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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «АиСД»

Тема:Деревья

Студент гр. 8304 _____ Порывай П.А

Преподаватель _____ Фирсов М.А

Санкт-Петербург

2019

Цель работы.

Ознакомится с такой структурой данных, как бинарное дерево, способами ее представления и реализации

Задание

(Вариант - 7д)

Для заданного бинарного дерева с произвольным типом элементов:

-получить лес естественно представленный этим деревом;

-вывести изображение леса и бинарного дерева;

-перечислить элементы леса в горизонтальном порядке(в ширину)

Описание алгоритма

Рекурсивный ввод дерева через файл или консоль и определение его верного задания , в случае удачи используя цикл, а также рекурсивно выводится лес заданный через бинарное дерево(передвигаемся по исходному дереву в котором правые сыновья становятся корнями деревьев, получившегося леса).Каждое дерево в лесу выводится в скобочном представлении. Итоговые данные выводятся в файл и на консоль

Основные функции и структуры

binTree ConsBT(const base& x, binTree& lst, binTree& rst);

Конструктор — создает дерево из левого и правого сына используя x, как информацию в корне. С помощью оператора new выделяется память под корень

void destroy(binTree&);

Рекурсивно удалает дерево через оператор delete

binTree enterBT(std::string str, int& i);

Функция для ввода бинарного дерева, i служит счетчиком считанных символов и используется далеее для проверки «правильности» дерева. Как только вызовы функций доходят до нижнего уровня вызывается конструктор

unInt sizeBT(binTree b);

Выдает длину дерева (Наибольший путь от корня до nullptr)

void printKLP(binTree b, std::ofstream& out);

Вывод дерева в клп порядке

void print_cons_write_file(std::string bin_tree, int i, std::ofstream& out);

Функция для вывода информации в файл и на консоль. Здесь используются функции ввода бинарного дерева, а также его проверки. В случае верного или ложного указания бд информация выводится в файл out или на консоль, i как счетчик количества введенных символов функцией ввода

void print_tree(binTree b, std::ofstream& out);

Функция печати дерева, которая работает рекурсивно, на каждом уровне дерева добавляется '('. в каждом ее вызове передается левый сын.

void print_forest(binTree b, std::ofstream& out);

Последовательно пока указатель на правого сына != nullptr вызывает void print_tree(binTree b, std::ofstream& out);

bool is_bt(std::string bin_tree,int i);

Функция проверяет введено ли бинарное дерево верно. Основная идея заключается в том, что если дерево введено верно то i равно bin_tree.length

Тестирование

Верный ввод

ad/ej/k//fl///bg/h//cim/n////

ab//c//

bc//a//

ab//c/d//

abc///d//

ab/d//c//

abd//e//cg//h//

ab///

ba///

Вывод

Бинарное дерево (повернутое):

a b c

```
      i
        m n
    g h
  d e f
       l
     j k
```

Размер (число узлов) дерева = 14

Бинарное дерево в КЛП-порядке:

adejkflbghcimn

Лес соответствующий данному бинарному дереву

(a(d)(e(j)(k))(f(l)))(b(g)(h))(c(i(m)(n)))

Бинарное дерево (повернутое):

```
 a c
   b
```

Размер (число узлов) дерева = 3

Бинарное дерево в КЛП-порядке:

abc

Лес соответствующий данному бинарному дереву

(a(b))(c)

Бинарное дерево (повернутое):

```
 b a
   c
```

Размер (число узлов) дерева = 3

Бинарное дерево в КЛП-порядке:

bca

Лес соответствующий данному бинарному дереву

(b(c))(a)

Бинарное дерево (повернутое):

```
 a c d
   b
```

Размер (число узлов) дерева = 4

Бинарное дерево в КЛП-порядке:

abcd

Лес соответствующий данному бинарному дереву

(a(b))(c)(d)

Бинарное дерево (повернутое):

```
 a d
   b
     c
```

Размер (число узлов) дерева = 4

Бинарное дерево в КЛП-порядке:

abcd

Лес соответствующий данному бинарному дереву

(a(b(c)))(d)

Бинарное дерево (повернутое):

```
 a c
   b d
```

Размер (число узлов) дерева = 4

Бинарное дерево в КЛП-порядке:

abdc

Лес соответствующий данному бинарному дереву

(a(b)(d))(c)

Бинарное дерево (повернутое):

```
 a c h
     g
   b e
```

d

Размер (число узлов) дерева = 7

Бинарное дерево в КЛП-порядке:

abdecgh

Лес соответствующий данному бинарному дереву

(a(b(d))(e))(c(g))(h)

Бинарное дерево (повернутое):

a

b

Размер (число узлов) дерева = 2

Бинарное дерево в КЛП-порядке:

ab

Лес соответствующий данному бинарному дереву

(a(b))

Бинарное дерево (повернутое):

b

a

Размер (число узлов) дерева = 2

Бинарное дерево в КЛП-порядке:

ba

Лес соответствующий данному бинарному дереву

(b(a))

Неверный ввод

/ld/ej/k//fl///bg/h//cim/n////

///nb//c//

/////////bc//a//

/ab//c/d//

/abc///d//

ab/

abd//e//cg//h////

Вывод

Бинарное дерево задано неверно

Бинарное дерево задано неверно

Бинарное дерево задано неверно

/ld/ej/k//fl///bg/h//cim/n////

Бинарное дерево задано неверно

///nb//c//

Бинарное дерево задано неверно

/////////bc//a//

Бинарное дерево задано неверно

/ab//c/d//

Бинарное дерево задано неверно

/abc///d//

Бинарное дерево задано неверно

ab/

Бинарное дерево задано неверно

abd//e//cg//h////

Приложение А. Исходный код

```
#include <iostream>
#include <cstdlib>
#include <fstream>
#include<string>

typedef char base;
typedef unsigned int unInt;

struct node {
      base info = -1;
      node* lt;
      node* rt;
      // constructor
      node() { lt = nullptr; rt = nullptr; }
};

typedef node* binTree; // "представитель" бинарного дерева

binTree Create(void);
bool isNull(binTree);
base RootBT(binTree); // для непустого бин.дерева
binTree Left(binTree);// для непустого бин.дерева
binTree Right(binTree);// для непустого бин.дерева
binTree ConsBT(const base& x, binTree& lst, binTree& rst);
void destroy(binTree&);

binTree enterBT(std::string str, int& i);
void outBT(binTree b);
void displayBT(binTree b, int n, std::ofstream& out);
unInt sizeBT(binTree b);
void printKLP(binTree b, std::ofstream& out);

void print_cons_write_file(std::string bin_tree, int i, std::ofstream& out);
void print_tree(binTree b, std::ofstream& out);
void print_forest(binTree b, std::ofstream& out);

bool is_bt(std::string bin_tree,int i);

//void is_btree(binTree b, bool &flag){
//    if( b != nullptr)
```

```
//          return true;
//    else
//          return false;
//}
binTree Create()
{
      return nullptr;
}
//-------------------------------------
bool isNull(binTree b)
{
      return (b == nullptr);
}
//-------------------------------------
base RootBT(binTree b)              // для непустого бин.дерева
{
      if (b == nullptr) {

            std::cerr << "Error: RootBT(null) \n"; exit(1);
      }
      else
            return b->info;
}
//-------------------------------------
binTree Left(binTree b)       // для непустого бин.дерева
{
      if (b == nullptr) {

            std::cerr << "Error: Left(null) \n"; exit(1);
      }
      else
            return b->lt;
}
//-------------------------------------
binTree Right(binTree b)            // для непустого бин.дерева
{
      if (b == nullptr) {
            std::cerr << "Error: Right(null) \n"; exit(1);
      }
      else
             return b->rt;
}
//-------------------------------------
binTree ConsBT(const base& x, binTree& lst, binTree& rst)
```

```
{
	binTree p;
	p = new node;
	if (p != nullptr) {
		p->info = x;
		p->lt = lst;
		p->rt = rst;
		return p;
	}
	else {
		std::cerr << "Memory not enough\n"; exit(1);
	}
}
//-------------------------------------
void destroy(binTree& b)
{
	if (b != nullptr) {

		destroy(b->lt);
		destroy(b->rt);
		delete b;
		b = nullptr;

	}
}
binTree enterBT(std::string str,int &i)
{
	char ch;//сюда добавить условие для неверного задания бинарного дерева
	binTree p, q;
	ch = str[i];
	i++;
	if(i <= str.length()){

		if (ch == '/')
			return nullptr;
		else {

			p = enterBT(str,i);
			q = enterBT(str, i );
			return ConsBT(ch, p, q);
		}
	}
	else{
		return nullptr;//если не хватает '/'
```

```
	}

}
//---------------------------------------
void outBT(binTree b)
{
	if (b != nullptr) {

		std::cout << RootBT(b);
		outBT(Left(b));
		outBT(Right(b));

	}
	else
		std::cout << '/';
}

//---------------------------------------
void displayBT(binTree b, int n,std::ofstream& out)
{	// n - уровень узла
	if (b != nullptr) {
		std::cout << ' ' << RootBT(b);
		out << ' ' << RootBT(b);

		if (!isNull(Right(b))) {

			displayBT(Right(b), n + 1, out);
		}
		else {
			std::cout << "\n";
			out << "\n";
		}// вниз
		if (!isNull(Left(b))) {
			for (int i = 1; i <= n; i++) {
				std::cout << "  "; // вправо
				out << "  ";
			}
			displayBT(Left(b), n + 1 , out);
		}
	}
	else {};
}
//---------------------------------------
```

```
//---------------------------------------
unInt sizeBT(binTree b)
{
	if (isNull(b))
		return 0;
	else
		return sizeBT(Left(b)) + sizeBT(Right(b)) + 1;
}
//---------------------------------------
void printKLP(binTree b, std::ofstream & out)
{
	if (!isNull(b)) {

		std::cout << RootBT(b);
		out << RootBT(b);
		printKLP(Left(b) , out);
		printKLP(Right(b) , out);

	}
}


void print_tree(binTree b, std::ofstream &out) {

	binTree p;

	if (b != nullptr) {
		std::cout << "( ";
		std::cout << b->info;
		out << "( ";
		out << b->info;
		p = b->lt;

		while (p != nullptr) {

			print_tree(p , out);
			p = p->rt;
		}

		std::cout << " )";
		out << " )";
	}
}
```

```
void print_forest(binTree b, std::ofstream &out) {

	binTree p = b;
	while (p != nullptr) {

		print_tree(p , out);
		std::cout << " ";
		p = p->rt;
	}
}

void print_cons_write_file(std::string bin_tree, int i,std::ofstream& out) {

		binTree b = enterBT(bin_tree, i);

		if(is_bt(bin_tree,i) == true){

			std::cout << "Бинарное дерево (повернутое): " << "\n";
			out<< "Бинарное дерево (повернутое): " << "\n";
			displayBT(b, 1, out);

			std::cout << "Размер (число узлов) дерева = " << sizeBT(b) <<
"\n";
			out<< "Размер (число узлов) дерева = " << sizeBT(b) << "\n";

			std::cout << "Бинарное дерево в КЛП-порядке: " << "\n";
			out<< "Бинарное дерево в КЛП-порядке: " << "\n";
			printKLP(b, out);
			std::cout << "\n";
			out<< "\n";
			std::cout << "Лес соответствующий данному бинарному дереву\n";
			out<< "Лес соответствующий данному бинарному дереву\n";
			print_forest(b, out);

			destroy(b);

			std::cout << "\n";
			std::cout << "\n";
			out << "\n";
			out << "\n";
		}
		else{
```

```
                        std::cout<<"Бинарное дерево задано неверно\n"<<bin_tree<<"\n";
                        out<<"Бинарное дерево задано неверно\n"<<bin_tree<<"\n";
                }
        }

        bool is_bt(std::string bin_tree,int i){

                if(bin_tree[0]  ==  '/'  ||  bin_tree  ==  "  "  ||  bin_tree  ==  ""  ||
bin_tree.length() != i )
                        return false;
                else
                        return true;

        }

        int main(int argc, char *argv[]) {

                setlocale(LC_ALL, "Russian");
                std::cout  <<  "Ввод  бинарного  дерева  в  клп  представлении  (/  =  nullptr)  из
файла или консоли(f,c)?\n";
                std::string arg, bin_tree;
                int i = 0;
                std::cin >> arg;
                std::ofstream out("wrong_out.txt");
                if (arg == "f") {
                        std::ifstream infile(argv[1]);
                        while (getline(infile, bin_tree)) {
                                        print_cons_write_file(bin_tree, i, out);
                        }
                }
                else if (arg == "c") {
                        while (getline(std::cin, bin_tree)) {
                                        print_cons_write_file(bin_tree, i, out);

                        }
                }
                else
                        std::cout << "Аргумент задан неверно\n";
                return 0;
        }
```