윈도우 프로그래밍이란 사용자가 발생시키는 이벤트에 대한 메시지를 처리하는 것
메시지 기반 프로그래밍, 이벤트 기반 프로그래밍

윈도우 프로그래밍하는 방법 (wind32 SDK Software Development Kit)
MFC(Mocrosoft Foundation Class)

실습 코드
이 코드는 RegisterClassEx, CreateWindow, WndProc 같은 저수준 API를 직접 사용
메시지 루프와 이벤트 처리까지 모두 수동으로 작성 구조가 매우 단순.
하지만 코드가 장황해지고, 실제 개발에서는 생산성이 낮음.

```
#include<Windows.h> //윈도우 어플 작성하는데 필요한 API 함수.
LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParm); //?

//메인함수
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstace, LPTSTR lpszCmdLine, int nCmdShow) {
    HWND hwnd; //윈도우를 담당할 핸들
    MSG msg; //메시지 정보를 담는 구조체
    WNDCLASSEX WndClass; //윈도우 클래스 구조체

    //1. 기본 외관 특성 정의
    WndClass.cbSize = sizeof(WNDCLASSEX); //구조체 크기
    WndClass.style = CS_HREDRAW | CS_VREDRAW | CS_DBLCLKS; //클래스 스타일
    WndClass.lpfnWndProc = WndProc; // 윈도우 프로시저
    WndClass.cbClsExtra = 0;//윈도우클래스 데이터 영역
    WndClass.cbWndExtra = 0; //윈도우의 데이터 영역
    WndClass.hInstance = hInstance; //인스턴스 핸들
    WndClass.hIcon = LoadIcon(NULL, IDI_APPLICATION); //아이콘 핸들
    WndClass.hCursor = LoadCursor(NULL, IDC_ARROW); //커서 핸들
    WndClass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH); //배경 브러시 핸들.
    WndClass.lpszMenuName = NULL; //메뉴 이름
    WndClass.lpszClassName = "EasyText"; //윈도우 클래스 이름
    WndClass.hIconSm = 0; //기본적인 작은 아이콘

    //윈도우 클래스를 등록한다.
    RegisterClassEx(&WndClass);

    //2. 프레임 윈도우를 생선한다 (== 윈도우 세부 정의)
    hwnd = CreateWindow( //윈도우 생성 API함수
        "EasyText", //등록된 윈도우 클래스 이름
        "20221057P1-1", //타이틀 바에 출력될 문자열
        WS_OVERLAPPEDWINDOW, //윈도우 스타일
        CW_USEDEFAULT,
        CW_USEDEFAULT,
        CW_USEDEFAULT,
        CW_USEDEFAULT,
        NULL,
        NULL,
        hInstance, //애플리케이션 인스턴스 핸들.
        NULL,
        );

    //윈도우를 화면에 표시
    ShowWindow(hwnd, nCmdShow);
```

```
    UpdateWindow(hwnd); //갱신한다?

    //3. 메시지 큐로부터 메시지를 받아와, 메시지를 윈도우 프로시저로 보냄.
    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return (int)msg.wParam;
    //메시지를 처리하는 함수.

}
LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParm) {
    HDC hdc;
    RECT rect; //사각형 좌표 지정 구조체
    PAINTSTRUCT ps;

    LPCSTR szMsg1 = "윈플재밌다?ㅎ";
    //c언어에서는 char*, const char*로 문자열을 지정
    //윈프에서는 LPSTR또는 LPCSTR가 위의 형과 같음.
    LPCSTR szMsg2 = "키보드가 눌러졌습니다.";
    LPCSTR szMsg3 = "키보드가 떼어졌습니다.";
    LPCSTR szMsg4 = "마우스가 눌러졌습니다.";
    LPCSTR szMsg5 = "마우스가 떼어졌습니다.";
    LPCSTR szMsg6 = "마우스가 이동중입니다.";
    LPSTR szMsg7 = new char[10];

    POINT MousePoint; //마우스 좌표값 저장 변수

    MousePoint.x = LOWORD(lParm);
    MousePoint.y = HIWORD(lParm);

    switch (message)
    {
    case WM_LBUTTONDOWN:
        hdc = GetDC(hwnd);
        GetClientRect(hwnd, &rect);
        wsprintf(szMsg7, "X:%ld, Y:%ld", MousePoint.x, MousePoint.y);
        DrawText(hdc, szMsg4, strlen(szMsg4), &rect, DT_SINGLELINE | DT_CENTER | DT_VCENTER);
        TextOut(hdc, MousePoint.x, MousePoint.y, szMsg7, strlen(szMsg7));
        ReleaseDC(hwnd, hdc); //DC해제
        break;

    case WM_LBUTTONUP:
                InvalidateRect(hwnd, NULL, TRUE); //화면 전체를 지우고 다시 그리기
        break;

    case WM_MOUSEMOVE:
        hdc = GetDC(hwnd);
        GetClientRect(hwnd, &rect);
        DrawText(hdc, szMsg6, strlen(szMsg6), &rect, DT_SINGLELINE | DT_CENTER | DT_VCENTER);
        ReleaseDC(hwnd, hdc); //DC해제
        break;
```

```
    case WM_KEYDOWN:
        hdc = GetDC(hwnd);
        GetClientRect(hwnd, &rect);
        DrawText(hdc, szMsg2, strlen(szMsg2), &rect, DT_SINGLELINE | DT_CENTER | DT_VCENTER);
        ReleaseDC(hwnd, hdc); //DC해제
        break;

    case WM_KEYUP:
        hdc = GetDC(hwnd);
        GetClientRect(hwnd, &rect);
        DrawText(hdc, szMsg3, strlen(szMsg3), &rect, DT_SINGLELINE | DT_CENTER | DT_VCENTER);
        ReleaseDC(hwnd, hdc); //DC해제
        break;

    case WM_CREATE:
        break;
    case WM_PAINT: //그리다..? 암튼 그럼
        hdc = BeginPaint(hwnd, &ps); //DC핸들을 얻어오기 위함.      paint에서는 무조건 beginpaint로 hdc 과정을 거쳐야함.
        //DC얻었기 때문에 TextOut으로 출력 가능
        TextOut(hdc, 10, 10, szMsg1, strlen(szMsg1));
        EndPaint(hwnd, &ps);
        break;
    case WM_DESTROY:
        PostQuitMessage(0);//종료메시지
        break;
    default:
        return DefWindowProc(hwnd, message, wParam, lParm);

    }
    return 0;

}
```

이 실습 코드의 중요한 개념은 DC 라는 개념인 것 같은데.
DC란 Devicd Conetext(장치 문맥)으로서 어떤 장치에 출력할지를 지정하는 핸들(HDC)이다.
ex) GetDC(hwnd); 는 hwnd 창의 클라이언트 영역에 출력한다는 뜻. 여기서 hwnd는 위의 코드에서

```
    //2. 프레임 윈도우를 생선한다 (== 윈도우 세부 정의)
    hwnd = CreateWindow( //윈도우 생성 API함수
        "EasyText", //등록된 윈도우 클래스 이름
        "20221057P1-1", //타이틀 바에 출력될 문자열
        WS_OVERLAPPEDWINDOW, //윈도우 스타일
        CW_USEDEFAULT,
        CW_USEDEFAULT,
        CW_USEDEFAULT,
        CW_USEDEFAULT,
        NULL,
        NULL,
        hInstance, //애플리케이션 인스턴스 핸들.
        NULL,
        );
```

이 부분으로 세팅을 마친 창을 말함.

이후 메인 함수에서는

```
    while (GetMessage(&msg, NULL, 0, 0)) { //계속하여 메시지 큐에서 값을 받아오는 반복문.
        TranslateMessage(&msg);
        DispatchMessage(&msg); //이 함수가 WndProc 함수를 호출해서 메시지를 해석하여 switch문을 거쳐 실행됨
    }
```

이 코드가 계속 반복될텐데 입력값이 0(종료)가 아닌 이상 계속 반복하여 GetMessage 함수가 메시지큐 라는 임시 저장공간에서 가장 오래된 입력을 가져와(선입선출) msg에 저장한다.
이후 DispatchMessage라는 함수로 WndProc으로 message 값을 넘겨서 메시지의 형태에 맞에 switch문을 실행하는 것을 반복하는 실습 코드이다.
조금만 더 들여다 보자면
가장 많은 것이 들어간

```
    case WM_LBUTTONDOWN:
        hdc = GetDC(hwnd);
        GetClientRect(hwnd, &rect);
        wsprintf(szMsg7, "X:%ld, Y:%ld", MousePoint.x, MousePoint.y);
        DrawText(hdc, szMsg4, strlen(szMsg4), &rect, DT_SINGLELINE | DT_CENTER | DT_VCENTER);
        TextOut(hdc, MousePoint.x, MousePoint.y, szMsg7, strlen(szMsg7));
        ReleaseDC(hwnd, hdc); //DC해제
        break;
```

이 case를 보자면
WM(윈도우 메시지)로 받아온 값이 LBUTTONDOWN(마우스 왼쪽 버튼 클릭)일 때
HDC hdc 라는 변수에 GetDC함수로 hwnd에 대한 권한(핸들)을 얻으며 시작
GetClientRect(hwnd, &rect); 코드로 rect라는 변수에 hwnd의 정보를 저장.
wsprintf 함수는 문자열 포맷팅 함수로서 szMsg7 문자열에 이 내용을 저장 할게 라는 소리이다.
DrawText로 rect의 정보에 기반하여 텍스트를 출력하며 함수의 매개변수로 오는DT_XXXXX는 DrawText함수에서만 사용가능하다.
TextOut 함수는 좌표 기반 메시지 출력을 해준다. 둘다 텍스트를 출력한다는점은 같지만 사용 방법이 달라 알아두면 좋다.
이 후 핸들을 해제해주면서 아무런 작없이 없다면 DC는 해제된 상태로 있는다.

핵심은 hwnd로 창을 설정하고 while문으로 메시지큐에 있는 값을 msg에 저장하여 DispatchMessage(&msg)
라는 함수로 WndProc으로 msg값을 보내서 switch 문에 맞게 실행한다. 이후 무한 반복 하다가 입력값이 종료라면
while문이 받는 값이 0이 되어 반복문이 종료되고 종료값을 리턴하며 메인함수가 종료된다.

1주차 실습 리뷰 끝