C Programming Assignment / Week 14

정리노트 #10

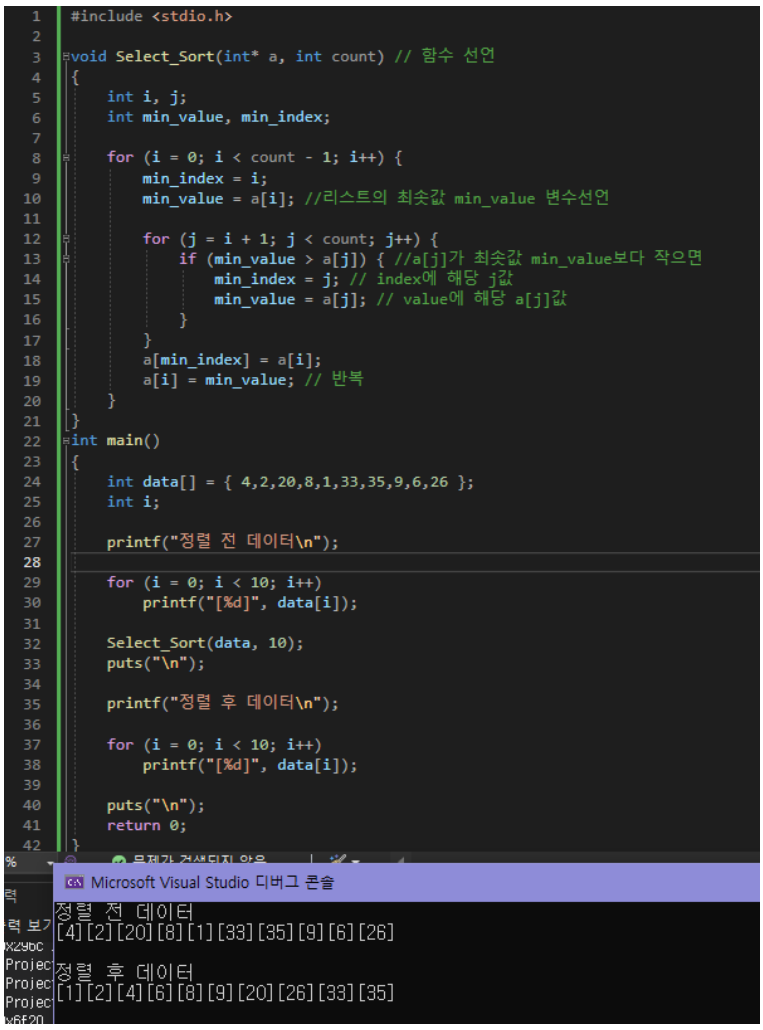
데이터 정렬 프로그램

데이터의 정렬 프로그램을 구성하는 5가지의 방식이 있다.

1. 선택 정렬
2. 버블 정렬
3. 삽입 정렬
4. 셸 정렬
5. 퀵 정렬

- **선택 정렬**

선택 정렬은, 반복문을 사용하여 리스트에서 가장 작은 원소를 첫 번째에 배치하고, 배치 완료된 원소를 제외한 나머지도 반복하여 정렬하는 방식이다.

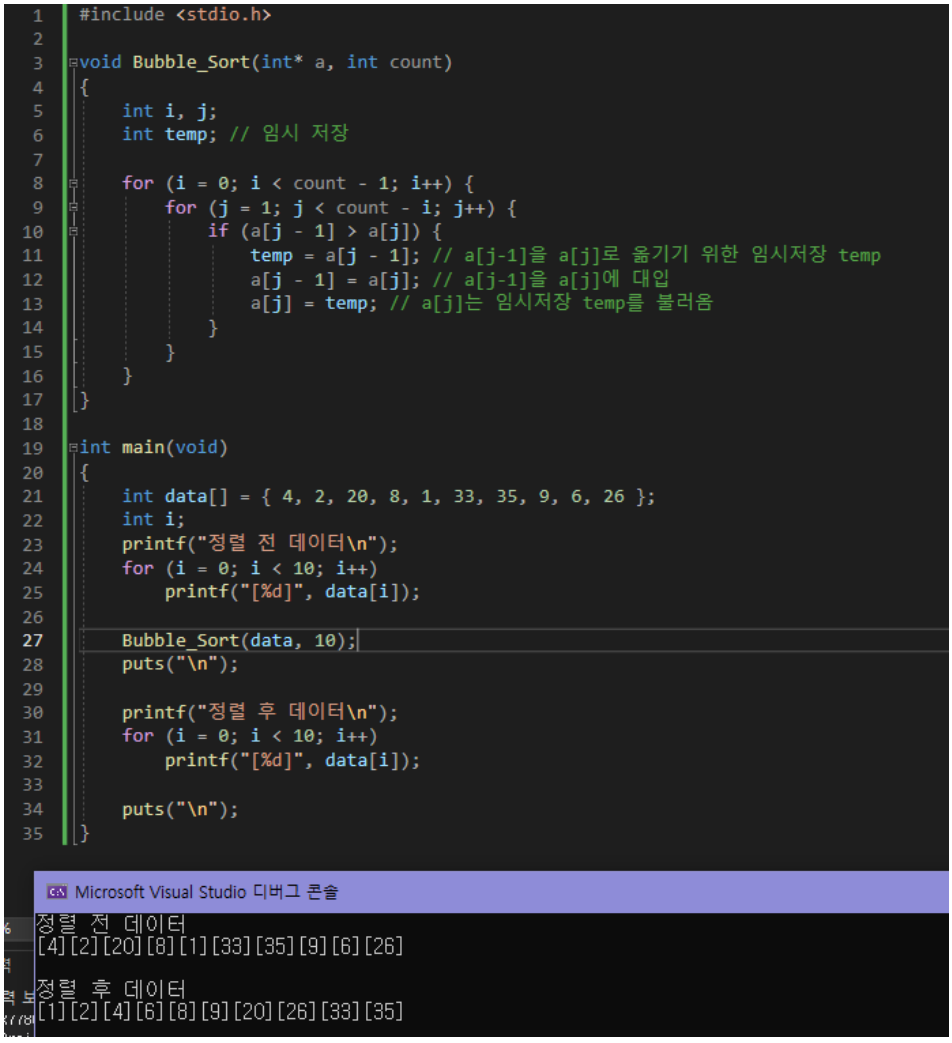


```c
#include <stdio.h>

void Select_Sort(int* a, int count) // 함수 선언
{
    int i, j;
    int min_value, min_index;

    for (i = 0; i < count - 1; i++) {
        min_index = i;
        min_value = a[i]; //리스트의 최솟값 min_value 변수선언

        for (j = i + 1; j < count; j++) {
            if (min_value > a[j]) { //a[j]가 최솟값 min_value보다 작으면
                min_index = j; // index에 해당 j값
                min_value = a[j]; // value에 해당 a[j]값
            }
        }
        a[min_index] = a[i];
        a[i] = min_value; // 반복
    }
}
int main()
{
    int data[] = { 4,2,20,8,1,33,35,9,6,26 };
    int i;

    printf("정렬 전 데이터\n");

    for (i = 0; i < 10; i++)
        printf("[%d]", data[i]);

    Select_Sort(data, 10);
    puts("\n");

    printf("정렬 후 데이터\n");

    for (i = 0; i < 10; i++)
        printf("[%d]", data[i]);

    puts("\n");
    return 0;
}
```

```
Microsoft Visual Studio 디버그 콘솔
정렬 전 데이터
[4][2][20][8][1][33][35][9][6][26]

정렬 후 데이터
[1][2][4][6][8][9][20][26][33][35]
```

- 버블 정렬

버블 정렬은, 정렬되지 않은 리스트의 뒤의 값과 앞의 값을 하나하나 비교하여 현재 값보다 작으면 바꾸는 방식을 반복문으로 반복하여 정렬하는 과정이다. 수업에서 가장 많이 쓰는 정렬 방법이다.



```c
#include <stdio.h>

void Bubble_Sort(int* a, int count)
{
    int i, j;
    int temp; // 임시 저장

    for (i = 0; i < count - 1; i++) {
        for (j = 1; j < count - i; j++) {
            if (a[j - 1] > a[j]) {
                temp = a[j - 1]; // a[j-1]을 a[j]로 옮기기 위한 임시저장 temp
                a[j - 1] = a[j]; // a[j-1]을 a[j]에 대입
                a[j] = temp; // a[j]는 임시저장 temp를 불러옴
            }
        }
    }
}

int main(void)
{
    int data[] = { 4, 2, 20, 8, 1, 33, 35, 9, 6, 26 };
    int i;
    printf("정렬 전 데이터\n");
    for (i = 0; i < 10; i++)
        printf("[%d]", data[i]);

    Bubble_Sort(data, 10);
    puts("\n");

    printf("정렬 후 데이터\n");
    for (i = 0; i < 10; i++)
        printf("[%d]", data[i]);

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

```
Microsoft Visual Studio 디버그 콘솔
정렬 전 데이터
[4][2][20][8][1][33][35][9][6][26]

정렬 후 데이터
[1][2][4][6][8][9][20][26][33][35]
```

- 삽입 정렬

삽입 정렬은 처음은 버블 정렬과 다를 것 없이 앞의 값과 뒤의 값을 비교하지만, 가상의 벽이라는 것을 놓아 이 가상의 벽 안에 뒤의 값을 크기에 맞게 집어넣는다. 벽의 앞에 있는 값들이 2개 이상이 되면 앞에 있는 값들과 가상의 벽 바로 뒤의 값을 비교하여 크기에 맞게 계속 집어넣는다.

```c
#include <stdio.h>

void Insert_Sort(int* a, int count)
{
    int i, j;
    int temp; // 임시 저장

    for (i = 1; i < count; i++) {
        temp = a[i]; // i = 1 일때를 가정하여 주석, temp에 a[1] 넣기
        j = i;

        while ((a[j - 1] > temp) && (j > 0)) // 예를 들어 a[0]이 a[1]보다 크다면,
        {
            a[j] = a[j - 1]; //a[1] 에 a[0]을 넣고
            j = j - 1; //
        }
        a[j] = temp; // a[0]에 temp값인 a[1]을 넣기
    }
}

int main(void)
{
    int data[] = { 4, 2, 20, 8, 1, 33, 35, 9, 6, 26 };
    int i;
    printf("정렬 전 데이터\n");
    for (i = 0; i < 10; i++)
        printf("[%d]", data[i]);

    Insert_Sort(data, 10);
    puts("\n");

    printf("정렬 후 데이터\n");
    for (i = 0; i < 10; i++)
        printf("[%d]", data[i]);

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

```
Microsoft Visual Studio 디버그 콘솔
정렬 전 데이터
[4][2][20][8][1][33][35][9][6][26]

정렬 후 데이터
[1][2][4][6][8][9][20][26][33][35]
```

- 셸 정렬

1959년에 만들어진 알고리즘으로 삽입 정렬을 개선한 방법

ㄴ 셸 정렬의 알고리즘

1. h의 계수를 정한다(원소의 개수와 상관없으며, 원소를 화학의 에너지 준위처럼 나열하는데 몇 줄로 할지 정하는 용도)

2. h의 간격만큼 떨어진 원소를 모은 부분집합에서 삽입 정렬을 각각 실행(이 후, 3번 파트를 실행하기 위해서 정렬했던 원소를 다시 리스트 형태로 모음)

3. h의 간격을 재설정하고 2번 구간을 다시 실행하는 구조

- 퀵 정렬

1960년에 제안된 방법이며 빠른 속도와 간단한 구현 방법으로 여러 분야에서 많이 사용하는 정렬 알고리즘이다

ㄴ 퀵 정렬의 알고리즘

1. 기준 원소 피벗을 고른다. (이때, 기준 원소 피벗은 중간값으로 하는 것이 최적일 것이다.)

2.피벗을 기준으로 작은 값은 왼쪽으로 이동, 큰 값은 오른쪽으로 이동시킨다. (원소를 이동만 시켰기 때문에 정렬은 되어있지 않은 상태)

3. 피벗은 가만히 두고 왼쪽과 오른쪽 원소 집합을 독립적으로 다시 퀵 정렬을 한다. (이 때, 설정하는 기준 피벗의 값도 중간값으로 설정하면서 하면 최적일 것이다.)