МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**ИНТЕЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ**

ОТЧЁТ

По лабораторной работе №6

Выполнил:

Студент группы ИИ-22

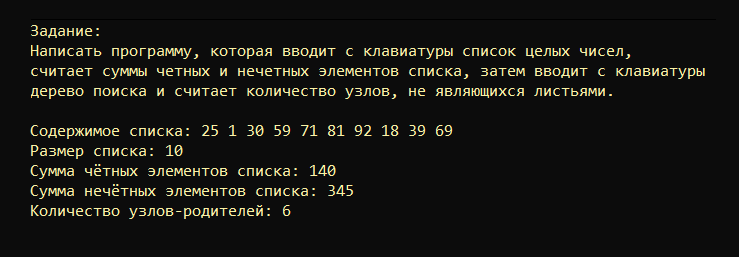
Копанчук Евгений Романович

Проверил:

Войцехович О. Ю.

Брест 2022

**Вариант 6**

****

**Код программы:**

***Файл BinaryTree.h***

#include "List.h"

#include "BinaryTree.h"

#include <ctime>

#include <Windows.h>

bool parity(int value) {

if (value % 2 == 0) return true;

return false;

}

bool odd(int value) {

if (value % 2 == 1) return true;

return false;

}

int main() {

srand(time(NULL));

setlocale(LC\_ALL, "Russian");

std::cout << "Задание:" << std::endl

<< "Написать программу, которая вводит с клавиатуры список целых чисел," << std::endl

<< "считает суммы четных и нечетных элементов списка, затем вводит с клавиатуры" << std::endl

<< "дерево поиска и считает количество узлов, не являющихся листьями." << std::endl << std::endl;

List list;

for (int i = 0; i < 10; i++) list.Add(rand() % 100);

std::cout << "Содержимое списка: ";

list.Show(0);

std::cout << "Размер списка: " << list.Size() << std::endl;

int\* n = list.CreateArray(n, list.Size()); // Создание массива элементов списка

std::cout << "Сумма чётных элементов списка: " << list.Sum(parity) << std::endl;

std::cout << "Сумма нечётных элементов списка: " << list.Sum(odd) << std::endl;

NodeTree\* test\_tree = NULL;

for (int i = 0; i < list.Size(); i++) test\_tree = insert(n[i], test\_tree);;

delete[] n;

std::cout << "Количество узлов-родителей: ";

std::cout << devour(test\_tree) << std::endl;

return 0;

}

***Файл List.h***

#pragma once

#include <iostream>

class List {

struct Node

{

Node() {}

int value;

Node\* Next;

Node\* Prev;

~Node() {}

};

Node\* Head;

Node\* Tail;

public:

List() {

Head = NULL;

Tail = NULL;

}

~List() {

while (Head)

{

Tail = Head->Next;

delete Head;

Head = Tail;

}

}

void Add(int value)

{

Node\* Temp = new Node;

Temp->Next = NULL;

Temp->value = value;

if (Head != NULL)

{

Temp->Prev = Tail;

Tail->Next = Temp;

Tail = Temp;

}

else

{

Temp->Prev = NULL;

Head = Tail = Temp;

}

}

void Show(bool reverse) {

if (reverse == false) for (Node\* Temp = Head; Temp != NULL; Temp = Temp->Next) std::cout << Temp->value << " ";

else for (Node\* Temp = Tail; Temp != NULL; Temp = Temp->Prev) std::cout << Temp->value << " ";

std::cout << std::endl;

}

int Size() {

int Count = 0;

for (Node\* Temp = Head; Temp != NULL; Temp = Temp->Next) Count++;

return Count;

}

int\* CreateArray(int\* (&n), int size) {

n = new int[size];

int i = 0;

for (Node\* Temp = Head; Temp != NULL; Temp = Temp->Next) {

n[i] = Temp->value;

i++;

}

return n;

}

int Sum(bool condition(int)) {

int Count = 0;

for (Node\* Temp = Head; Temp != NULL; Temp = Temp->Next) if (condition(Temp->value)) Count += Temp->value;

return Count;

}

};

***Файл Main.cpp***

#include "List.h"

#include "BinaryTree.h"

#include <ctime>

#include <Windows.h>

bool parity(int value) {

if (value % 2 == 0) return true;

return false;

}

bool odd(int value) {

if (value % 2 == 1) return true;

return false;

}

int main() {

srand(time(NULL));

setlocale(LC\_ALL, "Russian");

std::cout << "Задание:" << std::endl

<< "Написать программу, которая вводит с клавиатуры список целых чисел," << std::endl

<< "считает суммы четных и нечетных элементов списка, затем вводит с клавиатуры" << std::endl

<< "дерево поиска и считает количество узлов, не являющихся листьями." << std::endl << std::endl;

List list;

for (int i = 0; i < 10; i++) list.Add(rand() % 100);

std::cout << "Содержимое списка: ";

list.Show(0);

std::cout << "Размер списка: " << list.Size() << std::endl;

int\* n = list.CreateArray(n, list.Size()); // Создание массива элементов списка

std::cout << "Сумма чётных элементов списка: " << list.Sum(parity) << std::endl;

std::cout << "Сумма нечётных элементов списка: " << list.Sum(odd) << std::endl;

NodeTree\* test\_tree = NULL;

for (int i = 0; i < list.Size(); i++) test\_tree = insert(n[i], test\_tree);;

delete[] n;

std::cout << "Количество узлов-родителей: ";

std::cout << devour(test\_tree) << std::endl;

return 0;

}