**Отчет курсанта Громова Григория Андреевича группы 22.Б05 о выполнении практического задания на 17.11.2022**

1. Сортировка слиянием

Алгоритм (подробный алгоритм представлен в комментариях к тексту программы):

1. Алгоритм многократно делит массив на две половины, пока мы не достигнем этапа, на котором массив будет содержать только один элемент.
2. Применяем функцию слияния, которая объединяет отсортированные массивы в массивы большего размера, пока весь массив не будет объединен.

Разделим алгоритм на 2 функции: Слияния и сортировки

Текст программы:

#include <stdio.h>

#include <stdlib.h>

void merge(int arr[], int p, int q, int r); //объявляем функцию слияния

void sort(int arr[], int l, int r); //объявляем функцию сортировки

int main(){

printf("Введите длину массива: ");

int size;

scanf("%d", &size);

int arr[size];

printf("\nЗаполните массив:\n");

for (int i = 0; i < size; i++){

scanf("%d", &arr[i]);

}

printf("Начальный массив: \n");

for (int i = 0; i < size; i++){

printf("%d ", arr[i]);

}

printf("\n");

sort(arr, 0, size - 1);

printf("Отсортированный массив: \n");

for (int i = 0; i < size; i++){

printf("%d ", arr[i]);

}

}

void merge(int arr[], int p, int q, int r) { //функция слияния получает на вход копию массива A[p...r]

//первый и последний индекс первого подмассива и последний

//индекс второго подмассива (нам не нужен первый индекс второго

//подмассива, так как он равен последнему индексу первого +1)

int n1 = q - p + 1;

int n2 = r - q;

int L[n1], M[n2]; //объявляем подмассивы

for (int i = 0; i < n1; i++){ //записываем левую половину

L[i] = arr[p + i];

}

for (int j = 0; j < n2; j++){ //записываем правую половину

M[j] = arr[q + 1 + j];

}

//объявляем 3 указателя:

int i = 0; //текущий индекс L

int j = 0; //текущий индекс M

int k = p; //текущий индекс объединенного массива

// Пока мы не достигнем одного из концов L или M, выбираем больший из элементов L и M

// и помещаем их в соответствующую позицию в A[p...r]

while (i < n1 && j < n2) {

if (L[i] <= M[j]) {

arr[k] = L[i];

i++;

}

else{

arr[k] = M[j];

j++;

}

k++;

}

//Когда либо в L, либо в M закончились элементы, берем оставшиеся и вставляем A[p…r]

while (i < n1){

arr[k] = L[i];

i++;

k++;

}

while (j < n2){

arr[k] = M[j];

j++;

k++;

}

}

// Делим массив на два подмассива, сортируем их и объединяем

void sort(int arr[], int l, int r) {

if (l < r) {

// m это точка, в которой массив делится на два подмассива

int m = l + (r - l) / 2;

sort(arr, l, m);

sort(arr, m + 1, r);

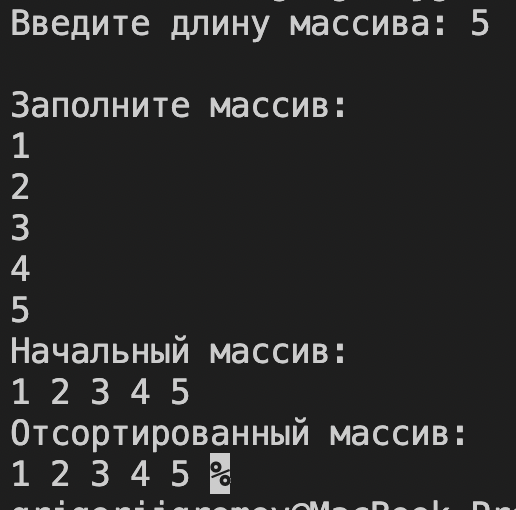
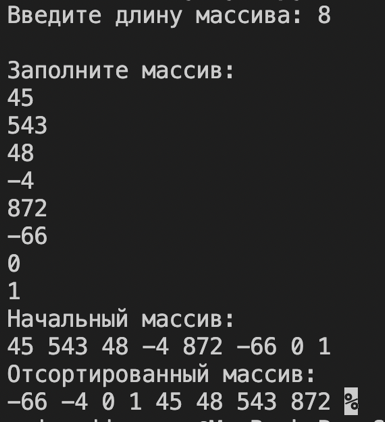
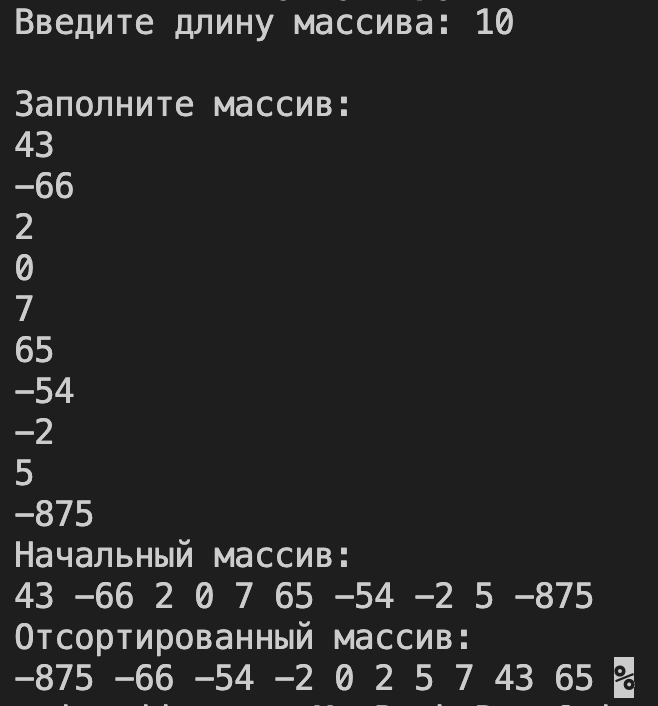
// Объединяем отсортированные подмассивы

merge(arr, l, m, r);

}

}

Протестируем:

1. Бинарный поиск

Дан отсортированный массив и значение, индекс которого нужно найти в этом массиве. Если же его там нет, необходимо сообщить об этом.

Алгоритм (подробный алгоритм представлен в комментариях к тексту программы):

1. Ищем значение элемента с серединным индексом
2. Сравниваем его с искомым значением
3. Если искомое значение меньше, возвращаемся к пункту 1, только вместо исходного массива берем ту его часть, которая расположена до середины. Если же искомое значение больше, возвращаемся к пункту 1, взяв часть массив, расположенную после середины. Если значение совпало, индекс середины – искомая величина.

Текст программы:

#include <stdio.h>

#include <stdlib.h>

int bin\_search(int a, int mass[], int n); //(искомый элемент, массив в котором нужно искать, длина этого массива)

int main()

{

int N = 8; //длина массива

int a;

int mass[] = {-4, -1, 1, 5, 8, 23, 58, 100} ; //произвольный отсортированный массив

printf("Sorted array:\n");

for (int i = 0; i < N; i++)

printf("%d ", mass[i]);

printf("\n");

printf("Введите значение, индекс которого вы хотите найти: ");

scanf("%d", &a);

int k;

k = bin\_search(a, mass, N); //бинарный поиск

if (k != -1){ //если функция не сообщила о том, что индекс не был найден

printf("Индекс искомого элемента равен %d\n", k);

}

else{ //в противном же случае

printf("Элемент в массиве не обнаружен\n");

}

return 0;

}

int bin\_search(int a, int mass[], int n)

{

int l = 0; //индекс первого элемента массива

int r = n - 1; //индекс последнего элемента массива

int mid; //индекс середин

while (l <= r){ //пока левая граница левве правой

mid = (l + r) / 2; //ищем индекс среднего элемента

if (a < mass[mid]) //если искомый элемент меньше элемена, хранящегося по данному индексу

r = mid - 1; //правой границей теперь будет середина

else if (a > mass[mid]){ //и наоборот

l = mid + 1;

}

else { //если же значение совпали, то этот индекс и есть искомый

return mid; //возвращаем его

}

}

return -1; //если не получилось найти, возвращаем -1

}

Проверка работоспособности:

