**IVS 2기 C언어 수업정리.**

24\_05\_27 ~ 24\_05\_29 까지의 전체적인 리뷰

C언어 기능 문법 특징등을 3일동안 학습

기본지식 보다는 내가 다시 깨달은 것,모르던것들을 정리

**1.삼항 연산자**

* 조건 ? 조건이 참일때 값 : 조건이 거짓일때 값

**2.난수발생**

* #include <stdlib.h>
* #include <time.h>
* srand((unsigned int)time(NULL))
* rand()%100 // 0~100

**3.switch문**

* switch(num) { case 1: ~~ break; ... default: ~~ break;

}

**4.Call by value와 Call by reference 비교**

* 인수 전달 방식
  + Call by Value: 인수의 값을 복사하여 전달. 함수 내에서 인수 값을 변경해도 원본 데이터는 변경되지 않음.
  + Call by Reference: 인수의 주소(참조)를 전달. 함수 내에서 인수 값을 변경하면 원본 데이터도 변경됨.
  + 메모리 사용
    - Call by Value: 인수의 복사본을 사용하므로 메모리 사용량이 증가할 수 있음.
    - Call by Reference: 인수의 주소만 전달하므로 메모리 사용이 효율적임.
  + 함수의 부작용
    - Call by Value: 함수의 부작용이 없음. 함수 외부의 데이터에 영향을 미치지 않음.
    - Call by Reference: 함수의 부작용이 발생할 수 있음. 함수 외부의 데이터가 변경될 수 있음.
  + 사용 사례
    - Call by Value: 간단한 값 (예: 숫자, 문자) 을 함수로 전달할 때 주로 사용. 원본 데이터를 보호하고, 함수 호출 후에도 원본 데이터가 변경되지 않아야 할 때 사용.
    - Call by Reference: 배열, 구조체, 큰 데이터를 함수로 전달할 때 사용. 함수 내에서 원본 데이터를 직접 수정해야 할 때 사용.

**5. 깊은복사 VS 얕은복사**

주요 차이점

1. **메모리 관리**:
   * **얕은 복사**: 원본과 복사본이 동일한 메모리 주소를 가리킵니다.
   * **깊은 복사**: 원본과 복사본이 독립적인 메모리 주소를 가집니다.
2. **수정 영향**:
   * **얕은 복사**: 한 객체의 변경이 다른 객체에 영향을 미칩니다.
   * **깊은 복사**: 한 객체의 변경이 다른 객체에 영향을 미치지 않습니다.
3. **복사 속도**:
   * **얕은 복사**: 일반적으로 빠릅니다.
   * **깊은 복사**: 메모리 할당과 데이터 복사로 인해 더 느릴 수 있습니다.

**6.포인터**

포인터란 메모리 주소를 저장하는 변수

6\_1 포인터 선언:

* 포인터를 선언할 때는 데이터 타입뒤에 \*를 붙인다
* Int\* p 와 int \*p 모두 동일
* p가 int 타입 변수를 가리키는 포인터다

6\_2 주소 연산자(&)

* 변수의 주소를 얻기 위해 사용
* p = &a
* p에 a의 주소를 할당

6\_3 간접 참조 연산자(\*)

* 포인터가 가리키는 주소의 값을 참조
* Int value = \*p //p가 가리키는 주소의 값을 가져옴.
* \*p = 20 -> p가 가리키는 주소에 20 저장

동적으로 메모리를 할당하고 배열,구조체에 활용할 수 있다

6\_4 배열과 포인터

* 배열 이름은 배열의 첫 번째 요소를 가리키는 포인터
* \*ptr = arr
* \*ptr = ptr[0] = arr[0]
* \*(ptr+1) = ptr[1] = arr[1]

6\_5 구조체와 포인터

* 서로 다른 데이터 타입을 하나로 묶어주는 사용자 정의 데이터 타입.
* person 이라는 구조체가 있고 p1변수 가 선언 되었을때
* typedef struct person Person
* Person p2 = {~~~}
* p2.name , p2.age 등으로 접근
* 포인터로 접근한다면 Person \*p = &p2;
* p->name,p->age 등으로 접근
* \*(p).name , \*(p).age 등으로 접근
* 구조체 배열 ps 가 있다하면 ps에는 ps[0]의 주소가 있고, ps[0]에는 id,값,나이등이 저장되어있는 것 문자열이 있다면 문자열을 가리키는 주소가 있겠지
* (ps+idx)->name || (\*(ps+idx)).name 으로 접근한다

**7. 배열 반환에 관하여**

함수내 배열을 선언한 후 그냥 return시 쓰레기 값이 return

* **정적 배열**: 함수 내에서 정적 배열을 선언하면 함수가 종료된 후에도 배열이 유지되므로 반환할 수 있습니다.
* **동적 메모리 할당**: malloc을 사용하여 동적 배열을 할당하고 반환할 수 있습니다. 반환된 메모리는 free로 해제해야 합니다.
* **배열을 인수로 전달**: 배열을 함수의 인수로 전달하여 함수 내에서 배열을 채울 수 있습니다.