

# 11장 포인터 문제풀이

01 다음에서 인덱스에 의한 배열 요소 참조는 포인터에 의한 참조로 바꾸고 반대로 포인터에 의한 참조는 인덱스에 의한 참조로 바꾸시오. (a)번은 예이다.

(a) list[6]                    \*(list+6)                    (b) name[3]                    \_\_\_\_\_  
(c) \*(cost + 8)                    \_\_\_\_\_                    (d) message[0]                    \_\_\_\_\_

02 다음 코드의 빈칸에 주석에 알맞은 문장을 넣으시오.

```
char code;  
_____; // char형 포인터 p선언  
_____; // 포인터에 변수 code의 주소 대입  
_____; // 포인터를 통하여 변수 code에 'a' 대입하기
```

03 int a[]={10, 20, 30, 40, 50}으로 정의되었다고 가정하자. \*(a+2)의 값은?

① 10                    ② 20                    ③ 30                    ④ 40                    ⑤ 50

04 아래 문장이 실행되었다고 가정하자. 다음 중 다른 문장들과 실행 결과가 다른 것은?

```
int i;  
int *p = &i;
```

① i = i + 1;                    ② i++;                    ③ \*p++;                    ④ \*p = \*p + 1;

05 다음 프로그램의 출력은?

```
int x = 6;  
int *p = &x;  
printf("%d\n", --(*p));  
printf("%d\n", (*p)++);
```

06 다음 프로그램의 출력은?

```
int *p = (int *)1000;  
double *q = (double *)2000;  
printf("%d\n", p+2);  
printf("%d\n", q+1);
```

1.

(a) \*(list+6)

(b) \*(name+3)

(c) cost[8]

(d) \*(message+0)

2.

char \*p;

p = &code;

\*p = 'a';

3. (3)

4. (3)

5.

5

5

계속하려면 아무 키나 누르십시오 . . .

6.

1008

2008

계속하려면 아무 키나 누르십시오 . . .

07 다음 프로그램의 출력은?

```
int list[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
int *p;
p = list;
printf("%d\n", *list);
printf("%d\n", *p + 1);
printf("%d\n", *(p + 1));
```

08 double형 배열을 매개 변수 a로 전달받는 함수 print\_array()의 헤더를 다음과 같은 방법으로 작성하라. 반환값은 없다.

- (a) b를 배열로 선언 \_\_\_\_\_  
(b) b를 포인터로 선언 \_\_\_\_\_

09 다음 프로그램에서 ip의 값이 변경되지 않는 이유는 무엇인가?

```
void f(int *p)
{
    static int data = 5;
    p = &data;
}
int main(void)
{
    int *ip=NULL;
    f(ip);
}
```

7.

```
0
1
1
계속하려면 아무 키나 누르십시오 . . .
```

8.

- (a) void print\_array(double a[])  
(b) void print\_array(double \*a)

9.

ip의 값이 전달되었기 때문에 ip를 변경할 수 없다. ip를 변경하려면 ip의 주소를 전달하여야 한다.

01 포인터를 이용하여 자기가 사용하는 CPU의 바이트 순서를 살펴보는 프로그램을 작성해보자. 바이트 순서(byte ordering, endian)은 컴퓨터의 메모리에 바이트를 배열하는 방법이다. 바이트 순서는 큰 단위가 앞에 나오는 빅 엔디언(Big-endian)과 작은 단위가 앞에 나오는 리틀 엔디언(Little-endian)으로 나눌 수 있다. 아래의 프로그램에 주석을 추가하라.

```
#include <stdio.h>
int main(void) {
    int x = 0x12345678;
    unsigned char *xp = (char *)&x;

    printf("바이트순서: %x %x %x %x\n", xp[0], xp[1], xp[2], xp[3]);
    return 0;
}
```

**HINT** 위의 프로그램을 실행시켜 보라. 인텔 CPU는 어떤 엔디언인가?

```
#include <stdio.h>
int main(void) {
    int x = 0x12345678;
    unsigned char* xp = (char*)&x;
    printf("바이트순서: %x %x %x %x\n", xp[0], xp[1], xp[2], xp[3]);
    return 0;
}
```

C:\Windows\system32\cmd.exe

바이트순서: 78 56 34 12  
계속하려면 아무 키나 누르십시오

종류	0x12345678의 표현
빅 엔디언	12 34 56 78
리틀 엔디언	78 56 34 12

02 2개의 정수의 합과 차를 동시에 반환하는 함수를 작성하고 테스트하라. 포인터 매개 변수를 사용한다.

```
void get_sum_diff(int x, int y, int *p_sum, int *p_diff) {
    ...
}
```

실행결과

C:\WINDOWS\system32\cmd.exe

원소들의 합=300  
원소들의 차=-100

**HINT** 함수 매개 변수에 포인터를 사용하면 2개 이상의 값을 반환할 수 있다. 본문에서 직선의 기울기와 절편을 반환하는 예제를 참고하라.

```
#include <stdio.h>
void get_sum_diff(int x, int y, int* p_sum, int* p_diff);
```

```
int main(void)
{
    int sum = 0, diff = 0;
    get_sum_diff(100, 200, &sum, &diff);
    printf("원소들의 합=%d\n", sum);
    printf("원소들의 차=%d\n", diff);
    return 0;
}

void get_sum_diff(int x, int y, int* p_sum, int* p_diff)
{
    *p_sum = x + y;
    *p_diff = x - y;
}
```

03 정수 배열을 받아서 요소들을 난수로 채우는 함수를 작성하고 테스트하라. 난수는 라이브러리 함수인 rand()를 사용하여 생성한다.

```
void array_fill(int *A, int size) {
    int i;
    for(i=0 ; i<size ; i++) {
        ...
    }
}
```

실행결과

C:\WINDOWS\system32\cmd.exe

41 18467 6334 26500 19169 15724 11478 29358 26962 24464

<

**HINT** 포인터 A는 A[0], A[1],...과 같이 배열 이름처럼 사용될 수 있다. 난수는 rand()로 발생한다. 배열의 크기는 sizeof(A)/sizeof(A[0])와 같이 구해도 된다.

```
#include <stdio.h>
#include <stdlib.h>
```

```
void array_fill(int* A, int size);
void array_print(int* A, int size);
```

```
int main(void)
```

```
{
    int data[10];
    array_fill(data, 10);
    array_print(data, 10);
    return 0;
}
```

```
void array_fill(int* A, int size)
```

```
{
    int i;
    for (i = 0; i < size; i++) {
        A[i] = rand();
    }
}
```


```
void array_print(int* A, int size)
```

```
{
    int i;
    for (i = 0; i < size; i++) {
        printf("%d ", A[i]);
    }
    printf("\n");
}
```

04 정수 배열의 요소들을 화면에 출력하는 함수를 작성하고 테스트하라. 출력 형식은  $A[] = \{ 1, 2, 3, 4, 5 \}$ 와 같은 형식이 되도록 하라.

```
void array_print(int *A, int size) {
    int i;
    printf("A[] = { ");
    for(i=0 ; i<size ; i++) {
        ...
    }
    printf("A[] = }\n");
}
```

실행결과



```
C:\WINDOWS\system32\cmd.exe
A[]={ 1 2 3 4 0 0 0 0 0 }
```

**HINT** 배열의 크기는  $\text{sizeof}(A)/\text{sizeof}(A[0])$ 와 같이 구해도 된다.

```
#include <stdio.h>
```

```
void array_print(int* A, int n);
```

```
int main(void)
```

```
{
```

```
    int list[10] = { 1, 2, 3, 4 };
```

```
    array_print(list, 10);
```

```
    return 0;
```

```
}
```

```
void array_print(int* A, int n)
```

```
{
```

```
    int i;
```

```
    printf("A[]={ ");
```

```
    for (i = 0; i < n; i++) {
```

```
        printf("%d ", A[i]);
```

```
    }
```

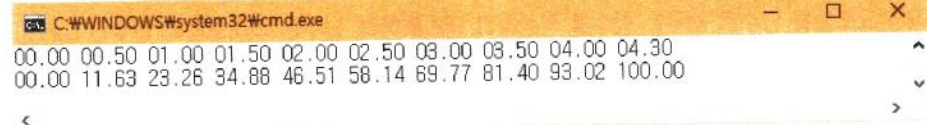
```
    printf(" }\n");
```

```
}
```

05 학생들의 평점은 4.3점이 만점이라고 하자. 배열 grades[]에 학생 10명의 학점이 저장되어 있다. 이것을 100점 만점으로 변환하여서 배열 scores[]에 저장하는 함수를 작성하고 테스트하라.

```
void convert(double *grades, double *scores, int size) {
    int i;
    for(i=0 ; i<size ; i++) {
        ...
    }
}
```

실행결과



**HINT** 점수 변환은 비례식을 사용한다. 즉  $100:4.3 = x:y$  라는 비례식을 풀면 된다.

```
#include <stdio.h>
void convert(double* grades, double* scores, int size);
void print(double* a, int size)
{
    int i;
    for (i = 0; i < size; i++) {
        printf("%05.2f ", a[i]);
    }
    printf("\n");
}
int main(void)
{
    double grades[10] = { 0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.3 };
    double scores[10] = { 0 };
    print(grades, 10);
    convert(grades, scores, 10);
    print(scores, 10);
    return 0;
}
void convert(double* grades, double* scores, int size) {
    //void convert(double grades[], double scores[], int size) {
    int i;
    for (i = 0; i < size; i++) {
        scores[i] = 100.0 * (grades[i] / 4.3);
        /*(scores + i) = 100.0 * (*(grades + i) / 4.3);
        /*scores++ = 100.0 * (*grades++ / 4.3);
    }
}
```

06 정수 배열 A[]를 다른 정수 배열 B[]에 복사하는 함수를 작성하고 테스트하라.

```
void array_copy(int *A, int *B, int size) {  
    int i;  
    for(i=0 ; i<size ; i++) {  
        ...  
    }  
}
```

실행결과

C:\WINDOWS\system32\cmd.exe

A[] = 1 2 3 0 0 0 0 0 0  
A[] = 1 2 3 0 0 0 0 0 0

```
#include <stdio.h>  
#include <stdio.h>  
#define N_DATA 10  
void array_copy(int* a, int* b, int size);  
void array_print(int* a, int size);  
  
int main(void)  
{  
    int A[N_DATA] = { 1, 2, 3 };  
    int B[N_DATA] = { 0 };  
  
    array_print(A, N_DATA);  
    array_copy(A, B, N_DATA);  
    array_print(B, N_DATA);  
    return 0;  
}  
  
void array_copy(int* a, int* b, int size)  
{  
    int i;  
    for (i = 0; i < size; i++) {  
        b[i] = a[i];  
    }  
}
```

```
void array_copy(int* a, int* b, int size)  
{  
    int i;  
    for (i = 0; i < size; i++) {  
        b[i] = a[i];  
    }  
}  
  
void array_print(int* a, int size)  
{  
    int i;  
    printf("A[] = ");  
    for (i = 0; i < size; i++) {  
        printf("%d ", a[i]);  
    }  
    printf("\n");  
}
```



07 직원들의 기본급이 배열 A[]에 저장되어 있다. 배열 B[]에는 직원들의 보너스가 저장되어 있다. 기본급과 보너스를 합하여 이번달에 지급할 월급의 총액을 계산하고자 한다. A[]와 B[]를 더하여 배열 C[]에 저장하는 함수를 작성하고 테스트하라. 즉 모든 i에 대하여  $C[i] = A[i] + B[i]$ 가 된다.

```
void array_add(int *A, int *B, int *C, int size) {
    int i;
    for(i=0 ; i<size ; i++) {
        ...
    }
}
```

실행결과

```
C:\WINDOWS\system32\cmd.exe
A[] = 1 2 3 0 0 0 0 0 0 0
B[] = 0 0 0 0 0 0 0 0 0 0
C[] = 1 2 3 0 0 0 0 0 0 0
```

[제목 없음]

```
#include <stdio.h>
#define N_DATA 10
void array_add(int a[], int b[], int c[], int size);
void array_print(char* name, int* a, int size)
{
    int i;
    printf("%s[] = ", name);
    for (i = 0; i < size; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}
int main(void)
{
    int A[N_DATA] = { 1, 2, 3 };
    int B[N_DATA] = { 0 };
    int C[N_DATA] = { 0 };

    array_print("A", A, N_DATA);
    array_print("B", B, N_DATA);
    array_add(A, B, C, N_DATA);
    array_print("C", C, N_DATA);
    return 0;
}
```

```
void array_add(int a[], int b[], int c[], int size)
{
    int i;
    for (i = 0; i < size; i++) {
        c[i] = a[i] + b[i];
    }
}
```

08 직원들의 월급이 배열 A[]에 저장되어 있다고 가정하자. 이번 달에 회사에서 지급할 월급의 총액을 계산하고자 한다. 정수형 배열 요소들의 합을 구하여 반환하는 함수를 작성하고 테스트하라.

```
int array_sum(int *A, int size) {
    int i, sum=0;
    for(i=0 ; i<size ; i++) {
        ...
    }
    return sum;
}
```

실행결과

```
C:\WINDOWS\system32\cmd.exe
A[] = 1 2 3 0 0 0 0 0 0 0
월급의 합=6
```

```
#include <stdio.h>
#define N_DATA 10
int array_sum(int a[], int size);
void array_print(char* name, int* a, int size)
{
    int i;
    printf("%s[] = ", name);
    for (i = 0; i < size; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}
int main(void)
{
    int A[N_DATA] = { 1, 2, 3 };
    int sum;

    array_print("A", A, N_DATA);
    sum = array_sum(A, N_DATA);
    printf("월급의 합=%d \n", sum);
    return 0;
}
```

```
int array_sum(int a[], int size)
{
    int i;
    int sum = 0;
    for (i = 0; i < size; i++) {
        sum += a[i];
    }
    return sum;
}
```

- 09 직원들의 월급이 저장된 배열에서 월급이 200만 원인 사람을 찾고 싶을 때가 있다. 주어진 값을 배열 A[]에서 탐색하여 배열 요소의 인덱스를 반환하는 함수를 작성하고 테스트하라.

```
int search(int *A, int size, int search_value) {
    int i;
    for(i=0 ; i<size ; i++) {
        if( A[i] == search_value ) ...
    }
}
```

실행결과

```
C:\WINDOWS\system32\cmd.exe
월급이 200만원인 사람의 인덱스=1
```

```
#include <stdio.h>
int search(int* A, int size, int x);

int main(void)
{
    int data[10] = { 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000 };
    int index = search(data, 10, 200);
    printf("월급이 200만원인 사람의 인덱스=%d\n", index);
    return 0;
}

int search(int* A, int size, int x)
{
    int i;
    for (i = 0; i < size; i++) {
        if (A[i] == x) return i;
    }
    return -1;
}
```

- 10 2개의 정수를 입력받아서 최대 공약수와 최소 공배수를 반환하는 함수를 작성하고 테스트하라. 최대 공약수는 유클리드의 방법을 사용하여서 계산한다.

```
void get_lcm_gcd(int x, int y, int *p_lcm, int *p_gcd) {
    ...
}
```

실행결과

```
C:\WINDOWS\system32\cmd.exe
두개의 정수를 입력하시오: 24 36
최소공배수는 72입니다.
최대공약수는 12입니다.
```

**HINT** 유클리드의 방법은 큰 수를 작은 수로 반복적으로 나누는 방법으로 7장 반복문을 참조한다. 최소 공배수는 다음과 같이 계산한다.  
 $lcm = (x * y) / gcd$

```
void get_gcd_lcm(int x, int y, int* gcd, int* lcm);
int main(void)
{
    int x, y, g, l;
    printf("두개의 정수를 입력하시오: ");
    scanf("%d %d", &x, &y);
    get_gcd_lcm(x, y, &g, &l);
    printf("최소공배수는 %d입니다.\n", l);
    printf("최대공약수는 %d입니다.\n", g);
    return 0;
}

void get_gcd_lcm(int org_x, int org_y, int* gcd, int* lcm)
{
    int x = org_x;
    int y = org_y;
    int tmp;

    while (y != 0)
    {
        tmp = y;
        y = x % y;
        x = tmp;
    }
    *gcd = x;
    *lcm = org_x * org_y / *gcd;
}
```

11 2개의 정렬된 정수 배열 A[]와 B[]가 있다고 가정하자. 이 2개의 배열을 합쳐서 하나의 정렬된 배열 C[]로 만드는 함수를 작성하고 테스트한다. 다음과 같은 함수 원형을 가진다고 가정하라.

```
void merge(int *A, int *B, int *C, int size) {  
    ...  
}
```

여기서 배열 A[], B[]는 똑같은 크기로 정의되어 있다고 가정한다. 배열 C[]에는 충분한 공간이 확보되어 있다고 가정하자. 합치는 알고리즘은 다음과 같다. 먼저 A[0]와 B[0]를 비교한다. 만약 A[0]가 B[0]보다 작으면 A[0]를 C[0]에 복사한다. 다음에는 A[1]과 B[0]를 비교한다. 이번에는 B[0]가 A[1]보다 작다면 B[0]를 C[1]에 저장한다. 똑같은 방식으로 남아있는 요소들을 비교하여 더 작은 요소를 C[]로 복사한다. 만약 A[]나 B[]중에서 어느 하나가 먼저 끝나게 되면 남아있는 요소들을 전부 C[]로 이동한다.

실행결과

```
C:\WINDOWS\system32\cmd.exe  
A[] = 2 5 7 8  
B[] = 1 3 4 6  
C[] = 1 2 3 4 5 6 7 8
```

```
#include <stdio.h>  
#define N_DATA 4  
void merge(int* A, int* B, int* C, int size);  
void array_print(char* name, int* a, int size)  
{  
    int i;  
    printf("%s[] = ", name);  
    for (i = 0; i < size; i++) {  
        printf("%d ", a[i]);  
    }  
    printf("\n");  
}
```

```
int main(void)  
{  
    int i;  
  
    int A[] = { 2, 5, 7, 8 };  
    int B[] = { 1, 3, 4, 6 };  
    int C[8];  
    array_print("A", A, N_DATA);  
    array_print("B", B, N_DATA);  
    merge(A, B, C, 4);  
    array_print("C", C, 2 * N_DATA);  
    for (i = 0; i < 8; i++)  
        printf("%d ", C[i]);  
    return 0;  
}  
  
void merge(int* A, int* B, int* C, int size)  
{  
    int i, a, b, c;  
  
    for (a = 0, b = 0, c = 0; a < size && b < size;) {  
        if (A[a] <= B[b])  
            C[c++] = A[a++];  
        else  
            C[c++] = B[b++];  
    }  
    for (i = a; i < size; i++)  
        C[c++] = A[i];  
    for (i = b; i < size; i++)  
        C[c++] = B[i];  
}
```

- 12 우리가 프로그램을 하다보면 사용자로부터 2개의 정수를 받아오는 경우가 많다. 이것을 함수로 구현해두고 필요할 때마다 사용하면 편리할 것이다. 하지만 한 가지 문제가 있다. C에서 함수는 하나의 값만 반환할 수 있다. 2개 이상의 값을 반환하려면 다른 방법을 사용해야 하는데 다음과 같이 포인터도 사용할 수 있다.

```
void get_int(int *px, int *py);
```

위와 같은 원형을 가지는 함수를 작성하고 이것을 이용해서 정수의 합을 계산하는 프로그램을 작성해보자.

실행결과



```
C:\WINDOWS\system32\cmd.exe
2개의 정수를 입력하시오(예: 10 20): 10 20
정수의 합은 30
```

```
#include <stdio.h>
void get_int(int* px, int* py);

int main(void)
{
    int x, y;
    get_int(&x, &y);
    printf("정수의 합은 %d \n", x + y);
    return 0;
}

void get_int(int* px, int* py)
{
    printf("2개의 정수를 입력하시오(예: 10 20): ");
    scanf("%d %d", px, py);
}
```