

제4장 윈도우 메뉴

2024년 1학기 윈도우 프로그래밍

학습 목표

- 학습목표
 - 리소스 파일을 만들고 윈도우에 메뉴를 등록하는 방법을 배운다.
 - 메뉴에서 발생하는 메시지를 처리하는 방법을 배운다.
- 내용
 - 메뉴 만들기
 - 메뉴 사용하기

리소스 개요

- 리소스(resource)
 - 메뉴, 아이콘, 커서, 다이얼로그, 액셀러레이터, 비트맵 등 사용자 인터페이스를 구성하는 자원들로 읽기 전용 정적 데이터를 말한다.
 - 리소스는 프로그램 실행 중 변경되지 않는 정적 데이터
 - C/C++과 같은 언어로 관리하지 않고 리소스 스크립트(Resource Script; .rc)파일로 관리한다.
- 윈도우 프로그램의 큰 특징 중 하나가 소스코드와 리소스가 분리
 - DOS기반 프로그래밍에서는 리소스를 정의할 때 복잡한 배열 등을 만들어 사용하거나 외부 파일로 만들어 둔 후 프로그램 실행 시에 읽어 들이도록 했다.
 - 그러나 윈도우 프로그래밍에서는 이런 데이터들을 리소스로 만들어 놓고 소스코드와 별도로 컴파일하며, 링크 시 최종 실행 파일에 합쳐진다.
 - 리소스 파일들:
 - 리소스 스크립트 파일: 확장자가 rc인 파일 - 사용할 리소스를 설정: 비주얼 스튜디오에서 열어 편집 가능
 - 리소스 헤더파일: 확장자가 h인 헤더파일 - 사용할 리소스 id 정의: 비주얼 스튜디오에서 열어 편집 가능
 - 다양한 리소스 파일들: 실제 리소스 파일들 - 메뉴, 아이콘, 커서, 다이얼로그 등 실제 리소스 파일들

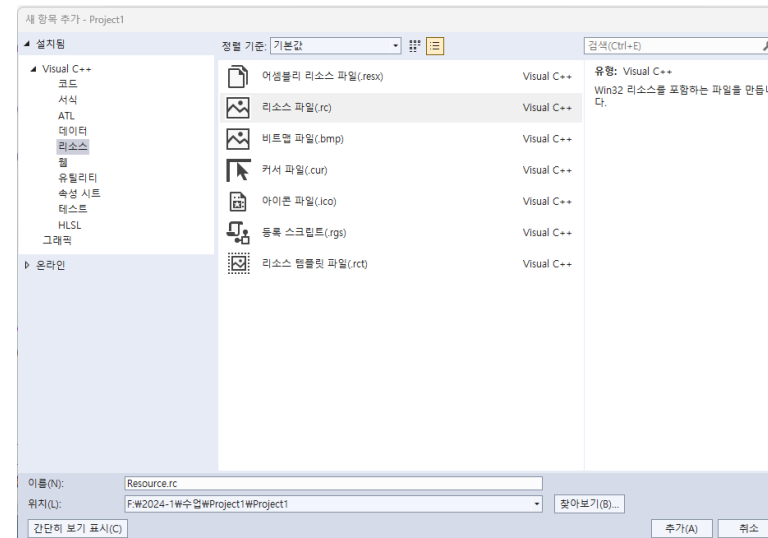
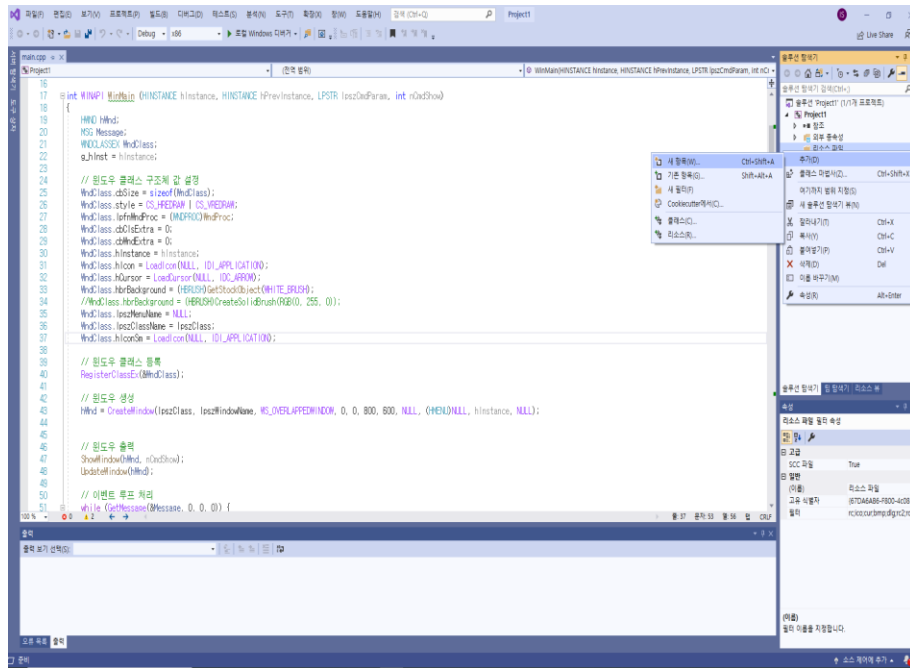
리소스 만들기: 메뉴

- 리소스 스크립트 파일 만들기
 - 방법: 소스 파일 작성과 유사
 - C++ 소스 대신에 Resource Script 선택하여 생성
 - 리소스 스크립트 파일과 헤더 파일 생성:
 - resource.rc: 리소스 스크립트 파일로 리소스 정의
 - resource.h: 헤더파일로 리소스 ID 정의
- 사용할 리소스 생성하고 편집
 - 리소스 도구상자를 이용하여 필요한 리소스를 추가
 - 리소스 편집창을 통해 추가된 리소스 편집
 - 메뉴인 경우
 - 메뉴 리소스 id: IDR_MENU1으로 정의됨 (속성창에서 확인)
 - 메뉴 캡션: 메뉴바에 추가될 메뉴항목들의 캡션 입력
 - 메뉴의 속성들: 메뉴의 id는 프로그램과 연계되어 프로그램에서 사용됨
- 프로그램에서 리소스 연결하고 메시지 처리하여 리소스 사용
 - 리소스 헤더파일 추가
 - **include "resource.h"**
 - 리소스에서 발생하는 메시지 처리하기
 - 메뉴인 경우
 - 메뉴ID 등록: 윈도우 클래스 생성 시 또는 윈도우 생성 시 또는 메뉴 생성 함수를 통하여 메뉴 등록
 - 메뉴 처리: 메뉴를 눌렀을 때 WM_COMMAND 메시지 발생, 처리

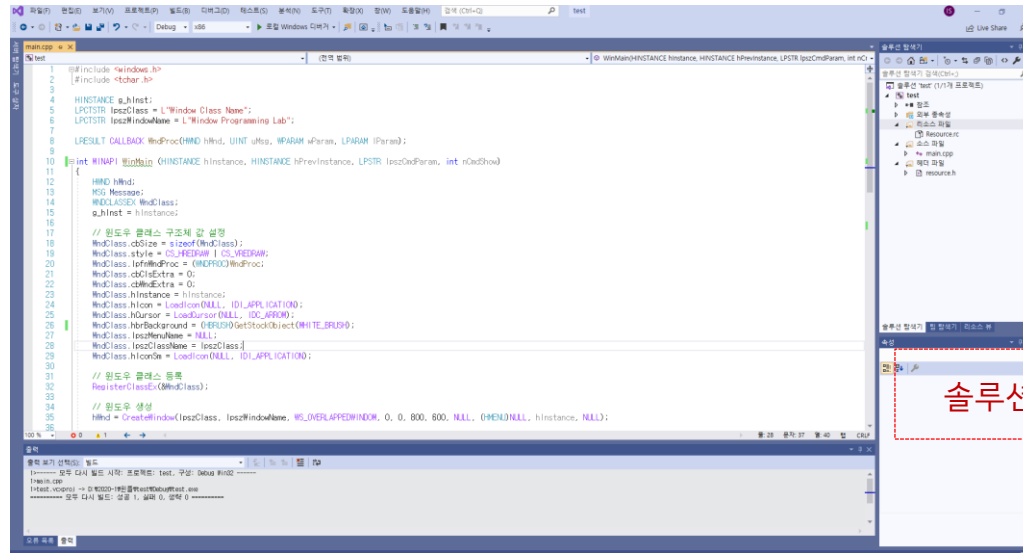
1. 리소스 스크립트 파일 만들기

- Visual Studio 2022 환경

- 윈도우 만들기
- 프로젝트에 리소스 추가하기
- [프로젝트] → [리소스 파일] → [새 항목 추가]

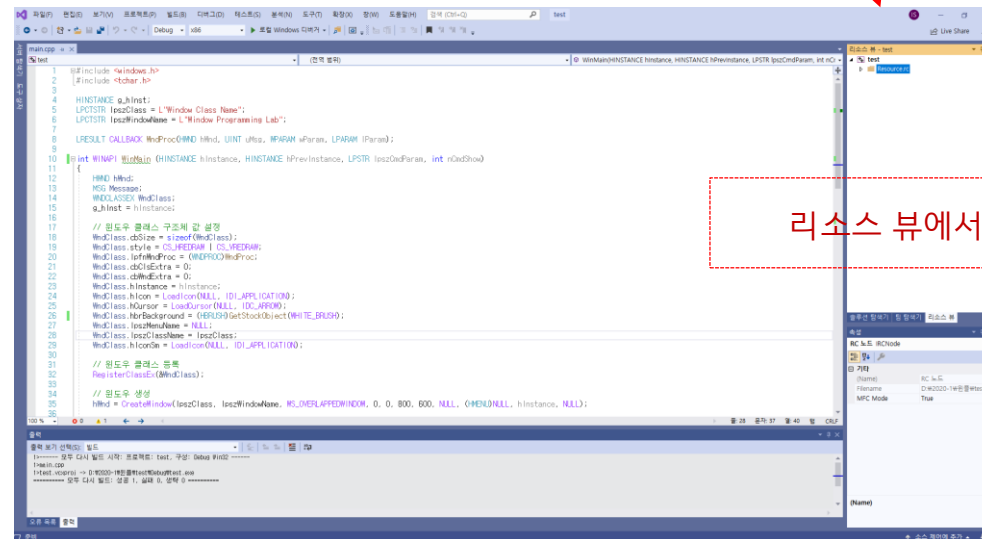


생성된 리소스 파일

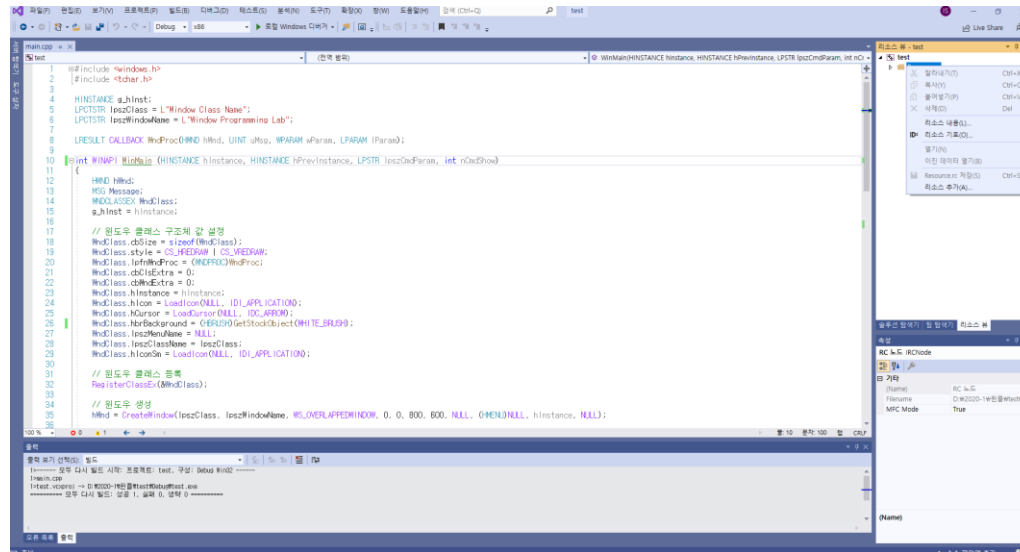


리소스 파일

솔루션 탐색기에서 볼 때

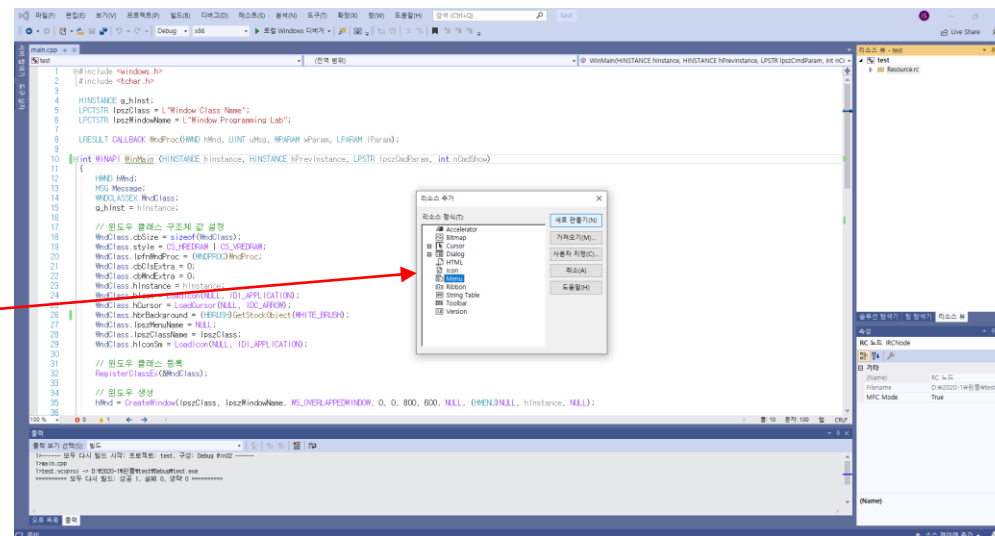


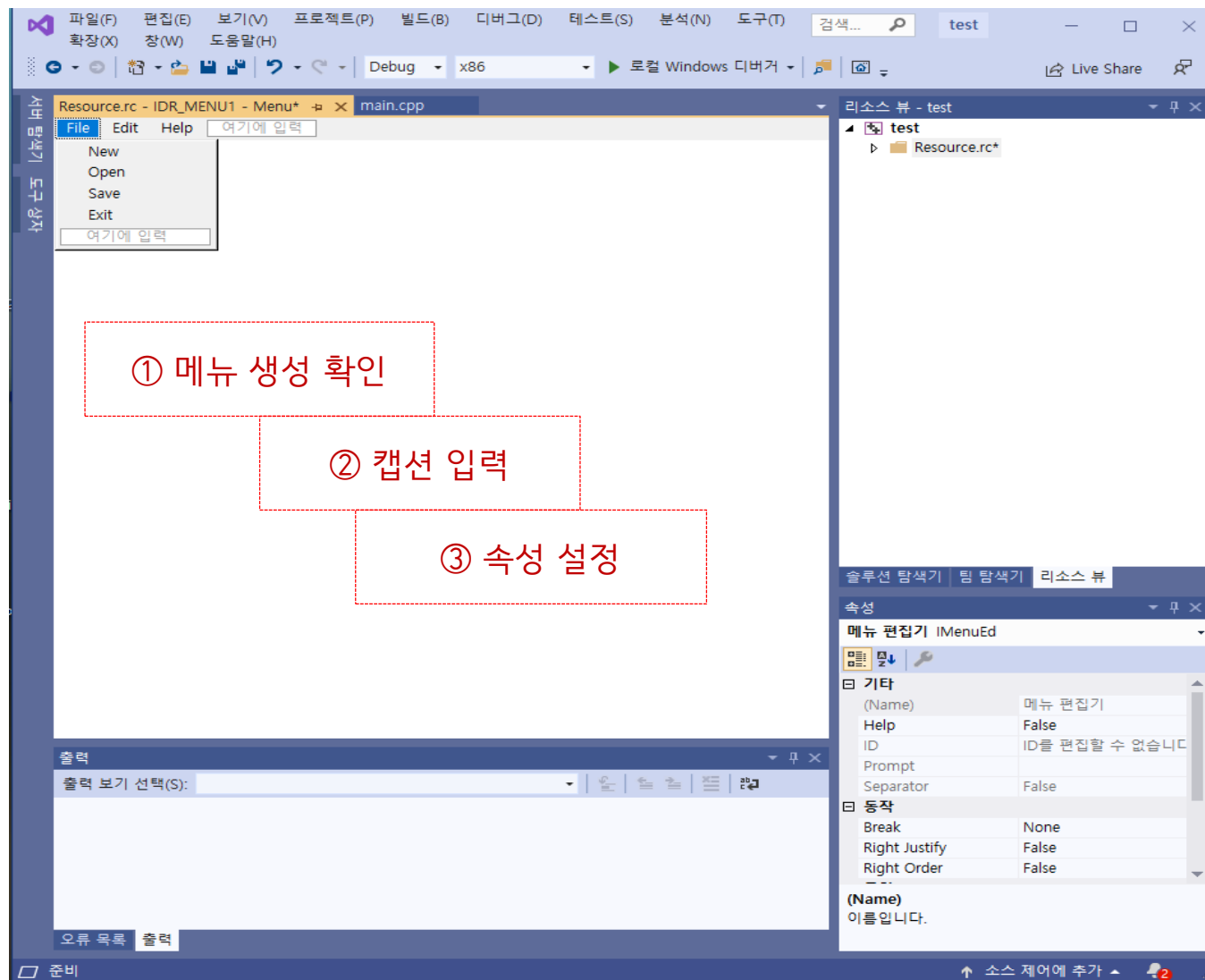
2. 리소스 생성하고 편집하기: 메뉴 만들기



- 1) 리소스 파일 폴더를 선택한 후 오른쪽 마우스 클릭
- 2) "리소스 추가" 선택

메뉴 리소스 선택



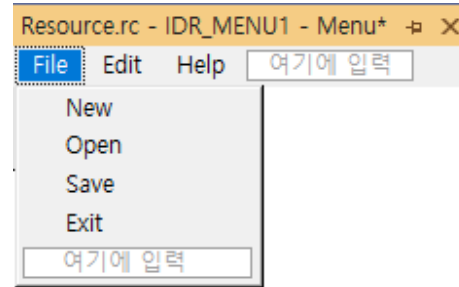


메뉴 만들기

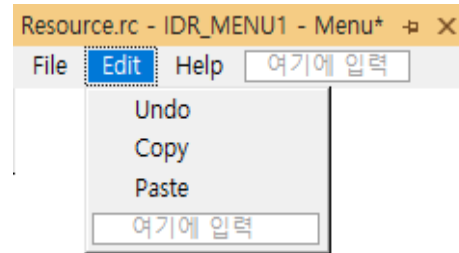
예제 내용

- 다음의 메뉴 항목 설정표를 참고하여 기본 메뉴를 작성하기.

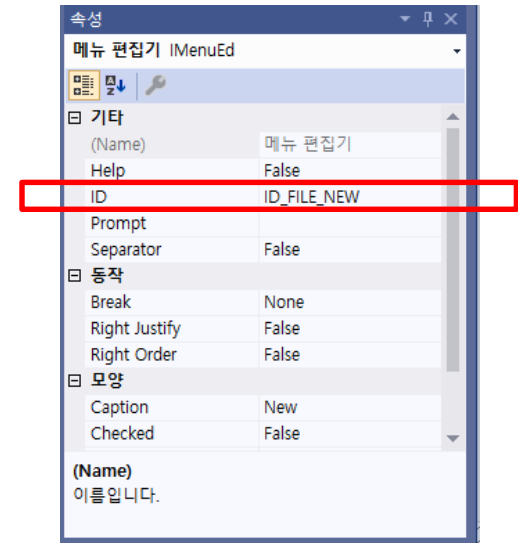
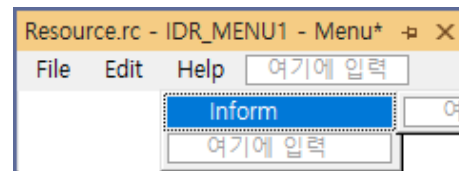
Caption	ID
File	
New	ID_FILE_NEW
Open	ID_FILE_OPEN
Save	ID_FILE_SAVE
Exit	ID_FILE_EXIT



Caption	ID
Edit	
Undo	ID_EDIT_UNDO
Copy	ID_EDIT_COPY
Paste	ID_EDIT_PASTE

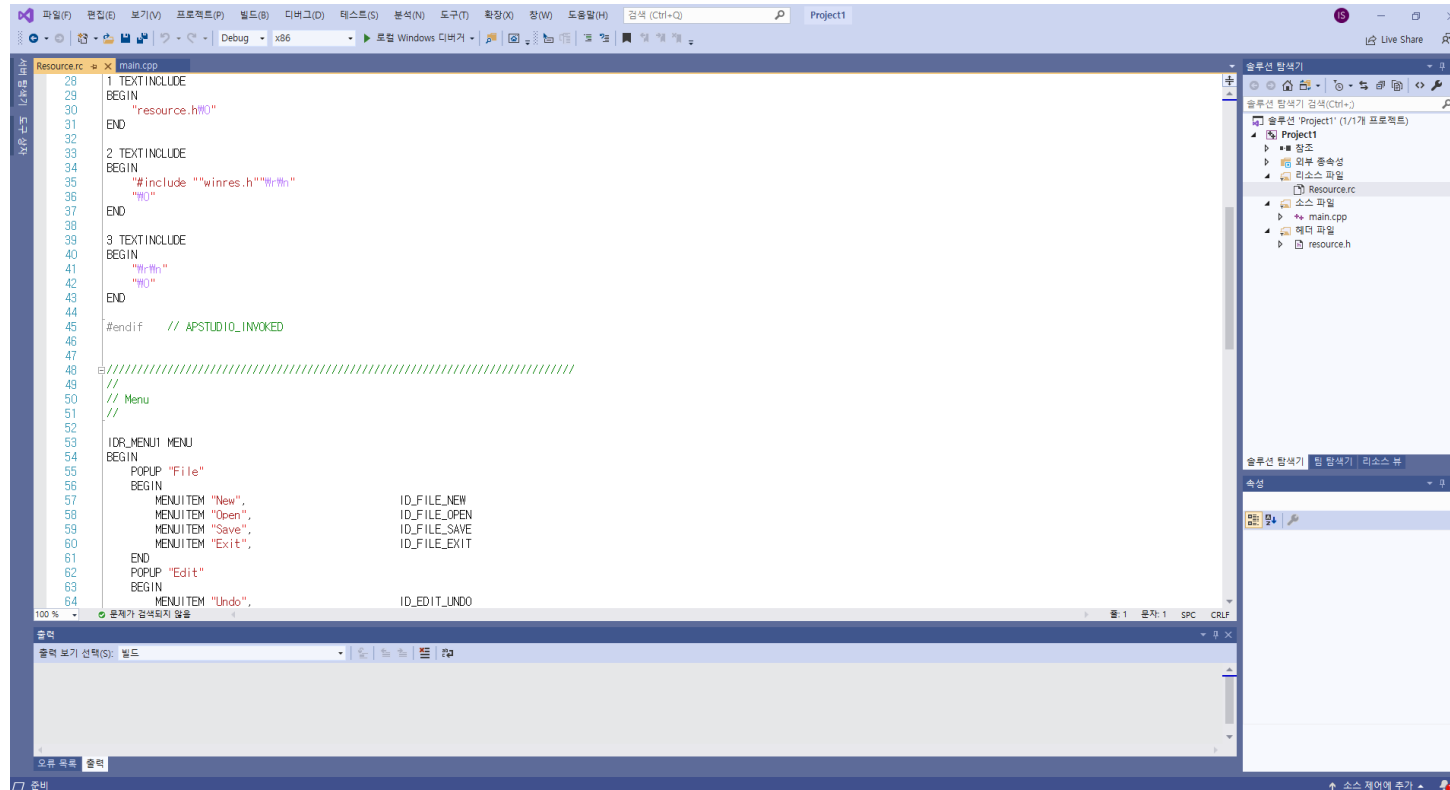


Caption	ID
Help	
Inform	ID_HELP_INFORM



메뉴 만들기

- 리소스.rc 파일
 - 비주얼 스튜디오에서 열어서 편집 가능



메뉴 만들기

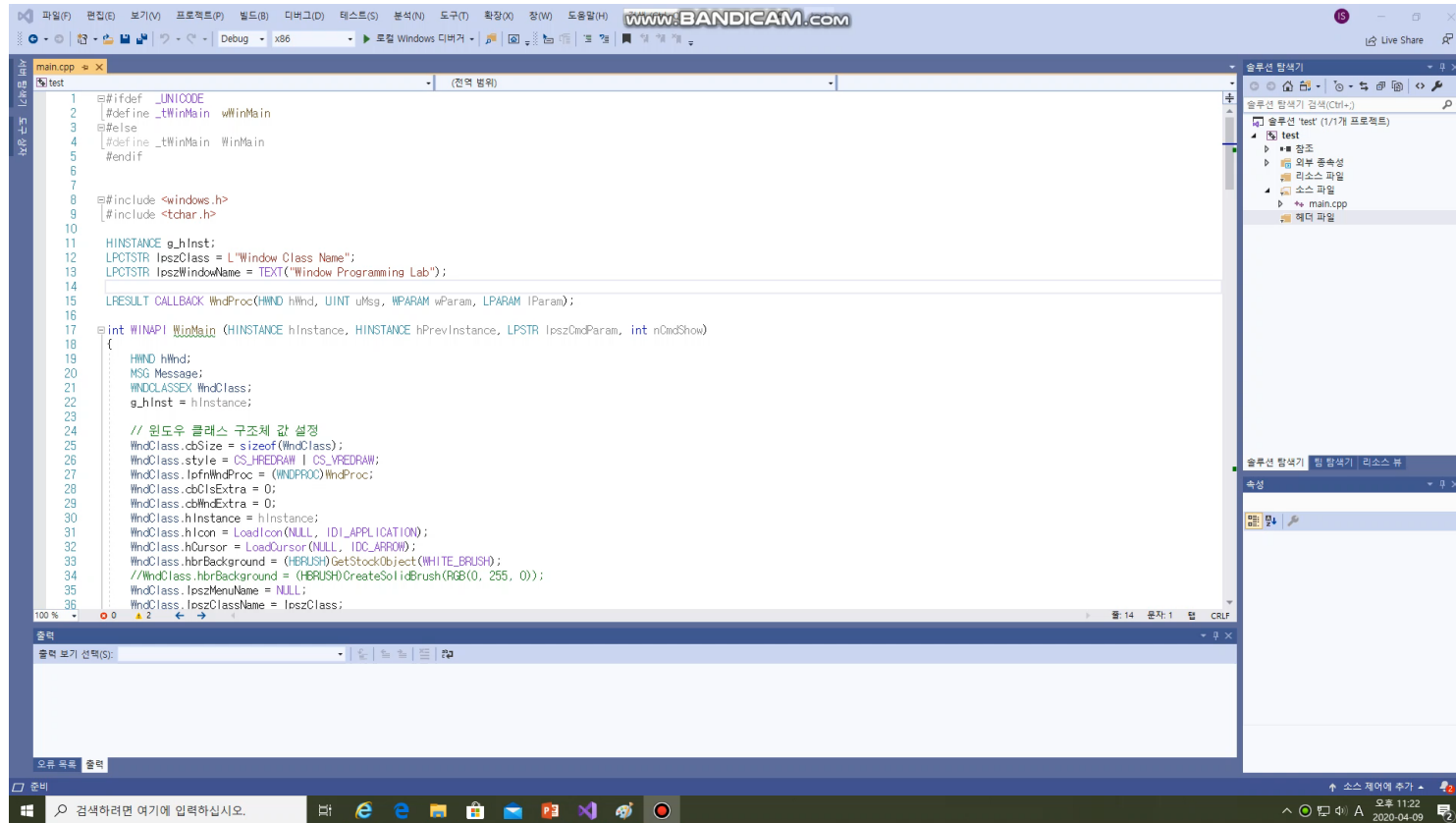
- 리소스.h 파일
 - 비주얼 스튜디오에서 열어서 편집 가능

```
resource.h  main.cpp
test
1  //{{NO_DEPENDENCIES}}
2  // Microsoft Visual C++에서 생성한 포함 파일입니다.
3  // Resource.rc에서 사용되고 있습니다.
4  //
5  #define IDR_MENU1 101
6  #define ID_FILE_NEW 40001
7  #define ID_FILE_OPEN 40002
8  #define ID_FILE_SAVE 40003
9  #define ID_FILE_EXIT 40004
10 #define ID_EDIT_UNDO 40005
11 #define ID_EDIT_COPY 40006
12 #define ID_EDIT_PASTE 40007
13 #define ID_HELP_INFORM 40008
14
15 // Next default values for new objects
16 //
17 #ifdef APSTUDIO_INVOKED
18 #ifndef APSTUDIO_READONLY_SYMBOLS
19 #define _APS_NEXT_RESOURCE_VALUE 102
20 #define _APS_NEXT_COMMAND_VALUE 40009
21 #define _APS_NEXT_CONTROL_VALUE 1001
22 #define _APS_NEXT_SYMED_VALUE 101
23 #endif
24 #endif
25
```

앞 페이지에서 설정한 메뉴의 id 값들이 define 되어 있다.

메뉴 만들기

- 메뉴 만들기 과정



3. 리소스 연결하고 사용하기: 윈도우에 리소스 연결하여 메뉴 붙이기

- 응용 프로그램에 메뉴 리소스를 불러오는 방법 3가지

방법	함수 사용
1) 윈도우 클래스를 만들 때 메뉴를 정의	<pre>//--- 윈도우 클래스 만들 때 추가 winclass.lpszMenuName = MAKEINTRESOURCE (IDR_MENU1);</pre>
2) 메인 윈도우를 생성할 때 메뉴를 첨부	<pre>//--- 윈도우 클래스의 메뉴에는 NULL 설정 winclass.lpszMenuName = NULL; //--- 윈도우 생성할 때 추가 CreateWindow (szWindowClass, szTitle, WS_OVERLAPPEDWINDOW, 0, 0, 800, 600, NULL, LoadMenu (hInstance, MAKEINTRESOURCE (IDR_MENU1)), hInstance, NULL);</pre>
3) 초기 윈도우 생성이 끝난 후에 붙인다.	<pre>//--- 메뉴를 만들어서 따로 추가 HMENU hMymenu; hMymenu = LoadMenu (hInstance, MAKEINTRESOURCE (IDR_MENU1)); SetMenu (hwnd, hMymenu);</pre>

윈도우에 리소스 연결하여 메뉴 붙이기

- 리소스에 대한 정수형 상수를 문자열로 변환하는 매크로 함수: **MAKEINTRESOURCE ()**

LPTSTR MAKEINTRESOURCE (WORD wInt);

- WORD wInt: 리소스에 대한 정수형 상수

- 메뉴 리소스를 불러오는 함수: **LoadMenu ()**

HMENU LoadMenu (HINSTANCE hInstance, LPCTSTR lpMenuName);

- HINSTANCE hInstance: 메뉴 리소스를 가진 인스턴스 핸들
- LPCTSTR lpMenuName: 메뉴 리소스의 이름을 지정하는 문자열

- 메뉴를 설정/변경하기: **SetMenu ()**

BOOL SetMenu (HWND hWnd, HMENU hMenu);

- HWND hWnd: 메뉴를 붙일 윈도우의 핸들, 차일드 윈도우에는 메뉴를 붙일 수 없다.
- HMENU hMenu: 메뉴 핸들, 이 핸들을 NULL로 주면 윈도우의 메뉴를 제거한다.

윈도우에 메뉴 붙이기

//--- 1) 윈도우 클래스를 만들 때 메뉴를 정의

```
#include "resource.h"
```

```
//--- 리소스 헤더 파일 첨부
```

```
LRESULT WINAPI WinMain ( ... )
```

```
{
```

```
    WNDCLASSEX WndClass;
```

```
    //--- wndclass 속성 설정
```

```
    WndClass.cbSize = sizeof(WndClass);
```

```
    WndClass.style=CS_HREDRAW | CS_VREDRAW;
```

```
    WndClass.lpfnWndProc=(WNDPROC)WndProc;
```

```
    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(BLACK_BRUSH);
```

```
    WndClass.lpszMenuName= MAKEINTRESOURCE (IDR_MENU1);    //--- 메뉴 id 등록
```

```
    WndClass.lpszClassName=lpszClass;
```

```
    WndClass.hIconSm = LoadIcon(NULL,IDI_APPLICATION);
```

```
    RegisterClassEx (&WndClass);
```

```
    ...
```

```
}
```

윈도우에 리소스 연결하여 메뉴 붙이기

//--- 2) 메인 윈도우를 생성할 때 메뉴를 첨부

```
#include "resource.h"
```

```
//--- 리소스 헤더 파일 첨부
```

```
LRESULT WINAPI WinMain ( ... )
```

```
{
```

```
    WNDCLASSEX WndClass;
```

```
    //--- wndclass 속성 설정
```

```
    WndClass.cbSize = sizeof(WndClass);
```

```
    WndClass.style=CS_HREDRAW | CS_VREDRAW;
```

```
    WndClass.lpfnWndProc=(WNDPROC)WndProc;
```

```
    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(BLACK_BRUSH);
```

```
    WndClass.lpszMenuName= NULL;
```

```
//--- 윈도우 클래스의 메뉴에는 NULL 설정
```

```
    WndClass.lpszClassName=lpszClass;
```

```
    WndClass.hIconSm = LoadIcon(NULL,IDI_APPLICATION);
```

```
    RegisterClassEx (&WndClass);
```

```
    //--- 윈도우 생성할 때 추가
```

```
    hWnd = CreateWindow ( szWindowClass, szTitle, WS_OVERLAPPEDWINDOW, 0, 0, 800, 600, NULL,  
                        LoadMenu ( hinstance, MAKEINTRESOURCE (IDR_MENU1) ), hInstance, NULL);
```

```
    ...
```

```
}
```


윈도우에 메뉴 붙이기

//--- 3) 초기 윈도우 생성이 끝난 후에 객체를 만들어 붙인다

```
#include "resource.h"
```

//--- 리소스 헤더 파일 첨부

```
LRESULT WINAPI WinMain ( ... )  
{
```

```
    WNDCLASSEX WndClass;
```

```
    HMENU hMymenu;
```

//--- 메뉴 핸들 선언

//--- wndclass 속성 설정

```
    WndClass.cbSize = sizeof(WndClass);
```

```
    WndClass.style=CS_HREDRAW | CS_VREDRAW;
```

```
    WndClass.lpfnWndProc=(WNDPROC)WndProc;
```

```
    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(BLACK_BRUSH);
```

```
    WndClass.lpszMenuName= NULL;
```

//--- 윈도우 클래스의 메뉴에는 NULL 설정

```
    WndClass.lpszClassName=lpszClass;
```

```
    WndClass.hIconSm = LoadIcon(NULL,IDI_APPLICATION);
```

```
    RegisterClassEx (&WndClass);
```

//--- 윈도우 생성: CreateWindow의 메뉴에는 NULL 설정

```
    hWnd = CreateWindow ( szWindowClass, szTitle, WS_OVERLAPPEDWINDOW, 0, 0, 800, 600, NULL, NULL, hInstance, NULL);
```

//--- 따로 만들어서 추가

```
    hMymenu = LoadMenu (hInstance, MAKEINTRESOURCE ( IDR_MENU1 ) );
```

```
    SetMenu (hwnd, hMymenu);    //--- 따로 만들어서 추가
```

```
    ...
```

```
}
```

메시지 처리하기: 커맨드 메시지 처리하기

- 메뉴를 선택했을 때 메시지 발생: WM_COMMAND 메시지가 전달
 - **WM_COMMAND** 메시지
 - 메뉴의 메뉴항목을 선택하면 발생하는 메시지
 - 커맨드메시지(Command message)라 부름
 - 윈도우 프로시저로 보내지는 파라미터에는
 - **wParam**:
 - » **LOWORD(wParam)**: 선택된 메뉴 항목의 ID가 정수로 들어 있음
 - » **HWORD(wParam)**: 0 (이벤트 소스)
 - **lParam**: 0

메시지 처리하기: 커맨드 메시지 처리하기

- 사용 예) New 메뉴를 선택했을 때, Exit 메뉴를 선택했을 때

LRESULT CALLBACK **WndProc** (HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)

```
{
    int answer;

    switch (iMsg)
    {
        case WM_COMMAND:
            switch (LOWORD(wParam)) {
                case ID_FILE_NEW:
                    //--- 메뉴를 선택했을 때
                    //--- 선택된 메뉴의 id가 저장되어 있다.
                    //--- New 메뉴 선택
                    MessageBox (hWnd, " 새파일을 열겠습니까 ?", "새파일 선택" , MB_OKCANCEL );
                    break;

                case ID_FILE_EXIT:
                    //--- Exit 메뉴 선택
                    answer = MessageBox (hWnd, "파일을 저장하고 끝내겠습니까 ?", "끝내기 선택", MB_YESNOCANCEL );
                    if (answer == IDYES || answer == IDNO)
                        PostQuitMessage (0);
                    break;
            }
        break;
    }
}
```

메시지박스

- 메시지박스: 사용자에게 경고나 알림 메시지를 주는 대화상자

int MessageBox (HWND hWnd, LPCTSTR lpText, LPCTSTR lpCaption, UINT uType);

- HWND hWnd: 부모 윈도우 핸들, 부모 윈도우가 없을 경우 NULL로 지정
- lpText: 메시지 박스에 표시될 글
- lpCaption: 메시지 박스의 타이틀바에 표시될 글
- uType과 함수 반환값

uType		반환 값
MB_OK	1개 버튼	IDOK
MB_OKCANCEL	2개 버튼	IDOK, IDCANCEL
MB_YESNO	2개 버튼	IDYES, IDNO
MB_YESNOCANCEL	3개 버튼	IDYES, IDNO, IDCANCEL

• 아이콘:

- 프로그램의 메인 윈도우가 최소화(아이콘화)되었을 때나 배경화면에 등록 될 때 응용 프로그램을 나타내는 작은 그래픽 이미지
- 손(IDI_HAND), 느낌표(IDI_WARNING) 등 9가지가 있으며 지금까지 사용한 표준 내장 아이콘(IDI_APPLICATION)은 다음과 같은 윈도우 모양을 갖는다.

값	아이콘모양	같은 값
IDI_APPLICATION (디폴트)		IDI_WINLOGO
IDI_ASTERISK		IDI_INFORMATION
IDI_ERROR		IDI_HAND
IDI_EXCLAMATION		IDI_WARNING
IDI_QUESTION		

```
wc.hIcon = LoadIcon ( NULL, IDI_APPLICATION );  
wc.hIconSm = LoadIcon ( NULL, IDI_APPLICATION );
```

HICON LoadIcon (HINSTANCE hInstance, LPCTSTR lplconName);

- 표준 아이콘 또는 리소스에 정의되어 있는 아이콘을 읽어온다.
- hInstance: 아이콘 리소스를 가지고 있는 인스턴스 핸들, 표준 아이콘 핸들일 경우 NULL
- lplconName: 읽을 아이콘을 가리키는 문자열 포인터

- 커서:
 - 마우스의 위치를 나타내는 작은 그래픽 이미지
 - 내장 커서는 아래의 표와 같은 모양이 제공된다.
 - 화살표, 모래시계 등 11가지가 있으며 지금까지 사용한 표준 내장 커서는 화살표 모양을 갖는다.

값	커서	모양
IDC_APPSTARTING	프로그램이 시작될 때 사용된다.	
IDC_ARROW	표준 화살표 커서	
IDC_CROSS	십자 모양의 커서. 정확한 선택을 해야 할 때 사용된다.	
IDC_IBEAM	I자 모양의 커서. 주로 문자열 입력 영역에 사용된다.	
IDC_ICON	Win32에서는 사용되지 않음	
IDC_NO	원 안의 빗금이 쳐진 커서이며 드래그 금지 영역을 나타낸다.	
IDC_SIZE	Win32에서는 사용되지 않음	
IDC_SIZEALL	4방향 화살표	
IDC_SIZENESW	좌하우상 크기조절 커서	
IDC_SIZENS	수직 크기조절 커서	
IDC_SIZENWSE	좌상우하 크기조절 커서	
IDC_SIZEWE	수평 크기조절 커서	
IDC_UPARROW	수직 화살표	
IDC_WAIT	모래 시계 커서. 시간이 오래 걸리는 작업을 할 때 사용된다.	

```
wc.hCursor = LoadCursor ( NULL ,IDC_ARROW);
```

HCURSOR LoadCursor (HINSTANCE hInstance, LPCTSTR lpCursorName);

- 표준 커서 또는 리소스에 정의되어 있는 커서를 읽어온다.
- hInstance: 커서 리소스를 가지고 있는 인스턴스 핸들, 표준 커서 핸들일 경우 NULL
- lpIconName: 읽을 커서를 가리키는 문자열 포인터

실습 4-1

- 3장의 실습 중 3-1 또는 3-2의 실습에 메뉴 붙이기
- 실습 3-1 원 꼬리 달기 실습에 메뉴 붙이기
 - 실습 3-1에 다음의 메뉴를 붙여 실행하도록 한다.
 - **Game**: Start/reset/end (원 이동하기 시작/보드를 다시 리셋/프로그램 종료)
 - **Speed**: Fast/Medium/Slow (원의 이동 속도가 빠르게/중간/느리게)
 - **Color**: Cyan/Magenta/Yellow (머리 원의 색상)
 - **Shape**: 원/네모 (머리 원의 모양을 원/네모)
 - **Number**: 20/25/30 (꼬리원의 개수)

또는

- 실습 3-2 벽돌깨기 실습에 메뉴 붙이기
 - 실습 3-2에 다음의 메뉴를 붙여 실행하도록 한다.
 - **Game**: Start/reset/end (원 움직이기 시작/벽돌 리셋/프로그램 종료)
 - **Speed**: Fast/Medium/Slow (원의 이동 속도가 빠르게/중간/느리게)
 - **Color**: Cyan/Magenta/Yellow (벽돌의 색상)
 - **Size**: Small/Big (튕기는 원의 크기)
 - **Number**: 3/4/5 (벽돌의 줄 개수)

실습 4-2

• 메뉴가 있는 그림판 만들기

- 화면에 보드 (모눈종이 형태)를 그린다. (기본 40 X 40 보드)
- 왼쪽 마우스를 누르고 드래그한 후 마우스를 놓으면 사각형이 그려진다. 최대 5개의 도형이 그려진다.
 - 도형은 보드의 칸에 맞춰져서 그려진다.
 - 마우스를 드래그할 때 래스터연산을 적용하여 고무줄효과가 있는 그리기로 실행한다.
 - 이미 색상이 있는 칸에 다시 그리기를 하면 색상이 합해져서 그려진다.
 - 예) 빨강색 (255, 0, 0) + 파랑색 (0, 0, 255) → 보라색 (255, 0, 255)

- 마우스 명령:

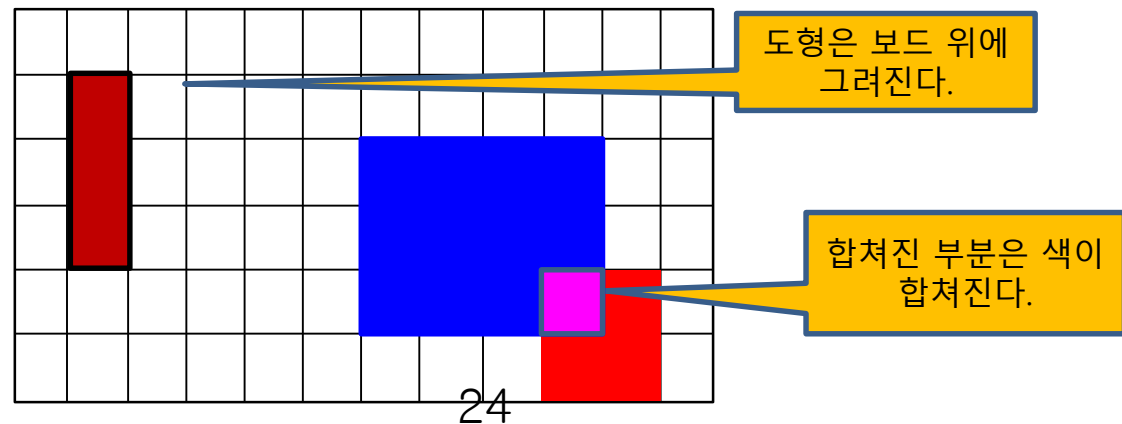
- **왼쪽 마우스:** 도형 그리기
- **오른쪽 마우스:** 도형이 그려진 칸에 오른쪽 마우스를 클릭하고 드래그 하면 도형의 위치가 변경된다. 변경된 도형도 보드 칸에 맞춰진다.

- 키보드 명령:

- **1/2/3/4/5:** 도형 그려진 순서대로 도형을 선택한다. 선택된 도형은 테두리에 표현한다.
- **좌우상하 화살표 키보드:** 선택된 도형을 좌우상하로 한 칸씩 이동하게 한다. 가장자리에 도달하면 반대편으로 나타난다.

- 메뉴:

- **Grid:** 실선/점선 (보드판의 선들을 실선/점선으로 그리기)
- **Color:** 본인이 설정한 6개의 색상
- **Border:** On/Off (도형의 테두리를 그리기/안그리기)
- **Move:** On/Off (오른쪽 마우스로 도형 위치 변경 가능/불가능)
- **Inform:** On/Off (화면에 도형의 개수, 각 도형의 가로세로 보드크기, 각 도형의 색상을 출력/지우기)



4장 학습 내용

- 학습내용
 - 메뉴 사용하기
 - 아이콘/커서 사용하기