

C Programming Assignment / Week 10

정리노트 #7

→ 포인터

메모리의 주소와 같은 개념이며 이 주소로 일반 변수의 값으로 접근할 수 있다.

```
#include <stdio.h>

int main(void)
{
    int i = 10;
    int* ptr = &i;

    printf("i의 주소 값 : %p\n", &i);
    printf("i의 주소 값 : %p\n", ptr); // 포인터로 주소값을 출력하는 법

    printf("i의 값 : %d\n", i);
    printf("i의 값 : %d\n", *ptr); // 포인터로 주소값의 변수 값을 출력하는 법
}
```

Microsoft Visual Studio 디버그

i의 주소 값 : 000000E1316FFAD4
i의 주소 값 : 000000E1316FFAD4
i의 값 : 10
i의 값 : 10

위에 보이는 대로 *를 붙이냐 아니냐에 따라 나타내는 값이 다르다

ptr = 주소 출력, *ptr = 그 주소의 변수 값 출력

*ptr은 위에 나온대로 변수 값을 출력한다고 했는데 그럼 이것으로 연산을 할 수 있는가?

```
#include <stdio.h>

int main(void)
{
    int i = 3;
    int* ptr = &i;
    *ptr = i + 2;
    printf("*ptr의 값 : %d\n", *ptr);
}
```

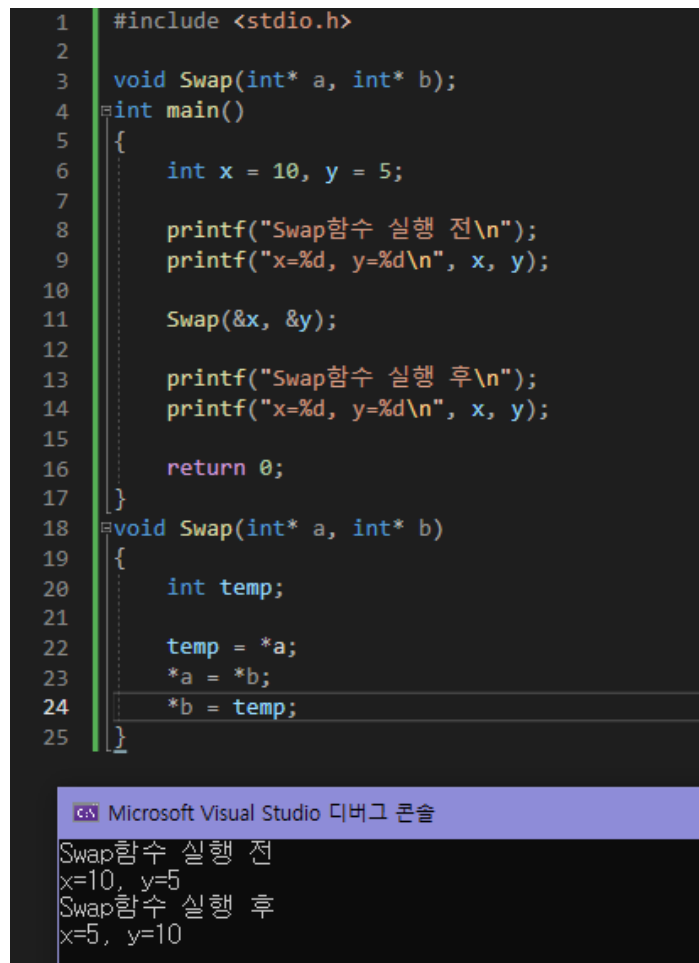
Microsoft Visual Studio 디버그

*ptr의 값 : 5

결과에 보이는데로 포인터를 이용해 간단한 연산을 할 수 있다.

(Swap 을 이용한 포인터 두 수 바꾸기)

```
1  #include <stdio.h>
2
3  void Swap(int* a, int* b);
4  int main()
5  {
6      int x = 10, y = 5;
7
8      printf("Swap함수 실행 전\n");
9      printf("x=%d, y=%d\n", x, y);
10
11     Swap(&x, &y);
12
13     printf("Swap함수 실행 후\n");
14     printf("x=%d, y=%d\n", x, y);
15
16     return 0;
17 }
18 void Swap(int* a, int* b)
19 {
20     int temp;
21
22     temp = *a;
23     *a = *b;
24     *b = temp;
25 }
```

The image shows a screenshot of a C program in Visual Studio. The code defines a swap function that takes two integer pointers and swaps their values. The main function initializes x=10 and y=5, prints them, calls swap, and prints them again. The output in the console shows that the values are successfully swapped to x=5 and y=10.

Microsoft Visual Studio 디버그 콘솔

```
Swap함수 실행 전
x=10, y=5
Swap함수 실행 후
x=5, y=10
```

swap 함수를 이용하여 x 와 y 를 바꾸었다.

이 코딩에서 주소표시인 &와 포인터 변수를 쓰지 않으면 값이 바뀌지 않는다.

```
1  #include <stdio.h>
2
3  void Swap(int* a, int* b);
4  int main()
5  {
6      int x = 10, y = 5;
7
8      printf("Swap함수 실행 전\n");
9      printf("x=%d, y=%d\n", x, y);
10
11     Swap(x, y);
12
13     printf("Swap함수 실행 후\n");
14     printf("x=%d, y=%d\n", x, y);
15
16     return 0;
17 }
18 void Swap(int a, int b)
19 {
20     int temp;
21
22     temp = a;
23     a = b;
24     b = temp;
25 }
```

Microsoft Visual Studio 디버그 콘솔

```
Swap함수 실행 전
x=10, y=5
Swap함수 실행 후
x=10, y=5
```

이게 바뀌지 않는 이유는, 일단 함수 내부에서 지역변수 temp 를 이용하여 값을 교환하고 있다. 이러면 함수 내부에서 값을 교환하더라도, 함수 내 지역변수에만 영향을 미치며, 함수의 호출한 곳에는 영향을 주지 않는다.

그래서 포인터 변수와, 주소를 이용하여 함수를 호출하여 계산하여야 한다.

➔ 인자를 값으로 전달

C 언어의 대표적 인자 전달 방식이며 함수가 호출되면 인자 값을 스택(Stack)에 복사한다.

(*Stack = LIFO(Last In First Out) 구조를 가지는 선형적인 자료구조)

이것은 값을 복사하기 때문에 함수에서 인자 값을 바꾸더라도 main()함수에 영향을 받지 않음.

- 두 함수는 독립적이고 안전함,

➔ 인자를 주소로 전달

전달하려는 변수의 주소를 함수에 입력하는 것이며 이 때, 포인터를 이용해 전달을 한다.

- 주소 연산자(&)를 통해 변수의 주소 값을 함수에 넘김 > 간접 연산자(*)를 이용해,

주소 값이 가리키는 값을 읽거나 저장함.

이는 전에 수업시간에 쓰인 함수를 통해 정리했다.

```
#include <stdio.h>

void CountIncrement1(int n); //값을 전달(스택에 복사해 보내는 식)
void CountIncrement2(int* n); //주소를 전달 (int에 포인터를 사용했음)
int main(void)
{
    int a = 10;
    printf("a의 초깃값 : %d\n", a);

    CountIncrement1(a);
    printf("CountIncrement1 함수 실행 후 a의 값 : %d\n", a);

    CountIncrement2(&a);
    printf("CountIncrement2 함수 실행 후 a의 값 : %d\n", a);

    return 0;
}

void CountIncrement1(int n) //main함수에 영향을 받지 않으므로 10으로 출력
{
    n++;
}

void CountIncrement2(int* n) //포인터로 인한 주소로 전달 받았으므로
{
    (*n)++; //변수로 출력할거면 앞에 간접 연산자를 이용해야함
}
```

Microsoft Visual Studio 디버그

a의 초깃값 : 10
CountIncrement1 함수 실행 후 a의 값 : 10
CountIncrement2 함수 실행 후 a의 값 : 11

간소한 설명은 주석으로 설명했음

➔ 배열을 이용해서 포인터를 초기화 하는 방법(2 가지)

배열 첨자를 이용하는 방법

(배열을 선언하고 포인터에다가 그 배열 원소 중 하나의 주소를 대입하는 식 Ex)int *p; > p = &a[0])

배열 명을 이용하는 방법

(포인터에 배열 이름을 입력함으로써 배열의 첫 원소를 대입: Ex)int *p; p=a;

```
#include <stdio.h>

int main(void)
{
    int a[] = { 10, 20, 30, 40, 50 };
    int* p = a; // *p는 포인터 변수 p가 가리키는 곳에 들어있는 값을 의미해서 a[0]과 같음
    int i;

    printf("배열명 a를 이용한 주소 표현\n"); // a[0]부터 a[4]까지의 주소 출력
    for (i = 0; i < 5; i++)
        printf("a[%d]의 주소 %p\n", i, a + i);

    printf("\n포인터 p를 이용한 배열 주소 표현\n"); // 포인터를 이용한 주소 출력(위에 나온대로 a[0]부터라 위에 코드랑 똑같음)
    for (i = 0; i < 5; i++)
        printf("a[%d]의 주소 %p\n", i, p + i);

    printf("\n포인터 p를 이용한 배열 값 표현\n"); // 포인터를 이용한 배열 변수 값 출력(a[0] + i 값 출력이라 생각하면 됨)
    for (i = 0; i < 5; i++)
        printf("a[%d] = %d\n", i, *(p + i));

    return 0;
}
```

Microsoft Visual Studio 디버그

배열명 a를 이용한 주소 표현
a[0]의 주소 0000005BAB0FFA48
a[1]의 주소 0000005BAB0FFA4C
a[2]의 주소 0000005BAB0FFA50
a[3]의 주소 0000005BAB0FFA54
a[4]의 주소 0000005BAB0FFA58

포인터 p를 이용한 배열 주소 표현
a[0]의 주소 0000005BAB0FFA48
a[1]의 주소 0000005BAB0FFA4C
a[2]의 주소 0000005BAB0FFA50
a[3]의 주소 0000005BAB0FFA54
a[4]의 주소 0000005BAB0FFA58

포인터 p를 이용한 배열 값 표현
a[0] = 10
a[1] = 20
a[2] = 30
a[3] = 40
a[4] = 50

간소한 포인터와 배열 관계의 설명은 주석으로 설명했음.

➔ 포인터와 배열명에서 증감 연산자의 의미

p++ : 포인터가 가리키는 곳의 다음 주소 의미(반대로 p--는 가리키는 곳의 이전 주소이다.)

*p++ : 포인터가 가리키는 곳의 다음 주소 안에 들어 있는 값을 의미

a++ : 배열명 a는 상수이므로 증감 연산자를 이용할 수 없다.

```
#include <stdio.h>

int main(void)
{
    int a[] = { 11, 22, 33, 44, 55 };
    int* p = a; //포인터에다가 배열명 대입

    printf("p의 값 = %d\n", *p); //a[0]의 값을 출력
    p++; //a[0 + 1]의 값을 대입

    printf("*(p+1)의 값 = %d\n", *p); //a[1]의 값을 출력
    p++; //a[1 + 1]의 값을 대입

    printf("*(p+2)의 값 = %d\n", *p); //a[2]의 값을 출력
    p--; //a[2 - 1]의 값을 대입

    printf("*(p+1)의 값 = %d\n", *p); //a[1]의 값을 출력

    return 0;
}
```

Microsoft Visual Studio 디버그

```
p의 값 = 11
*(p+1)의 값 = 22
*(p+2)의 값 = 33
*(p+1)의 값 = 22
```

간소한 설명은 주석으로 표시했음.

➔ 배열의 합 구하기

배열의 합을 구하는 것은 조금 어렵고 이해가 가지 않아 페어프로그래밍으로 진행해보았다.

정민수: 포인터를 이용하여, 배열을 함수 인자로 전달한 뒤, 그 배열의 합을 구하기를 해볼게요.
정수형 포인터 pA 정수 Size 를 매개변수로 받아 배열의 합을 반환하는 SumArray 함수를 사용할게요.

```

1  #include <stdio.h>
2
3  int SumArray(int* pA, int Size);
4  int main()
5  {
6      int a[] = { 10,5,15,25,7 };
7      int Sum;
8      Sum = SumArray(a, 5);
9      printf("배열의 합 : %d \n", Sum);
10
11     return 0;
12 }

```

정재현: 배열 a 의 주소와, 배열의 크기 5 를 전달하여 SumArray 함수를 호출하고, Sum 에 저장했습니다.

정민수: 이제 배열의 합을 계산하는 SumArray 의 정의를 시작할게요. 반복문을 사용하여 배열의 각 요소를 합칠게요. i 를 0 부터 Size 까지 1 씩 증가시키며 정의하는 반복문을 작성해주세요. 후에, result 를 도입하여 각 요소의 합을 구하는 return 값을 출력해주세요.

```

14 int SumArray(int* pA, int Size)
15 {
16     int result = 0, i;
17
18     for (i = 0; i < Size; i++)
19         result += pA[i];
20
21     return result;
22 }

```

정재현: result = result + pA[i]; 로 각 요소를 더하는 반복문 for 을 작성 완료했습니다. 이대로 실행하면 SumArray 의 함수를 호출하여, 포인터를 이용하여 배열의 합을 구하는 식이 작성 완료되었습니다.

정민수: 이제 저희가 수업시간에 이해가 잘 되지 않았던 포인터변수, 주소변수의 사용법을 익히게 되었네요. 함수에 인자로 전달된 변수의 값은, 함수 내부에서 지역변수로 복사되어 사용되기 때문에, 함수를 호출한 곳에서의 사용을 위해서는 포인터 변수가 필수라는 것을 배웠습니다.