**Задание 2 задачи 2. Сравнение сортировок.**

Сравним быструю сортировку с сортировкой слиянием.

1. Сортировка слиянием

#include <stdio.h>

#include <stdlib.h>

#include <clocale>

//Сортировка слиянием

void mergeSort(int\* arr, int length)

{

//середина массива

int mid = length / 2;

if (length % 2 == 1)

{

mid++;

}

int h = 1; //шаг

int\* newArr = (int\*)malloc(length \* sizeof(int));

int step; // шаг

while (h < length)

{

step = h;

int i = 0; // первая половина последовательности (от начала последовательнсти до середины)

int j = mid; // вторая половина последовательности (от середины последовательности и до конца)

int index = 0; //

while (step <= mid)

{

while ((i < step) && (j < length) && (j < (mid + step))) //пока не дошли до конца последовательности

{

//если элемент первой половины меньше элемента второй половины

if (arr[i] < arr[j])

{

//заполняем новый массив элементом первой половины

newArr[index] = arr[i];

//идём по первой последовательности

i++; index++;

}

else {

//иначе заполняем массив элементом второй половины

newArr[index] = arr[j];

//идём по второй последовательности

j++; index++;

}

}

while (i < step)

{

//записываем оставшиеся элементы первой последовательности в новый массив

newArr[index] = arr[i];

i++; index++;

}

while ((j < (mid + step)) && (j < length))

{

//записываем оставшиеся элементы второй последовательности в новый массив

newArr[index] = arr[j];

j++; index++;

}

step = step + h;

}

h = h \* 2;

//запись промежуточного массива в исходный массив

for (i = 0; i < length; i++)

arr[i] = newArr[i];

}

}

int main()

{

setlocale(LC\_ALL, "RUssian");

int array[8];

//заполнение массива

for (int i = 0; i < 8; i++)

array[i] = rand() % 20 - 5;

//массив до сортировки

printf("Массив до сортировки слиянием: \n");

for (int i = 0; i < 8; i++)

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

printf("\n");

//вызов функции сортировки

mergeSort(array, 8);

//массив после сортировки

printf("Массив после сортировки слиянием: \n");

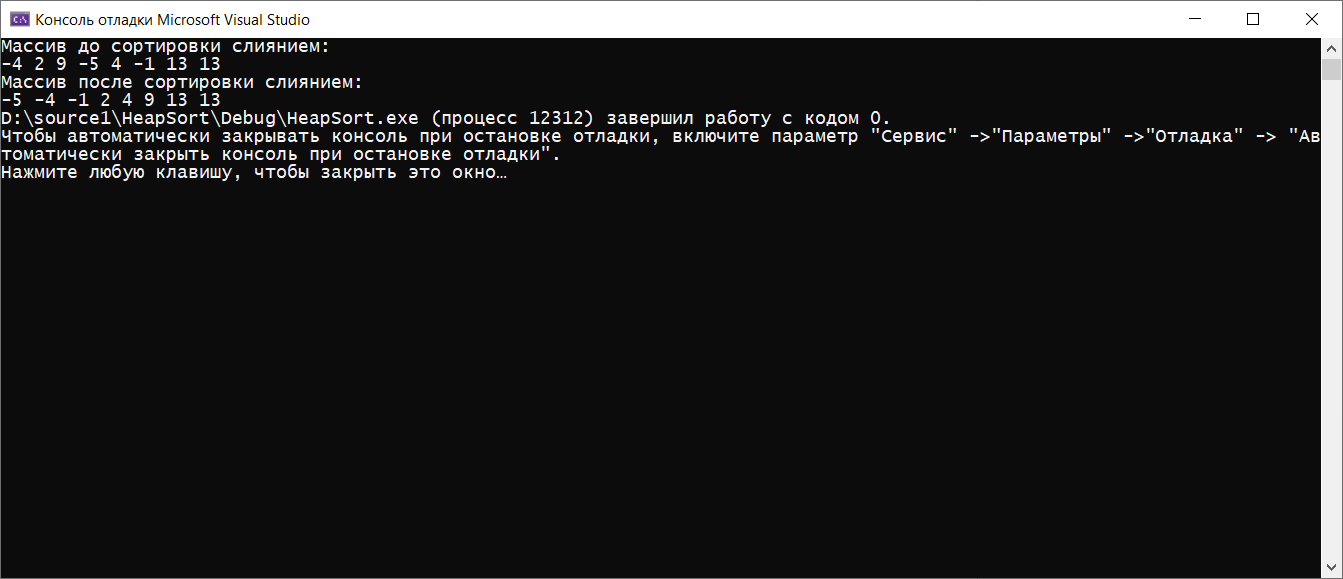
for (int i = 0; i < 8; i++)

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

return 0;

}

Результат выполнения при n = 8:



Сравнительная таблица:

|  |  |
| --- | --- |
| Быстрая сортировка | Сортировка слиянием |
| Скорость:  O(nlog) – лучший случай  O(n2) – худший случай | Скорость:  O(nlog) – лучший случай  O(n2) – худший случай |
| Наиболее часто применяется в работе с массивами | Применяется для работы со структурами с последовательным доступом (связные списки) |
| Недостатки: Зависимость быстродействия от выбора опорного элемента, возможность переполнения стека и вызова ошибки из-за рекурсивных вызовов функций | Недостатки: необходима дополнительная память размером с сортируемую последовательность |
| Плюсы: лаконичность алгоритма и понятность, работает быстрее сортировки слиянием или сортировки пирамидой | Плюсы: не зависит от перестановок элементов в последовательности |