KG 아이티뱅크

C 언어

C LANGUAGE

포인터

포인터 (Pointer)

❖ **주소값을 자료형으로 가지는 변수**

- 값이 저장된 **메모리주소**를 이용하여 **공간을 관리**하는 변수
- 서로 간섭할 수 없는 **공간을 넘어다니기 위한 방법**
- 변수의 저장공간의 **크기는 고정 4 Bytes**이며 자료형과 무관

❖ **변수의 선언 및 사용에 참조연산자(*)를 사용**

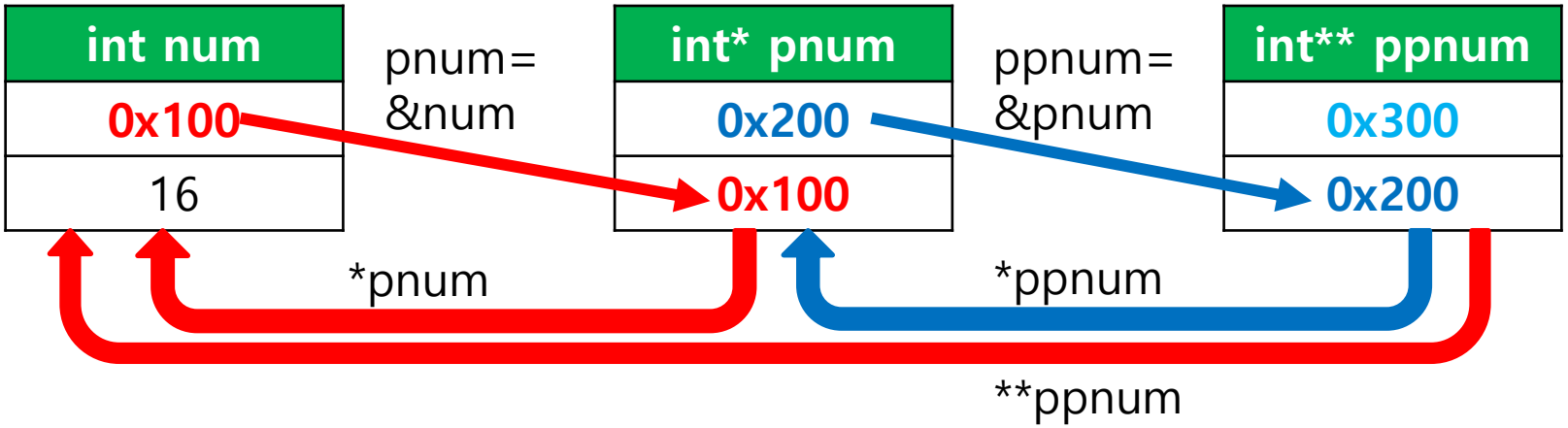
- 선언 : **주소값 자료형**을 가지는 변수임을 알려주는 기호
- 사용 : 변수에 저장된 **주소값을 이용해 공간을 사용**하는 것

❖ **코드. 선언과 초기화 예시.**

```
int num = 10;
int* pnum;              // 변수 선언
pnum = &num;            // 변수의 초기화
printf("*pnum : %d\n", *pnum);
*pnum = 20;             // 변수의 사용
printf("*pnum : %d\n", *pnum);
```

포인터 (Pointer)

❖ 개념도



❖ **변수 : 메모리주소에 이름을 붙여 쓰는 것**

- 메모리주소를 활용하는 것은 컴퓨터 내부의 처리작업
- 사용할 수 없음

❖ **포인터변수 : 메모리주소에 새로운 이름을 추가하는 것**

- 새로운 이름이 붙었으나 **어디까지나 주소 '값' 임**
- 사용하기 위해선 **'값' 이 아닌 '공간' 이라고 인식시켜야 함**

▪ **참조연산자 : 값을 공간으로 활용해 사용하는 연산자**

포인터 (Pointer)

< 파일이름 : pointerEX1.c >

❖ 실습예제1. 아래의 조건을 달성하는 코드를 작성하세요.

조건

1. 일반 문자, 정수, 실수 변수를 선언합니다.
2. 각 변수는 'A', 123, 3.14로 초기화합니다.
3. 문자, 정수, 실수 변수의 주소를 담을 수 있는 포인터 변수를 선언합니다.
4. 포인터 변수에 일반 변수의 주소를 담아 사용합니다.
5. **포인터 변수로만 처리합니다.**

결과

```
문자 포인터 변수의 참조결과     : A
정수 포인터 변수의 참조결과     : 123
실수 포인터 변수의 참조결과     : 3.14
123 + 3.14      = 126.14
A + 3.14        = 68.14
```

< 파일이름 : pointerEX2.c >

❖ **실습예제2. 아래의 조건을 달성하는 코드를 작성하세요.**

조건

1. 문자, 정수, 실수 변수를 준비하고
   각각을 A, 123, 3.14로 초기화합니다.
2. 문자, 정수, 실수 포인터 변수를 준비하고
   이를 이용하여 값을 G, 246, 6.28로 변경합니다.
3. 원본 변수를 이용하여 변경 전 값과
   변경 후 값을 모두 출력합니다.

결과

```
변경 전 값들 : A, 123, 3.14
변경 후 값들 : G, 246, 6.28
```

< 파일이름 : pointerEX3.c >

❖ 실습예제3. 아래의 조건을 달성하는 코드를 작성하세요.

조건

1. 정수 포인터 변수를 2개 이용합니다.
2. 포인터 변수를 이용해 값을 입력을 받습니다.
3. 입력을 받은 값을 포인터 변수를 이용해 연산을 진행합니다.

결과

```
정수1 입력 : (값1)
정수2 입력 : (값2)
크기 비교 : (값1 또는 값2)가 더 큽니다.
두 정수의 합          : (결과1)
두 정수의 곱          : (결과2)
두 정수의 몫          : (결과3)
두 정수의 나누기      : (결과4)
두 정수의 나머지      : (결과5)
```

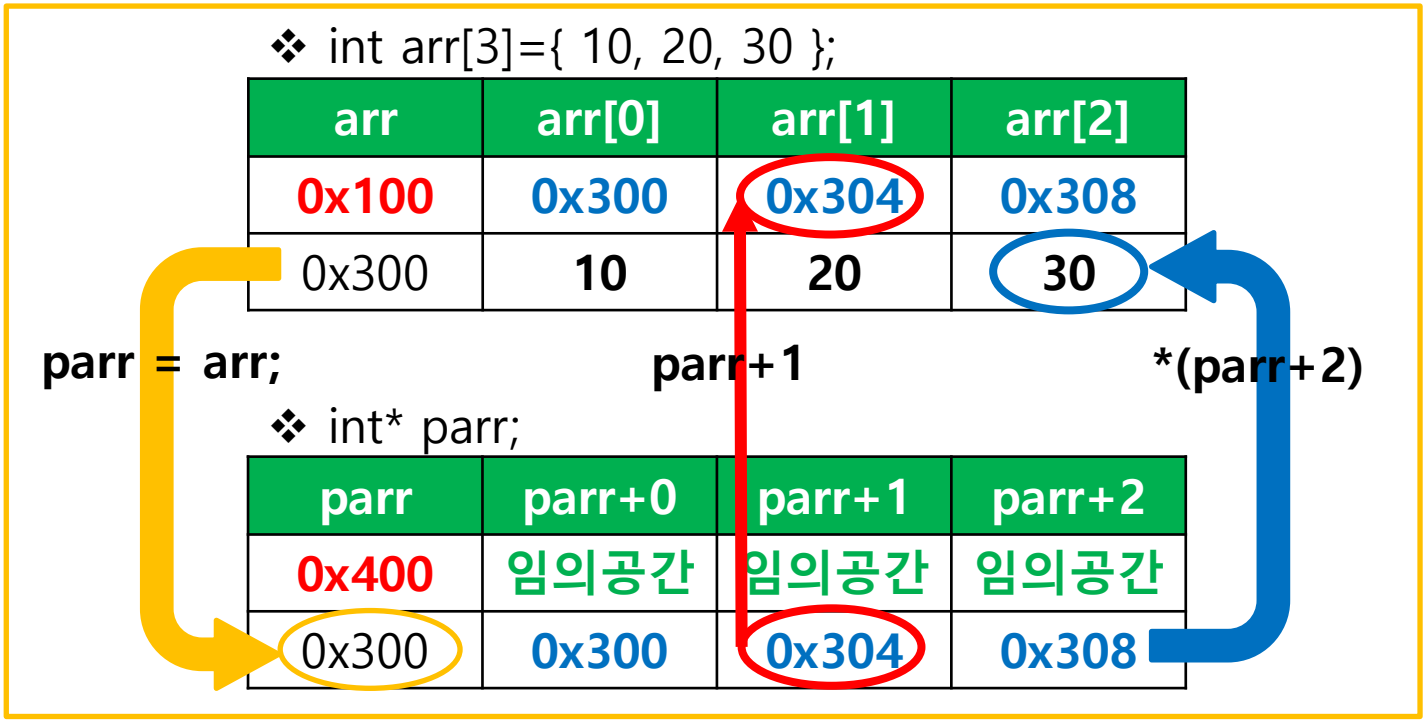
포인터 (Pointer)

❖ **배열은 연속되는 값들의 시작위치를 보관하는 변수**

- ➢ 포인터 변수로 받을 경우 **주소연산을 통해 위치를 지정**
- ➢ 연산된 주소값에 대해 **참조하여 공간을 사용**

❖ **배열에 사용되는 [ ] 연산자 : 배열첨자 연산자**

- ➢ 배열변수가 가지는 **주소값을 이용하는 연산자**
- ➢ 동일한 요령으로 포인터변수에서 사용이 가능



포인터 (Pointer)

< 파일이름 : pointerEX4.c >

❖ 실습예제4. 아래의 조건을 달성하는 코드를 작성하세요.

조건

1. 크기가 5인 정수 배열을 선언합니다.
2. 포인터 변수를 이용해 값을 초기화합니다.
3. 포인터 변수를 이용해 배열의 값들을 출력합니다.

결과

```
*(parr+0) : (임의값1 출력)
*(parr+1) : (임의값2 출력)
*(parr+2) : (임의값3 출력)
*(parr+3) : (임의값4 출력)
*(parr+4) : (임의값5 출력)
```

< 파일이름 : pointerEX5.c >

❖ **실습예제5. 아래의 조건을 달성하는 코드를 작성하세요.**

조건

1. 크기가 10 인 정수배열을 준비합니다.
2. 포인터변수를 이용해 정수를 입력을 받습니다.
3. 입력을 받은 결과를 출력합니다.

결과(1 ~ 10을 입력했을 경우)

```
1 2 3 4 5
6 7 8 9 10
```

< 파일이름 : pointerEX6.c >

❖ **실습예제6. 아래의 조건을 달성하는 코드를 작성하세요.**

조건

1. 크기가 7인 정수배열을 준비합니다.
2. 포인터 변수를 이용하여 처리합니다.
3. 임의의 값을 입력을 받아 저장합니다.
4. 포인터 변수를 이용해 입력을 받은 값들의 합을 구합니다.
5. 합을 10으로 나눈 몫을 출력합니다.

결과

(입력은 임의로 구성합니다.)

입력을 받은 값들의 합 : (결과1)

구한 합을 10으로 나눈 몫 : (결과2)

포인터 (Pointer)

❖ 포인터 변수는 격리된 공간으로 인한 제약을 해결하는 변수

❖ 결과는 하나의 변수에만 가능

```
STACK3. int fnc2(int n2) {
    n2 *= 10;
    return n2;
}
```

```
STACK2. int fnc1(int n1) {
    n1 *= 5;
    return n1;
}
```

```
STACK1. main

int num1=fnc1(8);
int num2=fnc2(5);
```

❖ 결과를 동시에 여러 변수에 대응

```
STACK2. void fnc(int *n1, int *n2) {
    *n1 = 8 * 5;
    *n2 = 5 * 10;
}
```

함수 내에서 동시에
여러 변수에 대한
처리가 가능

```
STACK1. main
int num1;
int num2;
fnc(&num1, &num2);
```

포인터 (Pointer)

< 파일이름 : pointerEX7.c >

❖ 실습예제7. 아래의 조건을 달성하는 코드를 작성하세요.

조건

1. **매개변수는 있지만 return이 없는 void 함수입니다.**
2. 이하의 조건을 만족하는 함수를 구성합니다.
   - ① 정수 변수 2개의 주소를 받도록 매개변수 구성
   - ② 받은 값을 이용해 각각 25, 50을 저장
   - ③ 받은 값을 이용해 함수에서 25와 50의 합을 출력
3. 변수에 저장된 값의 변화는 main에서 확인합니다.

결과

```
호출 전 num1, num2 : 0, 0

함수에서 초기화한 두 변수에 저장된 값의 합 : 75

호출 후 num1, num2 : 25, 50
```

포인터 (Pointer)

< 파일이름 : pointerEX8.c >

❖ **실습예제8. 아래의 조건을 달성하는 코드를 작성하세요.**

조건

1. **return이 없고 매개변수가 있는 함수입니다.**
2. 아래의 조건을 만족하는 함수를 구성합니다.
   ① 두 정수 변수의 주소를 받는 매개변수 준비
   ② 두 정수 변수가 가진 값을 서로 교환
3. 결과는 main에서 선언한 변수를 통해 확인합니다.

결과

```
-호출전-
num1, num2 : 10, 20
-호출후-
num1, num2 : 20, 10
```

< 파일이름 : pointerEX9.c >

❖ 실습예제9. 아래의 조건을 달성하는 코드를 작성하세요.

조건

1. 크기 8의 실수배열을 1.1~8.8으로 초기화합니다.
2. 크기 6의 문자배열을 A~E로 초기화합니다.
3. 이하의 조건을 만족하는 함수 두개를 준비합니다.
   ① 배열을 받을 수 있는 매개변수를 준비
   ② 함수에서 배열에 있는 값을 출력

결과

```
-함수호출-
실수배열 출력함수
1.1, 2.2, 3.3, 4.4
5.5, 6.6, 7.7, 8.8

문자배열 출력함수
ABCDE
```