МИНИСТЕРСТВО ОБРАЗОВАНИЯ и НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Коломенский институт (филиал)**

федерального государственного бюджетного образовательного учреждения

высшего образования

**«Московский политехнический университет»**

Профиль «**Программное обеспечение вычислительной техники и автоматизированных систем**»

**Направление подготовки:**

**«Информатика и вычислительная техника»**

Форма обучения: очная

**лабораторная работа № 6**

по дисциплине «Структуры и алгоритмы обработки данных»

Выполнил студент группы ИВТ-21 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /М.А. Крупин

(подпись) И.О. Фамилия

Проверил преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ И.Н. Филоненко /

(подпись) И.О. Фамилия

Дата:

Коломна 2020

**Лабораторная работа № 5**

**Деревья , как динамические структуры данных**

Программно организовать построение дерева Фибоначчи(Ф-дерево), как пример AVL-дерева.

Замечание: дерево Фибоначчи определяется следующим образом:

1) пустое дерево есть Ф-дерево высотой 0;

2)единственная вершина есть дерево высотой 1;

3)если Th-1 и Th-2 -Ф-деревья с высотами (h-1) и (h-2), то

Th =(Th-1,x,Th-2) также Ф-дерево высотой h;

4)других деревьев Фибоначчи не существует.

***Код:***

#include <iostream>

using namespace std;

struct Node

{

int Data;

Node\* Left;

Node\* Right;

};

class Tree

{

private:

Node\* Root;

public:

Tree() { Root = NULL; };

void PrintTree(Node\*, int, char pren);

void FibonTreeBuild(int, Node\*\*);

void FibonTreeFilling(Node\*\*, int\*);

Node\*\* GetTree() { return &Root; };

Node\* GetTree1() { return Root; };

};

void Tree::PrintTree(Node\* W, int l, char pren)

{

int i;

if (W != NULL)

{

PrintTree(W->Right, l + 1, 'r');

for (i = 1; i <= l; i++) cout << " ";

if (pren == 'l')

cout <<"|\_\_"<< W->Data <<endl;

else if (pren == 'r')

cout << "/--- " << W->Data << endl;

else

cout << W->Data << pren << endl;

//cout << W->Data << pren << endl;

PrintTree(W->Left, l + 1, 'l');

}

}

void Tree::FibonTreeBuild(int k, Node\*\* T)

{

if (k == 0)

{

(\*T) = NULL;

}

else

if (k == 1)

{

(\*T) = new (Node);

(\*T)->Left = (\*T)->Right = NULL;

}

else

{

(\*T) = new (Node);

FibonTreeBuild(k - 1, &((\*T)->Left));

FibonTreeBuild(k - 2, &((\*T)->Right));

}

}

void Tree::FibonTreeFilling(Node\*\* T, int\* i)

{

if ((\*T) != NULL)

{

FibonTreeFilling(&((\*T)->Left), i);

(\*T)->Data = (\*i)++;

FibonTreeFilling(&((\*T)->Right), i);

}

}

void main()

{

setlocale(LC\_ALL, "");

Tree A;

int h;

cout << "Input h:";

cin >> h;

int i = 1;

A.FibonTreeBuild(h, A.GetTree());

A.FibonTreeFilling(A.GetTree(), &i);

A.PrintTree(A.GetTree1(), 0, 'm');

}





