МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

КАФЕДРА ИИТ

ОТЧЁТ

по лабораторной работе №14

**«Динамические структуры: списки и деревья»**

Выполнил:

студент 1 курса

группы ПО-9

Мисиюк Алексей Сергеевич

Проверила:

Хацкевич М. В.

Брест 2022

**Цель работы:** Приобретение навыков работы с динамической памятью и указателями на С/C++. Изучение принципов работы с динамическими структурами данных: списками и деревьями.

**Порядок выполнения работы**

**Вариант 4**

*Текст задания*

Написать программу, которая вводит с клавиатуры список целых чисел, строит копию списка, затем вводит с клавиатуры дерево поиска и считает количество положительных и отрицательных элементов дерева.

**Блок-схема алгоритма**

**Код программы:**

#include <iostream>

using namespace std;

struct list {

int value;

struct list\* next;

};

struct tree

{

int value;

struct tree\* left;

struct tree\* right;

struct tree\* parent;

};

void readList(struct list\* head, int N);

void pushToList(struct list\*\* head, int data);

struct list\* copyList(const struct list\* head);

void freeList(struct list\* head);

void readTree(struct tree\* root, int N);

struct tree\* createTree(struct tree\* root, int value);

struct tree\* addToTree(struct tree\* root, int value);

int countNodes(struct tree\* root);

void freeTree(struct tree\* node);

int main() {

cout << "Написать программу, которая вводит с клавиатуры список целых чисел, строит копию списка, затем вводит с клавиатуры дерево поиска и считает количество положительных и отрицательных элементов дерева\n";

cout << "Enter amount of elements:\n";

int n;

cin >> n;

struct list\* list = NULL;

readList(list, n);

struct list\* newList = copyList(list);

freeList(list);

cout << "Enter amount of elements:\n";

cin >> n;

struct tree\* tree = NULL;

tree = createTree(tree, 0);

readTree(tree, n);

cout << "Positive - " << countNodes(tree->right) << endl;

cout << "Negative - " << countNodes(tree->left) << endl;

freeTree(tree);

return 0;

}

void readList(struct list\* head, int N) {

cout << "Enter elements:\n";

for (int i = 0; i < N; i++) {

int value;

cin >> value;

pushToList(&head, value);

}

}

void pushToList(struct list\*\* head, int data) {

struct list\* tmp = (struct list\*)malloc(sizeof(struct list));

tmp->value = data;

tmp->next = (\*head);

(\*head) = tmp;

}

struct list\* copyList(const struct list\* head) {

if (head == NULL) return NULL;

struct list\* node = (struct list\*)malloc(sizeof(struct list));

node->value = head->value;

node->next = head->next;

return node;

}

void freeList(struct list\* head) {

struct list\* head\_old;

while (head) {

head\_old = head;

head = head->next;

free(head\_old);

}

};

void readTree(struct tree\* root, int N) {

cout << "Enter elements:\n";

for (int i = 0; i < N; i++) {

int value;

cin >> value;

addToTree(root, value);

}

}

struct tree\* createTree(struct tree\* root, int value) {

struct tree\* tmp = (struct tree\*)malloc(sizeof(struct tree));

tmp->value = value;

tmp->parent = NULL;

tmp->left = tmp->right = NULL;

root = tmp;

return root;

}

struct tree\* addToTree(struct tree\* root, int value) {

struct tree\* root2 = root, \* root3 = NULL;

struct tree\* tmp = (struct tree\*)malloc(sizeof(struct tree));

tmp->value = value;

while (root2 != NULL)

{

root3 = root2;

if (value < root2->value)

root2 = root2->left;

else

root2 = root2->right;

}

tmp->parent = root3;

tmp->left = NULL;

tmp->right = NULL;

if (value < root3->value) root3->left = tmp;

else root3->right = tmp;

return root;

}

int countNodes(struct tree\* root) {

int tmp = 0;

if (root == NULL)

return tmp;

if (root->value) tmp++;

tmp += countNodes(root->left);

tmp += countNodes(root->right);

return tmp;

}

void freeTree(struct tree\* node) {

if (node) {

freeTree(node->right);

freeTree(node->left);

free(node);

}

};

**Пример работы программы**

Написать программу, которая вводит с клавиатуры список целых чисел, строит копию списка, затем вводит с клавиатуры дерево поиска и считает количество положительных и отрицательных элементов дерева  
Enter amount of elements:

5

Enter elements to list:

1 2 4 6 7

Enter amount of elements:

5

Enter elements into tree:

-3 -2 1 2 3

Positive - 3

Negative – 2

**Вывод:** были изучены принципы работы с динамической памятью и указателями на С/C++. Изучены принципы работы с динамическими структурами данных: списками и деревьями.