REPORT

(알고리즘 2차 과제)



|  |  |
| --- | --- |
| 제목 | 합병 정렬 |
| 제출일자 | 20.10.09 |
| 소속학과 | 컴퓨터공학과 |
| 학번 | 2017305039 |
| 성명 | 신동민 |

문제1)

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

const int MAX = 10000; //필요한 배열의 수를 MAX로 정의한다.

int main() {

srand((unsigned)time(NULL));

int\* Array = new int[MAX]; //랜덤 값을 10000개를 받는 배열과 변수 선언한다.

int index = 0;

int a;

for (int i = 0; i < MAX; i++) { // 0~49999의 값을 랜덤으로 Array에 넣어준다.

Array[i] = ((rand() << 15) | rand()) % 50000;

}

cout << "찾을 값을 입력하시오.(0~49999)"; //찾는 값을 입력 받아 변수a에 넣는다.

cin >> a;

for (int i = 0; i < MAX; i++) //선행탐색을 하는 for문이다.

{

if (Array[i] == a)

{

cout << i+1 << "번째에 있습니다." << endl;

++index;

}

if (index == 1) //1개라도 찾으면 중단한다.

break;

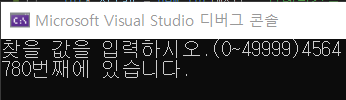
}

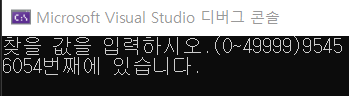
if (index == 0) //찾지 못하면 출력한다.

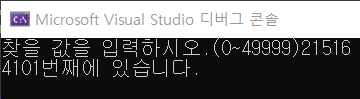
cout << "찾는 값이 없습니다.";

delete[] Array;

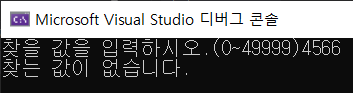
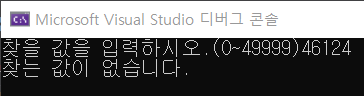
}

찾아지는 경우





찾아지지 않는 경우



문제2)

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

const int MAX = 10000; //필요한 배열의 수를 MAX로 정의한다.

int Temp[MAX]; //합병할 때에 필요한 공간의 배열을 전역변수로 선언한다.

void Merge(int data[], int start, int mid, int start2, int end) {

int Tindex = start;

int k = start;

while (start <= mid && start2 <= end) { //배열 영역 중 하나라도 영역을 넘으면 중단하고

if (data[start] <= data[start2]) //비교하는 값이 작은 쪽을 Temp에 저장한다.

Temp[Tindex++] = data[start++];

else

Temp[Tindex++] = data[start2++];

}

if (start > mid) { //while에서 한 쪽이 Temp에 다 넣어지면 다른 쪽에 남은

for (int i = start2; i <= end; i++) //값들을 Temp에 저장한다.

Temp[Tindex++] = data[i];

}

else {

for (int i = start; i <= mid; i++)

Temp[Tindex++] = data[i];

}

for (int i = k; i <= end; i++) //임시로 값을 저장한 Temp에서 data로 넣어준다.

data[i] = Temp[i];

}

void Mergesort(int data[], int start, int end) {

int mid;

if (start < end) //배열의 원소의 수가 2개 이상이면 실행한다

{

mid = (start + end) / 2 //mid=반으로 나누기 위한 중간 원소의 인덱스이다.

Mergesort(data, start, mid); //앞부분 재귀 호출한다.

Mergesort(data, mid + 1, end); //뒷부분 재귀 호출한다.

Merge(data, start, mid, mid + 1, end); //합병 함수 호출한다.

}

}

void location(int data[], int low, int high, int key) {

int mid;

if (low > high)

cout << "찾는 값이 없습니다." << endl; //찾지 못하면 출력한다.

else {

mid = (low + high) / 2;

if (key == data[mid])

cout << mid + 1 << "번째에 있습니다." << endl; //찾으면 중단한다.

else if (key < data[mid])

location(data, low, mid - 1, key); // 앞부분 재귀 호출한다.

else

location(data, mid + 1, high, key); // 뒷부분 재귀 호출한다

}

}

int main() {

srand((unsigned)time(NULL));

int\* Array = new int[MAX]; //동적할당으로 랜덤 값을 10000개를 받는

int a; //배열 및 변수 선언한다.

for (int i = 0; i < MAX; i++) //0~49999의 값을 랜덤으로 Array에 넣어준다.

Array[i] = ((rand() << 15) | rand()) % 50000;

cout << "찾을 값을 입력하시오.(0~49999)"; //찾는 값을 입력 받아 변수a에 넣는다.

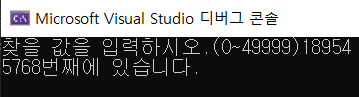
cin >> a;

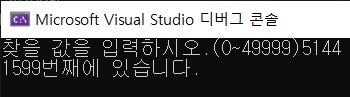
Mergesort(Array, 0, MAX-1); //합병 정렬의 함수 호출한다,

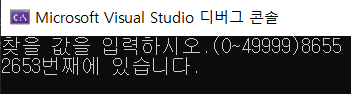
location(Array, 0, MAX-1, a); //이진탐색의 함수 호출한다.

delete[] Array;

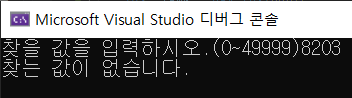
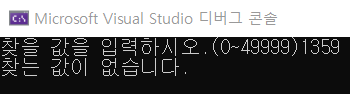
}

찾아지는 경우





찾아지지 않는 경우



과제 수행 소감

문제1에서 만든 선형탐색문제는 랜덤으로 값을 받고 그 배열에서 원하는 값을 하나씩 비교하면서 찾는 프로그램이라 문제2보다 상대적으로 쉽게 만들 수 있었습니다. 문제2는 합병 정렬을 한 후에 이진탐색으로 원하는 값을 찾는 문제인데 이진탐색은 이전에 해보았기에 코드를 짜는데 어렵지 않았지만 합병 정렬을 짜는데 시간이 많이 들었습니다. 분할은 재귀함수 호출로 n/2로 크기가 1이 될 때까지 나누었는데 합병할 때 처음에 data의 값을 Temp에 정렬하기 위해 이중for문을 사용해서 작성하려고 생각했으나 이중for문의 시간 복잡도가 O(n^2)으로 효율적이지 않은것같아서 생각하다가 While문을 써서 완성하였습니다. 프로그램을 짜면서 효율적인 생각보다 실행이 되기 만하면 좋다고 생각하였는데 알고리즘을 배우고 과제를 하니 프로그램을 작성할 때도 사용되는 시간, 공간, 가독성 등 많은 것을 생각하면서 설계해야 된다는 것을 알게 되었습니다.