자료구조 3장

part5. 포인터

포인터란? 다른 변수의 주소를 가지고 있는 변수. 모든 변수는 메모리 공간에 저장되고, 메모리의 각 바이트에는 고유한 주소가 매개져 있고 이 주소가 저장되는 곳이 포인터임.
주소는 컴퓨터에 따라 다를 수 있기에 정확한 숫자보다 화살표로 그려짐.

포인터와 관련한 연산자
1) & : 주소 연산자. 어떤 변수 앞에 &를 붙이면, 그 변수의 메모리 주소를 얻을 수 있음.
2) * : 간접 참조. 포인터가 가리키는 주소에 있는 값을 저장하거나 수정할 때 사용하는 연산자.

<예>
int a = 100; // int형, 정수형의 변수 a를 선언하는 동시에 100이라는 값을 할당함.
int *p; // 포인터 p를 선언함. *는 이 변수가 포인터라는 것을 나타냄. 포인터 p는 int 타입의 변수 주소를 저장할 수 있음.
p = &a; // p에 a의 메모리 주소를 저장함.

-> *p와 a는 동일한 메모리 위치를 참조함. 실질적인 값만 같은 것이 아니고 동일한 객체를 가리키기에 *p 값을 변경하면 a의 값도 동일하게 변경됨.

<예>
항목 값 설명
a, 10, 변수 a가 저장한 값
&a, ~~, 변수 a의 메모리 주소
p, ~~, 포인터 p가 저장한 주소(a의 주소)
*p , 10, 포인터 p가 가리키는 값(a의 값)

Q1. a의 값은?

int a = 10;
int *p;
p = &a;
*p = 20; A. 20

포인터의 종류
1) 정수형 포인터 : int *p
2) 실수형 포인터 : float *p
3) 문자형 포인터 : char *p
4) 이중 포인터 : int *pp 포인터를 가리키는 포인터

널 포인터란?
- 어떤 객체도 가리키지 않는 포인터
- 널 포인터는 NULL이라는 매크로로 표시함.

널 포인터가 중요한 이유

1) 포인터 초기화
포인터를 선언만 하고 초기화하지 않을 경우 쓰레기 값을 가지게 되는데, 이 값을 가진 포인터를 사용하면 프로그램이 메모리 충돌을 일으킬 수 있음. 이를 방지하기 위해 포인터를 선언할 때 반드시 초기화해야하며, 초기화할 값이 없으면 NULL로 설정해야 함.

2) 유효성 검사
포인터를 사용하기 전에 그 포인터가 유효한 메모리 주소를 가리키는지 확인해야 하는데 이를 NULL인지 확인함으로써 검사 가능함.

<유효성 검사 코드>
```
if (p == NULL) { // p가 널 포인터인지 확인함.
    fprintf(stderr, "오류 : 포인터가 아무 것도 가리키지 않습니다.");
    return;
} // stderr : 표준 에러. 오류 메시지를 출력할 때 사용하는 스트림임.
```

* NULL과 0이 같다고 생각할 수도 있지만 NULL른 특정 주소를 가리키지 않음을 뜻하지만, 0은 정수값 0을 의미하는 것이기 때문에 다른 것임.

함수 매개변수로 포인터 사용하기
- 특정 변수를 가리키는 포인터가 함수의 매개변수로 전달되면 그 초인터를 이용하여 함수 안에서 외부 변수의 값을 변경할 수 있음.

<예>
```
#include <stdio.h>

void swap(int *px, int *py) // 포인터 px, py를 매개변수로 갖는 함수 설정. 함수 설정할 때 앞부분에 void를 사용한 이유는 반환값이 필요없기 때문에 사용한 것임.
{
    int tmp;  // 정수 tmp 정의함.
    tmp = *px;  //   *px는 px가 가리키는 값, 이 값을 tmp에 저장함.
    *px = *py;  // *py는 py가 가리키는 값, 이 값을 *px에 저장함.
    *py = tmp;  // tmp는 px가 가리키는 값, 이 값을 *py에 저장함.
}

int main(void)
{
    int a = 1, b = 2;

    printf("swap을 호출하기 전: a=%d, b=%d\n", a, b);
    swap(&a, &b);  // swap 함수는 a,b의 주소 전달받아 a,b 값 바꿈.
    printf("swap을 호출한 다음: a=%d, b=%d\n", a, b);

    return 0;
}
```

배열과 포인터

<예시 3.7>

```
#include <stdio.h>
#define SIZE 6              // 배열의 크기를 6으로 정의(배열의 인덱스는 0부터 시작함.)

// 1. 배열에 정수를 입력받는 함수
void get_integers(int list[]) {  // list 배열을 매개변수로 전달함.
    printf("6개의 정수를 입력하세요: ");
    for (int i = 0; i < SIZE; ++i) {
        scanf("%d", &list[i]);    // 입력받은 정수를 배열의 i번째 위치에 저장
    }
}

// 2. 배열의 합을 계산하는 함수
int cal_sum(int list[]) {
    int sum = 0;                    // 합을 저장할 변수 초기화
    for (int i = 0; i < SIZE; ++i) {
        sum += *(list + i);        // 배열의 i번째 값을 더함 (포인터 연산 사용)
    }
    return sum;                     // 최종 합을 반환
}

// 3. 메인 함수
int main(void) {
    int list[SIZE];                // 크기가 6인 정수 배열 선언
    get_integers(list);            // 배열에 사용자 입력을 받음
    printf("합 = %d \n", cal_sum(list));  // 배열의 합을 계산하여 출력
    return 0;                      // 프로그램 종료
}
```