

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ
по лабораторной работе №5
по дисциплине «Алгоритмы и структуры данных»
Тема: Случайное БДП с рандомизацией

Студент гр. 9304

Афанасьев А.

Преподаватель

Филатов А.Ю.

Санкт-Петербург

2020

Цель работы.

Реализовать случайное БДП с рандомизацией и соответствующие методы.

Постановка задачи.

Вариант 10.

По заданной последовательности элементов *Elem* построить случайное БДП с рандомизацией. Для построенной структуры данных проверить, входит ли в неё элемент *e* типа *Elem*, и если входит, то в скольких экземплярах. Добавить элемент *e* в структуру данных

Выполнение работы.

Программа на вход ожидает строку для конструирования дерева и элемент *e*.

Тестирование.

Программу можно собрать командой *make*, после этого создается исполняемый файл *lab4*. Его можно запустить, передав в него строку. Также можно запустить тестирующий скрипт *testScript.py*, конфигурационный файл которого лежит в папке с исполняемым файлом. В конфигурационном файле можно настроить многие параметры, включая количество тестов и директорию, в которой они находятся. В тестовом файле должна находиться только лишь одна строка – сам тест. Программа возвращает сообщение об синтаксической ошибке ввода, если такая была, либо ответ. Тестирующий скрипт выводит на экран поданную строку, результат работы программы, правильный ответ и *success* или *fail* в зависимости от совпадения того, что вернула программа, и правильного ответа. Пример его работы можно посмотреть на рисунке 1. А в таблице 1 можно посмотреть примеры строк-тестов.

```
strx@strxpc:~/gitreps/main/Programs/ETU/3SEM/AaDS/lb4$ python testScript.py
Make sure that this script is in the same directory as the program execute file.
```

```
test0:
Input: "12"
CorrectAnswer: 1 2
Answer: 1 2
Result: success

test1:
Input: "3251784"
CorrectAnswer: 1 2 3 4 5 7 8
Answer: 1 2 3 4 5 7 8
Result: success

Total: Successes: 2. Fails: 0
```

Рисунок 1 - Пример вызова скрипта

Таблица 1. Примеры входных и выходных данных

№	Входные данные	Выходные данные
1	1 2	1 2
2	3 2 5 1 7 8 4	1 2 3 4 5 7 8
3	2 5 5 4 1 2 3 1 2 5	1 1 2 2 2 3 4 5 5 5
4	4 6 5 6 8 7 8 9 0 9 0 8 9 7 5 4 3 6 7 8	0 0 3 4 4 5 5 6 6 6 7 7 7 8 8 8 8 9 9 9
5	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
6	9 8 7 6 5 4 3 2 1	1 2 3 4 5 6 7 8 9
7	a b c d z x y	a b c d x y z

Выводы.

В ходе выполнения лабораторной работы было реализовано случайное БДП с рандомизацией. Выяснили, что алгоритмы вставки, поиска и удаления работают в нем быстрее, чем в обычном БДП, но худший случай все еще имеет место быть.

ПРИЛОЖЕНИЕ А

main.cpp

```
#include "../libs/IterQSort.h"

#include <iostream>
#include <string>
#include <algorithm>

int main(int argc, char const *argv[])
{
    std::string str(argv[1]), strCopy = str;

    iterQSort<std::string::iterator>(std::begin(str),
std::end(str));

    std::sort(strCopy.begin(), strCopy.end());

    std::cout << "std::sort:\n";
    for (auto it = strCopy.begin(); it != strCopy.end(); +
it)
        std::cout << *it << ' ';

    std::cout << '\n';
    return 0;
}
```

IterQSort.h

```
#ifndef __ITERQSORT__H__
#define __ITERQSORT__H__

#include <iostream>
#include <stack>
#include <utility>

template <typename RandomIt>
void swapObjs(RandomIt left, RandomIt right)
{
    auto tmpSwap = *left;
    *left = *right;
    *right = tmpSwap;
}

template <typename RandomIt>
RandomIt partition(RandomIt left, RandomIt right)
{
    RandomIt pivotPosSelector = left;
    RandomIt pivot = right;

    for (auto it = left; it != pivot; ++it)
    {
        if (*it < *pivot)
```

```

        {
            if (it != pivotPosSelector)
                swapObjs<RandomIt>(it, pivotPosSelector);
            ++pivotPosSelector;
        }
    }
    // ставим pivot на свое место
    swapObjs<RandomIt>(pivot, pivotPosSelector);
    return pivotPosSelector;
}

template <typename RandomIt>
void iterQSort(RandomIt start, RandomIt end)
{
    if (start < end)
    {
        std::stack<std::pair<RandomIt, RandomIt>> stck;
        stck.push(std::make_pair<RandomIt,
RandomIt>(std::move(start), std::move(end)));

        // счетчик шагов
        int counter = 0;

        // вывод на экран шага
        std::cout << "Step 0:\n";
        for (auto it = start; it != end; ++it)
            std::cout << *it << ' ';
        std::cout << '\n';

        while (!stck.empty())
        {
            ++counter;
            std::pair<RandomIt, RandomIt> borders =
stck.top();
            stck.pop();

            RandomIt pivot = partition(borders.first,
borders.second - 1);

            // вывод на экран шага
            std::cout << "Step " + std::to_string(counter) +
":\n";
            for (auto it = start; it != end; ++it)
                std::cout << *it << ' ';
            std::cout << '\n';

            // если оба отрезка длиннее 1-го элемента
            if ((pivot - 1 > borders.first) && (pivot + 1 <
borders.second - 1))
            {
                // больший отрезок первым кладем на стек. Так
мы в среднем уменьшим наибольшую высоту стека на один.
                if (((pivot - 1) - borders.first) >
(borders.second - 1) - (pivot + 1))
                {

```

```

        stck.push(std::make_pair<RandomIt,
RandomIt>(std::move(std::get<0>(borders)),
std::move(pivot)));
        stck.push(std::make_pair<RandomIt,
RandomIt>(pivot + 1, std::move(std::get<1>(borders))));
    }
    else
    {
        stck.push(std::make_pair<RandomIt,
RandomIt>(pivot + 1, std::move(std::get<1>(borders))));
        stck.push(std::make_pair<RandomIt,
RandomIt>(std::move(std::get<0>(borders)),
std::move(pivot)));
    }
}
else
{
    if (pivot - 1 > borders.first)
        stck.push(std::make_pair<RandomIt,
RandomIt>(std::move(std::get<0>(borders)),
std::move(pivot)));
    if (pivot + 1 < borders.second - 1)
        stck.push(std::make_pair<RandomIt,
RandomIt>(pivot + 1, std::move(std::get<1>(borders))));
}
}
}

#endif //!__ITERQSORT__H__

```

Makefile

```

compiler = g++
flags = -c -g -std=c++17 -Wall
appname = lab4
lib_dir = Sources/libs/
src_dir = Sources/srcs/

src_files := $(wildcard $(src_dir)*)
obj_files := $(addsuffix .o, $(basename $(notdir $
(src_files))))

define compile
    $(compiler) $(flags) $<
endef

programbuild: $(obj_files)
    $(compiler) $^ -o $(appname)

%.o: $(src_dir)/%.cpp $(lib_dir)/*.h
    $(call compile)

clean:
    rm -f *.o $(appname)

```