데이터 구조

C프로그래밍 복습

1차원 배열

- 같은 자료형을 가진 데이터들을 나열하여 저장하기 좋은 구조, 인덱스를 이용해서(반복문을 이용해서) 접근이 용이

```
int arr[10];

for(int i =0; i< 10; i++)
//인덱스는 0부터 시작하기 때문에 (배열사이즈)보다 작게
{
		arr[i] = i *2; // i가 0~9까지 증가하면서 한번씩 접근
}
```

1차원 배열

❖ **문제 : 10명의 학생의 성적을 입력받고 합계/평균 내는 프로그램을 제작**

1. 숫자 10개 입력 (0~100 사이로 입력)
2. 만약 범위 밖의 값이 들어오면 다시 입력 받도록
3. 입력을 다 받으면 합계 및 평균을 출력

- 1번 코드 확인

포인터

- **주소 접근을 위한 변수의 주소 값을 저장하기 위한 자료형 및 연산자**
  - 포인터 변수 = (자료형 *)
    - int * p 의 의미 : 변수명은 p이고 p가 다루는 정보가 메모리 주소값(int용)
  - 연산자(*, &) – 변수명을 p라고 할 때
    - *p 의 의미 : p에 저장되어 있는 주소값의 해당 메모리에 저장되어 있는 값
    - &p의 의미 : 변수 p가 저장되어 있는 메모리 주소
  - 언제 사용?
    - 함수의 한계
      - 리턴 값이 배열이 될 수 없음, 여러 값을 리턴할 수 없음
      - 매개변수를 값으로 보냈을 때(call by value) 변수값 변경 불가
    - 동적 할당
  - 2번, 3번 코드 확인

동적할당(Dynamic Allocation)

❖ 정적할당(Static Allocation)의 정반대 개념

- 정적할당 : 컴파일 시점에 코드를 읽고 메모리 공간을 확보하는 것
  - 프로그램의 실행상태에 따라 달라지지 않음
- 동적할당 : 런타임에 필요한 만큼 메모리를 확보하는 것
  - 프로그램의 실행상태에 따라 달라질 수 있음

- 동적할당의 장점 : 메모리를 유동적으로 사용할 수 있음

동적할당(Dynamic Allocation)

❖ C 언어에 사용하는 동적할당 함수

- 메모리 할당 함수 malloc, calloc
  - (void *) malloc(size_t size)
  - (void *) calloc(size_t number, size_t size) – 초기화를 시켜줌
  - (void *) realloc((void *) ptr, size_t size) – 사이즈 재할당

- 메모리 반환 함수 free
  - (void) free(void *ptr)

동적할당(Dynamic Allocation)

❖ 문제 1 응용

1. 숫자 N을 입력 받아 N명의 학생의 성적 다루기
2. 숫자 N개 입력 (0~100 사이로 입력)
3. 만약 범위 밖의 값이 들어오면 다시 입력 받도록
4. 입력을 다 받으면 합계 및 평균을 출력

❖ N개짜리 배열 만들기 (어떻게?)

- 동적할당을 이용해서..

- 4번, 5번 코드 확인

2차원 배열

❖ 데이터가 [열]과 [행]으로 N * M 형태를 띄고 있는 배열

- 선언 방법 : 자료형 변수명[열][행]
- 2차원 배열의 이해 : 자료형[행] 의 배열을 [열]만큼 가지고 있는 배열
- int my_array[3][4]; ➔ X my_array[3]

int	int	int	int
int	int	int	int
int	int	int	int

X = int * 4
X = int * 4
X = int * 4

```
int arr[3][4];

for(int i =0; i< 3; i++) //일반적으로 행을 먼저 접근
{
        for(int j =0; j <4; j++)
        {
                arr[i][j] = i*j;
        }
}
```

더블 포인터(이중 포인터)

- **포인터 변수를 가르키는 포인터 변수**
  - int ** : int* 변수의 주소값을 가르키는 변수
  - 언제 사용될까?
    - 포인터 변수를 함수로 보내야 할 때
    - 2차원 배열을 다룰 때

  - 6번 코드 확인

더블 포인터(6번 코드 분해)

Main
Aptr=

주소	값
200	0
204	1
208	2
212	3
216	4

Main
BPtr=

주소	값
250	9
254	8
258	7
262	6
266	5

주소	값
500	200
600	250
700 (1_C)	
750 (1_A)	
800 (1_B)	
850 (2_C)	
900 (2_A)	
950 (2_B)	

Swap1 함수의 _A와 _B 안에는 어떤 값?
_A = ?, _B= ?

Int * C = NULL;
C = *_A;
>> _A안에 있는 주소의 공간의 값을 C에 넣어라.

***_A = *_B;**
>> _B안에 있는 주소의 공간의 값을 _A안에 들어 있는 주소의 공간에 넣어라

***_B = C;**
> C안에 있는 값을 _ B안에 들어있는 주소의 공간에 넣어라.

Swap2 함수의 _A와 _B 안에는 어떤 값?
_A = ?, _B= ?

Int * C = NULL;
C = *_A;
>> _A안에 있는 주소의 공간의 값을 C에 넣어라.

***_A = *_B;**
>> _B안에 있는 주소의 공간의 값을 _A안에 들어 있는 주소의 공간에 넣어라

***_B = C;**
> C안에 있는 값을 _ B안에 들어있는 주소의 공간에 넣어라.

포인터를 통한 배열 접근

- **배열은 어떤 형태로 저장되어 있을까?**
- **접근은 어떻게 해야 할까?**
  - 인덱스를 통해서
  - 포인터를 이용해서(주소값을 통해)
    - [] 하나와 * 하나가 매칭된다고 생각
- **1차원 배열일 때 주소값의 형태는?**
- **2차원 배열일 때 주소값의 형태는?**

- **7번 코드 확인**

2차원 배열

❖ 2차원 배열의 동적할당? >> my_array[3][4]를 만들고 싶다

- 배열을 접근하는 방식과 똑같이 생각

```
int arr[3][4];

for(int i =0; i< 3; i++) //일반적으로 행을 먼저 접근
{
        for(int j =0; j <4; j++)
        {
                arr[i][j] = i*j;
        }
}

i =0 일때,  j가 0~3까지 증가  >> arr[0][0~3]
i =1 일때,  j가 0~3까지 증가  >> arr[1][0~3]
i =2 일때,  j가 0~3까지 증가  >> arr[2][0~3]
```

2차원 배열

❖ 2차원 배열의 동적할당? >> my_array[3][4]를 만들고 싶다

X = int * 4
X = int * 4
X = int * 4

- X라는 자료형의 3개짜리 배열을 만들어야된다.
  - X my_array[3] → X* my_array → my_array = (X *)malloc(sizeof(X) * 3)
  - my_array[0], my_array[1], my_array[2] 가 생성된 상태 자료형 X
  - 자료형 X는 무엇일까?? int 4개짜리 자료형 >>int * 형태

  - int** my_array >> my_array = (int **)malloc(sizeof(int *) * 3))
  - my_array[0~2] 각각에 4개짜리 배열을 재생성해주어야함
  - my_array[0~2] = (int *)malloc(sizeof(int) * 4)

구조체

- **0~9번의 학생의 국어 성적을 산출한다고 하자**
  - 필요한 자료형은?
    - int Kor[10];
- **0~9번의 학생의 국어,영어,수학 성적을 산출한다고 하자**
  - 필요한 자료형은?
    - int Kor[10], Eng[10], Math[10];
    - int Score[10][3];
- **0~9번의 학생의 이름, 국/영/수, 평균을 산출한다고 하자**
  - 필요한 자료형은?
    - Char* name[10]; int Kor[10], Eng[10], Math[10]; double ave[10];
    - 또는 ???? << 이게 구조체

구조체

- **다양한 형태의 자료형을 하나로 묶어서 관리하기 위한(캡슐화) 자료형**
  - 내가 사용하고 싶은 자료형들을 묶고 전체를 대표할 수 있는 자료형을 만드는 것

```
struct student
{
        char name[10];
        int kor;
        int eng;
        int math;
        double ave;
};

void main()
{
   struct student stu;

}
```

```
typedef struct
{
        char name[10];
        int kor;
        int eng;
        int math;
        double ave;
}student;

void main()
{
   student stu;

}
```

실습 (일부분- 나머지는 과제)

- **동적할당을 이용해서, 데이터를 계속 입력 받을 수 있는 배열을 제작하여라. - realloc 사용 금지**
  - Unsigned char 형의 동적할당 배열 제작
  - 1~255 사이의 값을 매번 입력 (0이 들어오는 경우 종료)
  - 처음에 배열 사이즈는 2개로 정의
    - 꽉 차면 4, 8, 16, 32 로 계속 증가