**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Информационной безопасности**

отчет

**по лабораторной работе №4**

**по дисциплине «Программирование»**

Тема: Построение, заполнение и обход бинарного дерева

|  |  |  |
| --- | --- | --- |
| Студент гр. 6361 |  | Нерсисян А. |
| Преподаватель |  | Халиуллин Р.А. |

Санкт-Петербург

2017

**Постановка задачи**

Необходимо построить бинарное дерево, каждый узел которого хранит один символ — букву латинского алфавита или цифру. Значение символа должно генерироваться случайным образом при построении дерева.

После построения бинарного дерева необходимо выполнить обход дерева всеми следующими способами:

- прямой обход - сверху вниз (префиксный);

- симметричный обход - слева направо (инфиксный);

- обратный обход - снизу в верх (постфиксный).

При обходе дерева необходимо вывести в консоль символ, значение которого хранится в текущем узле.

Количество узлов дерева и связи между ними — по выбору студента.

В отчет необходимо включить схему дерева, которое строится в реализованной программе. В схеме узлы дерева должны содержать те случайные значения, которые выведены в рисунках с результатом работы реализованной программы.

**Схема дерева**

**Исходный код**

#include <iostream>

using namespace std;

struct Node

{

char info;

Node \*left;

Node \*right;

};

int i = 0;

void Add\_Node(int x, Node \*&p)

{

if (NULL == p)

{

p = new Node;

p->info = x;

p->left = p->right = NULL;

}

if (x < p->info)

{

if (p->left != NULL) Add\_Node(x, p->left);

else

{

p->left = new Node;

p->left->left = p->left->right = NULL;

p->left->info = x;

}

}

if (x > p->info)

{

if (p->right != NULL) Add\_Node(x, p->right);

else

{

p->right = new Node;

p->right->left = p->right->right = NULL;

p->right->info = x;

}

}

}

void Show\_Prefics(Node \*p)

{

if (p)

{

cout << p->info << ' ';

Show\_Prefics(p->left);

Show\_Prefics(p->right);

}

}

void Show\_Infics(Node \*p)

{

if (p)

{

Show\_Infics(p->left);

cout << p->info << ' ';

Show\_Infics(p->right);

}

}

void Show\_Postfics(Node \*p)

{

if (p)

{

Show\_Postfics(p->left);

Show\_Postfics(p->right);

cout << p->info << ' ';

}

}

void Delete\_Tree(Node \*&p)

{

if (p != NULL)

{

Delete\_Tree(p->left);

Delete\_Tree(p->right);

delete p;

p = NULL;

}

}

int main()

{

Node \*Root, \*p;

char x = 0;

Root = 0;

p = 0;

while (i < 8)

{

i++;

x = rand() % 62;

if (x < 10) x += 48;

else

{

x -= 10;

if (x < 26) x += 65;

else

{

x -= 26;

if (x < 26) x += 97;

}

}

Add\_Node(x, Root);

}

cout << "Prefics\n";

Show\_Prefics(Root);

cout << "\nInfics\n";

Show\_Infics(Root);

cout << "\nPostfics\n";

Show\_Postfics(Root);

cin.get();

Delete\_Tree(Root);

return 0;

}

**Результаты тестирования**

Было построено бинарное дерево, каждый узел которого хранит один символ — букву латинского алфавита или цифру. Значение символа генерируется случайным образом при построении дерева.

После построения бинарного дерева выполняется обход дерева всеми следующими способами:

- прямой обход - сверху вниз (префиксный);

- симметричный обход - слева направо (инфиксный);

- обратный обход - снизу в верх (постфиксный).

При обходе дерева выводятся в консоль символы, значение которых хранится в узлах дерева.

Количество узлов дерева и связи между ними — 8.

На рисунке 1 программа построила дерева и обходила тремя способами (Префикс, Инфикс, Постфикс).

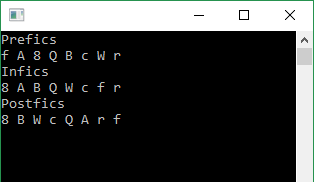


рисунок 1