

C Programming Language

$$C = \sum f(x)$$

Part II

Object oriented programming in C

<어두운 밤 AT & T 벨 연구소 근처>



<스컬리, 멀더의 차에서 맞은편에 헤드라이트가 켜져 있는 차가 보인다>

<멀더와 스컬리는 차에서 내리고 맞은편으로 향한다>

<맞은편 차에서 문이 열리고 중절모와 긴 코트를 입은 남자가 내리고는 담배를 물고 피우기 시작한다>

멀더 당신이 나를 보자고 했습니까?

담배 피는 남자 글썄, 나보다는 자네가 나를 보고 싶어 했을 것 같구만, 멀더.
 뭔가 골치 아픈 일이 있지 않은가?

멀더 그 말이 맞다고 칩시다.
 지금 골치 아픈 일이 있긴 하죠.

 그걸 해결할 뭔가를 당신이 알고 있을까 싶군요.

담배 피는 남자 자네가 USB 메모리를 갖고 있기에 여기까지 왔겠군.
 거기에 Part I 이라고 있었지 아마?

멀더 그렇군요.
 그럼 당신이 Part II 라도 갖고 있다는 생각이 드는군요.
 그 Part II 가 문제의 실마리를 해결할 수 있다는 뜻이겠군요.

담배 피는 남자 빙고!
 자네에게 Part II USB 를 넘겨 줄 수는 있네.
 단, 조건이 있네.

멀더 무슨 조건이 필요한 지 모르겠군요.
 여튼 조건을 말씀해 보세요.

담배 피는 남자 이 내용을 책으로 썼으면 하네.
 많은 이들이 이 내용을 알았으면 하네.
 그것뿐이네.

멀더 책을 쓸 시간이 있을까 싶습니다만.
 책으로 쓸만한 내용이라면 기꺼이 그렇게 하겠습니다.

담배 피는 남자 자, 여기 있네 (코트 안주머니에서 USB 메모리 하나를 꺼내 멀더에게 건넨다)
 내 조건을 들어 줄 만한 내용이기를 바라네.

그럼 잘 가게. (담배를 끄고 자기 차로 돌아가 사라진다)

멀더 Part II 라 ...

스컬리 멀더, 저 사람을 정말 믿어도 될까요?

 Part II 라니요. C 언어는 Part II 가 없는 걸로 알고 있어요.

멀더 믿고 말고는 이제 우리 손에 달렸군요.

 사무소를 돌아가 우리의 믿음을 판단해 봅시다.

<스컬리, 멀더는 차로 돌아가 다시 사무소로 돌아온다>

#Scene 1 스컬리 & 멀더의 탐정 사무소

<멀더가 계단을 후다닥 올라 사무실 문을 열고 있다. 스컬리는 숨을 헐떡거리며 오르고 있다>

스컬리 멀더, 좀 천천히 가요.

멀더 스컬리, 시간이 없어요. 빨리 올라와요.

(문을 열쇠로 열고 자신의 PC 를 켜며 USB 장치를 꽂는다)

(숨을 헐떡이며) 빨리 빨리.

스컬리 멀더, 조바심은 금물이라고 했죠.

(역시 숨을 헐떡이며) PC 가 부팅이 다 될 때까지 기다려요.

<USB 저장소 안에는 part_2 폴더가 보인다>

<part21 폴더 안에는 c_sigma_fx_part_2_00 ~ c_sigma_fx_part_2_01 폴더들이 보인다>

#Scene 2

[c_sigma_fx_part_2_00](#)

<c_sigma_fx_part_2_00.c>

```
#include <stdio.h>
#include "Sum.h"

void    main()
{
    TYPE_SUM Sum;
    TYPE_LPSUM pSum;

    Sum_Init(&Sum);
    Sum_A_B(&Sum, 2, 7);

    printf("%d + %d = %d\n", Sum.nA, Sum.nB, Sum.nSum);

    pSum = Sum_New();
    Sum_Init(pSum);
    Sum_A_to_B_Fast(pSum, 1, 7);

    printf("%d + ... + %d = %d\n", pSum->nA, pSum->nB, pSum->nSum);

    Sum_Delete(pSum);
}
```

<Sum.h>

```
#ifndef __SUM_H__
#define __SUM_H__

#include "Define.h"

////////////////////////////////////
// Attributes
////////////////////////////////////

typedef struct tagSum
{
    int nA;
    int nB;
```

```

        int nSum;
    } TYPE_SUM, *TYPE_LPSUM;

    //////////////////////////////////////
    // Methods
    //////////////////////////////////////

    TYPE_LPSUM Sum_New();
    void      Sum_Delete(TYPE_LPSUM pSum);
    void      Sum_Init(TYPE_LPSUM pSum);
    void      Sum_A_B(TYPE_LPSUM pSum, int nA, int nB);
    void      Sum_A_to_B(TYPE_LPSUM pSum, int nA, int nB);
    void      Sum_A_to_B_Fast(TYPE_LPSUM pSum, int nA, int nB);
    int       Sum_GetResult(TYPE_LPSUM pSum);

    #endif // !_SUM_H_

```

<Sum.c>

```

#include <malloc.h>
#include <memory.h>
#include <string.h>
#include "Sum.h"

    //////////////////////////////////////
    // Attributes
    //////////////////////////////////////

    //////////////////////////////////////
    // Methods - declaration
    //////////////////////////////////////

    void      Sum_Two(TYPE_LPSUM pSum);
    void      Sum_Set_A_B_Ordering(TYPE_LPSUM pSum, int nA, int nB);
    void      Sum_Range(TYPE_LPSUM pSum);
    void      Sum_Range_Fast(TYPE_LPSUM pSum);

    //////////////////////////////////////
    // Methods - public
    //////////////////////////////////////

```

```

TYPE_LPSUM Sum_New()
{
    TYPE_LPSUM pTemp = (TYPE_LPSUM)malloc(sizeof(TYPE_SUM));
    if (null == pTemp)        return null;

    memset(pTemp, 0, sizeof(*pTemp));

    return pTemp;
}

void Sum_Delete(TYPE_LPSUM pSum)
{
    SAFE_FREE(pSum);
}

void Sum_Init(TYPE_LPSUM pSum)
{
    if (null == pSum) return;

    memset(pSum, 0, sizeof(*pSum));
}

void Sum_A_B(TYPE_LPSUM pSum, int nA, int nB)
{
    if (null == pSum) return;

    pSum->nA = nA;
    pSum->nB = nB;

    Sum_Two(pSum);
}

void Sum_A_to_B(TYPE_LPSUM pSum, int nA, int nB)
{
    if (null == pSum) return;

    Sum_Set_A_B_Ordering(pSum, nA, nB);
}

```



```

        Sum_Range(pSum);
    }

void    Sum_A_to_B_Fast(TYPE_LPSUM pSum, int nA, int nB)
{
    if (null == pSum) return;

    Sum_Set_A_B_Ordering(pSum, nA, nB);

    Sum_Range_Fast(pSum);
}

int      Sum_GetResult(TYPE_LPSUM pSum)
{
    return pSum->nSum;
}

/////////////////////////////////////////////////////////////////
// Methods - private
/////////////////////////////////////////////////////////////////

void    Sum_Two(TYPE_LPSUM pSum)
{
    pSum->nSum = pSum->nA + pSum->nB;
}

void    Sum_Set_A_B_Ordering(TYPE_LPSUM pSum, int nA, int nB)
{
    if (nA < nB)
    {
        pSum->nA = nA;
        pSum->nB = nB;
    }
    else
    {
        pSum->nA = nB;
        pSum->nB = nA;
    }
}

```

```

void Sum_Range(TYPE_LPSUM pSum)
{
    pSum->nSum = 0;

    for (int i = pSum->nA; i <= pSum->nB; i++)
    {
        pSum->nSum += i;
    }
}

void Sum_Range_Fast(TYPE_LPSUM pSum)
{
    int nAdd = pSum->nA + pSum->nB;
    int nRange = (pSum->nB - pSum->nA + 1) / 2;

    pSum->nSum = nAdd * nRange;

    if ((pSum->nB - pSum->nA) != (nRange * 2))        return;

    pSum->nSum += (nAdd / 2);
}

```

- 멀더 main 함수에는 별다른 건 없어 보이는군요.
 다만, TYPE_SUM, TYPE_LPSUM 이 보이는군요.
 이름으로 보아 합산을 하는 것으로 보이는군요.
- 스컬리 TYPE_SUM 등으로 보아 사용자 정의 데이터 형 이네요.
 typedef 으로 선언했을 꺼예요.
 Sum_A_B(&Sum, 2, 7) 로 보아 2 와 7 의 합산이겠쥬. 답은 뭐 9 이고요.
 Sum_A_to_B_Fast(pSum, 1, 7) 는 1 에서부터 7 까지의 합산이고요.
 답은 음... 28 일거예요.
- 멀더 역시 내 파트너 답군요. 28 맞습니다.
 TYPE_SUM 은 합산에 왜 필요한 걸 까요?
 Sum 함수 하나면 충분해 보이는데요. Sum_A_to_B_Fast 도 마찬가지고요.
 이해할 수 없군요.
- 스컬리 아마도 Sum.h, Sum.c 에 해답이 있을꺼예요.
 같이 살펴 보아요.

멀더 Sum.h 를 보니 스컬리 말이 맞군요.

```
typedef struct tagSum
{
    int nA;
    int nB;
    int nSum;
} TYPE_SUM, *TYPE_LPSUM;
```

Typedef 으로 시작해서 TYPE_SUM, TYPE_LPSUM 인걸 보면
TYPE_SUM 은 새로운 사용자 데이터 형이군요.
TYPE_LPSUM 은 TYPE_SUM 의 포인터 형이군요.

함수에서는

TYPE_LPSUM Sum_New() 이 있군요. 이걸 왜 필요한 건가요?

스컬리 Sum_New() 함수는 동적 할당 이랍니다.

저번에 이야기한 것처럼 프로세스 메모리 중 스택이 아니라 힙에 할당하는 거에
요. 스택은 오버플로우가 날 수 있으니 최대한 힙 메모리를 사용해야 한답니다.

Sum_Delete() 함수에서 할당된 메모리를 해제하는 거고요.

할당된 메모리를 제대로 해제하지 않으면 메모리 누수(Memory Leak) 현상이 발
생해요. 그럼 힙 메모리가 점점 부족해져서 나중에는 메모리 할당이 안된답니다.
꼭 염두에 두어야 해요. 멀더.

멀더 그렇군요. 스컬리. 꼭 염두에 둘게요.

그리고 주석을 보니까

```
// Attributes
```

```
// Methods
```

라는 것이 보이는군요.

이 주석은 무엇을 의미하는 걸까요?

스컬리 내, 멀더. 아주 중요한 부분을 지적했어요.

위의 주석은 객체 지향 언어에서 볼 수 있는 거예요.

```
// Attributes
```

라는 것은 객체의 속성을 의미하는 거예요.

즉, 구조체를 객체로 인식한다고 보면 되요.

```
// Methods
```

라는 것은 객체의 행동을 의미하는 거예요.

Sum_A_B() 는 이제 함수가 아니라 A,B 두 개의 정수를 합산하는 메서드가 되는
거예요.

Sum_A_to_B() 는 A ~ B 까지의 정수 합산을 정의한 메서드이고요.

즉, 구조체는 객체 속성, 함수는 객체 메서드로 재정의하는 거예요.

C 언어로도 객체 지향적으로 프로그래밍할 수 있다는 거예요.

멀더 아! 그렇군요.
 C 언어로도 객체 지향적 프로그래밍이 가능해졌군요.
 그럼 객체 지향적 프로그래밍의 장점을 알아야 겠군요. 스컬리.
 스컬리 물론, C 언어로 완벽한 객체 지향적 프로그래밍을 할 수는 없어요.
 그럼, C++ 나 JAVA 를 배울 필요가 없겠죠.
 하지만 C 언어로 개발해야 하는 환경에서는 아주 유용할 거예요.
 다음과 같은 장점이 있을꺼 같네요.


1) 코드의 이해도 상승
 이전에 보았던 스파게티 코드를 탈피할 수 있다는 거예요.
 내가 코팅한 소스를 다른 프로그래머가 쉽게 이해할 수 있다는 장점이 있어요.
 이건 아주 중요한 부분이에요.

2) 유지/보수의 상승
 요즘의 프로그램들은 대부분 대규모의 프로젝트로 이루어져 있어요.
 코드의 라인(Line) 수가 수만 라인에서 수십만 라인 이상으로 이루어져 있어요.
 만약에 모든 코드가 C 언어로 되어 있고 스파게티 코드라면 상상만 해도 아찔
 해져요.
 하지만, 모든 코드가 C 언어로 되어 있다 하더라도 객체 지향적으로 되어 있다
 면 훨씬 수월하게 수정이나 업그레이드가 가능하겠네요.

3) 대규모 코드 제작 가능
 위의 1)과 2)의 설명에서 이어 이제 C 언어로도 대규모 프로젝트 코드를 작성할
 수 있게 된다는 거예요. 정말 대단한 거예요.

4) 객체 지향 언어의 이해
 뿐만 아니라, C 언어의 객체 지향적 프로그래밍 방법을 통해 C++ 이나 JAVA 의
 객체 지향적 특성을 보다 쉽게 이해할 수 있다는 거예요.
 물론 C++, JAVA 의 객체 지향적 특성은 더 무한하지만요.
 C 언어에서 객체 지향적 특성을 보다 근본적으로 이해할 수 있다면 멋지겠네요.
 멀더 C 언어를 통해 객체 지향적 특성을 이해할 수 있다는 이야기이군요.
 이것이 C 언어가 단순히 구조적 프로그래밍만 가능한 게 아니라 객체 지향적
 프로그래밍도 어느 정도는 가능하다는 거고요. 그리고 객체 지향적 특성을 이해
 할 수 있어 다른 객체 지향 언어를 배우는 데에도 도움이 된다는 거군요.
 이제 그것을 알게 되었네요.
 하하하
 스컬리 호호호

<출력 결과>



```
D:\Research\c_sigma_fx\part_2\c_sigma_fx_part_2_00\Debug\c_sigma_fx_part_2_00.exe
2 + 7 = 9
1 + ... + 7 = 28
```

#Scene 3 [c_sigma_fx_part_2_01_drinks](#)

<c_sigma_fx_part_2_01_drinks.c>

```
#include "DrinksMachine.h"

void main(void)
{
    TYPE_DRINKSMACHINE DrinksMachine;

    if (DrinksMahcine_Open(&DrinksMachine))
    {
        DrinksMahcine_Run(&DrinksMachine);
        DrinksMahcine_Close(&DrinksMachine);
    }
}
```

<Drink.h>

```
#ifndef __DRINK_H__
#define __DRINK_H__

#include "Define.h"

/////////////////////////////////////////////////////////////////
// Attributes
/////////////////////////////////////////////////////////////////

#define DRINK_NAME_MAX      256

typedef struct tagDrink
{
    char    szName[DRINK_NAME_MAX];
    int     nPrice;
    int     nProfit;
} TYPE_DRINK, *TYPE_LPDRINK;

/////////////////////////////////////////////////////////////////
// Methods
/////////////////////////////////////////////////////////////////
```

```

TYPE_LPDRINK    Drink_New();

void              Drink_Delete( TYPE_LPDRINK pDrink );


boolean Drink_Open( TYPE_LPDRINK pDrink );
void        Drink_Close( TYPE_LPDRINK pDrink );


void        Drink_Set( TYPE_LPDRINK pDrink, char* pszName, int nPrice, int nProfit );


////////////////////////////////////
#endif // !__DRINK_H__

```

<Drink.c>

```

#include "Drink.h"
#include <malloc.h>
#include <string.h>


////////////////////////////////////
// Attributes
////////////////////////////////////


////////////////////////////////////
// Methods
////////////////////////////////////


TYPE_LPDRINK    Drink_New()
{
    TYPE_LPDRINK    pTemp = (TYPE_LPDRINK)malloc( sizeof(TYPE_DRINK) );
    if ( null == pTemp )        return null;

    memset( pTemp, 0, sizeof(*pTemp) );

    return pTemp;
}


void              Drink_Delete( TYPE_LPDRINK pDrink )
{
    SAFE_FREE( pDrink );
}

```

```

boolean Drink_Open( TYPE_LPDRINK pDrink )
{
    if ( null == pDrink )        return false;

    memset( pDrink->szName, 0, sizeof(pDrink->szName) );
    pDrink->nPrice = 0;
    pDrink->nProfit = 0;

    return true;
}

void    Drink_Close( TYPE_LPDRINK pDrink )
{
    if ( null == pDrink )        return;

    memset( pDrink, 0, sizeof(*pDrink) );
}

void    Drink_Set( TYPE_LPDRINK pDrink, char* pszName, int nPrice, int nProfit )
{
    strcpy_s( pDrink->szName, sizeof(pDrink->szName), pszName );
    pDrink->nPrice = nPrice;
    pDrink->nProfit = nProfit;
}

```

<DrinkSlot.h>

```

#ifndef __DRINKSLOT_H__
#define __DRINKSLOT_H__

#include "Drink.h"

////////////////////////////////////
// Attributes
////////////////////////////////////

#define DRINKSLOT_DRINK_MAX      10

```



```

typedef struct tagDrinkLink*      TYPE_LPDRINKLINK;

typedef struct tagDrinkLink
{
    TYPE_LPDRINK      pDrink;
    TYPE_LPDRINKLINK  pNext;

} TYPE_DRINKLINK;

typedef struct tagDrinkSlot
{
    TYPE_LPDRINKLINK  pBottom;
    TYPE_LPDRINKLINK  pTop;

    int               nCount;

} TYPE_DRINKSLOT, *TYPE_LPDRINKSLOT;

////////////////////////////////////
// Methods
////////////////////////////////////

TYPE_LPDRINKSLOT      DrinkSlot_New();
void                  DrinkSlot_Delete( TYPE_LPDRINKSLOT pDrinkSlot );

boolean DrinkSlot_Open( TYPE_LPDRINKSLOT pDrinkSlot );
void      DrinkSlot_Close( TYPE_LPDRINKSLOT pDrinkSlot );

char*      DrinkSlot_GetDrinkName( TYPE_LPDRINKSLOT pDrinkSlot );
int        DrinkSlot_GetDrinkPrice( TYPE_LPDRINKSLOT pDrinkSlot );

boolean DrinkSlot_IsEmpty( TYPE_LPDRINKSLOT pDrinkSlot );
boolean DrinkSlot_IsFull( TYPE_LPDRINKSLOT pDrinkSlot );
int      DrinkSlot_GetCount( TYPE_LPDRINKSLOT pDrinkSlot );

void      DrinkSlot_AddDrink( TYPE_LPDRINKSLOT pDrinkSlot, TYPE_LPDRINK pDrink );
TYPE_LPDRINK      DrinkSlot_OutputDrink( TYPE_LPDRINKSLOT pDrinkSlot );

////////////////////////////////////

```

```
#endif // !_DRINKSLOT_H_
```

<DrinkSlot.c>

```
#include "DrinkSlot.h"
#include <malloc.h>
#include <string.h>

////////////////////////////////////
// Attributes
////////////////////////////////////

////////////////////////////////////
// Methods
////////////////////////////////////

TYPE_LPDRINKLINK      DrinkLink_New()
{
    TYPE_LPDRINKLINK pTemp = (TYPE_LPDRINKLINK)malloc( sizeof(TYPE_DRINKLINK) );
    if ( null == pTemp )      return null;

    memset( pTemp, 0, sizeof(*pTemp) );

    return pTemp;
}

void                  DrinkLink_Delete( TYPE_LPDRINKLINK pDrinkLink )
{
    SAFE_FREE( pDrinkLink->pDrink );
    SAFE_FREE( pDrinkLink );
}

void                  DrinkLink_AddDrink(      TYPE_LPDRINKLINK      pDrinkLink,
TYPE_LPDRINK pDrink )
{
    pDrinkLink->pDrink = pDrink;
}

TYPE_LPDRINK          DrinkLink_GetDrink( TYPE_LPDRINKLINK pDrinkLink )
```

```

{
    return pDrinkLink->pDrink;
}

/////////////////////////////////////////////////////////////////

TYPE_LPDRINKSLOT      DrinkSlot_New()
{
    TYPE_LPDRINKSLOT      pTemp                                =
(TYPE_LPDRINKSLOT)malloc( sizeof(TYPE_DRINKSLOT) );
    if ( null == pTemp )      return null;

    memset( pTemp, 0, sizeof(*pTemp) );

    return pTemp;
}

void                    DrinkSlot_Delete( TYPE_LPDRINKSLOT pDrinkSlot )
{
    SAFE_FREE( pDrinkSlot );
}

boolean DrinkSlot_Open( TYPE_LPDRINKSLOT pDrinkSlot )
{
    if ( null == pDrinkSlot )      return false;

    pDrinkSlot->pTop = null;
    pDrinkSlot->pBottom = null;

    pDrinkSlot->nCount = 0;

    return true;
}

void      DrinkSlot_Close( TYPE_LPDRINKSLOT pDrinkSlot )
{
    if ( null == pDrinkSlot )      return;

    memset( pDrinkSlot, 0, sizeof(*pDrinkSlot) );
}

```

```

}

char*   DrinkSlot_GetDrinkName( TYPE_LPDRINKSLOT pDrinkSlot )
{
    TYPE_LPDRINK pDrink;

    if ( null == pDrinkSlot )    return null;

    pDrink = DrinkLink_GetDrink( pDrinkSlot->pBottom );
    if ( null == pDrink )        return null;

    return pDrink->szName;
}

int      DrinkSlot_GetDrinkPrice( TYPE_LPDRINKSLOT pDrinkSlot )
{
    TYPE_LPDRINK pDrink;

    if ( null == pDrinkSlot )    return 0;

    pDrink = DrinkLink_GetDrink( pDrinkSlot->pBottom );
    if ( null == pDrink )        return 0;

    return pDrink->nPrice;
}

boolean DrinkSlot_IsEmpty( TYPE_LPDRINKSLOT pDrinkSlot )
{
    if ( null == pDrinkSlot )    return true;

    return pDrinkSlot->nCount <= 0;
}

boolean DrinkSlot_IsFull( TYPE_LPDRINKSLOT pDrinkSlot )
{
    if ( null == pDrinkSlot )    return true;

    return pDrinkSlot->nCount >= DRINKSLOT_DRINK_MAX;
}

```

```

int      DrinkSlot_GetCount( TYPE_LPDRINKSLOT pDrinkSlot )
{
    if (null == pDrinkSlot)    return 0;

    return pDrinkSlot->nCount;
}

void      DrinkSlot_AddDrink( TYPE_LPDRINKSLOT pDrinkSlot, TYPE_LPDRINK pDrink )
{
    TYPE_LPDRINKLINK pDrinkLink = DrinkLink_New();
    DrinkLink_AddDrink( pDrinkLink, pDrink );

    if ( null == pDrinkSlot->pTop )
    {
        pDrinkSlot->pTop = pDrinkLink;
        pDrinkSlot->pBottom = pDrinkLink;
    }
    else
    {
        pDrinkSlot->pTop->pNext = pDrinkLink;
        pDrinkSlot->pTop = pDrinkLink;
    }

    pDrinkSlot->nCount++;
}

TYPE_LPDRINK      DrinkSlot_OutputDrink( TYPE_LPDRINKSLOT pDrinkSlot )
{
    TYPE_LPDRINKLINK pDrinkLink = pDrinkSlot->pBottom;
    TYPE_LPDRINK pDrink = pDrinkLink->pDrink;

    if ( null == pDrinkLink )    return null;

    pDrinkSlot->pBottom = pDrinkLink->pNext;

    pDrinkLink->pDrink = null;
    DrinkLink_Delete( pDrinkLink );
}

```

```

        pDrinkSlot->nCount--;

        return pDrink;
    }

```

<DrinksMachineManager.h>

```

#ifndef __DRINKSMACHINEMANAGER_H__
#define __DRINKSMACHINEMANAGER_H__

#include "DrinksMachine.h"

/////////////////////////////////////////////////////////////////
// Attributes
/////////////////////////////////////////////////////////////////

typedef struct tagDrinksMachineManager
{
    boolean bOpened;

    int nMenu_Main;

} TYPE_DRINKSMACHINEMANAGER, *TYPE_LPDRINKSMACHINEMANAGER;

/////////////////////////////////////////////////////////////////
// Methods
/////////////////////////////////////////////////////////////////

TYPE_LPDRINKSMACHINEMANAGER      DrinksMachineManager_New();
void
    DrinksMachineManager_Delete(          TYPE_LPDRINKSMACHINEMANAGER
pDrinksMachineManager );

boolean DrinksMachineManager_Open(          TYPE_LPDRINKSMACHINEMANAGER
pDrinksMachineManager );
void    DrinksMachineManager_Run(          TYPE_LPDRINKSMACHINEMANAGER
pDrinksMachineManager, TYPE_LPDRINKSMACHINE pDrinksMachine );
void    DrinksMachineManager_Close(        TYPE_LPDRINKSMACHINEMANAGER
pDrinksMachineManager );

```

```

////////////////////////////////////
#endif // !__DRINK_H__

```

<DrinksMachineManager.c>

```

#include "DrinksMachineManager.h"
#include <stdio.h>
#include <malloc.h>
#include <string.h>
#include <conio.h>
#include <stdlib.h>

////////////////////////////////////
// Attributes
////////////////////////////////////

////////////////////////////////////
// Methods - Declarations
////////////////////////////////////

void    DrinksMachineManager_Menu_Main(          TYPE_LPDRINKSMACHINEMANAGER
pDrinksMachineManager, TYPE_LPDRINKSMACHINE pDrinksMachine );
void    DrinksMachineManager_DrinkSlot_Print(    TYPE_LPDRINKSMACHINE    pDrinksMachine,
boolean bManagement );

void    DrinksMachineManager_Menu_DrinkAdd(      TYPE_LPDRINKSMACHINEMANAGER
pDrinksMachineManager, TYPE_LPDRINKSMACHINE pDrinksMachine );
void    DrinksMachineManager_Menu_Sales(        TYPE_LPDRINKSMACHINEMANAGER
pDrinksMachineManager, TYPE_LPDRINKSMACHINE pDrinksMachine );

////////////////////////////////////
// Methods
////////////////////////////////////

TYPE_LPDRINKSMACHINEMANAGER    DrinksMachineManager_New()
{
    TYPE_LPDRINKSMACHINEMANAGER    pTemp    =
(TYPE_LPDRINKSMACHINEMANAGER)malloc( sizeof(TYPE_LPDRINKSMACHINEMANAGER) );
    if ( null == pTemp )    return null;

```

```

        memset( pTemp, 0, sizeof(*pTemp) );

        return pTemp;
    }

void
    DrinksMachineManager_Delete(                                TYPE_LPDRINKSMACHINEMANAGER
pDrinksMachineManager )
{
    SAFE_FREE( pDrinksMachineManager );
}

////////////////////////////////////////////////////////////////

boolean DrinksMachineManager_Open(                                TYPE_LPDRINKSMACHINEMANAGER
pDrinksMachineManager )
{
    if ( null == pDrinksMachineManager )        return false;

    pDrinksMachineManager->bOpened = true;

    return true;
}

void    DrinksMachineManager_Close(                                TYPE_LPDRINKSMACHINEMANAGER
pDrinksMachineManager )
{
    if ( null == pDrinksMachineManager )        return;
    if ( false == pDrinksMachineManager->bOpened )    return;

    memset( pDrinksMachineManager, 0, sizeof(*pDrinksMachineManager) );
}

////////////////////////////////////////////////////////////////

void    DrinksMachineManager_Run(                                TYPE_LPDRINKSMACHINEMANAGER
pDrinksMachineManager, TYPE_LPDRINKSMACHINE pDrinksMahchine )
{
    boolean bLoop;

```



```

do {
    bLoop = true;

    DrinksMachineManager_Menu_Main(
pDrinksMahchine );

    switch ( pDrinksMachineManager->nMenu_Main )
    {
        case 1:
            DrinksMachineManager_Menu_DrinkAdd(
pDrinksMahchine );
            break;
        case 2:
            DrinksMachineManager_Menu_Sales(
pDrinksMahchine );
            break;
        case 3: case 0:
            bLoop = false;
            break;
    }

} while ( bLoop );
}

////////////////////////////////////

void    DrinksMachineManager_Menu_Main(
TYPE_LPDRINKSMACHINEMANAGER
pDrinksMachineManager, TYPE_LPDRINKSMACHINE pDrinksMahchine )
{
    boolean bSelected = false;

    do {
        int nCh;

        printf( "[ 관리자 ] ===== 메뉴를 선택하세요 =====\n" );
        printf( "[ 관리자 ] 1. 음료 추가  2. 정산하기  3. 나가기  0. 전원 끄기\n" );

        bSelected = false;

```

```

        pDrinksMachineManager->nMenu_Main = 0;

        nCh = _getch();

        switch ( nCh )
        {
        case '1': case '2': case '3':
            pDrinksMachineManager->nMenu_Main = nCh - '0';
            bSelected = true;
            break;
        case '0':
            pDrinksMachineManager->nMenu_Main = 0;
            pDrinksMachineManager->bRun = false;
            bSelected = true;
            break;
        }

    } while ( false == bSelected );
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

void    DrinksMachineManager_Menu_DrinkAdd(                TYPE_LPDRINKSMACHINEMANAGER
pDrinksMachineManager, TYPE_LPDRINKSMACHINE pDrinksMachine )
{
    TYPE_LPDRINKSLOT pDrinkSlot;

    boolean bSelected = false;

    do {
        int nCh;

        printf( "[ 관리자 ] 음료 슬롯을 선택하세요\n" );
        DrinksMachineManager_DrinkSlot_Print( pDrinksMachine, false );

        bSelected = false;

        nCh = _getch();
    } while ( bSelected == false );
}

```

```

        if ( '0' == nCh )
        {
            bSelected = true;
        }
        else if ( 0 <= nCh - '1' && nCh - '1' < DRINKSLOT_MAX )
        {
            pDrinkSlot = &pDrinksMachine->DrinkSlot[ nCh - '1' ];
            if ( DrinkSlot_IsFull( pDrinkSlot ) )
            {
                printf( "슬롯이 꽉 찼습니다 !\n" );
                bSelected = true;
            }
            else
            {
                TYPE_DRINK aDrink;
                Drink_Open( &aDrink );

                printf( "[ 관리자 ] 음료 이름을 입력하세요\n" );
                scanf_s( "%s", aDrink.szName,
(unsigned)_countof( aDrink.szName ) );

                printf( "[ 관리자 ] 음료 가격(원)을 입력하세요\n" );
                scanf_s( "%d", &aDrink.nPrice );

                printf( "[ 관리자 ] 음료 이윤(원)을 입력하세요\n" );
                scanf_s( "%d", &aDrink.nProfit );

                do {
                    printf("[ 관리자 ] 음료를 추가하시겠습니까?\n1.예
2.아니오\n");

                    nCh = _getch();

                    if ( '1' == nCh )
                    {
                        TYPE_LPDRINK pDrink = Drink_New();
                        if ( null != pDrink )
                        {
                            Drink_Set( pDrink, aDrink.szName,

```

```

aDrink.nPrice, aDrink.nProfit );

DrinkSlot_AddDrink(      pDrinkSlot,
pDrink );

printf( "[ 관리자 ] 음료(총 %d개)가
추가되었습니다.\n", DrinkSlot_GetCount( pDrinkSlot ) );

if ( DrinkSlot_IsFull( pDrinkSlot ) )
{
    printf( "[ 관리자 ] 더 이상
추가할 수 없습니다.\n" );

    nCh = '2';
    bSelected = true;
}
}
else if ( '2' == nCh )
{
    bSelected = true;
}

} while ( '2' != nCh );

}

} while ( false == bSelected );
}

////////////////////////////////////

void      DrinksMachineManager_Menu_Sales(      TYPE_LPDRINKSMACHINEMANAGER
pDrinksMachineManager, TYPE_LPDRINKSMACHINE pDrinksMachine )
{
    printf( "[ 관리자 ] 정산 정보\n" );
    printf( "[ 관리자 ] 판매 총액 = %d 원\n", pDrinksMachine->nSales );
    printf( "[ 관리자 ] 이윤 총액 = %d 원\n", pDrinksMachine->nProfit );
}

```

```

////////////////////////////////////
void    DrinksMachineManager_DrinkSlot_Print(    TYPE_LPDRINKSMACHINE    pDrinksMachine,
boolean bManagement )
{
    int i;

    if ( bManagement )        printf( "[ 관리자 ] " );

    for ( i = 0; i < DRINKSLOT_MAX; i++ )
    {
        TYPE_LPDRINKSLOT pDrinkSlot = &pDrinksMachine->DrinkSlot[ i ];
        if ( DrinkSlot_IsEmpty( pDrinkSlot ) )
        {
            printf( "%d. 음료 없음  ", i + 1 );
        }
        else
        {
            printf( "%d. %s (%d원)  ", i + 1, DrinkSlot_GetDrinkName( pDrinkSlot ),
DrinkSlot_GetDrinkPrice( pDrinkSlot ) );
        }
    }

    if ( bManagement )        printf( "0. 관리자₩n" );
    else                        printf( "₩n" );
}

```

<DrinksMachine.h>

```

#ifndef __DRINKSMACHINE_H__
#define __DRINKSMACHINE_H__

#include "DrinkSlot.h"

////////////////////////////////////
// Attributes
////////////////////////////////////

#define DRINKSLOT_MAX 4

```

```

typedef struct tagDrinksMachine
{
    boolean bOpened;

    boolean bRun;

    int      nMoney;

    TYPE_DRINKSLOT DrinkSlot[ DRINKSLOT_MAX ];
    int      nSelecctDrinkSlot;

    boolean bSelectDrink;
    boolean bSelectManager;

    int      nSales;
    int      nProfit;

} TYPE_DRINKSMACHINE, *TYPE_LPDRINKSMACHINE;

////////////////////////////////////
// Methods
////////////////////////////////////

TYPE_LPDRINKSMACHINE DrinksMachine_New();
void                  DrinksMachine_Delete(    TYPE_LPDRINKSMACHINE
pDrinksMachine );

boolean DrinksMahcine_Open( TYPE_LPDRINKSMACHINE pDrinksMachine );
void    DrinksMahcine_Close( TYPE_LPDRINKSMACHINE pDrinksMachine );

void    DrinksMahcine_Run( TYPE_LPDRINKSMACHINE pDrinksMachine );

////////////////////////////////////
#endif // !__DRINKSMACHINE_H__

```

<DrinksMachine.c>

```

#include "DrinksMachine.h"
#include "DrinksMachineManager.h"
#include <malloc.h>

```

```

#include <memory.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>

////////////////////////////////////
// Attributes
////////////////////////////////////

////////////////////////////////////
// Methods - Declarations
////////////////////////////////////

boolean DrinksMachine_IsOpened( TYPE_LPDRINKSMACHINE pDrinksMachine );

void    DrinksMachine_DrinkSlot_Print(  TYPE_LPDRINKSMACHINE  pDrinksMachine,  boolean
bManagement );

void    DrinksMachine_Management( TYPE_LPDRINKSMACHINE pDrinksMachine );

void    DrinksMachine_SelectDrink( TYPE_LPDRINKSMACHINE pDrinksMachine );
void    DrinksMachine_CheckMoney( TYPE_LPDRINKSMACHINE pDrinksMachine );
void    DrinksMachine_DispenseDrink( TYPE_LPDRINKSMACHINE pDrinksMachine );

////////////////////////////////////

TYPE_LPDRINKSMACHINE DrinksMachine_New()
{
    TYPE_LPDRINKSMACHINE pTemp
    (TYPE_LPDRINKSMACHINE)malloc( sizeof(TYPE_LPDRINKSMACHINE) );
    if ( null == pTemp )        return null;

    memset( pTemp, 0, sizeof(*pTemp) );

    return pTemp;
}

```

```

void                                DrinksMachine_Delete(    TYPE_LPDRINKSMACHINE
pDrinksMachine )
{
    SAFE_FREE( pDrinksMachine );
}

boolean DrinksMahcine_Open( TYPE_LPDRINKSMACHINE pDrinksMachine )
{
    if ( null == pDrinksMachine )    return false;

    pDrinksMachine->bOpened = true;

    for ( int i = 0; i < DRINKSLOT_MAX; i++ )
    {
        DrinkSlot_Open( &pDrinksMachine->DrinkSlot[ i ] );
    }

    printf("음료 자동판매기를 시작합니다.\n");

    return true;
}

void    DrinksMahcine_Close( TYPE_LPDRINKSMACHINE pDrinksMachine )
{
    if ( false == DrinksMachine_IsOpened( pDrinksMachine ) )    return;

    pDrinksMachine->bOpened = false;

    for ( int i = 0; i < DRINKSLOT_MAX; i++ )
    {
        DrinkSlot_Close( &pDrinksMachine->DrinkSlot[ i ] );
    }

    printf("음료 자동판매기를 중지합니다.\n");
}

void    DrinksMahcine_Run( TYPE_LPDRINKSMACHINE pDrinksMachine )
{
    if ( false == DrinksMachine_IsOpened( pDrinksMachine ) )    return;

```



```

printf( "음료 자동판매기를 실행합니다.\n" );

pDrinksMachine->nMoney = 0;
pDrinksMachine->nSales = 0;
pDrinksMachine->nProfit = 0;

pDrinksMachine->bRun = true;
do {

    DrinksMachine_SelectDrink( pDrinksMachine );

    if ( pDrinksMachine->bSelectDrink )
    {
        DrinksMachine_CheckMoney( pDrinksMachine );
        DrinksMachine_DispendeDrink( pDrinksMachine );
    }
    else if ( pDrinksMachine->bSelectManager )
    {
        DrinksMachine_Management( pDrinksMachine );
    }

} while ( pDrinksMachine->bRun );
}

////////////////////////////////////////////////////////////////

boolean DrinksMachine_IsOpened( TYPE_LPDRINKSMACHINE pDrinksMachine )
{
    if ( null == pDrinksMachine )    return false;

    return ( pDrinksMachine->bOpened ) ;
}

////////////////////////////////////////////////////////////////

void    DrinksMachine_SelectDrink( TYPE_LPDRINKSMACHINE pDrinksMachine )
{
    boolean bSelected;

```

```

do {
    int nCh;
    int nSelectDrinkSlot;

    printf( "===== 음료를 선택하세요 =====\n" );

    DrinksMachine_DrinkSlot_Print( pDrinksMachine, true );

    bSelected = false;
    pDrinksMachine->bSelectDrink = false;
    pDrinksMachine->bSelectManager = false;

    nCh = _getch();

    if ('0' == nCh)
    {
        bSelected = true;
        pDrinksMachine->bSelectManager = true;
    }
    else
    {
        nSelectDrinkSlot = nCh - '1';

        if (0 <= nSelectDrinkSlot && nSelectDrinkSlot < DRINKSLOT_MAX)
        {
            if (false == DrinkSlot_IsEmpty( &pDrinksMachine->DrinkSlot[ nSelectDrinkSlot ] ) )
            {
                pDrinksMachine->bSelectDrink = true;
                pDrinksMachine->nSelectDrinkSlot = nSelectDrinkSlot;

                bSelected = true;
            }
        }
    }

} while ( false == bSelected );

```

```

}

void DrinksMachine_DrinkSlot_Print( TYPE_LPDRINKSMACHINE pDrinksMachine, boolean
bManagement )
{
    int i;

    if ( bManagement )        printf( "[ 관리자 ] " );

    for ( i = 0; i < DRINKSLOT_MAX; i++ )
    {
        TYPE_LPDRINKSLOT pDrinkSlot = &pDrinksMachine->DrinkSlot[ i ];
        if ( DrinkSlot_IsEmpty( pDrinkSlot ) )
        {
            printf( "%d. 음료 없음 ", i + 1 );
        }
        else
        {
            printf( "%d. %s (%d원) ", i + 1, DrinkSlot_GetDrinkName( pDrinkSlot ),
DrinkSlot_GetDrinkPrice( pDrinkSlot ) );
        }
    }

    if ( bManagement )        printf( "0. 관리자₩n" );
    else                        printf( "₩n" );
}

void DrinksMachine_Management( TYPE_LPDRINKSMACHINE pDrinksMachine )
{
    TYPE_DRINKSMACHINEMANAGER DrinksMachineManager;

    DrinksMachineManager_Open( &DrinksMachineManager );
    DrinksMachineManager_Run( &DrinksMachineManager, pDrinksMachine );
    DrinksMachineManager_Close( &DrinksMachineManager );
}

void DrinksMachine_CheckMoney( TYPE_LPDRINKSMACHINE pDrinksMachine )
{
    int nCh;

```

```

boolean bSelected;

do {
    int nMoney = 0;

    bSelected = false;

    printf( "===== 다음을 선택하세요 =====\n" );
    printf( "1. 1000원 투입  2. 500원 투입  3. 100원 투입  0. 환불하기\n" );

    nCh = _getch();

    switch ( nCh )
    {
        case '1': nMoney = 1000; break;
        case '2': nMoney = 500; break;
        case '3': nMoney = 100; break;
    }

    if (nMoney > 0)
    {
        TYPE_LPDRINKSLOT    pDrinkSlot    =    &pDrinksMachine-
>DrinkSlot[ pDrinksMachine->nSelecctDrinkSlot ];

        pDrinksMachine->nMoney += nMoney;
        if ( pDrinksMachine->nMoney >=
DrinkSlot_GetDrinkPrice( pDrinkSlot ) )
        {
            bSelected = true;
        }
    }
    else if ( '0' == nCh )
    {
        pDrinksMachine->nMoney = 0;
        bSelected = true;
    }

    printf("[투입 금액] %d 원\n", pDrinksMachine->nMoney);

```

```

    } while ( false == bSelected );
}

void    DrinksMachine_DispendDrink( TYPE_LPDRINKSMACHINE pDrinksMachine )
{
    TYPE_LPDRINKSLOT  pDrinkSlot  =  &pDrinksMachine->DrinkSlot[ pDrinksMachine->nSelecctDrinkSlot ];
    TYPE_LPDRINK  pDrink = DrinkSlot_OutputDrink( pDrinkSlot );

    if ( pDrinksMachine->nMoney <= 0 )        return;

    printf( "%s (%d원) 가 배출되었습니다~\n", pDrink->szName, pDrink->nPrice );

    pDrinksMachine->nMoney -= pDrink->nPrice;

    pDrinksMachine->nSales += pDrink->nPrice;
    pDrinksMachine->nProfit += pDrink->nProfit;
}

```

멀더 이 소스 코드는 이전 part_1_21 의 drinks 의 객체화된 버전인 것 같군요.
main 함수를 보니 아주 간단하군요.

TYPE_DRINKSMACHINE DrinksMachine

이건 스택에 객체를 선언하고

DrinksMachine_Open()

DrinksMachine_Run()

DrinksMachine_Close()

3개의 메서드만 사용하는군요.

```
#ifndef __DRINK_H__
```

```
#define __DRINK_H__
```

```
...
```

```
#endif // !__DRINK_H__
```

그리고 이런 형태가 많이 나오는데요. 프리프로세서란 거죠?

스컬리 네, 맞네요.

위의 프리프로세서 형태는 헤더 파일 정보를 한번만 인식하도록 하는 거예요.

__DRINK_H__ 가 선언이 안된 상태에서는 __DRINK_H__ 를 선언하고 내용을 인식하도록 하죠.

다음에는 __DRINK_H__ 가 선언되었으니 다시 인식하지 않도록 하는 거예요.

이런 방식은 아주 정형화된 형태예요.

멀더 잘 알아두면 좋아요.
 그렇군요.
 대부분의 헤더 파일이 이런 형태군요. 염두에 둘게요.
 이제 C 언어로도 다수의 객체를 만들어서 사용할 수 있어서 좋군요.
 정말 좋은 방법 같군요.
 스컬리 네, 이런 방법을 쓰면 큰 프로젝트도 C 언어로만으로도 해결할 수 있네요.
 호호호
 멀더 하하하
 이제 분석은 다 된 것 같군요.
 스컬리, 정말 훌륭했어요.
 스컬리 멀더, 덕분에 새로운 세상을 본 것 같아요.
 고마워요.

<출력 결과>

```

D:\Research\wc_sigma_fx\part_2\wc_sigma_fx_part_2_01_drinks\Debug\wc_sigma_fx_part_2_01_drinks.exe
[ 관리자 ] 1. 음료 추가 2. 정산하기 3. 나가기 0. 전원 끄기
[ 관리자 ] 음료 슬롯을 선택하세요
1. Cola (1000원) 2. 음료 없음 3. 음료 없음 4. 음료 없음
슬롯이 짝 찻습니다!
[ 관리자 ] ===== 메뉴를 선택하세요 =====
[ 관리자 ] 1. 음료 추가 2. 정산하기 3. 나가기 0. 전원 끄기
===== 음료를 선택하세요 =====
[ 관리자 ] 1. Cola (1000원) 2. 음료 없음 3. 음료 없음 4. 음료 없음 0. 관리자
===== 다음을 선택하세요 =====
1. 1000원 투입 2. 500원 투입 3. 100원 투입 0. 환불하기
[ 투입 금액] 1000 원
Cola (1000원) 가 배출되었습니다~
===== 음료를 선택하세요 =====
[ 관리자 ] 1. Cola (1000원) 2. 음료 없음 3. 음료 없음 4. 음료 없음 0. 관리자
[ 관리자 ] ===== 메뉴를 선택하세요 =====
[ 관리자 ] 1. 음료 추가 2. 정산하기 3. 나가기 0. 전원 끄기
===== 음료를 선택하세요 =====
[ 관리자 ] 1. Cola (1000원) 2. 음료 없음 3. 음료 없음 4. 음료 없음 0. 관리자
===== 음료를 선택하세요 =====
[ 관리자 ] 1. Cola (1000원) 2. 음료 없음 3. 음료 없음 4. 음료 없음 0. 관리자
===== 다음을 선택하세요 =====
1. 1000원 투입 2. 500원 투입 3. 100원 투입 0. 환불하기
[ 투입 금액] 500 원
===== 다음을 선택하세요 =====
1. 1000원 투입 2. 500원 투입 3. 100원 투입 0. 환불하기
[ 투입 금액] 1000 원
Cola (1000원) 가 배출되었습니다~
===== 음료를 선택하세요 =====
[ 관리자 ] 1. Cola (1000원) 2. 음료 없음 3. 음료 없음 4. 음료 없음 0. 관리자
  
```

<End of Part 2>