Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

**Кафедра автоматизированных систем управления (АСУ)**

**БИНАРНЫЕ ДЕРЕВЬЯ**

Отчет по лабораторной работе № 1 по дисциплине «Структуры и алгоритмы обработки данных в ЭВМ»

Выполнил: студент

специальности 09.03.01, группа   
з-436У-а, поток 75

Красноперов В.С.

«30» сентября 2017 г.

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

**г. Калининград 2017**

Содержание

[Тема работы 3](#_Toc495770748)

[Цель работы 3](#_Toc495770749)

[Индивидуальное задание 3](#_Toc495770750)

[Алгоритм решения задачи 3](#_Toc495770751)

[Результаты работы программы 3](#_Toc495770752)

[Выводы 4](#_Toc495770753)

[Приложение А. Листинг программы 4](#_Toc495770754)

# Тема работы

Представление в памяти ЭВМ структуры данных “Бинарное дерево”

# Цель работы

Получить практические навыки представления в памяти ЭВМ структуры данных «бинарное дерево», реализовать на языке программирования C/C++ алгоритмы работы с деревьями.

# Индивидуальное задание

Даны две последовательности чисел. Построить бинарное дерево, содержащее числа первой последовательности. Для каждого числа второй последовательности узнать, входит ли оно в дерево. После выполнения программы очистить память, занятую древовидной структурой.

# Алгоритм решения задачи

1. Осуществить ввод чисел первой последовательности
2. На основе введенных чисел построить бинарное дерево
3. Ввести число второй последовательности и по мере ввода чисел:
   1. Методом обхода в глубину обойти дерево с целью поиска введенного числа
   2. Вывести сообщение с результатом поиска
4. Очистить память, занимаемую деревом

# Результаты работы программы

Число элементов в первой последовательности: 10

Первая последовательность: 3 10 25 0 -6 123 8 67 9 -20

Дерево выглядит так:

   123

    67

  25

 10

   9

  8

3

 0

  -6

   -20

Число элементов во второй последовательности: 4

Вторая последовательность: 5

5 НЕ содержится в дереве

7

7 НЕ содержится в дереве

9

9 содержится в дереве

0

0 содержится в дереве

# Выводы

Выполняя данную лабораторную работу я получил практические навыки работы с бинарным деревом с памяти ЭВМ

# 

# Приложение А. Листинг программы

#include <stdlib.h>

#include <string.h>

#include <iostream>

using namespace std;

typedef struct bTree {

int data; //поле данных

struct bTree\* left; //указатель на левого потомка

struct bTree\* right; //указатель на правого потомка

} bTree;

//создание бинарного дерева

void create\_tree(bTree\*\* p, int x)

{

if (!(\*p)) //если указатель на корень дерева не равен NULL

{

bTree\* pnew = (bTree\*)

malloc(sizeof(bTree)); // выделяем память

pnew->data = x; //заносим значение

pnew->left = pnew->right = NULL;

\*p = pnew;

}

else {

if ((\*p)->data > x)

create\_tree(&((\*p)->left), x);

else

create\_tree(&((\*p)->right), x);

}

}

void print\_tree(bTree\*\* p, int l)

{

int i;

if (\*p != NULL) {

print\_tree(&((\*p)->right), l + 1);

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

printf(" ");

printf("%d\n", (\*p)->data);

print\_tree(&((\*p)->left), l + 1);

}

}

//прямой обход бинарного дерева

bool search(bTree\* node, int x)

{

if (node != NULL) {

if(node->data == x || search(node->left, x) || search(node->right, x)) {

return true;

}

}

return false;

}

//освобождение памяти, выделенной под бинарное дерево

bTree\* drop\_tree(bTree\* Node)

{

if (Node != NULL) {

drop\_tree(Node->left);

drop\_tree(Node->right);

free(Node);

}

return NULL;

}

int main()

{

int i, n, temp, x;

bTree\* root;

root = NULL;

printf("Число элементов в первой последовательности: ");

scanf("%d", &n);

printf("Первая последовательность: ");

for (i = 0; i < n; i++) {

scanf("%d", &temp);

create\_tree(&root, temp);

}

printf("Дерево выглядит так: \n");

print\_tree(&root, 0);

printf("Число элементов во второй последовательности: ");

scanf("%d", &n);

printf("Вторая последовательность: ");

for (i = 0; i < n; i++) {

scanf("%d", &x);

if (search(root, x)) {

printf("\n%d содержится в дереве\n", x);

} else {

printf("\n%d НЕ содержится в дереве\n", x);

}

}

drop\_tree(root);

return 0;

}