(lParam);
    int left = 0, right = 0;
    left = GetDlgItemInt(hDlg, IDC_EDIT1, NULL, TRUE);
    right = GetDlgItemInt(hDlg, IDC_EDIT2, NULL, TRUE);
    switch (message)
    {
    case WM_INITDIALOG:
        return (INT_PTR)TRUE;

    case WM_COMMAND:
        if (LOWORD(wParam) == IDOK) {
            SetDlgItemInt(hDlg, IDC_EDIT3, left + right, TRUE);
        }
        else if (LOWORD(wParam) == IDCANCEL)
        {
            EndDialog(hDlg, LOWORD(wParam));
```

```
            return (INT_PTR)TRUE;
        }
        break;
    }
    return (INT_PTR)FALSE;
}

//减、乘、除同上，代码重复度较大不重复放了。

INT_PTR CALLBACK Letter(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    UNREFERENCED_PARAMETER(lParam);
    WCHAR ch;
    ch = GetDlgItemText(hDlg, IDC_EDIT1, NULL, TRUE);
    switch (message)
    {
    case WM_INITDIALOG:
        return (INT_PTR)TRUE;

    case WM_COMMAND:
        if (LOWORD(wParam) == IDOK) {
            SetDlgItemInt(hDlg, IDC_EDIT2, ch, TRUE);
        }
        else if (LOWORD(wParam) == IDCANCEL)
        {
            EndDialog(hDlg, LOWORD(wParam));
            return (INT_PTR)TRUE;
        }
        break;
    }
    return (INT_PTR)FALSE;
}

INT_PTR CALLBACK C_coord(HWND hDlg, UINT message, WPARAM wParam, LPARAM
lParam)
{
    double points[100][2];
    int count = 0;
    UNREFERENCED_PARAMETER(lParam);
    switch (message)
    {
    case WM_INITDIALOG:
        return (INT_PTR)TRUE;
```

```
        case WM_LBUTTONDOWN:
            points[count][0] = LOWORD(lParam);
            points[count][1] = HIWORD(lParam);
            SetDlgItemInt(hDlg, IDC_EDIT1, points[count][0], TRUE);
            SetDlgItemInt(hDlg, IDC_EDIT2, points[count][1], TRUE);
            count += 1;
            break;

        case WM_DISTANCE:
            double distance = 0;
            for (int i = 0; i < count; i++) {
                for (int j = 0; j < count; j++) {
                    if (i == j)
                        continue;
                    double temp = pow(points[i][0] - points[j][0]) + pow(points[i][1] -
points[j][1]);
                    if (temp > distance)
                        distance = temp;
                }
            }
            SetDlgItemInt(hDlg, IDC_EDIT3, sqrt(distance), TRUE);
            break;
        case WM_COMMAND:
            if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)
            {
                EndDialog(hDlg, LOWORD(wParam));
                return (INT_PTR)TRUE;
            }
            break;
        }
        return (INT_PTR)FALSE;
}
```

2. 请基于MFC的事件驱动重新实现该上题中的窗口。（代码略）
   比较并说明基于API和基于MFC这两种实现方式的区别。

基于API的实现思路清晰，但是具体实现较烦琐，思维量小而代码量大。
基于MFC的实现思路不太清晰，WinMain函数等封装在类中，代码量小但思维量大。
基于API和基于MFC的实现在特点上符合面向过程编程和面向对象编程的区别。