

자료구조응용

chapter 01

1. 다음은 1차원 배열에 대해 배열원소의 합을 구하는 프로그램의 일부이다. 형식매개변수가 다른 세 가지 버전의 함수를 각각 정의하고 실행되도록 작성하라.

```
int main(void)
{
    int ary1D[ ] = {1, 2, 3, 4, 5, 6};

    printf("sumAry1D_f1() %d\n",    sumAry1D_f1(ary, size ));
    printf("sumAry1D_f2() %d\n\n", sumAry1D_f2(ary,  size));
    printf("sumAry1D_f3() %d\n\n", sumAry1D_f3(ary[6] ));

    return 0;
}
```

[프로그램 설명]

- int sumAry1D_f1(int ary[], int size); // 배열파라미터, 배열크기 // 권장
- int sumAry1D_f2(int *ary, int size); // 배열포인터, 배열크기
- int sumAry1D_f3(int ary[6]);

[실행결과]

2. 다음은 2차원 배열에 대해 배열원소의 합을 구하는 프로그램의 일부이다. 형식매개변수가 다른 세 가지 버전의 함수를 각각 정의하고 실행되도록 작성하라.

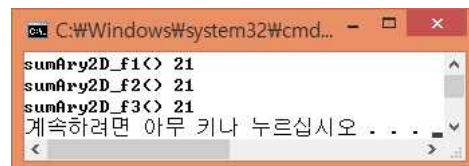
```
int main(void)
{
    int ary2D[ ][3] = { {1, 2, 3}, {4, 5, 6}};

    printf("sumAry2D_f1() %d\n", sumAry2D_f1(ary2D, ROW, COL));
    printf("sumAry2D_f2() %d\n\n", sumAry2D_f2(ary2D, ROW));
    printf("sumAry2D_f3() %d\n\n", sumAry2D_f3(ary2D));
    return 0;
}
```

[프로그램 설명]

- int sumAry2D_f1(int ary[][3], int ROW, int COL); // 배열파라미터 // 권장
- int sumAry2D_f2(int (*ary)[3], int ROW); // 배열포인터
- int sumAry2D_f3(int ary[2][3]);

[실행결과]



```
cmd C:\Windows\system32\cmd...
sumAry2D_f1<> 21
sumAry2D_f2<> 21
sumAry2D_f3<> 21
계속하려면 아무 키나 누르십시오 . . .
```

3. 아래의 메모리의 동적 할당과 해제에 대해, 보다 견고한 할당이 될 수 있도록 두 가지 버전으로 프로그램을 작성하라. 모두 제대로 main 함수를 구현하여 실행되도록 하여야 한다.

```
int i, *pi;
float f, *pf;
pi = (int *) malloc(sizeof(int));
pf = (float *) malloc(sizeof(float));
*pi = 1024;
*pf = 3.14;
printf("an integer = %d, a float = %f\n", *pi, *pf);
free(pi);
free(pf);
```

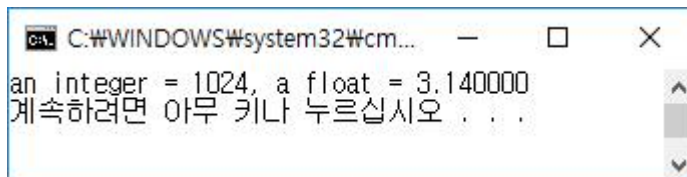
Program 1.1: Allocation and deallocation of memory

(1) 다음 코드를 이용

```
if ((pi = (int *) malloc(sizeof(int))) == NULL ||
    (pf = (float *) malloc(sizeof(float))) == NULL)
{fprintf(stderr, "Insufficient memory");
 exit(EXIT_FAILURE);
}
```

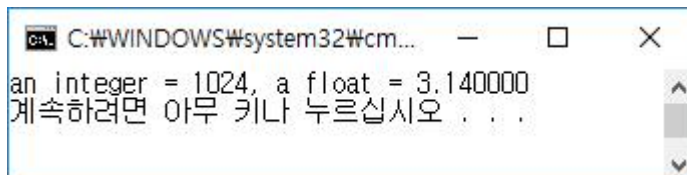
or by the equivalent code

```
if (!(pi = malloc(sizeof(int))) ||
    !(pf = malloc(sizeof(float))))
{fprintf(stderr, "Insufficient memory");
 exit(EXIT_FAILURE);
}
```



(2) 다음 매크로를 이용하여 Program1.1 을 수정

```
#define MALLOC(p,s) \
    if (!(p) = malloc(s)) {\
        fprintf(stderr, "Insufficient memory"); \
        exit(EXIT_FAILURE);\
    }
```



4. 다음과 같은 프로그램을 두 가지 버전(탐색시 반복문과 재귀적 호출이용)으로 작성하시오.

[실행순서]

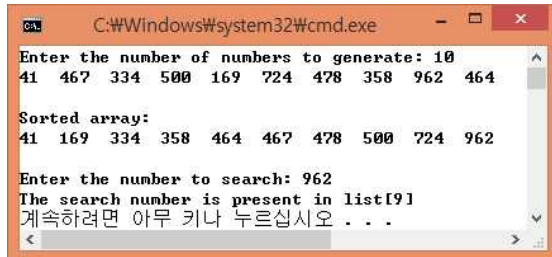
- ① 사용자로부터 난수생성 개수(n)를 입력받는다.
- ② 정수 난수를 n개 발생시켜 1차원 배열에 저장한다.
- ③ 1차원 배열에 대해 선택정렬(selection sort)을 수행한다.
- ④ 사용자로부터 임의의 정수를 입력받는다.
- ⑤ 입력받은 정수가 배열에 있는지 이진탐색(binary search)을 수행하여 그 결과를 출력한다.

난수생성조건:

난수생성 개수는 최대 100개, 난수범위는 0~999, 난수의 중복허용, SEED를 지정하지 않음

[실행결과]

(1) 함수 swap, compare와 반복문을 사용한 이진탐색을 구현한 버전

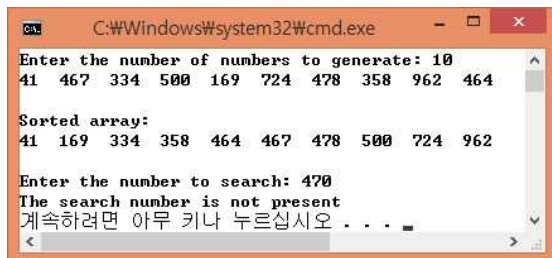


```
C:\Windows\system32\cmd.exe
Enter the number of numbers to generate: 10
41 467 334 500 169 724 478 358 962 464

Sorted array:
41 169 334 358 464 467 478 500 724 962

Enter the number to search: 962
The search number is present in list[9]
계속하려면 아무 키나 누르십시오 . . .
```

(2) 매크로 SWAP, COMPARE와 재귀호출을 사용한 이진탐색을 구현한 버전



```
C:\Windows\system32\cmd.exe
Enter the number of numbers to generate: 10
41 467 334 500 169 724 478 358 962 464

Sorted array:
41 169 334 358 464 467 478 500 724 962

Enter the number to search: 470
The search number is not present
계속하려면 아무 키나 누르십시오 . . .
```

■ 제출 형식

- 솔루션 이름 : DS_01
- 프로젝트 이름 : 1, 2, 3-1, 3-2, 4-1, 4-2
- 솔루션 폴더를 압축하여 제출할 것.
- 학습관리시스템에 과제를 올릴 때 제목:
1차 제출: 학번_이름_DS_01(1), 2차 제출: 학번_이름_DS_01(2)
제출은 2회걸쳐 가능(수정 시간 기준으로 처리)