

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

**ОТЧЕТ**  
**по лабораторной работе №5**  
**по дисциплине «Алгоритмы и структуры данных»**  
**Тема: Случайное бинарное дерево поиска с рандомизацией**

Студент гр. 9304

Преподаватель

\_\_\_\_\_  
\_\_\_\_\_

Прокофьев М.Д.

Филатов А.Ю.

Санкт-Петербург

2020

## **Цель работы.**

Узнать о БДП с рандомизацией и о их использовании в практике

## **Задание.**

*БДП: случайное БДП с рандомизацией; действие:*

- 1) По заданной последовательности элементов Elem построить структуру данных определённого типа – БДП или хеш-таблицу;*
- 2) Записать в файл элементы построенного БДП в порядке их возрастания; вывести построенное БДП на экран в наглядном виде.*

## **Выполнение работы.**

Для выполнения работы было создано несколько функций, а именно: `r_Insert()`, `Insert_to_begin()`, `to_rotate_left()`, `to_rotate_right()`, `chance()`, `print()`, `drawTree()`

`r_Insert()` служит для вставки элемента в полученное БДП, при выполнении этой функции есть вероятность запуска рекурсивной функции вставки элемента в корень, в остальных случаях функция `r_Insert()` вызывает саму себя. Вероятность определяется функцией `chance()`

`Insert_to_begin()` предназначена для рекурсивного вызова вставки элемента в корень. В теле функции присутствует выполнение функций `to_rotate_left()` и `to_rotate_right()` (которые “переворачивают” дерево, соответственно влево или вправо), посредством которых элемент “выталкивается в начало”.

Функции `to_rotate_left()` и `to_rotate_right()` предназначены для переворачивания дерева. “Переворачивание” происходит посредством определенной переменной указателей элементов БДП.

Т.к. итоговое дерево является БДП с рандомизацией, то в простом выводе его в ЛКП, элементы будут расположены по возрастанию. Этот вывод реализован в функции `print()`.

`drawTree()` используется для вывода изображения дерева.

### **Тестирование.**

Был написан python-скрипт для проведения тестов(testing.py). Посредством testing.py, в argv[] (“входной” строке в программе) последовательно передаются строки из файлов SuccessTests.txt и ErrorTests.txt. Таким образом, при “запуске” Makefile, программа обрабатывает строки из вышеуказанных файлов(SuccessTests.txt и ErrorTests.txt), после чего выдает ответы в консоли.

Результаты тестирования изложены в приложении В.

### **Выводы.**

Написана программа с использованием БДП с рандомизацией. Было реализовано создание соответствующего БДП, вывод элементов в порядке возрастания, а также вывод самого дерева.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

**Файл: main.cpp**

```
#include <iostream>
#include <memory>
#include "string.h"

class binSTree{
public:
    char info;
    int count;
    int number;
    std::shared_ptr<binSTree> lt {nullptr};
    std::shared_ptr<binSTree> rt {nullptr};
};

void drawTree(std::shared_ptr<binSTree> curNode, int level, int direction)
{
    if(curNode)
    {
        drawTree(curNode->rt, level + 1, 1);
        for(int i = 0; i < level; i++) std::cout << "  ";
        if(direction == 1) std::cout << "/" << " ";
        else if(direction == 2) std::cout << "\\ " << " ";
        std::cout << curNode->info << std::endl;
        drawTree(curNode->lt, level + 1, 2);
    }
}
```

```

void print(std::shared_ptr<binSTree> curNode){
    if(curNode == nullptr){
        return;
    }
    print(curNode->lt);
    std::cout << curNode->info;
    print(curNode->rt);
}

```

```

bool chance(int random)
{
    srand(time(0));
    return !(rand() % random);
}

```

```

void to_rotate_right(std::shared_ptr<binSTree>& t)
{
    std::shared_ptr<binSTree> x;
    if(t==NULL)
        std::cout<<"to_rotate_right(NULL)"<<std::endl;
    else
    {
        x=t->lt;
        t->lt=x->rt;
        x->rt=t;
        t=x;
    }
}

```

```

void to_rotate_left(std::shared_ptr<binSTree>& t)
{
std::shared_ptr<binSTree> x;
if(t==NULL)
    std::cout<<"to_rotate_right(NULL)"<<std::endl;
else
    {
        x=t->rt;
        t->rt = x->lt;
        x->lt=t;
        t=x;
    }
}

void Insert_to_begin(std::shared_ptr<binSTree>& b, char x)
{
if(b==NULL)
    {
        b = std::make_shared<binSTree>();
        if(b!=NULL)
            {
                b->info=x;
                b->count=1;
            }
        else
            {
                std::cout << "1 Memory not enough\n";
                exit(1);
            }
    }
}

```

```

        }
else
if(x<b->info)
    {
        Insert_to_begin(b->lt, x);
        to_rotate_right(b);
    }
else if(x>b->info)
    {
        Insert_to_begin(b->rt, x);
        to_rotate_left(b);
    }
else b->count++;
}

```

```

void r_Insert(std::shared_ptr<binSTree>& b, char x)
{
if(b==NULL)
    {
        b = std::make_shared<binSTree>();
        if(b!=NULL)
            {
                b->info=x;
                b->count=1;
                b->number=1;
                return;
            }
        else

```

```

        {
            std::cout << "1 Memory not enough\n";
            exit(1);
        }
    }

    if(chance(b->number+1))
    {
        Insert_to_begin(b,x);
        return;
    }

    if(x<b->info)
        r_Insert(b->lt, x);
    else
        r_Insert(b->rt, x);
    b->number++;
}

```

```

int main(int argc, char* argv[])
{
    char* text;
    strcpy(text, argv[1]);
    std::shared_ptr<binSTree> My_Tree = NULL;
    for(int i=0; text[i]; i++)
        r_Insert(My_Tree, text[i]);
    print(My_Tree);
    std::cout << "\n";
    drawTree(My_Tree, 0, 0);
    return 0;
}

```

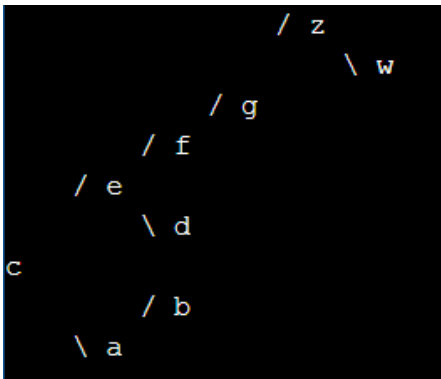

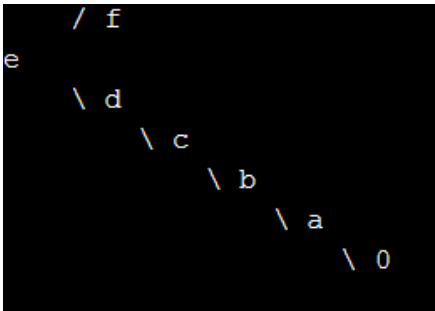
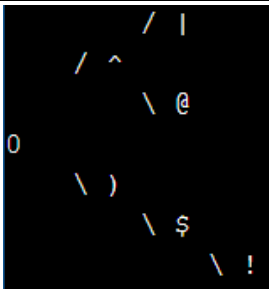
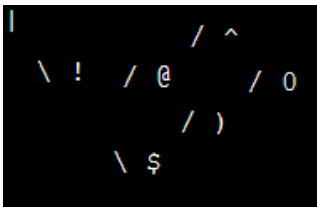


## ПРИЛОЖЕНИЕ В

### ТЕСТИРОВАНИЕ

Результаты тестирования представлены в таблице Б.1

Таблица Б.1 — Результаты тестирования

№ п/п	Входные данные	Выходные данные (изображение)	Выходные данные (элементы по возрастанию)
1.	ceafbdgzw		abcdefghijklmnopqrstuvwxyz
2.	a		a
3.	0abcdfe		0abcdef
4.	!@\$^0		!\$)0@^
5.	!@\$^0 (для проверки случайности)		!\$)0@^