03 프로그래밍 프로젝트

01 배열에 대하여 다음의 일련의 프로그램을 작성하고 테스트하라.

(1) 배열의 내용을 아래와 같이 초기화하고 배열의 내용을 다음과 같이 출력하는 프로그램을 작성하시오.

```
int day[12] = { 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
```

```
1월은 31일까지 있습니다.
2월은 29일까지 있습니다.
...
12월은 31일까지 있습니다.
```

(2) 크기가 10인 1차원 배열 a와 b를 임의의 값으로 초기화하고, 배열 a와 b의 차를 구해 배열 c에 저장하는 함수 matrix_diff(a,b,c)를 작성하시오.

▶힌트 배열을 함수의 파라미터로 전달할 수 있다.

(3) 크기가 10인 배열에 임의의 값을 읽어 들인 후 배열에서 최대값과 최소값을 출력하는 프로그램을 작성하시오.

▶힌트 임의의 값은 rand() 함수를 호출하여 생성하라.

(4) 크기가 10인 1차원 배열에 임의의 값을 읽어드린 후 임의의 값의 위치를 찾는 함수 search(배열 이름, 찾고자 하는값, 요소의 위치)를 작성하시오.

```
배열의 내용: 23, 45, 12, 34, 65, 25, 89, 61, 26, 11
```

(5) 1차원 배열의 각 원소의 주소를 출력하여 보자. 다음과 같이 1차원 배열을 선언한 다음, 다음의 프로그램을 수행하여 보라.

```
#include <stdio.h>
#define LIST_SIZE 10

int list[LIST_SIZE] = { 23, 45, 12, 34, 65, 25, 89, 61, 26, 11 };

main()
{
     int i;
     for(i=0;i<LIST_SIZE;i++){
         printf("list[%d]의 주소 = %p\n",i,&list[i]);
     }
}
```

▶힌트 Visual C++의 printf()의 포맷 중에서 변수의 주소를 출력하는 포맷은 %p이다.

(6) 2차원 배열 원소의 주소를 알아보기 위하여 다음의 프로그램을 수행시켜 보라. C언어에서의 2차원 배열은 행(row)우선인가, 아니면 열(column)우선인가? 즉 같은 행이 먼저 출력되는가 아니면 같은 열이 먼저 출력되는가?

```
#include < stdio.h>

void printArrayAddress(int [][3]);
//
void main(void)
```

```
{
    int array[2][3] = {{1, 2, 3}, {4, 5, 6}};

    printf("배열 array의 출력 : \n");
    printArrayAddress(array);
}
//
void printArrayAddress(int a[][3])
{
    int i, j;
    for(i=0; i<=1; i++) {
        for(j=0; j<=2; j++)
            printf("%p ", &a[i][j]);
        printf("\n");
    }
}
```

02 학생들의 성적을 처리하여 평균과 표준편차를 구하는 프로그램을 작성하여 보자.

(1) 먼저 학생들을 구조체 타입 student로 나타내자. 다음과 같은 항목을 갖도록 하라. typedef을 사용하여 student를 정의하라.

- name: 크기가 100인 문자열로 학생의 이름을 나타낸다.
- student_number: 정수로서 학번을 나타낸다.
- score: 정수 배열, 여기에 성적이 기록된다.

```
typedef struct student {
        ;
        ;
        ;
} student;
```

(2) 이 구조체의 배열 students를 만들어 여기에 학생들의 퀴즈 성적을 저장한다. 배열의 크기는 1000으로 하라. 학생들의 이름, 학번, 성적을 읽어 들여 구조체 안의 name, student_number, score 항목에 각각 저장한다.

(3) 평균값과 표준편차를 계산하는 함수 get_mean, get_deviation을 각각 호출한다. 이때 파라미터로 students 배열을 전달하라. get_mean 함수는 students 배열과 학생의 수를 입력으로 받아서 성적의 평균을 계산하여 반환한다. get_deviation 함수는 students 배열과 학생의 수를 입력으로 받아서 표준편차를 계산하여 반환한다. 표준편차란 평균을 이라고 했을 때 다음과 같이 정의된다.

$$S = \sqrt{S^2} = \sqrt{\frac{(x_1 - M)^2 + (x_2 - M)^2 + \cdots + (x_n - M)^2}{N}}$$

(4) 동일한 함수 get_mean과 get_deviation을 이번에는 파라미터로 배열 students의 포인터를 받아서 작업을 하도록 변경하여 보라.

03 포인터에 대하여 다음의 일련의 실험을 수행하라.

(1) 다음과 같이 여러 가지 타입의 포인터를 만들고 포인터 pc의 현재값을 출력해보라.

```
#include <stdio.h>
void main()
```

```
{
 char *pc;
 int *pi;
 float *pf;
 double *pd;

 char c;
 int i;
 float f;
 double d;

 printf("pc = %lu\n", pc);
}
```

(2) 아래의 문장을 추가하여 포인터를 초기화시키지 않은 상태에서 포인터가 가리키는 곳에다 데이터를 저장하여 보라. 어떠한 일이 발생하는가?

```
printf("pc = %lu\n", pc);
*pc = 100;
```

(3) pc를 NULL포인터로 만든 후에 데이터를 저장하여 보라. 어떠한 일이 발생하는가?

```
pc = NULL;
printf("pc = %lu\n", pc);
*pc = 100;
```

(4) 다음과 같이 char 포인터인 pc가 char 변수 c를 가리키도록 초기화한 상태에서 2번과 같은 동작을 하여 보라. char 변수 c의 값이 변경되었는가?

```
 printf("초기화하기전의 pc = %lu\n", pc);
 pc = &c;
 printf("초기화하기후의 pc = %lu\n", pc);
 *pc = 100;
 printf("c = %d\n", c);
```

(5) 나머지 포인터들 pi, pf, pd에 대해서도 똑같은 실험을 되풀이하라.

(6) [포인터와 배열] 다음 프로그램을 수행시켜 보라. 배열 이름이 포인터와 같은지를 확인한다.

```
#include <stdio.h>

void main(void)
{
     int a[10] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
     int *p;

     p = a; /* p에 배열 a의 시작 주소를 치환한다. */
     /* 이 문장은 포인터를 사용하여 배열 a의 첫째, 둘째, 셋째 원소들을 출력한다. */
     printf("%d %d %d\n", *p, *(p+1), *(p+2));

     /* 이 문장은 배열이름을 포인터처럼 사용하여 배열 a의 첫째, 둘째, 셋째 원소들을 출력한다. */
     printf("%d %d %d\n", *a, *(a+1), *(a+2));

     /* 이 문장은 배열 a를 사용하여 같은 것을 출력한다. */
```

```
    printf("%d %d %d\n", a[0], a[1], a[2]);

}
```

(7) [포인터와 배열] 다음은 포인터를 배열처럼 사용한 예제이다. 수행시켜 결과를 확인하라.

```
#include <stdio.h>

void main(void)
{
    char str[] = "What is Pointer?";
    char *p;
    int i;

    p = str;
    /* 널 문자가 발견될 때까지 반복한다. */
    for(i=0; p[i] != NULL ; i++)
        printf("%c",p[i]);
}
```

(8) [포인터의 배열] 다른 자료형처럼 포인터를 배열로 만들 수 있다. 예를 들어, 다음 문장은 20개의 원소를 갖는 정수형 포인터 배열을 선언한다.

```
    int *pa[20];
```

다음의 예를 수행시켜 보라.

```
#include <stdio.h>

char *pa[] = {
        "에러 1",
        "에러 2",
        "에러 3",
        "에러 4",
        "에러 5",
        "에러 6"
};
void error(int err_num)
{
        printf(pa[err_num]);
}

main()
{
 error(1);
}
```

(9) [포인터와 함수] 본문에서 설명한 대로 포인터를 함수에 전달하게 되면 포인터가 가리키는 변수의 내용을 변경할 수 있다. 다음 프로그램을 실행시켜 보라.

```
#include <stdio.h >

void swap1(int i, int j);
```

```
void swap2(int *i, int *j);
//
void main()
{
     int num1, num2;

     num1 = 100;
     num2 = 200;

     swap1(num1, num2);
     printf("num1 : %d num2 : %d\n", num1, num2);
     swap2(&num1, &num2);
     printf("num1 : %d num2 : %d\n", num1, num2);
}
//
void swap1(int i, int j)
{
     int temp;

     temp = i;
     i = j;
     j = temp;
}
//
void swap2(int *i, int *j)
{
     int temp;

     temp = *i;
     *i = *j;
     *j = temp;
}
```

(10) 다음 프로그램은 1 바이트 크기의 동적 메모리 할당을 받고 이 동적 메모리를 포인터가 가리키도록 다음 동적 메모리에 100을 저장하는 프로그램이다. 각 문장에 대하여 주석을 붙여라.

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

void main()
{
     char *pc;

     pc = (char *) malloc(1);
     *pc = 100;
     printf("c = %d\n", *pc);
     free(pc);
}
```

(11) 100 바이트만큼의 공간을 할당받고 싶으면 다음과 같이 하면 된다. 제대로 동작되는지 실습하라.

```
void main()
{
    int i;
    char *pc;

    pc = (char *) malloc(100);
    for(i=0;i<100;i++){
        *pc = 100;
    }
    free(pc);
}
```

(12) 다음과 같은 동적 메모리 할당에 대해서도 실습하라. 각 동적으로 할당된 메모리 공간을 테스트하는 코드를 추가하여 실습하라.

```
char *pc;
int *pi;
float *pf;
double *pd;

pc = (char *) malloc(100*sizeof(char));
pi = (int *) malloc(100*sizeof(int));
pf = (float *) malloc(100*sizeof(float));
pd = (double *) malloc(100*sizeof(double));
// 동적 메모리 공간 실습 코드를 넣으시오.


//
free(pc);
free(pi);
free(pf);
free(pd);
```

(13) free() 함수의 호출을 생략하면 어떻게 되는지를 실습하라.

(14) 포인터의 사칙 연산에 대하여 살펴보자. 어떤 결론을 내릴수 있는가?

```
char *pc;
int *pi;
float *pf;
double *pd;

pc = (char *) malloc(100*sizeof(char));
pi = (int *) malloc(100*sizeof(int));
pf = (float *) malloc(100*sizeof(float));
pd = (double *) malloc(100*sizeof(double));
// 동적 메모리 공간 실습 코드를 넣으시오.

printf("pc = %lu\n", pc);
```

```
printf("pc+1 = %lu\n", pc+1);
printf("pc-1 = %lu\n", pc-1);

printf("pi = %lu\n", pi);
printf("pi+1 = %lu\n", pi+1);
printf("pi-1 = %lu\n", pi-1);
```

(15) pf와 pd에 대해서도 똑같은 코드를 작성하여 실습해보라.

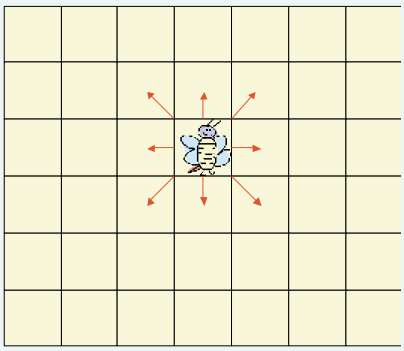
(16) 다음과 같은 구조체만큼의 공간을 동적할당할 수 있는지를 실습해보라. 구조체 안에 데이터를 저장하여 보라.

```
typedef struct test {
    int data;
    struct test *link;
} test_str;

void main()
{
    int i;
    test_str *ptest;

    ptest = (double *) malloc(sizeof(test_str));
    // 동적 메모리 공간 실습 코드를 넣으시오.

    //
    free(ptest);
}
```

(17) 동적할당된 공간과 포인터의 타입이 다른 경우, 즉 공간은 char 100개만큼의 공간을 받았는데 포인터는 int 포인터인 경우, 어떤 일이 발생하는지를 프로그램을 작성하여 실습하여 보라.

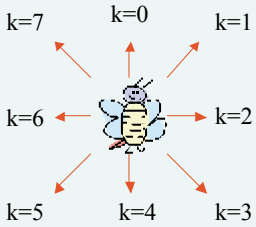
```
int *pi;
pi = (int *) malloc(100*sizeof(char));
```

04 수학에서의 "random walk"라 불리우는 문제를 배열을 이용하여 프로그래밍하여 보자. 문제는 다음과 같다. 술에 취한 딱정벌레가 크기의 타일로 구성된 방안에 있다. 딱정벌레는 임의의(랜덤) 위치를 선택하여 여기저기 걸어 다닌다. 현재의 위치에서 주의의 8개의 타일로 걸어가는 확률은 동일하다고 가정하자. 그러면 최종적인 문제는 "딱정벌레가 방안의 모든 타일을 한번씩 지나가는데 걸리는 시간은 얼마인가?" 이다.



방 전체를 2차원 배열 tile[n][m]로 모델링을 하고 처음에는 딱정벌레가 배열의 중앙에 있다고 가정하라. tile[][]의 초기값은 0이다. 딱정벌레가 타일을 지나갈 때마다 2차원 배열의 값을 1로 만들어서 딱정벌레가 지나

갔음을 나타낸다. 0부터 7까지의 랜덤한 숫자를 생성하여 다음과 같이 움직인다. 즉 0이면 북쪽으로 이동하고 4이면 남쪽으로 이동한다.



0부터 7까지의 랜덤한 숫자는 다음과 같이 생성할 수 있다. 즉 rand() 함수의 반환값을 8로 나누어 나머지를 취한다.

```
number = rand() % 8;
```

모든 타일값을 검사하여 전체 타일이 1이 되면 프로그램을 종료한다. n=20, m=20으로 하고 시작점은 중앙으로 하라. 그리고 이동의 최대 한도를 100,000으로 하라. 딱정벌레의 총 이동수, 수행 시간을 출력하여 제출한다.

05 배열로 다항식을 나타내는 2가지 방법 중에서 2번째 방법, 즉 0이 아닌 항만을 저장하는 방법을 채택하였다고 가정하자. 이 방법으로 표현된 다항식을 받아서 다항식의 뺄셈을 수행하는 함수 poly_sub2를 구현하여 보라.

06 배열로 희소행렬을 나타내는 2가지 방법 중에서 2번째 방법, 즉 0이 아닌 항만을 저장하는 방법을 채택하였다고 가정하자. 이 방법으로 표현된 다항식을 받아서 희소행렬의 뺄셈을 수행하는 함수 sparse_matrix_sub2를 구현하여 보라.