**알고리즘 및 실습 HW #3**

**B089020 김찬우**

|  |
| --- |
| # Visual Studio를 사용해서 안정적 선택 정렬을 구현한다.  # 강의록의 비안정적 선택정렬 알고리즘을 안정적 선택 정렬로 변환하라.  # 자리바꿈 확인을 편하게 하기 위해, 배열 번호를 보여줄 행렬을 반드시 하나 더 만들것.  # 다양한 input data를 만들고 그것에 대해서 결과 보고할 것.  # 보고서는 간단하게, 아이디어와 소스코드 설명, 비안정적 정렬이 안정적에게 되는 부분의 스크린 캡쳐 및 설명 그리고 코드 실행 결과 캡쳐만 넣을 것.  # 마지막으로 보고서와 프로그램 파일 (프로젝트 파일), 결과 텍스트 파일을 ZIP으로 묶어 제출한다. |

# 문제

# 안정적 선택 정렬을 구현 하기 위해 추가된 소스코드

|  |
| --- |
| void selectionSort(int[], int, int[])  for (i = 0; i < n - 1; i++)  {  //…생략  for (k = i + 1; k < MinIndex; k++) {  if (A[i] == A[k]) {  swap(&A[i], &A[k]);  swap(&B[i], &B[k]);  }  }  //…원래 정렬하는 부분 수행  } |
| 기본적인 선택정렬은 같은 값이 순서대로 나오는 것을 보장하지 못하기 때문에 안정적으로 바꾸기 위하여  위의 코드를 추가하여 정렬될 때 같은 값이 뒤섞이는 걸 방지했다.  배열B는 배열의 초기 index정보를 저장한 배열이다. swap이 일어나는 곳에선 모두 A배열과B배열의 값을 함께 swap해주었다.  sort함수에 들어오면서 배열의 크기만큼 전체적으로 반복문을 실행하면서  다시 정렬 완료된 자리부터 min index까지 중에 같은 값이 있는 경우 순서를 바꿔준다.  이렇게 되면 비안정적으로 되기 전에 같은 값인 원소끼리 미리 자리를 바꾸고  다른 값과 바뀌게 되기 때문에 결과가 안정적으로 나오게 된다. |

# 자리바꿈 확인을 편하게 하기 위해, 배열 번호를 보여줄 행렬을 반드시 하나 더 만들것.

|  |
| --- |
| void main()  {  int data; //각 데이터를 저장할 변수 선언  int i;  int size = 0;  int a[1000]; //읽은 파일의 데이터를 배열 선언  **int b[1000]; //index를 저장할 배열** |
| 배열 번호를 저장할 b[1000]배열을 추가함. |
| while (fscanf\_s(fi, "%d", &data) != EOF) //파일의 정수형 data를 끝까지 읽음  {  a[size] = data; //data를 배열에 넣음  **b[size] = size; //최초 배열 위치 저장**  size++;  } |
| 파일에서 읽은 값을 배열A에 저장하면서 그때 저장된 배열 번호를 B에 저장함. |
| printf("\n\n<<<<<<<<<< 정렬전 데이터 >>>>>>>>>>\n"); //정렬 되기 전의 data 출력  for (i = 0; i < size; i++){  printf("%3d ", a[i]);  }  printf("\n");  **for (i = 0; i < size; i++) {**  **printf("%3d ", b[i]);**  } |
| 추가한 배열 B를 이용하여 배열번호를 출력. |

# 제공된 input data에서의 실행결과

|  |  |
| --- | --- |
| 비안정 선택정렬에서의 실행 결과 | 안정적 선택정렬 추가 후 결과 |
|  |  |
|  |  |
| 추가한 코드가 정상적으로 30 두개를 순서대로 맞춰주었다. 하지만 이 케이스에서 비안정적으로 정렬된 부분이  A[0], A[6] 30뿐이었어서 다른 테스트 케이스를 넣어보았다. | |

# 추가한 input data에서 값 비교

|  |  |
| --- | --- |
| 비안정 선택정렬에서의 실행 결과 | 안정적 선택정렬 추가 후 결과 |
|  |  |
|  |  |
| 다른 테스트 케이스에서도 정상적으로 안정적으로 정렬되는 것을 확인하였다. | |

# 비안정적 정렬이 안정적에게 되는 부분의 스크린 캡쳐 및 설명

|  |
| --- |
|  |
| 4번째 과정 완료 후  보통의 선택정렬에서는 5번째에서 30(0)이 20(1)과 바뀌게되는데  그렇게 되기전에 30(0)과 30(6)을 교환해서  30(6)과 20(1)이 바뀌게 하여 안정성을 보장하는 부분이다. |

#결과 화면

|  |
| --- |
|  |