Пермский филиал федерального государственного автономного образовательного учреждения высшего образования

«Национальный исследовательский университет

«Высшая школа экономики» Пермь»

Факультет экономики, менеджмента и бизнес-информатики.

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7

студента образовательной программы «Программная инженерия»

по направлению подготовки *09.03.04 Программная инженерия*

Чепокова Елизара Сергеевича

Руководитель:

Кандидат технических наук

О.Л. Викентьева

Пермь, 2018 год

# **1.Постановка задачи:**

1. Сформировать однонаправленный список, тип информационного поля указан в варианте.
2. Распечатать полученный список.
3. Выполнить обработку списка в соответствии с заданием.
4. Распечатать полученный список.
5. Удалить список из памяти.
6. Сформировать двунаправленный список, тип информационного поля указан в варианте.
7. Распечатать полученный список.
8. Выполнить обработку списка в соответствии с заданием.
9. Распечатать полученный список.
10. Удалить список из памяти.
11. Сформировать идеально сбалансированное бинарное дерево, тип информационного поля указан в варианте.
12. Распечатать полученное дерево.
13. Выполнить обработку дерева в соответствии с заданием, вывести полученный результат.
14. Преобразовать идеально сбалансированное дерево в дерево поиска.
15. Распечатать полученное дерево.

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Однонаправленный | Двунаправленный | Бинарное дерево |
| 23 | Тип информационного поля double.  Добавить в список после каждого элемента с отрицательным информационным полем элемент с информационным полем равным 0. | Тип информационного поля int.  Удалить из списка первый элемент с четным информационным полем. | Тип информационного поля string.  Найти количество листьев в дереве. |

# **2.Анализ задачи:**

Основные функции

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Функция | Входные данные | Классы входных данных | Выходные данные | Классы выходных данных |
| 1 | Формирование однонаправленного списка | Количество элементов списка (size) | size – целое число > 0 | Однонаправленный список | Список (Point List1) |
| size – Символ, Целое число ≤ 0, Вещественное число | Сообщение об ошибке | Неверный ввод / Список пуст |
| 1.1 | Формирование однонаправленного списка с помощью ДСЧ | Количество элементов списка (size)  Нижний и верхний порог для ДСЧ (LowLimit и HighLimit) | size – целое число > 0  LowLimit и HighLimit – любое целое число | Однонаправленный список | Список (Point List1) |
| size – Символ, Целое число ≤ 0, Вещественное число  LowLimit и HighLimit – вещественное число или символ | Сообщение об ошибке | Неверный ввод / Список пуст |
| 1.2 | Формирование однонаправленного списка с клавиатуры | Количество элементов списка (size)  Число для каждого элемента | size – целое число > 0  Вещественное число | Однонаправленный список | Список (Point List1) |
| size – Символ, Целое число ≤ 0, Вещественное число | Сообщение об ошибке | Неверный ввод / Список пуст |
| 2 | Формирование двунаправленного списка | Количество элементов списка (size) | size – целое число > 0 | Двунаправленный список | Список (BiList List2) |
| size – Символ, Целое число ≤ 0, Вещественное число | Сообщение об ошибке | Неверный ввод / Список пуст |
| 2.1 | Формирование двунаправленного списка с помощью ДСЧ | Количество элементов списка (size)  Нижний и верхний порог для ДСЧ (LowLimit и HighLimit) | size – целое число > 0  LowLimit и HighLimit – любое целое число | Двунаправленный список | Список (BiList List2) |
| size – Символ, Целое число ≤ 0, Вещественное число  LowLimit и HighLimit – вещественное число или символ | Сообщение об ошибке | Неверный ввод / Список пуст |
| 2.2 | Формирование двунаправленного списка с клавиатуры | Количество элементов списка (size)  Число для каждого элемента | size – целое число > 0  Целые числа | Двунаправленный список | Список (BiList List2) |
| size – Символ, Целое число ≤ 0, Вещественное число  Вещественные числа | Сообщение об ошибке | Неверный ввод / Список пуст |
| 3 | Формирование бинарного дерева | Количество элементов списка (size) | size – целое число > 0 | Бинарное дерево | Дерево (Tree BiTree) |
| size – Символ, Целое число ≤ 0, Вещественное число | Сообщение об ошибке | Неверный ввод / Дерево пустое |
| 3.1 | Формирование бинарного дерева с помощью ДСЧ | Количество элементов списка (size) | size – целое число > 0 | Бинарное дерево | Дерево (Tree BiTree) |
| size – Символ, Целое число ≤ 0, Вещественное число | Сообщение об ошибке | Неверный ввод / Дерево пустое |
| 3.2 | Формирование бинарного дерева с клавиатуры | Количество элементов списка (size)  Каждый элемент | size – целое число > 0  Любые символы | Бинарное дерево | Дерево (Tree BiTree) |
| size – Символ, Целое число ≤ 0, Вещественное число | Сообщение об ошибке | Неверный ввод / Дерево пустое |

Дополнительные функции

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Функция | Входные данные | Классы входных данных | Выходные данные | Классы выходных данных |
| Ввод целого числа с клавиатуры | Proverkavvoda | Целое число | Число | Число |
| Символ  Целое число <=0  Вещественное число | Сообщение об ошибке | Неверный ввод |
| Печать диалога для ввода команд пользователя | Proverkavvoda | Целое число | Диалоговая таблица | Меню |
| Символ  Целое число <=0  Вещественное число | Сообщение об ошибке | Нужно выбрать из списка! |

# **3.Проектирование функций:**

Основные функции

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Функция** | **Входные данные** | **Выходные данные** |
| **Однонаправленный список** | | | |
| 1 | public Point | int data  Point next | Элемент двунаправленного списка |
| 2 | static Point FormNewList2 | int info  int size  BiList beg | Элемент двунаправленного списка с данными из info и указанной ссылкой на предыдущий элемент |
| 3 | static Point WorkList1 | Point List1 | Двунаправленный список |
| **Двунаправленный список** | | | |
| 1 | public BiList | int data  BiList next  BiList last | Элемент двунаправленного списка |
| 2 | static BiList FormNewList2 | int info  int size  BiList beg | Элемент двунаправленного списка с данными из info и указанной ссылкой на предыдущий элемент |
| 3 | static BiList MakePoint2 | int d | Двунаправленный список |
| 4 | static void ShowList2 | List2 | Вывод списка |
| 5 | static BiList WorkList2 | List2 | Список с добавленными нулями |
| **Бинарное дерево** | | | |
| 1 | public Tree | int data  Tree next  Tree last | Элемент дерева |
| 2 | static Tree CreateTree | int info | Вывод меню для выбора ввода |
| 3 | static Tree IdealTreeRnd | tree p  int count  Tree root | Дерево созданное с помощью ДСЧ |
| 4 | static Tree IdelaTree | Tree root, int[] arr | Дерево созданное с помощью ручного ввода |
| 5 | static Tree FormIdealTree | tree p  int size | Формирование идеальносбалансированного дерева |
| 6 | static int BeginSearchIdealTree | tree p  int count | Выод количества листьев в дереве |
| 7 | static void SearchIdealTree | tree p  ref int count | Поиск листьев в дереве |
| 8 | static void CompileTree | ref string[] arr  ref int pos  Tree p | Вывод дерева |
| 9 | static Tree CompileSearchTree | ref string[] arr  ref int pos | Ищет дерево поиска |
| 10 | static void CompileSearchTreeStep | Tree root  String data | Пошагово создает дерево поиска |
| 11 | static void ShowTree | Tree p | Вывод дерева |

Вспомогательные функции

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функции | Входные данные | Выходные данные |
| 1 | ProverkaVvoda() – проверка ввода меню | int menu  int how  int MainMenu | Число |
| Сообщение об ошбке |
| 2 | RemoveSpaces() – удаление пробелов в строке для дерева | string symbol | Строка без пробелов |
| 3 | IntRND() – проверка количества элементов списка | int count | Int число элементов списка |
| Сообщение об ошибке |
| 4 | Int() – проверка для двунаправленного списка | int n | Целое число |
| Сообщение об ошибке |
| 5 | Double() – проверка для однонаправленного списка | double n | Вещественное число |
| Сообщение об ошибке |
| 6 | Class Text\_Dialog – класс с менюшками | int menu  int how  int MainMenu | Меню |

# **5.Определение функции Main:**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функции | Входные данные | Выходные данные |
| 1 | static void Point | int MainMenu | Вывод меню работы с однонаправленным списком |
| 2 | static void PointTwo | int MainMenu | Вывод меню работы с двунаправленным списком |
| 3 | static void Tree | int MainMenu | Вывод меню работы с деревом |

# **6.Текст программы:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

namespace LAB\_7\_CH

{

//---Все менюшки----------------------------------------------

class Text\_Dialog

{

public static void MainError()

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Нужно выбрать из списка!");

Console.ResetColor();

}

public static void ErrorMenu()

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Ошибка, некорректный ввод");

Console.ResetColor();

}

public static void FormList()

{

Console.WriteLine("\n------------------------------");

Console.WriteLine("| Как сформировать список: |");

Console.WriteLine("| 1) Рандомно |");

Console.WriteLine("| 2) Вручную |");

Console.WriteLine("| 9) Вернуться в начало |");

Console.WriteLine("| 0) Выход из консоли |");

Console.WriteLine("------------------------------");

Console.Write("Действие: ");

}

}

//------------------------------------------------------------

class Search\_for\_Bugs

{

public static int ProverkaVvoda()

{

int number;

bool res;

do

{

res = int.TryParse(Console.ReadLine(), out number);

if (res == false)

{

Text\_Dialog.ErrorMenu();

}

} while (!res);

return number;

}

}

//---Выполнение задач--------------------------------------------------------------------

class Point

{

public double data; //информационное поле

public Point next; //адресное поле

public Point(double element) //конструктор с параметрами

{

data = element;

next = null;

}

public override string ToString()

{

return data.ToString() + " ";

}

}

class BiList

{

public int data;

public BiList next, //адрес следующего элемента

last; //адрес предыдущего элемента

public BiList(int d)

{

data = d;

next = null;

last = null;

}

public override string ToString()

{

return data.ToString() + " ";

}

}

class Tree

{

public string data;

public Tree left, //адрес левого поддерева

right; //адрес правого поддерева

public Tree(string d)

{

data = d;

left = null;

right = null;

}

public override string ToString()

{

return data.ToString() + " ";

}

}

//---------------------------------------------------------------------------------------

//---Вход в программу--------------------------------------------------------------------

class Program

{

const int MinNumber = -100;

const int MaxNumber = 100;

const int NumberOfElemenets = 5;

static Random rnd = new Random();

static Point MakePoint(double d)

{

Point p = new Point(d);

return p;

}

static BiList MakePoint2(int d)

{

BiList p = new BiList(d);

return p;

}

static Tree MakePoint3(string d)

{

Tree p = new Tree(d);

return p;

}

public static int IntRND(int LowerLimit, int HigherLimit, string Error)

{

int count;

bool confirmed;

do

{

confirmed = Int32.TryParse(Console.ReadLine(), out count);

if ((count < LowerLimit) || (count > HigherLimit) || (confirmed != true))

{

Console.WriteLine(Error);

}

} while ((count < LowerLimit) || (count > HigherLimit) || (confirmed != true));

return count;

}

public static int Int()

{

bool ok;

int n;

do

{

ok = Int32.TryParse(Console.ReadLine(), out n);

if (!ok) Console.WriteLine("Вы ввели не целое число. Пожалуйста, введите другое.");

} while (!ok);

return n;

}

public static double Double()

{

bool ok;

double n;

do

{

ok = double.TryParse(Console.ReadLine(), out n);

if (!ok) Console.WriteLine("Вы ввели не вещественное число. Пожалуйста, введите символ.");

} while (!ok);

return n;

}

//---Однонаправленный список---------------------------------------------------------------------------------

#region

// 4. Первое меню (однонаправленный список)

static void Menu1()

{

int SwitchNumber;

Point List1 = null;

do

{

Console.WriteLine("Выберите, что бы вы хотели сделать с однонаправленным списком");

Console.WriteLine("1 - Сформировать новый список");

Console.WriteLine("2 - Вставить после каждого отрицательного элемента списка нулевой элемент");

Console.WriteLine("3 - Вывести текущий список на экран");

Console.WriteLine("4 - Завершить работу с однонаправленным списком");

SwitchNumber = IntRND(1, 4, "Введенное вами число не соответствует ни одному из предложенных вариантов. Пожалуйста, введите другое.");

switch (SwitchNumber)

{

case 1:

{

List1 = FormNewList1();

ShowList1(List1);

break;

}

case 2:

{

List1 = WorkList1(List1);

ShowList1(List1);

break;

}

case 3:

{

ShowList1(List1);

break;

}

case 4:

{

break;

}

}

} while (SwitchNumber != 4);

}

// 4.1 - Формирование нового списка

static Point FormNewList1()

{

Console.WriteLine("Введите число элементов в списке");

int size = IntRND(1, 999999, "Введенное вами число не может являться числом элементов в списке");

double info = 0, LowLimit = 0, HighLimit = 0;

Console.WriteLine("Введите, как вы хотите заполнить список.");

Console.WriteLine("1 - с помощью ручного ввода");

Console.WriteLine("2 - случайными числами");

int SwitchNumber = IntRND(1, 2, "");

if (SwitchNumber == 2)

{

Console.WriteLine("Введите нижний порог для случайного значения элемента");

LowLimit = Double();

Console.WriteLine("Введите верхний порог для случайного значения элемента");

HighLimit = Double();

if (HighLimit < LowLimit)

{

double TempDouble = HighLimit;

HighLimit = LowLimit;

LowLimit = TempDouble;

}

}

if (SwitchNumber == 1)

{

Console.WriteLine("Введите элемент вещественного типа под номером 1");

info = Double();

}

else

{

info = rnd.Next(Convert.ToInt32(LowLimit), Convert.ToInt32(HighLimit));

}

Point beg = MakePoint(info);

Point r = beg;

for (int i = 1; i < size; i++)

{

if (SwitchNumber == 1)

{

Console.WriteLine("Введите элемент вещественного типа под номером {0}", i+1);

info = Double();

}

else

{

info = rnd.Next(Convert.ToInt32(LowLimit), Convert.ToInt32(HighLimit));

}

Point p = MakePoint(info);

r.next = p;

r = p;

}

Console.WriteLine("Список успешно сформирован");

return beg;

}

// 4.2 - Вставка нулевых элементов после отрицательных

static Point WorkList1(Point List1)

{

if (List1 == null)

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("Список пуст");

Console.ForegroundColor = ConsoleColor.Gray;

return List1;

}

Point p = List1;

while (p != null)

{

if (p.data < 0)

{

Point d = MakePoint(0);

d.next = p.next;

p.next = d;

}

p = p.next;

}

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("В список успешно добавлены нулевые элементы после каждого отрицательного элемента");

Console.ForegroundColor = ConsoleColor.Gray;

return List1;

}

// 4.3 - Вывод списка на экран.

static void ShowList1(Point List1)

{

Console.ForegroundColor = ConsoleColor.Yellow;

if (List1 == null)

{

Console.WriteLine("Список пуст");

Console.ForegroundColor = ConsoleColor.Gray;

return;

}

Point p = List1;

while (p != null)

{

Console.Write(p);

p = p.next;

}

Console.ForegroundColor = ConsoleColor.Gray;

Console.WriteLine();

}

#endregion

//-----------------------------------------------------------------------------------------------------------

//---Двунаправленный список----------------------------------------------------------------------------------

#region

static void Menu2()

{

int SwitchNumber;

BiList List2 = null;

do

{

Console.WriteLine("Выберите, что бы вы хотели сделать с двунаправленным списком");

Console.WriteLine("1 - Сформировать новый список");

Console.WriteLine("2 - Удалить из текущего списка первый четный элемент");

Console.WriteLine("3 - Вывести текущий список на экран");

Console.WriteLine("4 - Завершить работу с двунаправленным списком");

SwitchNumber = IntRND(1, 4, "Введенное вами число не соответствует ни одному из предложенных вариантов. Пожалуйста, введите другое.");

switch (SwitchNumber)

{

case 1:

List2 = FormNewList2();

ShowList2(List2);

break;

case 2:

List2 = WorkList2(List2);

ShowList2(List2);

break;

case 3:

ShowList2(List2);

break;

case 4:

break;

}

} while (SwitchNumber != 4);

}

// 5.1 - Формирование нового списка

static BiList FormNewList2()

{

Console.Write("\nВведите число элементов в списке:");

int size = IntRND(1, 999999, "Ошибка, при вводе, повторите!");

int info = 0, LowLimit = 0, HighLimit = 0;

Console.WriteLine("Введите, как вы хотите заполнить список.");

Console.WriteLine("1 - с помощью ручного ввода");

Console.WriteLine("2 - случайными числами");

int SwitchNumber = IntRND(1, 2, "");

if (SwitchNumber == 2)

{

Console.WriteLine("Введите нижнюю границу");

LowLimit = Int();

Console.WriteLine("Введите верхний порог для случайного значения элемента");

HighLimit = Int();

if (HighLimit < LowLimit)

{

int TempInt = HighLimit;

HighLimit = LowLimit;

LowLimit = TempInt;

}

}

if (SwitchNumber == 1)

{

Console.WriteLine("Введите элемент строкового типа под номером 1");

info = Int();

}

else

{

info = rnd.Next(LowLimit, HighLimit);

}

BiList beg = MakePoint2(info);

BiList r = beg;

for (int i = 1; i < size; i++)

{

if (SwitchNumber == 1)

{

Console.WriteLine("Введите элемент строкового типа под номером {0}", i+1);

info = Int();

}

else

{

info = rnd.Next(LowLimit, HighLimit);

}

BiList p = MakePoint2(info);

r.next = p;

p.last = r;

r = p;

}

Console.WriteLine("Список сформирован");

return beg;

}

// 5.2 - Обработка списка

static BiList WorkList2(BiList List2)

{

bool found = false;

if (List2 == null)

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("Список пуст");

Console.ForegroundColor = ConsoleColor.Gray;

return List2;

}

BiList p = List2;

while ((p != null)&&(found==false))

{

if (p.data%2==0)

{

found = true;

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("Первый четный элемент успешно удален из списка");

Console.ForegroundColor = ConsoleColor.Gray;

if (p.last!=null)

{

if (p.next!=null)

{

p.last.next = p.next;

p.next.last = p.last;

break;

}

else

{

p.last.next = null;

break;

}

}

else

{

if (p.next!=null)

{

List2 = p.next;

break;

}

else

{

List2 = null;

break;

}

}

}

p = p.next;

}

if (found==false)

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("В списке нет ни одного четного элемента");

Console.ForegroundColor = ConsoleColor.Gray;

}

return List2;

}

// 5.3 - Вывод списка на экран

static void ShowList2(BiList List2)

{

if (List2 == null)

{

Console.WriteLine("Список пуст");

Console.ResetColor();

return;

}

BiList p = List2;

while (p != null)

{

Console.WriteLine(p.data);

p = p.next;

}

Console.WriteLine();

}

#endregion

//-----------------------------------------------------------------------------------------------------------

//---Бинарное дерево-----------------------------------------------------------------------------------------

#region

static void Menu3(string[] args)

{

Tree Tree = null;

string[] arr = new string[999999];

int pos = 0;

do

{

Console.WriteLine("\n-----------------------------------------------------------");

Console.WriteLine("| Выберите следующее действие: |");

Console.WriteLine("| 1) Сформировать идеально сбалансированное дерево |");

Console.WriteLine("| 2) Найти количество листьев в дереве |");

Console.WriteLine("| 3) Преобразовать в дерево поиска |");

Console.WriteLine("| 4) Вывести дерево на экран |");

Console.WriteLine("| 9) В главное меню |");

Console.WriteLine("| 0) Выход |");

Console.WriteLine("-----------------------------------------------------------");

Console.Write("Действие: ");

int Menu = Search\_for\_Bugs.ProverkaVvoda();

switch (Menu)

{

case 1:

{

Tree = BeginFormIdealTree();

Console.Clear();

Console.WriteLine("-------------------Дерево-------------------");

ShowTree(Tree);

Console.WriteLine("--------------------------------------------");

break;

}

case 2:

BeginSearchIdealTree(Tree);

break;

case 3:

{

CompileTree(ref arr, ref pos, Tree);

Tree = CompileSearchTree(arr, pos);

Console.Clear();

Console.WriteLine("-------------------Дерево-------------------");

ShowTree(Tree);

Console.WriteLine("--------------------------------------------");

break;

}

case 4:

{

Console.Clear();

if (Tree == null)

{

Console.WriteLine("Дерево пустое, давайте для начала его создадим");

Tree = BeginFormIdealTree();

Console.Clear();

Console.WriteLine("-------------------Дерево-------------------");

ShowTree(Tree);

Console.WriteLine("--------------------------------------------");

}

else

{

Console.WriteLine("-------------------Дерево-------------------");

ShowTree(Tree);

Console.WriteLine("--------------------------------------------");

}

break;

}

case 9:

Console.Clear();

Main(args);

break;

case 0:

Console.Clear();

Console.WriteLine(" ");

Console.WriteLine("Завершение работы...");

Thread.Sleep(900);

Environment.Exit(0);

break;

default:

Text\_Dialog.MainError();

continue;

}

} while (true);

}

static Tree BeginFormIdealTree()

{

Console.WriteLine("Введите количество элементов в дереве");

int size = IntRND(1, 999999, "Ошибка, некорректный ввод!");

Tree List3 = null;

List3 = FormIdealTree(size, List3);

return List3;

}

static Tree FormIdealTree(int size, Tree p)

{

int i = 1;

Tree r;

int nl, nr;

if (size == 0)

{

p = null;

return p;

}

nl = size / 2;

nr = size - nl - 1;

Console.WriteLine("Введите {0} элемент дерева", i);

string symbol = Console.ReadLine();

r = new Tree(symbol);

r.left = FormIdealTree(nl, r.left);

r.right = FormIdealTree(nr, r.right);

i++;

return r;

}

static int BeginSearchIdealTree(Tree p)

{

int count = 1;

SearchIdealTree(p, ref count);

Console.ForegroundColor = ConsoleColor.Cyan;

string[] Padej = { "листьев", "листа", "лист" };

if ((count % 100 != 11) && (count % 10 == 1))

{

Console.WriteLine($"В дереве всего {count} {Padej[2]}");

}

else if (((count % 10 >= 2) && (count % 10 <= 4)) && (count % 100 != 12) && (count % 100 != 13) && (count % 100 != 14))

{

Console.WriteLine($"В дереве всего {count} {Padej[1]}");

}

else

{

Console.WriteLine($"В дереве всего {count} {Padej[0]}");

}

Console.ResetColor();

return count;

}

static void SearchIdealTree(Tree p, ref int count)

{

if ((p.left != null) && (p.right == null))

{

count++;

}

else

{

if (p.left != null)

{

SearchIdealTree(p.left, ref count);

}

if (p.right != null)

{

SearchIdealTree(p.right, ref count);

}

}

}

static void CompileTree(ref string[] arr, ref int pos, Tree p)

{

if (p != null)

{

arr[pos] = p.data;

pos++;

CompileTree(ref arr, ref pos, p.left);

CompileTree(ref arr, ref pos, p.right);

}

}

static Tree CompileSearchTree(string[] arr, int pos)

{

Tree root = MakePoint3(arr[0]);

Tree beg = root;

for (int i = 1; i < pos; i++)

{

beg = CompileSearchTreeStep(root, arr[i]);

}

return beg;

}

static Tree CompileSearchTreeStep(Tree root, string data)

{

if (root == null)

{

root = MakePoint3(data);

return root;

}

else

{

if (data.Length < root.data.Length)

{

root.left = CompileSearchTreeStep(root.left, data);

}

if (data.Length > root.data.Length)

{

root.right = CompileSearchTreeStep(root.right, data);

}

}

return root;

}

static void ShowTree(Tree p, int l = 0)

{

if (p != null)

{

ShowTree(p.left, l + 3);

for (int i = 0; i < l; i++) Console.Write(" ");

Console.WriteLine(p.data);

ShowTree(p.right, l + 3);

}

}

#endregion

//-----------------------------------------------------------------------------------------------------------

static void Main(string[] args)

{

Console.Clear();

do

{

Console.WriteLine("\n----------------------------------");

Console.WriteLine("| Выберите следующее действие: |");

Console.WriteLine("| 1) Однонаправленные списки |");

Console.WriteLine("| 2) Двунаправленные списки |");

Console.WriteLine("| 3) Бинарное дерево |");

Console.WriteLine("| 0) Выход из консоли |");

Console.WriteLine("----------------------------------");

Console.Write("Действие: ");

int MainMenu = Search\_for\_Bugs.ProverkaVvoda();

switch (MainMenu)

{

case 1:

Console.Clear();

Menu1();

break;

case 2:

Console.Clear();

Menu2();

break;

case 3:

Console.Clear();

Menu3(args);

break;

case 0:

Console.Clear();

Console.WriteLine(" ");

Console.WriteLine("Завершение работы...");

Thread.Sleep(900);

Environment.Exit(0);

break;

default:

Text\_Dialog.MainError();

continue;

}

} while (true);

}

}

//---------------------------------------------------------------------------------------

}

# **7.Тесты:**

Формирование однонаправленного списка (с помощью ДСЧ):

|  |  |  |  |
| --- | --- | --- | --- |
| **Тест** | **Входные данные** | **Ожидаемый результат** | **Реальный результат** |
| 1 | 5 элементов  нижняя граница -100,  верхняя 50 | 5 чисел от -100 до 50 | 17 28 -70 -44 -26 |
| 2 | 0 элементов | Сообщение об ошибке | Список пуст |
| 3 | -3 элементов | Сообщение об ошибке | Неверный ввод |
| 4 | 1,34 элементов | Сообщение об ошибке | Неверный ввод |
| 5 | Q элементов | Сообщение об ошибке | Неверный ввод |
| 6 | 1 элемент  нижняя граница -100,  верхняя 100 | 1 число от -100 до 100 | 39 |
| 7 | 1111111111111 элементов | Сообщение об ошибке | Неверный ввод |
| 8 | 1 элемент  нижняя граница -2,  верхняя -2 | -2 -2 -2 -2 -2 | -2 -2 -2 -2 -2 |
| 9 | 5 элементов  нижняя граница 1,  верхняя -5 | Сообщение об ошибке | Нижняя граница не может быть больше верхней |

Формирование однонаправленного списка (с клавиатуры):

|  |  |  |  |
| --- | --- | --- | --- |
| **Тест** | **Входные данные** | **Ожидаемый результат** | **Реальный результат** |
| 10 | 5 элементов  4 9,3 99 -6,5 1 | 4 9,3 99 -6,5 1 | 4 9,3 99 -6,5 1 |
| 11 | 0 элементов | Сообщение об ошибке | Список пуст |
| 12 | -3 элементов | Сообщение об ошибке | Неверный ввод |
| 13 | 1,34 элементов | Сообщение об ошибке | Неверный ввод |
| 14 | Q элементов | Сообщение об ошибке | Неверный ввод |
| 15 | 1 элемент  0 | 0 | 0 |
| 16 | 1111111111111 элементов | Сообщение об ошибке | Неверный ввод |

Формирование двунаправленного списка (с помощью ДСЧ):

|  |  |  |  |
| --- | --- | --- | --- |
| **Тест** | **Входные данные** | **Ожидаемый результат** | **Реальный результат** |
| 1 | 5 элементов  нижняя граница -100,  верхняя 100 | 5 чисел от -100 до 100 | 47  33  -61  -86  -16 |
| 2 | 0 элементов | Сообщение об ошибке | Список пуст |
| 3 | -3 элементов | Сообщение об ошибке | Неверный ввод |
| 4 | 1,34 элементов | Сообщение об ошибке | Неверный ввод |
| 5 | Q элементов | Сообщение об ошибке | Неверный ввод |
| 6 | 1 элемент  нижняя граница 2,  верхняя 2 | 2 | 2 |
| 7 | 1111111111111 | Сообщение об ошибке | Неверный ввод |
| 8 | 5 элементов  нижняя граница -2,  верхняя -2 | -2 -2 -2 -2 -2 | -2 -2 -2 -2 -2 |
| 9 | 1 строка 1 столбец  нижняя граница 1,  верхняя -5 | Сообщение об ошибке | Нижняя граница не может быть больше верхней |

Формирование двунаправленного списка (с клавиатуры):

|  |  |  |  |
| --- | --- | --- | --- |
| **Тест** | **Входные данные** | **Ожидаемый результат** | **Реальный результат** |
| 10 | 5 элементов  1  43  11  -3  1 | 1  43  11  -3  1 | 1  43  11  -3  1 |
| 11 | 0 элементов | Сообщение об ошибке | Список пуст |
| 12 | -3 элементов | Сообщение об ошибке | Неверный ввод |
| 13 | 1,34 элементов | Сообщение об ошибке | Неверный ввод |
| 14 | Q элементов | Сообщение об ошибке | Неверный ввод |
| 15 | 2 элемента  0  2 | 0  2 | 0  2 |
| 16 | 1111111111111 элементов | Сообщение об ошибке | Неверный ввод |

Формирование бинарного дерева (с помощью ДСЧ):

|  |  |  |  |
| --- | --- | --- | --- |
| **Тест** | **Входные данные** | **Ожидаемый результат** | **Реальный результат** |
| 1 | 10 элементов | впвп  ааа  ке  рвап344  ккк  ыва  и  мс  ва  и | впвп  ааа  ке  рвап344  ккк  ыва  и  мс  ва  и |
| 2 | 0 элементов | Сообщение об ошибке | Дерево пустое |
| 3 | -3 элементов | Сообщение об ошибке | Неверный ввод |
| 4 | 1,34 элементов | Сообщение об ошибке | Неверный ввод |
| 5 | Q элементов | Сообщение об ошибке | Неверный ввод |
| 7 | 1111111111111 | Сообщение об ошибке | Неверный ввод |

Формирование бинарного дерева (с клавиатуры):

|  |  |  |  |
| --- | --- | --- | --- |
| **Тест** | **Входные данные** | **Ожидаемый результат** | **Реальный результат** |
| 8 | 10 элементов  1  2  3  4  5  6  7  8  9  10 | Дерево из 10 элементов | 4  3  2  6  5  1  9  8  7  10 |
| 9 | 0 элементов | Сообщение об ошибке | Дерево пустое |
| 10 | -3 элементов | Сообщение об ошибке | Неверный ввод |
| 11 | 1,34 элементов | Сообщение об ошибке | Неверный ввод |
| 12 | Q элементов | Сообщение об ошибке | Неверный ввод |
| 13 | 4 элемента  Gf  sdf  wer  ccccccc | gf  sdf  wer  ccccccc | gf  sdf  wer  ccccccc |

Добавление нулей в однонаправленный список:

|  |  |  |  |
| --- | --- | --- | --- |
| **Тест** | **Входные данные** | **Ожидаемый результат** | **Реальный результат** |
| 1 | 17 28 -70 -44 -26 | 17 28 -70 0 -44 0 -26 0 | 17 28 -70 0 -44 0 -26 0 |
| 2 | 39 | Сообщение об ошибке | Нет отрицательных элементов |
| 3 | -2 -2 -2 -2 -2 | -2 0 -2 0 -2 0 -2 0 -2 0 | -2 0 -2 0 -2 0 -2 0 -2 0 |
| 4 | Пустой список | Сообщение об ошибке | Список пуст |

Удаление первого чётного из двунаправленного списка:

|  |  |  |  |
| --- | --- | --- | --- |
| **Тест** | **Входные данные** | **Ожидаемый результат** | **Реальный результат** |
| 1 | 47  33  -61  -86  -16 | 47  33  -61  -16 | 47  33  -61  -16 |
| 2 | 2 | Список пуст | Список пуст |
| 3 | -2 -2 -2 -2 -2 | -2 -2 -2 -2 | -2 -2 -2 -2 |
| 4 | Пустой список | Сообщение об ошибке | Список пуст |
| 5 | 1  43  11  -3  1 | Сообщение об ошибке | Нет четных элементов |

Поиск листьев в бинарном дереве:

|  |  |  |  |
| --- | --- | --- | --- |
| **Тест** | **Входные данные** | **Ожидаемый результат** | **Реальный результат** |
| 1 | 4  3  2  6  5  1  9  8  7  10 | 4 листа | 4 листа |
| 2 | вап | 1 лист | 1 лист |
| 3 | gf  sdf  wer  ccccccc | 2 листа | 2 листа |
| 4 | Пустое дерево | Сообщение об ошибке | Дерево пустое |

Преобразование в дерево поиска:

|  |  |  |  |
| --- | --- | --- | --- |
| **Тест** | **Входные данные** | **Ожидаемый результат** | **Реальный результат** |
| 1 | 4  3  2  6  5  1  9  8  7  10 | Дерево поиска | 1  10 |
| 2 | 4  3  2  6  5  1  9  8  7  10  gf  sdf  wer  ccccccc | Дерево поиска | 1  10  wer  ccccccc |

Печать однонаправленного списка:

|  |  |  |  |
| --- | --- | --- | --- |
| **Тест** | **Входные данные** | **Ожидаемый результат** | **Реальный результат** |
| 1 | пустой список | Сообщение об ошибке | Список пуст |
| 2 | 4 9,3 99 -6,5 1 | 4 9,3 99 -6,5 1 | 4 9,3 99 -6,5 1 |

Выбор в меню:

|  |  |  |  |
| --- | --- | --- | --- |
| **Тест** | **Входные данные** | **Ожидаемый результат** | **Реальный результат** |
| 1 | 1 | Меню с однонаправленным списком | Menu1() |
| 2 | 2 | Меню с двунаправленным списком | Menu2() |
| 3 | 3 | Меню с бинарным деревом | Menu3() |
| 4 | 0 | Выход из программы | Выход из программы |

Чёрный ящик:

Однонаправленный список:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Т1 | Т2 | Т3 | Т4 | Т5 | Т6 | Т7 | Т8 | Т9 | Т10 | Т11 | Т12 | Т13 | Т14 | Т15 | Т16 |
| Входные данные | | | | | | | | | | | | | | | | |
| Нули |  | + |  |  |  |  |  |  |  |  | + |  |  |  | + |  |
| Вещественные числа |  |  |  | + |  |  |  |  |  | + |  |  | + |  |  |  |
| Целые положительные числа | + |  |  |  |  | + | + | + | + | + |  |  |  |  | + |  |
| Целые отрицательные числа | + |  | + |  |  | + |  | + | + | + |  | + |  |  |  |  |
| Пустота |  | + |  |  |  |  | + |  |  |  | + |  |  |  |  | + |
| Символы |  |  |  |  | + |  |  |  |  |  |  |  |  | + |  |  |
| Ввод вручную |  |  |  |  |  |  |  |  |  | + | + | + | + | + | + | + |
| ДСЧ | + | + | + | + | + | + | + | + | + |  |  |  |  |  |  |  |
| Пустой набор |  | + |  |  | + |  |  |  |  |  | + |  |  | + |  |  |
| Единичный набор |  |  |  |  |  | + |  | + |  |  |  |  |  |  |  |  |
| Нормальный набор | + |  | + | + |  |  | + |  | + | + |  | + | + |  | + | + |
| Выходные данные | | | | | | | | | | | | | | | | |
| Сообщение об ошибке |  | + | + | + | + |  | + |  | + |  | + | + | + | + |  | + |
| Однонаправленный список | + |  |  |  |  | + |  | + |  | + |  |  |  |  | + |  |
| Список с добавленными нулями | + |  |  |  |  | + |  | + |  | + |  |  |  |  | + |  |

Двунаправленный список:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Т1 | Т2 | Т3 | Т4 | Т5 | Т6 | Т7 | Т8 | Т9 | Т10 | Т11 | Т12 | Т13 | Т14 | Т15 | Т16 |
| Входные данные | | | | | | | | | | | | | | | | |
| Нули |  | + |  |  |  |  |  |  |  |  | + |  |  |  | + |  |
| Вещественные числа |  |  |  | + |  |  |  |  |  | + |  |  | + |  |  |  |
| Целые положительные числа | + |  |  |  |  | + | + | + | + | + |  |  |  |  | + |  |
| Целые отрицательные числа | + |  | + |  |  | + |  | + | + | + |  | + |  |  |  |  |
| Пустота |  | + |  |  |  |  | + |  |  |  | + |  |  |  |  | + |
| Символы |  |  |  |  | + |  |  |  |  |  |  |  |  | + |  |  |
| Ввод вручную |  |  |  |  |  |  |  |  |  | + | + | + | + | + | + | + |
| ДСЧ | + | + | + | + | + | + | + | + | + |  |  |  |  |  |  |  |
| Пустой набор |  | + |  |  | + |  |  |  |  |  | + |  |  | + |  |  |
| Единичный набор |  |  |  |  |  | + |  | + |  |  |  |  |  |  |  |  |
| Нормальный набор | + |  | + | + |  |  | + |  | + | + |  | + | + |  | + | + |
| Выходные данные | | | | | | | | | | | | | | | | |
| Сообщение об ошибке |  | + | + | + | + |  | + |  | + |  | + | + | + | + |  | + |
| Однонаправленный список | + |  |  |  |  | + |  | + |  | + |  |  |  |  | + |  |
| Удаление четных | + |  |  |  |  | + |  | + |  |  |  |  |  |  | + |  |

Бинарное дерево:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Т1 | Т2 | Т3 | Т4 | Т5 | Т7 | Т10 | Т11 | Т12 | Т13 | Т14 | Т15 |
| Выходные данные | | | | | | | | | | | | |
| Нули |  | + |  |  |  |  |  | + |  |  |  | + |
| Вещественные числа |  |  |  | + |  |  | + |  |  | + |  |  |
| Целые положительные числа | + |  |  |  |  | + | + |  |  |  |  | + |
| Целые отрицательные числа | + |  | + |  |  |  | + |  | + |  |  |  |
| Пустота |  | + |  |  |  | + |  | + |  |  |  |  |
| Символы |  |  |  |  | + |  |  |  |  |  | + |  |
| Ввод вручную |  |  |  |  |  |  | + | + | + | + | + | + |
| ДСЧ | + | + | + | + | + | + |  |  |  |  |  |  |
| Пустой набор |  | + |  |  | + |  |  | + |  |  | + |  |
| Единичный набор |  |  |  |  |  |  |  |  |  |  |  |  |
| Нормальный набор | + |  | + | + |  | + | + |  | + | + |  | + |
| Выходные данные | | | | | | | | | | | | |
| Сообщение об ошибке |  | + | + | + | + | + |  | + | + | + | + |  |
| Однонаправленный список | + |  |  |  |  |  | + |  |  |  |  | + |
| Листья найдены | + |  |  |  |  |  | + |  |  |  |  | + |
| Листья не найдены |  | + | + | + | + | + |  | + | + | + | + |  |