МИНОБРНАУКИ 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. 9304 Сорин А.В.

Преподаватель Филатов А.Ю.

Санкт-Петербург

2020

Цель работы.

Узнать о сортировках и их использовании в практике.

Задание.

Реализовать сортировку с пошаговым выводом элементов.

Вариант 17 – реализация нитевидной сортировки.

Формат входных и выходных данных.

На вход подаются элементы списка, разделяемые пробелами. Например:

1 8 0 3.4 2.1 12 4.57 19

На выходе результат – отсортированный список и результат тестирования.

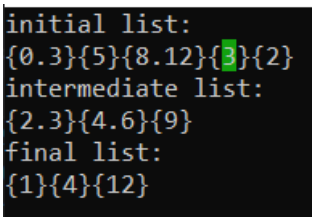
Выполнение работы.

Для выполнения задания был создан класс – sort_list.

В нем содержатся SortList – сортирующийся список. Также есть дефолтные конструктор и деструктор.

Также есть следующие методы: PrintList – выводит список, выделяя элемент. ReadList – считывает список. StrandSort – сама сортировка. ReadNumber – считывает число.

Пример пошагового вывода:



Тестирование.

Тестирование проводится при помощи сравнивая результата сортировки со встроенной сортировкой.

Элементы списка сравниваются по очереди, после чего выводится был пройден тест или нет.

Выводы.

Стало известно о сортировках и их использовании в практике.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <string>
#include "sort.h"

int main() {
	try
	{
		sort_list<double> SL;

		bool Res = 0;
		std::string Str;
		if (!std::getline(std::cin, Str))
			throw std::runtime_error("Error while reading from stream");
		std::stringstream Stream(Str);
		SL.ReadList(Stream);
		SL.StrandSort();

	}
	catch (const std::exception& Error)
	{
		std::cout << Error.what();
	}
	return 0;
}
```

Название файла: sort.h

```
#ifndef __SORT_H
#define __SORT_H

#include <iostream>
#include <stdexcept>
#include <sstream>
#include <list>
#include <chrono>
#include <thread>

template <typename base>
std::ostream& operator<<(std::ostream& os, std::list<base>& res) {
	for (auto it = res.begin(); it != res.end(); it++)
	{
		os << '{' << *it << '}';
	}
	return os << '\n';
}
```

```
template <typename base>
class sort_list {
public:
	sort_list() = default;
	~sort_list() = default;

	void PrintList(std::list<base>& res, int Ind) {
		int i = 0;
		for (auto it = res.begin(); it != res.end(); it++, i++)
		{
			if (i == Ind)
				std::cout << '{' << "\x1b[42m"<< *it << "\x1b[0m" <<
'}';
			else
				std::cout << '{' << *it << '}';
		}
		std::cout << '\n';
	}

	void ReadList(std::stringstream& Stream) {
		char c = 0;
		if (Stream.get(c))
		{
			if (!std::isdigit(c))
				throw std::invalid_argument("Error while entering
expression");
			else
			{
				SortList.push_back(ReadNumber(c, Stream));
				ReadList(Stream);
			}
		}
	}
	void StrandSort(void) {
		std::chrono::milliseconds T(std::chrono::milliseconds(550));
		std::list<base> CorrectSort = SortList;
		CorrectSort.sort();
		if (SortList.size() <= 1)
		{
			system("cls");
			std::cout << "final list:\n";
			std::cout << SortList;
			std::cout << "Test passed, sorting correct\n";
			std::this_thread::sleep_for(T);
```

```
	return;
}
std::list<base> tmp;
std::list<base> res;

while (!SortList.empty())
{
	system("cls");
	std::cout << "initial list:\n";
	PrintList(SortList, 0);
	std::cout << "intermediate list:\n";
	std::cout << tmp;
	std::cout << "final list:\n";
	std::cout << res;
	std::this_thread::sleep_for(T);

	tmp.push_back(SortList.front());
	SortList.pop_front();

	int i = 0;
	for (auto it = SortList.begin(); it != SortList.end();)
	{
		if (tmp.back() <= *it)
		{
			tmp.push_back(*it);
			it = SortList.erase(it);
		}
		else
			it++, i++;
		system("cls");
		std::cout << "initial list:\n";
		PrintList(SortList, i);
		std::cout << "intermediate list:\n";
		std::cout << tmp;
		std::cout << "final list:\n";
		std::cout << res;
		std::this_thread::sleep_for(T);
	}
	res.merge(tmp);

	system("cls");
	std::cout << "initial list:\n";
	std::cout << SortList;
	std::cout << "intermediate list:\n";
	std::cout << tmp;
```

```
            std::cout << "final list:\n";
            std::cout << res;
            std::this_thread::sleep_for(T);

        }

        system("cls");
        std::cout << "initial list:\n";
        std::cout << SortList;
        std::cout << "intermediate list:\n";
        std::cout << tmp;
        std::cout << "final list:\n";
        std::cout << res;

        system("cls");
        std::this_thread::sleep_for(T);
        std::this_thread::sleep_for(T);
        std::cout << "final list:\n";
        std::cout << res;

        SortList = std::move(res);
        bool IsTestPassed = true;
        for (auto it = SortList.begin(); it != SortList.end(); it++)
        {
            if (CorrectSort.front() == *it)
            {
                CorrectSort.pop_front();
            }
            else
            {
                IsTestPassed = false;
                break;
            }
        }
        if (IsTestPassed)
            std::cout << "Test passed, sorting correct\n";
        else
            std::cout << "Test failed, sorting uncorrect\n";
    }
private:
    std::list<base> SortList;

    base ReadNumber(char k, std::stringstream& Stream) {
        base Num;
```

```
          if (k == '0')
          {
               if (!Stream.get(k))
                    throw std::invalid_argument("Error while entering
expression");
               if (k == '.')
               {
                    int I = 0;
                    if (!Stream.get(k))
                         throw std::invalid_argument("Error while
entering expression");
                    if (!std::isdigit(k))
                         throw std::invalid_argument("Error while
entering expression");
                    Num = 0.1 * ((base)k - '0');
                    I++;
                    while (1)
                    {
                         if (!Stream.get(k))
                              throw std::invalid_argument("Error while
entering expression");
                         if (std::isdigit(k))
                         {
                              base deg = 0.1;
                              for (int i = 0; i < I; i++)
                                   deg /= 10;
                              Num += deg * ((base)k - '0');
                              I++;
                         }
                         else if (k == ' ')
                              return Num;
                         else
                              throw std::invalid_argument("Error while
entering expression");
                    }
               }
               else if (k == ' ')
                    Num = 0;
               else
                    throw std::invalid_argument("Error while entering
expression");

               return Num;
          }
          if (!std::isdigit(k))
```

```
                throw std::invalid_argument("Error while entering
expression");
           Num = ((base)k - '0');
           while (1)
           {
                if (!Stream.get(k))
                     throw std::invalid_argument("Error while entering
expression");
                if (std::isdigit(k))
                {
                     Num *= 10;
                     Num += ((base)k - '0');
                }
                else if (k == ' ')
                     return Num;
                else if (k == '.')
                {
                     int I = 0;
                     if (!Stream.get(k))
                          throw std::invalid_argument("Error while
entering expression");
                     if (!std::isdigit(k))
                          throw std::invalid_argument("Error while
entering expression");
                     Num += 0.1 * ((base)k - '0');
                     I++;
                     while (1)
                     {
                          if (!Stream.get(k))
                               throw std::invalid_argument("Error while
entering expression");
                          if (std::isdigit(k))
                          {
                               base deg = 0.1;
                               for (int i = 0; i < I; i++)
                                    deg /= 10;
                               Num += deg * ((base)k - '0');
                               I++;
                          }
                          else if (k == ' ')
                               return Num;
                          else
                               throw std::invalid_argument("Error while
entering expression");
                     }
```

```
                }
                else
                      throw std::invalid_argument("Error while entering
expression");
           }
     }
};


#endif // __SORT_H
```

Название файла: Makefile

```
Lab4: ./src/main.cpp
     g++ -std=c++17 ./src/main.cpp -o lab4
```