16. 사용자 정의형, 열거형

- 형을 정의할 때 사용하는 키워드에는 다음과 같은 것들이 있습니다.

  - typedef
  - enum
  - struct
  - union
  - class

typedef(TYPE DEFinition)

• typedef를 이용하여 byte를 아래와 같이 정의할 수 있습니다.

typedef unsigned char **byte**;

• typedef는 형을 정의하는 문장(statement)이므로 끝에 세미콜론 ;이 필요합니다. 마지막의 단어가 형으로 정의된다는 것을 주의하세요.

typedef unsigned int WORD TYPE;

• 위 문장은 unsigned int WORD를 TYPE으로 정의하려하기 때문에 컴파일 시간 에러가 발생합니다. 아래의 문장은 byte를 정의합니다.

typedef char **byte**;

• 길이가 40인 문자열을 관리하기 위해 41바이트의 문자열 배열을 형으로 정의할 수 없을까요? 아래와 같이 str40이라는 형을 정의할 수 있습니다.

```
typedef char str40[41];
```

• 그러므로 크기가 41인 문자 배열 s를 다음과 같이 선언할 수 있습니다.

```
str40 s;
```

• 좀 더 특별한 형 정의는 함수 포인터 형(function pointer type)을 정의하는 경우에도 발생합니다. 이것은 '**함수 포인터**'에서 자세히 다루도록 하겠습니다.

```
typedef long (*Fun)(long,long);
```

```
#include <stdio.h>
#include <conio.h>
#include <string.h>

typedef unsigned char byte;
typedef char str40[41];
typedef int (*Fun)();

void main() {
    byte i=65;//unsigned char i=65;와 같다.
    str40 s;//char s[41];와 같다.
    Fun f;//int (*f)();와 같다.

    strcpy(s,"Hello World");
    f=getch;//getch()의 시작 주소를 f에 대입한다.
    printf("%c,%s\n",i,s);
    (*f)();//f가 가리키는 곳, 즉 getch()를 호출한다.
}
```

enum: 열거형

• enum은 **열거형(ENUMeration type)**을 정의할 수 있고, 열거형 상수를 정의할 수도 있습니다. 또한 열거형 변수를 선언할 수도 있습니다.

enum [<tag_name>] {<constant_name> [= <value>], ...} [var_list];

• 아래의 문장은 sun부터 sat까지를 0부터 6까지 초기화합니다.

```
enum {sun, mon, tue, wed, thu, fri, sat};
```

• 이것은 다음과 같이 명시적인 초기화를 사용한 문장과 동일합니다.

```
enum {sun=0, mon=1, tue=2, wed=3, thu=4, fri=5, sat=6};
```

- 아래와 같은 #define 매크로 문장으로도 똑 같은 효과를 낼 수 있습니다.

```
#define  sun      0
#define  mon      1
#define  tue      2
#define  wed      3
#define  thu      4
#define  fri 5
#define  sat      6
```

• enum을 사용하는 몇 가지 예를 들어보면, 다음과 같습니다.

■ enum {FALSE, TRUE} b;

FALSE와 TRUE 상수를 선언하고, 변수 b를 선언합니다. b는 FALSE 혹은 TRUE값을 가질 수 있습니다.

■ enum BOOLEAN {FALSE2, TRUE2} b2,b3;

FALSE2와 TRUE2 상수를 선언하고, BOOLEAN 형을 정의했으며, BOOLEAN형의 변수 b2,b3을 선언했습니다.

■ enum BOOLEAN2 {FALSE3, TRUE3};

변수는 선언하지 않고 FALSE3과 TRUE3을 상수로 선언했으며, BOOLEAN2 형을 정의했습니다. 이제 BOOLEAN2 i,j; 처럼 변수 선언을 할 수 있습니다.

```
#include <stdio.h>

enum {sun, mon, tue, wed, thu, fri, sat};
enum {FALSE, TRUE} b;
enum BOOLEAN {FALSE2, TRUE2} b2,b3;
enum BOOLEAN2 {FALSE3, TRUE3};

void main() {
    char str[][6]={"false","true"};

    b=FALSE;
    b2=FALSE2;
    b3=TRUE2;

    BOOLEAN2 bValue;

    bValue=FALSE3;

    printf("%s,%s,%s,%s\n",str[b],str[b2],str[b3],str[bValue]);
```

```
    printf("%d,%d,%d\n",sun,sat,TRUE);
}//main
```

• 결과는 다음과 같습니다.

false,false,true,false
0,6,1

• enum의 상수값의 초기화가 생략되었을 때, 상수 값은 '*이전값+1*'로 초기화됩니다. 아래의 예에서 B,C,BB와 BBB는 각각 2,3, 11과 101로 초기화됩니다.

enum {A=1, B, C, AA=10, BB, AAA=100, BBB};

컴파일러의 입장

• enum형의 변수가 enum에서 정의한 상수 이외의 값을 가질 수 있을까요?

```
enum BOOLEAN {FALSE, TRUE};
BOOLEAN b;
```

• 위의 예에서 처럼 선언되었을 때, b에는 어떤 값을 대입할 수 있을까요? 물론 FALSE와 TRUE를 대입할 수 있으며, 각각은 정수 0과 1과 같습니다. 그렇다면 b에 100을 대입하는 다음의 문장은 어떤가요?

```
b=100;
```

클래스에서 enum의 사용

• 아래의 예제는 클래스 안에서 선언된 const, static과 enum상수를 초기화하는 방법을 보여주고 있습니다.

```
#include <iostream>
#include <stdio.h>

class CTest {
public:
    const int c;
    enum { A=100, B=200 };
    int i;
    static int s;
    static const int s2 = 30;
public:
    CTest(int t=0):c(10) {
        i=t;
```

```
    }
    void Print() {
        printf("i=%d\n",i);
        printf("c=%d\n",c);
        printf("s=%d\n",s);
    }
};//class CTest

int CTest::s=20;

void main() {
    CTest c;

    c.Print();
    printf("%d\n",CTest::A);
}//main
```

실습문제

1. MFC(Microsoft Foundation Class)의 AppWizard가 생성한 코드를 보면, 대화상자(dialog box)의 ID를 위해 class 안에서 enum을 사용합니다. 왜 MFC의 설계자들은 이러한 방법을 택한 것일까요?

2. enum으로 상수를 정의하면 #define으로 상수를 정의하는 것보다, 어떤 점에서 이득인가요?(힌트: 문서화)

3. 클래스를 정의하면서, 클래스 안에서 상수를 정의하는 방법이 enum말고도 있을까요?

@