1장
윈도우 프로그래밍 들어가기

김성영교수
금오공과대학교
컴퓨터공학부

예제

- 다음 프로그램은 언제 종료할까?

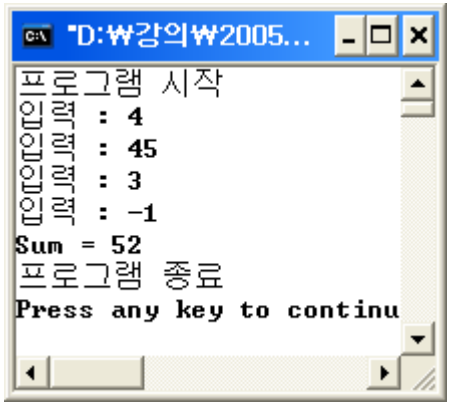
```
#include <iostream.h>
#define QUIT -1

int Func(void) {
    int i;
    cout << "입력 : ";
    cin >> i;
    return i;
}

void main(void) {
    int Sum = 0, i;
    cout << "프로그램 시작" << endl;

    while ((i = Func()) != QUIT) {
       Sum += i;
    }

    cout << "Sum = " << Sum << endl;
    cout << "프로그램 종료" << endl;
}
```



간단한 Win32 응용프로그램 (1)

Hello.c – win32 console application

```
#include <stdio.h>

int main()
{
    printf("Hello World\n" );

    return 0;
}
```

간단한 Win32 응용프로그램 (2)

Hello_Win.c – win32 console application

```
#include <windows.h>

int main()
{
    MessageBox( NULL, "Hello World", "Hello", MB_OK );

    return 0;
}
```

간단한 Win32 응용프로그램 (3)

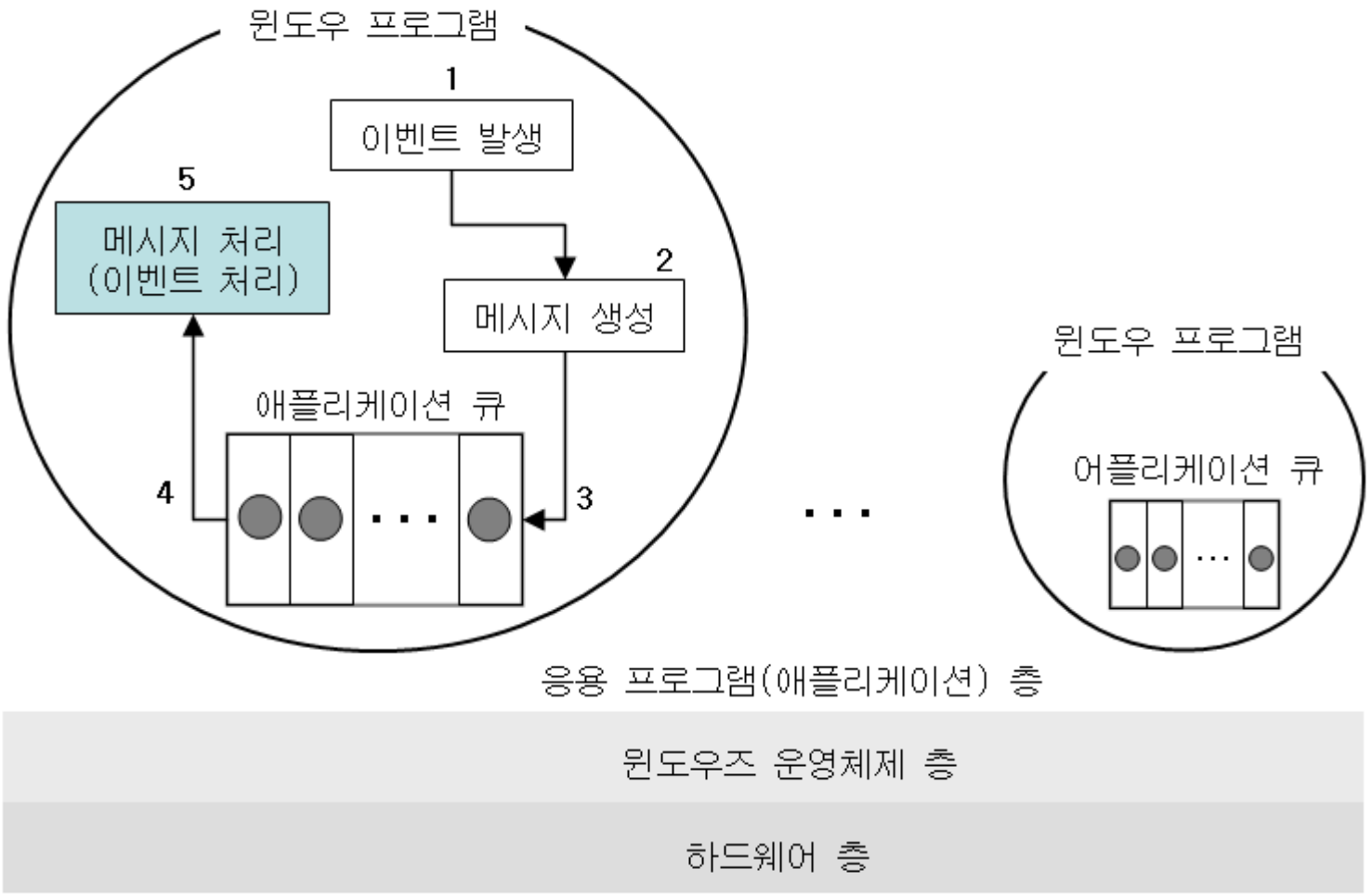
Hello_WinMain.c – win32 application

```
#include <windows.h>

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
                   PSTR szCmdLine, int iCmdShow)
{
    MessageBox( NULL, "Hello World", "Hello", MB_OK );

    return 0;
}
```

이벤트 처리 과정


윈도우 프로그램
1
이벤트 발생
5
메시지 처리
(이벤트 처리)
2
메시지 생성
애플리케이션 큐
4
3
윈도우 프로그램
어플리케이션 큐
응용 프로그램(애플리케이션) 층
윈도우즈 운영체제 층
하드웨어 층

메시지 구조체

```
typedef struct {
    HWND hwnd;      // 이벤트가 발생한 윈도우를 위한 식별자
    UINT message;   // 이벤트 구분을 위한 식별자
    WPARAM wParam; // 이벤트 발생 당시의 상황(부가) 정보를 기록
    LPARAM lParam; // 이벤트 발생 당시의 상황(부가) 정보를 기록
    DWORD time;     // 이벤트 발생 시간을 기록
    POINT pt;       // 이벤트 발생시의 마우스 좌표를 기록
} MSG
```

Windows Data Type

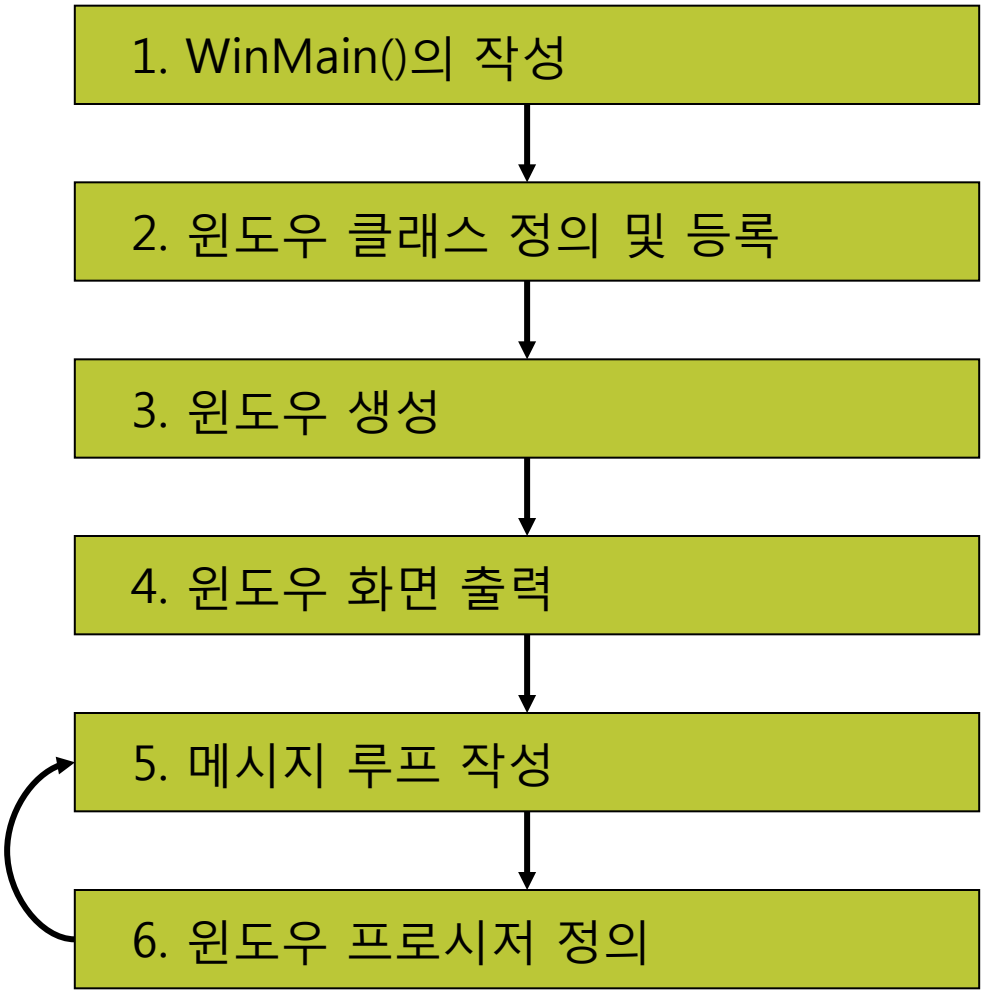
winnt.h와 windef.h에 재정의

```
typedef     char              CHAR;
typedef     short             SHORT;
typedef     long              LONG;
typedef     unsigned long     DWORD;
typedef     int               BOOL;
typedef     unsigned char     BYTE;
typedef     unsigned short    WORD;
typedef     float             FLOAT;
typedef     int               INT;
typedef     unsigned int      UINT;
typedef     UINT              WPARAM;
typedef     LONG              LPARAM;
typedef     LONG              LRESULT;
typedef     char *            LPSTR
typedef const char *          LPCSTR
```

<변수 명명법의 예>

a : 배열(Array)
b : BOOL(Bool)
ch : 문자(Character)
cb : 바이트 수(Count of Bytes)
dw : 부호없는 long형 정수(Double Word)
h : 핸들(Handle)
i : 인덱스(Index)
l : long int
n : int
sz : NULL로 끝나는 문자열
w : 부호없는 16bit 정수(Word)
m_ : 멤버 변수
g_ : 전역 변수

프로그램 작성 흐름


1. WinMain()의 작성
2. 윈도우 클래스 정의 및 등록
3. 윈도우 생성
4. 윈도우 화면 출력
5. 메시지 루프 작성
6. 윈도우 프로시저 정의

실습 1-1

※ p14의 HELLO.CPP 입력 후 실행

```
WinMain()  메인 함수
{
     - 윈도우생성
          1. 윈도우 클래스 정의
          2. 윈도우 클래스 등록
          3. 윈도우 생성 및 출력
     - 메시지루프
}

WndProc()  윈도우 프로시저
{
     - 이벤트(메시지) 처리
}
```

1. WinMain( ) 작성

```
#include <windows.h>

int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    LPSTR lpCmdLine, int nCmdShow )
{
    return 0;
}
```

2. 윈도우 클래스 정의 및 등록

```
#include <windows.h>

int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    LPSTR lpCmdLine, int nCmdShow )
{
    WNDCLASS WndClass;
    TCHAR szAppName[] = TEXT( "Hello" );

    // 윈도우 클래스 정의
    WndClass.style              = 0;
    WndClass.lpfnWndProc        = WndProc;   // 윈도우 프로시저 이름
    WndClass.cbClsExtra         = 0;
    WndClass.cbWndExtra         = 0;
    WndClass.hInstance          = hInstance;
    WndClass.hIcon              = LoadIcon( NULL, IDI_APPLICATION );
    WndClass.hCursor            = LoadCursor( NULL, IDC_ARROW );
    WndClass.hbrBackground      = (HBRUSH)GetStockObject( WHITE_BRUSH );
    WndClass.lpszMenuName       = NULL;
    WndClass.lpszClassName      = szAppName;

    // 윈도우 클래스 등록
    if ( !RegisterClass(&WndClass) ) return -1;

    return 0;
}
```

윈도우 클래스 구조체

```
typedef struct _WNDCLASS {
    UINT        style;          // 윈도우 클래스 스타일
    WNDPROC     lpfnWndProc;    // 윈도우 프로시저에 대한 포인터
    int         cbClsExtra;     // 윈도우 클래스의 데이터 영역 (기본값 0)
    int         cbWndExtra;     // 윈도우의 데이터 영역 (기본값 0)
    HINSTANCE   hInstance;      // 인스턴스 핸들 (프로그램 자신의 핸들값)
    HICON       hIcon;          // 윈도우에 사용될 아이콘 핸들
    HCURSOR     hCursor;        // 윈도우 영역에서 사용될 커서 핸들
    HBRUSH      hbrBackground;  // 클라이언트 영역 배경색을 표시할 브러시 핸들
    LPCTSTR     lpszMenuName;   // 윈도우에 사용될 메뉴 이름
    LPCTSTR     lpszClassName;  // 윈도우 클래스 이름
} WNDCLASS, *PWNDCLASS;
```

http://msdn.microsoft.com/ko-kr/library/ms633576.aspx

3. 윈도우 생성

```
#include <windows.h>

int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    LPSTR lpCmdLine, int nCmdShow )
{
    …
    TCHAR szAppName[] = TEXT( "Hello" );
    HWND hwnd;
    …
    // 윈도우 생성
    hwnd = CreateWindow(
                szAppName,              // window class name
                szAppName,              // window caption
                WS_OVERLAPPEDWINDOW,    // window style
                CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,
                NULL,                   // parent window handle
                NULL,                   // window menu handle
                hInstance,              // program instance handle
                NULL                    // creation parameters
        );

    return 0;
}
```

CreateWindow( )

```
HWND CreateWindow(
          LPCTSTR lpClassName,    // 등록한 윈도우 클래스
          LPCTSTR lpWindowName,   // 윈도우 캡션
          DWORD dwStyle,          // 윈도우 스타일
          int x;                  // 윈도우 좌측 상단의 x좌표
          int y;                  // 윈도우 좌측 상단의 y좌표
          int nWidth;             // 윈도우 폭
          int nHeight;            // 윈도우 높이
          HWND hWndParent;        // 부모 윈도우의 핸들
          HMENU hMenu;            // 메뉴 핸들
          HINSTANCE hInstance;    // 윈도우를 생성한 인스턴스 핸들
          LPVOID lpParam;         // CREATESTRUCT 구조체를 통해 전달되는 값
);
```

http://msdn.microsoft.com/ko-kr/library/ms632679.aspx

윈도우 스타일

- WS_BORDER : 경계를 갖는 윈도우
- WS_CAPTION : 캡션바를 갖는 윈도우
- WS_OVERLAPPED : 캡션바와 경계를 갖는 윈도우
- WS_CHILD : 특정 윈도우의 자식 윈도우
- WS_POPUP : POPUP 윈도우
- WS_MAXIMIZE : 최대화 버튼을 갖는 윈도우
- WS_MINIMIZE : 최소화 버튼을 갖는 윈도우
- WS_SYSMENU : 캡션바에 시스템 메뉴를 갖는 윈도우
- WS_THICKFRAME : 크기 조정이 가능한 윈도우
- WS_OVERLAPPEDWINDOW : WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU | WS_THICKFRAME | WS_MINIMIZEBOX | WS_MAXIMIZEBOX
- WS_POPUPWINDOW : WS_POPUP | WS_BORDER | WS_SYSMENU
- WS_CHILDWINDOW : WS_CHILD

4. 윈도우 화면 출력

```
#include <windows.h>

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine, int nCmdShow)
{
     …
     HWND hwnd;
     …
     hwnd = CreateWindow(
                    szAppName,
                    szAppName,
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL
        );
     // 윈도우 화면 출력
     ShowWindow( hwnd, nCmdShow );
     UpdateWindow( hwnd );

     while (1) {
     }

     return 0;
}
```

ShowWindow( )

```
BOOL ShowWindow (
        HWND hWnd,              // 출력할 윈도우 핸들
        int nCmdShow            // 초기 윈도우 상태
        );
```

```
nCmdShow
     SW_HIDE : 윈도우를 안보이게 함
     SW_SHOW : 화면에 표시
     SW_SHOWMAXIMIZED : 윈도우를 최대 화면으로 출력
     SW_SHOWMINIMIZED : 윈도우를 작업 표시줄의 아이콘으로 표시
```

5. 메시지 루프 작성

```
#include <windows.h>

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine, int nCmdShow)
{
    …
    HWND hwnd;
    MSG  msg;

    …

    ShowWindow( hwnd, iCmdShow );

    // 메시지 루프
    while ( GetMessage(&msg, NULL, 0, 0) )
    {
        TranslateMessage( &msg );
        DispatchMessage( &msg );
    }

    return msg.wParam;
}
```

6. 윈도우 프로시저 정의

```
#include <windows.h>

LRESULT CALLBACK WndProc( HWND, UINT, WPARAM, LPARAM );

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine, int nCmdShow)
{
    …
    WndClass.lpfnWndProc   = WndProc;
    …
}

// 윈도우 프로시저 정의
LRESULT CALLBACK WndProc( HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam )
{
    switch(message)
    {
    case WM_DESTROY:
        PostQuitMessage( 0 );
        return FALSE;  // 사용자 처리의 경우 반드시 FALSE을 리턴
    }

    // 응용 프로그램이 처리하지 않으면 윈도우 운영체제가 처리
    return DefWindowProc( hwnd, message, wParam, lParam );
}
```