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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «АиСД»

Тема:Бинарное дерево поиска

Студент гр. 8304 _____ Порывай П.А

Преподаватель _____ Фирсов М.А

Санкт-Петербург

2019

Цель работы.

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

Задание

(Вариант - 9)

Для заданного бинарного дерева поиска в котором все элементы различны:
-Записать в файл элементы в порядке их возрастания, а также вывести это на экран

Описание алгоритма

Рекурсивный ввод дерева через файл или консоль и определение его верного задания (для этого используется функции inp_bts, insert_key) Перебираются введенные элементы и в зависимости от значения представляются в памяти как левое или правое поддерево(лист)Ввод может быть осуществлен как из файла так и из консоли, а для этого применяется перегруженная функция inp_bts.

Основные функции и структуры

void destroy(binTree&);

Рекурсивно удалает дерево через оператор delete

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 inp_bts(binTree& bt,std::ifstream& inp)

Считывает последовательно элементы из файла, вызывает insert_key

void inp_bts(binTree& bt, std::istream& inp)

Считывает последовательно элементы из консоли, вызывает insert_key

void insert_key(binTree &bt, int key)

В зависимости от значения ключа переходит в правое или левое поддерево, в случае нахождения пустого указателя создает узел в info записывает key

bool check(std::string str)

Проверяет введенную строку на «правильность»

Тестирование

Верный ввод

12 11 13 14 15 7 6 3 88 1221 21

123 12 2144 425 23131 21312 12

423 124 1234 52423 5232 1223 542

213 425 4254 32442 3545 23442 54345

24552 54665 235534 34646 57573

2553 2352 354663 42626 32552 43463 23553 6226 47745

12441 36446856 56885 6363 43634 34663446 121244 1235533

142124 6475745 745475 54774 4546 32112 124412 21442 142142

124 5435 8677886 978957 45664 2344243 124234

Вывод

3 6 7 11 12 13 14 15 21 88 1221

12 123 425 2144 21312 23131

124 423 542 1223 1234 5232 52423

213 425 3545 4254 23442 32442 54345

24552 34646 54665 57573 235534

2352 2553 6226 23553 32552 42626 43463 47745 354663

6363 12441 43634 56885 121244 1235533 34663446 36446856

4546 21442 32112 54774 124412 142124 142142 745475 6475745

124 5435 45664 124234 978957 2344243 8677886

Неверный ввод

dassd adsasd asccsa

asccaslk saccsa

-1 d 12

21 124 123231 1232 21321 -1 -4

qwert

(пробел)

(пустая строка)

11212 133113 saddas

Вывод

dassd adsasd asccsa

Неверно введено дерево

asccaslk saccsa

Неверно введено дерево

-1 d 12

Неверно введено дерево

21 124 123231 1232 21321 -1 -4

Неверно введено дерево

qwert

Неверно введено дерево

Неверно введено дерево

Неверно введено дерево

11212 133113 saddas

Неверно введено дерево

Приложение А. Исходный код

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

typedef int base;
```

```
typedef unsigned int unInt;

struct node {
	base info;
	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);
bool check(std::istream& inp);

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;
```

```
	}
}

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

	if (!isNull(b)) {


		print(Left(b), out);

		std::cout << RootBT(b) << " ";
		out << RootBT(b);
		out << " ";
		print(Right(b), out);

	}
}



void insert_key(binTree &bt, int key) {
//	std::cout<<"insert key";
	if (bt == nullptr) {

		bt = new node;
```

```
            (bt)->info = key;
            bt->lt = nullptr;
            bt->rt = nullptr;

      }
      else if ((bt)->info > key) {

            insert_key( bt->lt, key);

      }
      else if ((bt)->info < key) {

            insert_key(bt->rt, key);

      }
}
void inp_bts( binTree& bt,std::ifstream& inp) {
      int key;

      while (inp.peek() != '\n' && !inp.eof()) {
            inp >> key;
            insert_key(bt, key);
      }

}

void inp_bts(binTree& bt, std::istream& inp) {
      int key;

      while (inp.peek() != '\n' && !inp.eof()) {
            inp >> key;
            insert_key(bt, key);
      }

}

bool check(std::string str){



      int i = 0;
      if(str == "" || str ==" ")
            return false;
      while( i < str.length() ){
```

```
                if( str[i] != '1' && str[i] !='0' && str[i] !='2' && str[i] !='3' &&
str[i] !='4' && str[i] !='5' && str[i] !='6' && str[i] !='7' && str[i] !='8' && str[i] !
='9'  )
                      if(str[i] != ' ')
                            return false;

                i++;
          }


          return true;

     }




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

          setlocale(LC_ALL, "Russian");
          std::cout << "Ввод бинарного дерева из файла или консоли(f,c)?\n";
          std::string arg, bin_tree;
          int i = 0;
          char ch = '0';
          getline(std::cin ,  arg);
          std::string str;

          std::ofstream out("wrong_out.txt");

          if (arg == "f") {

                std::ifstream inp(argv[1]);


                while (!inp.eof()) {
                      getline(inp,str);

                      if( check(str) == true){

                                  inp.putback('\n');

                                  for(i = str.length() -1; i >= 0; i--)
                                        inp.putback(str[i]);
```

```
                binTree bt = nullptr;
                inp_bts(bt, inp);
                print(bt, out);
                destroy(bt);
                getline(inp , str);
                std::cout << "\n";
                out << "\n";
          }
          else{
                if(!inp.eof()){

                      std::cout<<str<<"\n";
                      std::cout<<"Неверно введено дерево\n\n";
                      out<<str<<"\n";
                            out<<"Неверно введено дерево\n\n";}
          }

     }



}
else if (arg == "c") {
     while (!std::cin.eof()) {
          getline(std::cin , str);

          if(check(str) == true){

                std::cin.putback('\n');
                for(i = str.length() -1; i >= 0; i--)
                      std::cin.putback(str[i]);

                binTree bt = nullptr;
                inp_bts(bt, std::cin);
                print(bt, out);
                destroy(bt);
                std::cin.get(ch);
                std::cout << "\n";
                out << "\n";
          }
          else{
                if(!std::cin.eof()){
                      std::cout<<"Неверно введено дерево\n";
```

```
                        out << "Неверно введено дерево\n";
                    }
                }
            }


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

    return 0;
}
```