자료구조 9주차 과제

제출일: 2022-11-01
학번/이름: 2016121150 / 윤준영

문1) 일반적인 선택정렬 함수를 사용하여 다음 실수 배열 a를 작은 수부터 큰 수의 순서대로 정렬하려고 한다. 필요한 함수와 main 함수를 작성하여 프로그램을 완성하시오. (3)

HW > HW9 > C hw9_1.c > ...

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

#define BASE(i)     ((char*)base + (i)*width)
typedef int (*FCMP)(const void*, const void*);

void select_sort(void *base, size_t nelem, size_t width,
                 int (*fcmp)(const void*, const void*)) {
    void *min;
    int minindex, i, j;
    min = malloc(width);
    for (i = 0; i < nelem - 1; i++)     {
            minindex = i;
            for (j = i + 1; j < nelem; j++)
                if (fcmp(BASE(minindex), BASE(j)) > 0)
                    minindex = j;
            memcpy(min, BASE(minindex), width);
            memcpy(BASE(minindex), BASE(i), width);
            memcpy(BASE(i), min, width);
        }
    free(min);
}

int double_cmp(const void *a, const void *b){
    /* compare float */
    return (*(double *)a - *(double *)b + 1);
}


int main() {
    double a[] = {4.2, 3.4, 5.6, 1.2, 3.3, 7.7};
    select_sort(a, 6, sizeof(double), double_cmp);
    for (int i=0; i<6; i++)
        printf("%lf ", a[i]);
    printf("\n");
}
```

select_sort 함수는 배열의 첫 주소와 원소의 개수, 한 원소의 데이터 크기, 비교 함수를 인자로 받는다. a 배열은 double형으로 이루어져 있으므로 먼저 double형의 두 값을 비교하는 double_cmp를 작성하였다. 16줄의 if 문에서 만약 앞의 값이 더 크다면 1이상(int형 이므로)의 값을 반환하여야 하므로 27줄에 두 수를 뺀 후 +1하였다. 배열의 원소는 6개이므로 select_sort(a, 6, sizeof(double), float_cmp)를 사용하면 선택정렬을 할 수 있다.

실행 결과는 다음과 같다.

```
1.200000 3.300000 3.400000 4.200000 5.600000 7.700000
 터미널이 작업에서 다시 사용됩니다. 닫으려면 아무 키나 누르세요.
```

제대로 정렬이 된 것을 확인할 수 있다.

문2) 삽입 정렬 함수는 안정성 있는(stable) 정렬인가? 주장을 정당화 하시오. (2)
삽입 정렬 함수는 안정성 있는 정렬이다.
안정성 있는 정렬은 같은 키값을 가진 원소들의 상대적인 위치가 유지된다. 삽입 정렬은 정렬되지 않은 구역의 원소 중 가장 작은 원소를 앞의 정렬된 구역의 뒷부분으로 삽입하는 정렬로, 같은 킷값을 가진 원소라면 해당 원소들이 정렬된 순서로 다시 정렬된 구역으로 추가되게 된다. 예를 들면 6 2(1) 2(2) 5 이라는 배열이 있을 때, 2(1)과 2(2)는 순서대로 정렬된 구역으로 추가되게 된다.
6 2(1) 2(2) 5 > 2(1) 6 2(2) 5 > 2(1) 2(2) 6 5 > 2(1) 2(2) 5 6

문3) 다음 거품 정렬 함수를 일반적인 자료형을 위한 거품 정렬 함수로 변경하시오. (위 문 1의 main 함수와 추가 정의된 함수를 사용하여 수행해 보시오.) (5)

HW > HW9 > C hw9_2.c > ...

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

#define BASE(i)     ((char*)base + (i)*width)
typedef int (*FCMP)(const void*, const void*);

void bubble_sort(void *base, size_t n, size_t width,
                 int (*fcmp)(const void*, const void*)) {
    void *tmp = malloc(width);
    int i, j;
    for (i = n-1; i > 0; i--)
        for (j = 1; j <= i; j++)
            if (fcmp(BASE(j-1), BASE(j)) > 0){
                memcpy(tmp, BASE(j-1), width);
                memcpy(BASE(j-1), BASE(j), width);
                memcpy(BASE(j), tmp, width);
            }
    free(tmp);
}

int double_cmp(const void *a, const void *b){
    /* compare float */
    return (*(double *)a - *(double *)b + 1);
}

int main() {
    double a[] = {4.2, 3.4, 5.6, 1.2, 3.3, 7.7};
    bubble_sort(a, 6, sizeof(double), double_cmp);
    for (int i=0; i<6; i++)
        printf("%lf ", a[i]);
    printf("\n");
}
```

버블 정렬 함수는 n개의 원소를 가진 배열에서 (0, 1)번째 원소를 비교하여 앞의 원소가 크다면 자리를 바꾸고, (1, 2), (2, 3), ..., (n−2, n−1)번째에 대해서 반복하여 가장 큰 원소가 맨 뒤로 가게한 뒤, 맨 마지막 원소를 제외한 나머지 n−1개의 원소에 대해 다시 버블 정렬을 하는 것을 n−2, n−3, ... 2까지 반복하는 함수이다. 따라서 임시 저장공간인 tmp를 할당한 후 비교하는 두 원소 a, b가 a>b라면 b, a순서로 정렬하게 된다.
실행 결과는 다음과 같다.

```
1.200000 3.300000 3.400000 4.200000 5.600000 7.700000
 터미널이 작업에서 다시 사용됩니다. 닫으려면 아무 키나 누르세요.
```

결과를 보면 오름차순으로 제대로 정렬된 것을 확인할 수 있다.