ММИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Алгоритмы и структуры данных»

Тема: Сортировки

Студент гр. 9384 _____ Николаев А.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Реализовать алгоритм быстрой сортировки итеративным методом.

Задание.

ВАРИАНТ 12.

Реализовать быструю сортировку итеративным методом.

Выполнение работы.

Была реализована быстрая сортировка итеративным методом. То есть все операции происходят в циклах. Основная идея такого алгоритма состоит в разделении массива на 2 части. Переменные h и l хранят индексы левого и правого конца последовательности.

Далее выполняется следующий перечень действий:

- Будем двигать указатель l с шагом в 1 элемент по направлению к концу массива, пока не будет найден элемент a[l] >= p.
- Затем аналогичным образом начнем двигать указатель h от конца массива к началу, пока не будет найден a[h] <= p.
- Далее, если l <= h, меняем a[l] и a[h] местами и продолжаем двигать l, h по тем же правилам...

Повторяем шаг 3, пока i <= j.

Сравнение сортировок:

Qsort (n log(n)) Сложность

Timsort (n) Сложность

Selectionsort (n^2) Сложность

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

Входные данные	Выходные данные
54321 a < b edcba a < b 12345 a > b abcde a > b	12345 abcde 54321 edcba

Выводы.

В ходе выполнения лабораторный работы была реализована быстрая сортировка итеративным методом.

ПРИЛОЖЕНИЕ

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <functional>
#include <iostream>
#include <string>
#include <vector>
#include <stack>
#include <list>

template<typename T>
void printArr(T arr, int size)
{
	for (int index = 0; index < size; index++)
	{
		std::cout << arr[index] << ' ';
	}
	std::cout << '\n';
}

template<typename T>
bool cmp(T a, T b)
{
	return a < b;
}

template<typename T>
int partition(T* arr, int l, int h, std::function<bool(T, T)> comp)
{
	T x = arr[h];
	int i = (l - 1);

	for (int j = l; j <= h - 1; j++)
	{
		//if (arr[j] <= x) {
		if (cmp(arr[j], x))
		{
			i++;
			//std::cout << "swap: " << arr[i] << " with " << arr[h] << '\n';
			std::swap(arr[i], arr[j]);
		}
	}
	//std::cout << "swap: " << arr[i + 1] << " with " << arr[h] << '\n';
```

```
        std::swap(arr[i + 1], arr[h]);
        printArr(arr, h + 1);
        return (i + 1);
}

template<typename T>
void qSort(T* arr, int count, std::function<bool(T, T)> comp)
{
        std::stack<int> st;
        st.push(0);
        st.push(count);

        while (!st.empty())
        {
                int h = st.top();
                st.pop();
                int l = st.top();
                st.pop();

                int p = partition(arr, l, h, comp);

                if (p - 1 > l)
                {
                        st.push(l);
                        st.push(p - 1);
                }
                if (p + 1 < h)
                {
                        st.push(p + 1);
                        st.push(h);
                }

        }
}


int main()
{
        std::cout << "Enter array size: ";
        int size;
        std::cin >> size;

        int* arr = new int[size];
```

```
for (int index = 0; index < size; index++)
{
	std::cout << "Enter [" << index << "] element: ";
	std::cin >> arr[index];
}

std::cout << "\nStart Array:\n";
printArr(arr, size);

std::function<bool(int, int)> comp = cmp<int>;

qSort(arr, size - 1, comp);

std::cout << "\nEnd Array:\n";
printArr(arr, size);

delete[] arr;

return 0;

/*int arrInt[] = { 78, 1037, 102, 2, -317, 1000000, 3587, 94, -377 };
int sizeInt = sizeof(arrInt) / sizeof(arrInt[0]);

char arrChar[] = "agsdahsdghlfgsdfjhsludhfahsdu";
int sizeChar = sizeof(arrChar) / sizeof(arrChar[0]);

float arrFloat[] = { 0.1, 0.234234, -1.23423, -111, -5.123, 77.655 };
int sizeFloat = sizeof(arrFloat) / sizeof(arrFloat[0]);

std::cout << "Start ArrayInt:\n";
printArr(arrInt, sizeInt);
std::cout << "Start ArrayChar:\n";
printArr(arrChar, sizeChar);
std::cout << "Start ArrayFloat:\n";
printArr(arrFloat, sizeFloat);

qSort(arrInt, sizeInt - 1);
qSort(arrChar, sizeChar - 1);
qSort(arrFloat, sizeFloat - 1);

std::cout << "\nEnd ArrayInt:\n";
printArr(arrInt, sizeInt);

std::cout << "End ArrayChar:\n";
```

```
        printArr(arrChar, sizeChar);

        std::cout << "End ArrayFloat:\n";
        printArr(arrFloat, sizeFloat);

        return 0;*/
}
```