<제2강 console 게임 제작>

7. 콘솔 함수 다루기(console.h, console.cpp)

1) 콘솔 창 관련

```
_DCRTIMP int __cdecl system(
    _In_opt_z_ char const* _Command
    );
```

: system("calc"), system("notepad")

① 제목 설정:

- system("title 제목"); ex. system("title test");
- SetConsoleTitle("제목"); ex. SetConsoleTitle(L"미로 게임"); SetConsoleTitle("my game");
  SetConsoleTitle(TEXT("겜프는 꿀잼"));

② 콘솔 창 크기 설정:

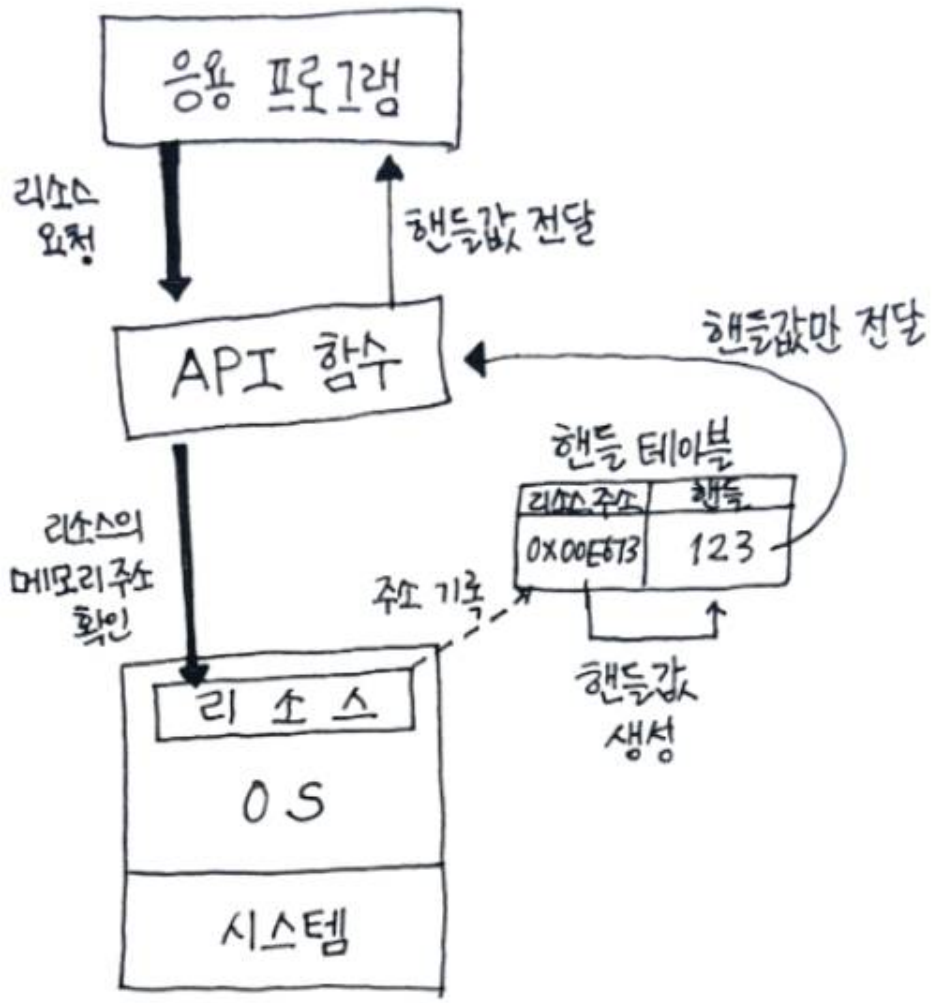
- system("mode con cols=가로길이 lines=세로길이");

=> 응용 system("mode con cols=가로길이 lines=세로길이 | title 제목");

ex. system("mode con cols=50 lines=50 | title test");

※ 가로가 세로보다 짧음. 가로를 50문자가 출력될 정도, 세로를 50줄이 출력될 정도. 띄어쓰기 주의!!

※ 핸들이란? 리소스의 메모리 주소를 정수로 치환한 값



③ 콘솔 창 전체화면 설정:

```
BOOL
APIENTRY
SetConsoleDisplayMode(
    _In_ HANDLE hConsoleOutput,
    _In_ DWORD dwFlags,
    _Out_opt_ PCOORD lpNewScreenBufferDimensions
    );
```

```
HANDLE
WINAPI
GetStdHandle(
    _In_ DWORD nStdHandle
    );
```

: 지정된 표준 디바이스에 대한 '핸들'을 검색함.
※ STD_OUTPUT_HANDLE: 표준 출력 디바이스

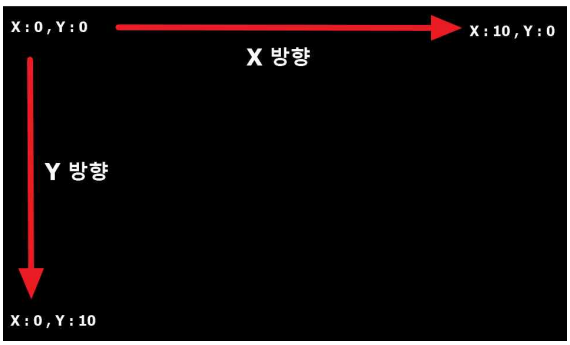
```
ex. void Fullscreen()
    { //CONSOLE_FULLSCREE_MODE가 전체화면모드, CONSOLE_WINDOWED_MODE가 윈도우모드
    //SetConsoleDisplayMode(GetStdHandle(STD_OUTPUT_HANDLE), CONSOLE_FULLSCREEN_MODE, 0);
            ShowWindow(GetConsoleWindow(), SW_MAXIMIZE);
    }
```

2) 프로그램 관련

- 종료: exit(0);
- 일시정지: system("pause");

3) 커서 제어

※ 콘솔 좌표계



① 좌표 이동 함수(Gotoxy)

```
BOOL
WINAPI
SetConsoleCursorPosition(
    _In_ HANDLE hConsoleOutput,
    _In_ COORD dwCursorPosition
    );
```

: "콘솔의 핸들 값"과 "좌표 값"을 받아서, 해당 위치로 콘솔의 커서를 이동시키는 함수.

```
typedef struct _COORD {
    SHORT X;
    SHORT Y;
} COORD, *PCOORD;
```

: 커서의 위치를 저장하는 구조체.
=> #include<Windows.h> 필요

```
ex. void Gotoxy(int x, int y)
    {
      HANDLE hOut = GetStdHandle(STD_OUTPUT_HANDLE); // 콘솔창 핸들
      COORD Cur = {x,y }; // {x*2, y};처럼 게임마다 2칸씩 이동하는게 자연스러울 수도 있음.
      SetConsoleCursorPosition(hOut, Cur);
    }
```

② 깜빡거리는 커서 보이지 않게 하기

```
BOOL
WINAPI
SetConsoleCursorInfo(
    _In_ HANDLE hConsoleOutput,
    _In_ CONST CONSOLE_CURSOR_INFO* lpConsoleCursorInfo
    );
```

: 지정된 콘솔 화면에 대한 커서의 크기와 표시 유형을 설정함.

```
typedef struct _CONSOLE_CURSOR_INFO {
    DWORD   dwSize;
    BOOL    bVisible;
} CONSOLE_CURSOR_INFO, *PCONSOLE_CURSOR_INFO;
```

: 콘솔 커서의 정보를 저장하는 구조체.

```
ex. void Setcursor(bool _bVis, DWORD _size)
    {
      CONSOLE_CURSOR_INFO curinfo;
      curinfo.dwSize = _size; // 커서 굵기(1~100)
      curinfo.bVisible = _bVis; // True: 보임, FALSE: 숨김
      SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &curinfo);
    }
```

4) 색깔 칠하기

```
BOOL
WINAPI
SetConsoleTextAttribute(
    _In_ HANDLE hConsoleOutput,
    _In_ WORD wAttributes
    );
```

: "콘솔의 핸들 값"과 "색상 값"을 받아서 글자 색깔을 변경해주는 함수.

```
ex. void Setcolor(int color, int bgcolor)
      {
            SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), (bgcolor << 4) | color);
      }
```

- 글자 배경색은 글자색의 16배이다. ex. 글자색의 2와 글자 배경색의 32는 같은 색깔
- WORD는 short형(2바이트=16비트)임. 배경색상을 바꾸려면 5~8번째 4개의 비트를 조작해야 함.

ex. 겜프는 허니잼 : 밝은 흰색 바탕에 연한 녹색 글씨 => 0000 0000 1010 1111

※ 색상표 => 아래와 같이 enum class 활용!

```
enum class COLOR
{
    BLACK, BLUE, GREEN, SKYBLUE, RED,
    VOILET, YELLOW, LIGHT_GRAY, GRAY, LIGHT_BLUE,
    LIGHT_GREEN, MINT, LIGHT_RED, LIGHT_VIOLET, LIGHT_YELLOW, WHITE
};
```

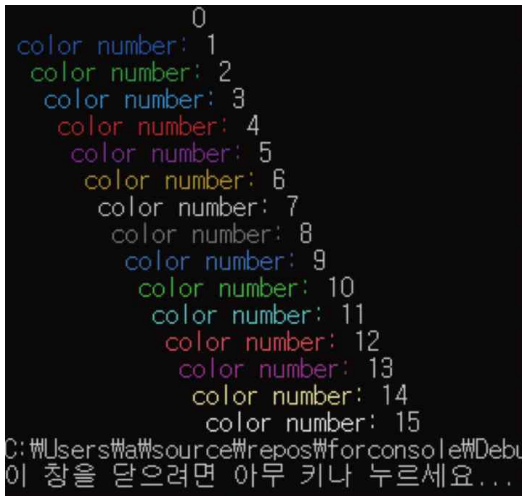
0: 검은색, 1: 파란색, 2: 초록색, 3: 하늘색, 4: 빨간색, 5: 자주색, 6: 노란색, 7: 연한 회색 8: 회색
9: 연한 파란색, A: 연한 초록색, B: 민트색, C: 연한 빨간색, D: 연한 자주색, E: 연한 노란색, F: 흰색(기본)

< 예제 ② Gotoxy()와 Setcolor() 함수로 아래 화면을 만들어보세요. >

- 실행 후 첫 화면

→

- 3초 후 화면



5) 키보드 처리

① _getch(), _putch()

↑: -32 -> 72

←: -32 -> 75

→: -32 -> 77

↓: -32 -> 80

※ 스페이스바: 32, 엔터키: 13

=> 두 개 이상의 키를 한번에 입력받지 못해 대각선을 못함.

Q) _kbhit()는 왜 써야 할까??

=> _getch()로 입력을 받는 과정에서 키보드가 입력되지 않으면 키보드 입력을 기다리면서 입력할 때까지 게임이 진행되지 않는 문제점이 생김.

② GetAsyncKeyState: 동시에 입력받을 때 사용. ex. 대각선 이동

```
SHORT
WINAPI
GetAsyncKeyState(
    _In_ int vKey);
```

: 인자 값으로 키보드의 가상키코드를 받음.

- 키가 눌려진 상태에서는 최상위 비트(0x8000)이 1이 되며, 처음 입력되었을 때는 0x8001 비트가 1임.
- 이 함수는 키의 상태에 따라 아래와 같이 비트형태로 반환함.

ex. if (GetAsyncKeyState(VK_UP) & 0x8000)

- 가상 키코드

값	가상키 코드	설명
0x01	VK_LBUTTON	마우스 왼쪽 버튼
0x02	VK_RBUTTON	마우스 오른쪽 버튼
0x04	VK_MBUTTON	마우스 가운데 버튼
0x08	VK_BACK	Backspace
0x09	VK_TAB	Tab
0x0D	VK_RETURN	Enter
0x10	VK_SHIFT	Shift
0x11	VK_CONTROL	Ctrl
0x12	VK_MENU	Alt
0x1B	VK_ESCAPE	Esc
0x20	VK_SPACE	Space Bar
0x21	VK_PRIOR	Page Up
0x22	VK_NEXT	Page Down
0x23	VK_END	End
0x24	VK_HOME	Home
0x25	VK_LEFT	←
0x26	VK_UP	↑
0x27	VK_RIGHT	→
0x28	VK_DOWN	↓
...		

반환 값	설명
0(0x0000)	이전에 누른 적이 없고 호출 시점에서 안눌린 상태
0x8000	이전에 누른 적이 없고 호출 시점에서 눌린 상태
0x8001	이전에 누른 적이 있고 호출 시점에서 눌린 상태
1(0x0001)	이전에 누른 적이 있고 호출 시점에서 안눌린 상태

=> 이 시점을 판단하는 기준은! 이전 GetAsyncKeyState 후 ~ 호출시의 GetAsyncKeyState의 바로 전까지의 기간임.

※ 시간 관련 함수

- clock(): c 함수(#include<time.h>). 응용 프로그램이 시작된 이후 CPU 틱수를 반환함.(ms)
- GetTickCount64(): win32 api 함수(#include<Windows.h>). 시스템이 시작된 이후 경과된 시간을 반환함. 1초에 1000씩 틱 카운트를 증가시킴.

※ 64가 붙은 이유: 윈도우가 시작되고 1초에 1000틱씩 카운트를 증가시키는데, 카운트는 32비트 값이라. 최대 49.7일간만 유지할 수 있어 이후에는 오버플로우라 0으로 초기화됨.
64비트로 증가하여 오버플로우 시점이 약 5억 8천정도로늘어남.