*Федеральное государственное бюджетное образовательное учреждениевысшего профессионального образования*

|  |  |
| --- | --- |
|  | ***«Московский государственный технический университет***  ***имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ: Информатики и систем управления

КАФЕДРА: Компьютерные системы и сети

**Отчет**

**по усложнённой лабораторной работе №** 6

**Название лабораторной работы:** *Динамические структуры. Деревья.*

**Дисциплина:** Объектно-ориентированное программирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент гр. ИУ6-22 |  |  | | Боярских Никита |
|  | (Подпись, дата) | |  | (И. О. Фамилия) |
| Преподаватель |  | |  | Черноусова Т. Г. |
|  | (Подпись, дата) | |  | (И. О. Фамилия) |

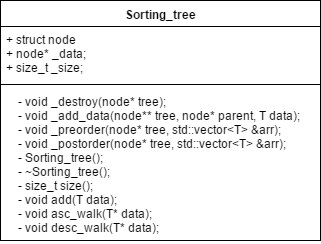
**Вариант 1**

Москва, 2017

**Задание:**

Реализовать программу сортировки строк с использованием дерева. Обеспечить вывод элементов по возрастанию и убыванию.

Для поставленной задачи разработан класс Sorting\_tree (диаграмма классов представлена на рис. 1)



*Рисунок 1 - диаграмма классов программы, работающей с деревом*

**Код программы, работающей с деревом**

#include <iostream>

#include <string>

#include <vector>

// Дерево ссортировки

template <class T>

class Sorting\_tree {

private:

struct node {

node\* parent;

node\* left;

node\* right;

std::string value;

};

node\* \_data;

size\_t \_size;

void \_destroy(node\* tree);

void \_add\_data(node\*\* tree, node\* parent, T data);

void \_preorder(node\* tree, std::vector<T>&arr);

void \_postorder(node\* tree, std::vector<T>&arr);

public:

Sorting\_tree();

~Sorting\_tree();

size\_t size();

void add(T data);

void asc\_walk(T\* data);

void desc\_walk(T\* data);

};

(Продолжение кода на следующей странице)

(Продолжение кода)

int main(int argc, char \*\*argv)

{

using std::cout;

using std::cin;

using std::endl;

Sorting\_tree<std::string> tree;

std::string s;

cout <<"Введите строки для сортировки:"<< endl;

while( getline(cin, s) ) {

tree.add(s);

}

cout << endl <<"Отсортированные по возрастанию данные:"<< endl;

std::string\* asc = new std::string[tree.size()];

tree.asc\_walk(asc);

for(size\_t i = 0; i < tree.size(); i++) {

cout << asc[i] << ' ';

}

cout << endl <<"Отсортированные по убыванию данные:"<< endl;

std::string\* desc = new std::string[tree.size()];

tree.desc\_walk(desc);

for(size\_t i = 0; i < tree.size(); i++) {

cout << desc[i] << ' ';

}

cout << endl;

delete[] asc;

delete[] desc;

return 0;

}

template <class T>

Sorting\_tree<T>::Sorting\_tree()

{

\_size = 0;

\_data = NULL;

}

template <class T>

size\_t Sorting\_tree<T>::size()

{

return \_size;

}

template <class T>

void Sorting\_tree<T>::add(T data)

{

\_add\_data(&\_data, NULL, data);

\_size++;

}

template <class T>

void Sorting\_tree<T>::\_add\_data(node\*\* tree, node\* parent, T data)

{

(Продолжение кода на следующей странице)

(Продолжение кода)

if(\*tree == NULL) {

\*tree = new node;

(\*tree)->value = data;

(\*tree)->parent = parent;

(\*tree)->left = NULL;

(\*tree)->right = NULL;

return;

}

if(data < (\*tree)->value) {

\_add\_data(&(\*tree)->left, \*tree, data);

} else {

\_add\_data(&(\*tree)->right, \*tree, data);

}

}

template <class T>

void Sorting\_tree<T>::asc\_walk(T\* data)

{

if(data == NULL) return;

if(\_data == NULL) return;

std::vector<T> vec;

\_preorder(\_data, vec);

for(size\_t i = 0; i < vec.size(); i++) data[i] = vec[i];

}

template <class T>

void Sorting\_tree<T>::\_preorder(node\* tree, std::vector<T>&vec)

{

if(tree == NULL) return;

\_preorder(tree->left, vec);

vec.push\_back(tree->value);

\_preorder(tree->right, vec);

}

template <class T>

void Sorting\_tree<T>::desc\_walk(T\* data)

{

if(data == NULL) return;

if(\_data == NULL) return;

std::vector<T> vec;

\_postorder(\_data, vec);

for(size\_t i = 0; i < vec.size(); i++) data[i] = vec[i];

}

template <class T>

void Sorting\_tree<T>::\_postorder(node\* tree, std::vector<T>&vec)

{

if(tree == NULL) return;

\_postorder(tree->right, vec);

vec.push\_back(tree->value);

\_postorder(tree->left, vec);

}

(Продолжение кода на следующей странице)

(Продолжение кода)

template <class T>

void Sorting\_tree<T>::\_destroy(node\* tree)

{

if(tree == NULL) return;

\_destroy(tree->left);

\_destroy(tree->right);

delete tree;

}

template <class T>

Sorting\_tree<T>::~Sorting\_tree()

{

\_destroy(\_data);

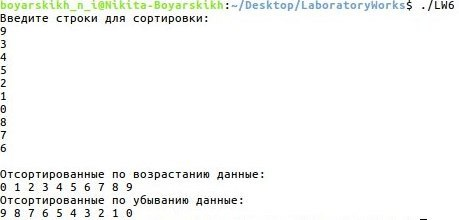
}

Проведено тестирование программы (см. табл. 1)

Таблица 1 – результаты тестирования программы, работающей с деревом

|  |  |  |  |
| --- | --- | --- | --- |
| № | Исходные данные | Ожидаемый результат | Полученный результат |
| 1 | 9 3 4 5 2 1 0 8 7 6 | 0 1 2 3 4 5 6 7 8 9 | 0 1 2 3 4 5 6 7 8 9 |
| 2 | 1 6 4 74 325 45 99 0 124 -23 вмпви 13p43 je;gq | -23 0 1 124 13p43 325 4 45 6 74 99 je;gq вмпви  вмпви je;gq 99 74 6 45 4 325 13p43 124 1 0 -23 | -23 0 1 124 13p43 325 4 45 6 74 99 je;gq вмпви  вмпви je;gq 99 74 6 45 4 325 13p43 124 1 0 -23 |
| 3 | string some string new string(1) other string enoth one many | enoth one many new string(1) other string some string string | enoth one many new string(1) other string some string string |

**Скриншоты, демонстрирующие работу программы**



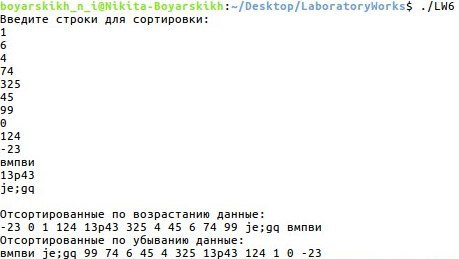
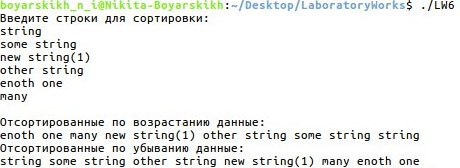
*Рисунок 2 – тестирование программы при исходных данных №1 (см. табл. 1)*

Рисунок 3 – тестирование программы при исходных данных №2 (см. табл. 1)



*Рисунок 4 – тестирование программы при исходных данных №3 (см. табл. 1)*

**Вывод:**

1) Разработаны классы для решения поставленной задачи. Составлена диаграмма в среде [draw.io](http://draw.io) (рис. 1)

2) Создан код программы по алгоритму в среде Vim

3) Проведено тестирование (см. табл. 1)

4) Тестирование показало корректность работы при заданных исходных данных (см. табл. 1 и рис. 2 - 4)