Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №4

по теме «Бинарное дерево поиска»

Выполнили:

Студенты группы 21ВВ2

Назаров Е.А.

Макаров И.С.

Принял:

Митрохин М.А.

Юрова О.В.

Пенза, 2022

**Цель работы:**

Изучить структуру данных бинарных деревьев поиска, разработать класс для работы с ней и получить практический навык его использования.

**Лабораторные работы:**

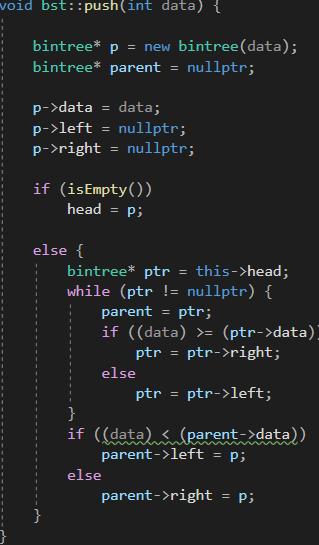
**Задание:**

1. Реализовать алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве.
2. Реализовать функцию подсчёта числа вхождений заданного элемента в дерево.
3. \* Изменить функцию добавления элементов для исключения добавления одинаковых символов.
4. \* Оценить сложность процедуры поиска по значению в бинарном дереве.

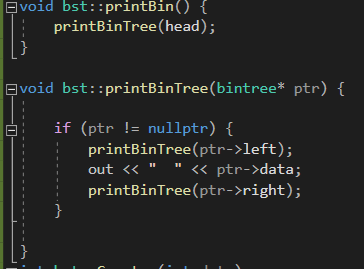
**Пояснительный текст к программе:**



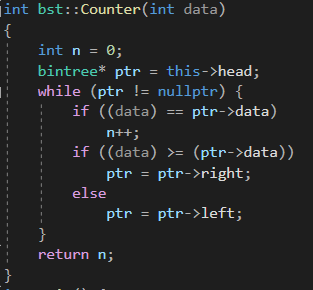
Был создан класс bst в котором находятся все функции и операции, связанные с нашим бинарным деревом, в privet части создана структура элемента бинарного дерева и указатели.



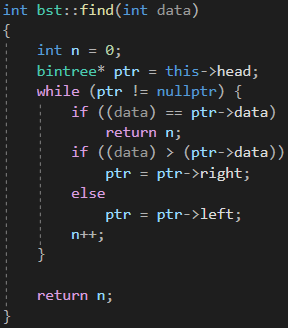
Функция push отвечает за добавления элемента в дерево. Сначала проводится проверка на пустоту списка, а потом операция добавления. Меньший элемент идет в левую часть дерева, больший в правую. Головой списка является самый первый добавленный элемент, добавленный в наше дерево.



Функции вывод дерева на экран или в файл. Из main части кода вызывается изначально функция printBin, которая передает в основную функцию вывода голову нашего списка. Вывод происходит по возрастание, из глубины дерева.



Функция Counter нужна для подсчета числа вхождений заданного элемента в дерево. На вход она получает интересуемое нами число, после чего возвращает кол-во его вхождений.



Функция поиска происходит следующим способом: в функцию мы получаем число, после чего идет проверка на наличие его в бинарном дереве. Мы движемся по дереву только по нужным веткам, что улучшает время поиска. На выход мы получаем его порядок нахождения в дереве.

Сложность функции поиска: O(n).

**Листинг:**

#include<iostream>

#include <sstream>

#include <string>

#include <locale.h>

using namespace std;

class bst {

struct bintree {

int data;

bintree\* left;

bintree\* right;

bintree(int data, bintree\* pleft = nullptr, bintree\* pright = nullptr) {

this->data = data;

this->left = pleft;

this->right = pright;

}

};

bintree\* head;

public:

bst() {

head = nullptr;

}

bool isEmpty() { return(head == nullptr); }

int find(int data);

void push(int data);

void printBin(int l);

void printBinTree(bintree\* ptr, int l);

int Counter(int data);

};

int bst::find(int data)

{

int n = 0;

bintree\* ptr = this->head;

while (ptr != nullptr) {

if ((data) == ptr->data)

return n;

if ((data) > (ptr->data))

ptr = ptr->right;

else

ptr = ptr->left;

n++;

}

return n;

}

void bst::push(int data) {

bintree\* p = new bintree(data);

bintree\* parent = nullptr;

p->data = data;

p->left = nullptr;

p->right = nullptr;

if (isEmpty())

head = p;

else {

bintree\* ptr = this->head;

while (ptr != nullptr) {

parent = ptr;

if ((data) >= (ptr->data))

ptr = ptr->right;

else

ptr = ptr->left;

}

if ((data) < (parent->data))

parent->left = p;

else

parent->right = p;

}

}

void bst::printBin(int l) {

printBinTree(head, l);

}

void bst::printBinTree(bintree\* ptr, int l) {

if (ptr != nullptr) {

printBinTree(ptr->right, l+1);

for (int i = 0; i < l; i++)

{

cout << "\t";

}

cout << ptr->data<<endl;

printBinTree(ptr->left, l+1);

}

}

int bst::Counter(int data)

{

int n = 0;

bintree\* ptr = this->head;

while (ptr != nullptr) {

if ((data) == ptr->data)

n++;

if ((data) >= (ptr->data))

ptr = ptr->right;

else

ptr = ptr->left;

}

return n;

}

int main() {

setlocale(0, "");

srand(time(NULL));

bst b;

while (true)

{

cout<< "Чтобы закончить ввод, введите \*"<<endl;

string s;

cin >> s;

if (s == "\*")

break;

b.push(atoi(s.c\_str()));

}

b.printBin(0);

cout << endl << "Результат поиска: " << b.find(100);

cout << endl << "Кол-во совпадений: " << b.Counter(100);

cin.get();

return 0;

}

**Вывод:** Мы изучили структуру данных бинарных деревьев поиска, разработали класс для работы с ней и получили практический навык его использования.