***Сортировка слиянием.***

Рассмотрим вначале *слияние* двух *упорядоченных* массивов на примере.

Пусть задан массив *A,* элементы которого будем обозначать через *a:*

|  |  |  |
| --- | --- | --- |
| *2* | *4* | *6* |

И массив *B,* элементы которого будем обозначать через *b:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *3* | *5* | *7* | *8* | *9* |

Результируюший массив *C* будет содержать *8* ячеек.

*1 шаг*. Сравниваем *первые* элементы двух массивов: *a<b,* если сравнениевыпооняется, в результирующий массив записываем  *a,* в противном соучае записываем  *b* (сюда и входит случай, когда  *a = b* ).

У нас  *a* меньше, поскольку мы взяли элемент из *A*, передвигаем указатель для сравнения на следующий элемент массива *A.* Массивы *A, B* после первого прохола будут иметь вид:

|  |  |  |
| --- | --- | --- |
|  | *4* | *6* |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *3* | *5* | *7* | *8* | *9* |

Массив *C:*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *2* |  |  |  |  |  |  |  |

*2 шаг*. В массиве *A* первый элемент убрали, поэтому сравниваем второй элемент массива *A* с первым элементом *B* и записываем меньший элемент в результирующий массив, в данном случае это *3(из массива В)* и передвигаем указатель по массиву *B:*

|  |  |  |
| --- | --- | --- |
|  | *4* | *6* |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *5* | *7* | *8* | *9* |

Массив *C:*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *2* | *3* |  |  |  |  |  |  |

*3 шаг*. Сравниваем *4* и *5*, записываем *4* в массив *C,* передвигаем указатель для сравнения в массиве *A.*

|  |  |  |
| --- | --- | --- |
|  |  | *6* |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *5* | *7* | *8* | *9* |

Массив *C:*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *2* | *3* | *4* |  |  |  |  |  |

*4 шаг.* Сравниваем 6 и *5*, записываем *5* в массив *C,* передвигаем указатель для сравнения в массиве *B.*

|  |  |  |
| --- | --- | --- |
|  |  | *6* |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | *7* | *8* | *9* |

Массив *C:*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *2* | *3* | *4* | *5* |  |  |  |  |

*5 шаг.* Сравниваем *6* и *7*, записываем *6* в массив *C,* поскольку в массиве *A* нет больше элементов, записываем в массив *C* оставшиеся элементы массива *B.* Получаем упорядоченный результирующий массив.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* |

Теперь рассмотрим *сортировку* *слиянием*, где используется рассмотренное ранее слияние двух упорядоченных массивов. Сортировка слиянием относится к стратегии *«разделяй и властвуй»* и это *рекурсивный* алгоритм.

Рассмотрим также нп примере. Пусть дан массив из *9* элементов:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 1 | 5 | 7 | 6 | 8 | 2 | 3 |

1. Делим наш массив на *2*, затем каждую половинку на *2* и так до тех пор пока не получим *одноэлементные* массивы. Будет выглядеть так:

*|2* | | *3* | |*1* | | *5 | |7 | | 6 | | 8 | | 2 | |3 |*

Теперь каждые *2 одноэлементных* массива сливаем, считая что они *упорядоченные*. Получаем:

*|2 3* | |*1 5* | *|6 7 | |2 8 | |3 |*

Опять сливаем *2-х* *элементные* и получаем *четырехэлементные*:

*|1 2 3 5* | *|2 6 7 8 | |3 |*

*Сливаем четырехэлементные,* получаем *8*-*элементный* массив и *1-элементный.*

*|1 2 2 3 5 6 7 8* |  *|3 |*

Наконец сливаем и их и получаем *отсортированный* массив:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *1* | *2* | *2* | *3* | *3* | *5* | *6* | *7* | *8* |

Ниже приведена возможная реализация *сортировки слиянием*.

#include <iostream>

#include <string.h>

#include<cstdlib>

#include <ctime>

using namespace std;

int random(int dataSize);

int\* getSourceArray(int arraySize);

int\* getResultArray(int\* sourceArray, int sourceLength);

int\* merge(int\* firstArray, int firstLength, int\* secondArray, int secondLength);

int main() {

int arraySize = 11;

int\* sourceArray = getSourceArray(arraySize);

int\* resultArray = getResultArray(sourceArray, arraySize);

cout << "source array :" << endl;

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

cout << sourceArray[i] << ", ";

}

cout << endl;

cout << "result array :" << endl;

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

cout << resultArray[i] << ", ";

}

cout << endl;

return 0;

}

int\* getSourceArray(int arraySize) {

int\* sourceArray = new int[arraySize];

srand(time(0));

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

sourceArray[i] = rand();

}

return sourceArray;

}

int\* getResultArray(int\* sourceArray, int sourceLength) {

int firstLength = sourceLength / 2;

int secondLength = sourceLength - firstLength;

int\* firstArray = new int[firstLength];

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

firstArray[i] = sourceArray[i];

}

int\* secondArray = new int[secondLength];

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

secondArray[i] = sourceArray[firstLength + i];

}

if (firstLength > 1) {

firstArray = getResultArray(firstArray, firstLength); // рекурсивный вызов

} // функции

if (secondLength > 1) {

secondArray = getResultArray(secondArray, secondLength); // рекурсия

}

return merge(firstArray, firstLength, secondArray, secondLength);

}

int\* merge(int\* firstArray, int firstLength, int\* secondArray, int secondLength) {

int\* resultArray = new int[firstLength + secondLength];

int resultPointer = 0;

int firstPointer = 0;

int secondPointer = 0;

int firstElement = 0;

int secondElement = 0;

while (firstPointer < firstLength && secondPointer < secondLength)

{

firstElement = firstArray[firstPointer];

secondElement = secondArray[secondPointer];

if (firstElement < secondElement)

{

resultArray[resultPointer] = firstElement;

firstPointer++;

}

else

{

resultArray[resultPointer] = secondElement;

secondPointer++;

}

resultPointer++;

}

while (firstPointer < firstLength) // добовляем в результирующий массив

{ //оставшиеся элементы первого, второй закончился

resultArray[resultPointer] = firstArray[firstPointer];

resultPointer++;

firstPointer++;

}

while (secondPointer < secondLength) // добовляем в результирующий массив

{ //оставшиеся элементы второго, первый заеончился

resultArray[resultPointer] = secondArray[secondPointer];

resultPointer++;

secondPointer++;

}

return resultArray;

}