Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Институт информационных технологий

Факультет компьютерных технологий

Кафедра информационных систем и технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

«ПОИСК БИНАРНЫХ ДЕРЕВЬЕВ В ВИДЕ МАТЕМАТИЧЕСКИХ ФОРМУЛ»

Студент:

гр. 281071 Полуянович Д. В.

Руководитель: Савенко А. Г.

Минск 2023

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Институт информационных технологий

Факультет компьютерных технологий

УТВЕРЖДАЮ

Заведующий кафедрой ИСиТ

––––––––––––––––––––––––

(подпись)

––––––––––––––––– 2023 г.

ЗАДАНИЕ

на курсовую работу

студенту      Полуянович Даниилу Вячеславовичу

1. Тема работы    «Представление математических формул в виде бинарных деревьев»

2. Срок сдачи студентом законченной работы––15.05.2023 г.–––

3. Исходные данные к работе

4. Содержание пояснительной записки

Введение

1. Моделирование программного средства

2. Проектирование программного средства

3. Оценка работы (тестирование) программного средства и анализ результатов

Выводы

Список используемых источников

Приложение А. Фрагменты программного кода

5. Перечень графического материала

1. "Math binary-tree", схема программы, чертеж – формат А3, лист 1.

2. "Представление математических формул в виде бинарных деревьев", схема алгоритма, чертеж – формат А3, лист 1.

3. "Предоставление поиска поддеревьев в бинарных деревьях", схема алгоритма, чертеж – формат А3, лист 1.

6. Консультант по курсовой работе А.Г.Савенко

7. Дата выдачи задания – 18.02.2023 г.

8. Календарный график работы над курсовой работой на весь период   
(с обозначением сроков выполнения и процентом от общего объёма работы):

|  |  |  |  |
| --- | --- | --- | --- |
| № | Содержание работ | Срок  выполнения | % от общего объёма работы |
| 1 | Раздел 1 | 04.03.2023 | 15 % |
| 2 | Раздел 2 | 01.04.2023 | 30 % |
| 3 | Раздел 3 | 15.04.2023 | 60 % |
| 4 | Раздел 4 | 06.05.2023 | 90 % |
| 5 | Оформление пояснительной записки и графического материала | 15.05.2023 | 100 % |
| 6 | Защита курсовой работы |  |  |

РУКОВОДИТЕЛЬ–––––– А.Г.Савенко

(подпись)

Задание принял к исполнению –––\_\_\_\_––

(дата и подпись студента)

**РЕФЕРАТ**

Автора: Полуянович Д. В.

Руководитель:Савенко А. Г.

Данной целью курсового проекта является реализация поиска бинарных деревьев в виде математических формул в виде бинарных деревьев. Определение и анализ алгоритма поиска бинарных деревьев её архитектуру и работу.

В этом курсовом проекте будет представлен алгоритм поиска бинарных деревьев в виде математических прилагающим к нему исходным кодом и диаграммами работы.

Данный документ будет состоять из содержания, введения, моделирования, тестирования, и вывода, вместе с исходным кодом и диаграммами соответственно.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc10613)

[МОДЕЛИРОВАНИЕ 7](#_Toc8288)

[ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 8](#_Toc9678)

[ТЕСТИРОВАНИЕ 11](#_Toc18594)

[ВЫВОДЫ 13](#_Toc12731)

# **ВВЕДЕНИЕ**

Объектом исследования предметной области является алгоритм бинарных деревьев.

Алгоритм бинарных деревьев или же двоичное дерево поиска был разработан был разработан Эндрю Дональдом Бутом в 1960-м году. Данный алгоритм представляет собой древовидную структуру разделений данных на вершину, ветви, листья.

Вершина - представляет собой начальный элемент данных от которого начинаются ответвления и начало дерева.

Ветви - представляют ответвление от вершины и ответвление от родительских ветвь являющие их дочерними.

Листья - представляют собой конечные значения вершин и ветвь и являются одиночными то есть конечными значениями дерева.

Бинарное дерево представляет собой запись вида левого значения, правого значения и данных родительской ветви.

Поиск бинарного дерева осуществляется путём исключения в зависимости от сравнений правого значения и левого значения ориентируясь по значению родительской ветви или родительской вершины.

Предметом исследование является реализация представление математических формул в виде бинарных деревьев. Данный алгоритм будет представлять набор методов для поиска оптимальных математически формул в бинарном дереве. Каждый элемент дерева будет представлять структуру из объектов класса содержащих в себе значение и математическую функцию для обработки текущего значения.

# **МОДЕЛИРОВАНИЕ**

Алгоритм поиска бинарных деревьев в виде математических должен состоять из нескольких методов:

* Функция ввода - представляет собой метод ввода значения числа и ввод веса для ветви деревьев.
* Функция генерации дерева - представляет собой метод создания бинарного дерева с указанием веса, значения, и математической функции для работы с водимым значением.
* Функция поиска - представляет собой метод поиска математической функции по бинарному дереву переходя всё дальше и дальше от вершины к ветвям до конечного результата.
* Функция вывода - представляет собой функцию вывода формулы и результата обработки вводимого значения.

Вершины, ветви и листья будут представлять собой объекты класса в котором будут хранится ключ, математическая формула, функция обработки числа, и дочерний элемент для ветви или листьев.

# **ПРОГРАММНАЯ РЕАЛИЗАЦИЯ**

Структура запуска программы будет состоять из основного класса Program и статического метода Main где будет запускаться работа программы.

Реализация алгоритма поиска бинарных деревьев в виде математических формул будет представляться в виде класса BinaryTreeElement.

Класс BinaryTreeElement представляет собой узлы бинарного дерева. Данный класс содержит в себе параметры:

* Параметр key(string) - представляет собой ключ дерева по которому мы будет обращатся к дочерним узлам.
* Параметр formula(string)- представляет собой строку формулы.
* Параметр childrens(BinaryTreeElement[]) - представляет дочерние элементы дерева.
* Параметр value(double) - представляет значение текущего числа.

Так же класс BinaryTreeElement имеет следующие методы.

* Метод add - представляет собой метод добавления узла.
* Метод remove - представляет собой метод удаления узла.
* Метод find - представляет собой поиск узла по ключу.
* Метод calculate - представляет собой метод расчёта значения по заданной математической формуле.

class BinaryTreeElement{

        public int key;

        public string formula;

        public List<dynamic> childrens = new List<dynamic>();

        public double value = 0;

        public BinaryTreeElement(int \_key, string \_formula, List<dynamic> \_childrens = null)

        {

            this.key = \_key;

            this.formula = \_formula;

            if(\_childrens!=null){

            this.childrens = new List<dynamic>();

            } else{

            this.childrens = \_childrens;

            }

        }

        public bool add(BinaryTreeElement child){

            bool has\_child = childrens.Exists(item=>item.key==child.key);

            if(has\_child){

                Console.WriteLine("Текущий узел существует!");

                return false;

            } else{

                Console.WriteLine("Добавлен узел!");

                childrens.Add(child);

                return true;

            }

        }

        public bool remove(int key\_child){

            int index = childrens.FindIndex(item=>item.key==key\_child);

            if(index>0){

                Console.WriteLine("Удалён узел!");

                childrens = childrens.FindAll(item => item.key != index);

                return true;

            } else{

                Console.WriteLine("Данный узел отсутствует!");

                return false;

            }

        }

        public BinaryTreeElement? find(int key\_child){

            BinaryTreeElement child = childrens.Find(item => item.key == key\_child);

            if(child!=null){

                Console.WriteLine("Найден узел!");

                return child;

            } else{

                Console.WriteLine("Данный узел не существует!");

                return null;

            }

        }

        public double calculate(double data){

            try

            {

                string math = formula.Replace("$", data.ToString());

                string value = new DataTable().Compute(math, null).ToString();

                Console.WriteLine("Применена формула: "+math);

                Console.WriteLine("Результат: "+value);

                return Convert.ToDouble(value);

            }

            catch (System.Exception)

            {

                Console.WriteLine("Не корректная формула: "+formula.Replace("$", data.ToString()));

                return data;

            }

        }

    }

В данной программе так же существует метод генерации дерева GenerateTree.

BinaryTreeElement GenerateTree(){

var tree = new BinaryTreeElement(1,"$+2",

new List<dynamic>() {

new BinaryTreeElement(2,"$\*3",

new List<dynamic>() {

new BinaryTreeElement(4,"$/2"),

new BinaryTreeElement(5,"$+13"),

}

),

new BinaryTreeElement(3,"$-1",

new List<dynamic>() {

new BinaryTreeElement(6,"$+7"),

new BinaryTreeElement(7,"$-4"),

}

)

}

);

return tree;

}

Метод ввода значения SetValue.

double SetValue(string val){

double value = 0;

try{

Console.WriteLine("Введите число: ");

value = Convert.ToDouble(val);

}

catch(System.Exception){

Console.WriteLine("Неверно указанно число");

}

return value

}

Метод вывода значения GetValue.

void GetValue(tree){

var subtree = tree;

try{

foreach (string k in keys){

subtree = subtree.find(Convert.ToInt32(k));

}

Console.WriteLine(subtree.calculate(value));

}

catch(System.Exception){

Console.WriteLine("Не удалось выполнить функцию!");

}

}

# **ТЕСТИРОВАНИЕ**

Таблица 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Модуль/функция | Шаги выполнения | Ожидаемый результат | Фактический результат |
| 1 | Генерация дерева | 1. Запустить приложение | Вывод дерева в виде json | Вывод дерева через консоль в формате json |
| 2 | Ввод значения | 1. Запустить приложение 2. Ввод отрицательного числа -5 | Вывод ошибки что данной число не подходит | Вывод ошибки “Данное значение не подходит” |
| 3 | Ввод значения | 1. Запустить приложение 2. Ввод отрицательного строки “rgt” | Вывод ошибки что данной число не подходит | Вывод ошибки “Данное значение не подходит” |
| 4 | Ввод ключа | 1. Запустить приложение 2. Ввод числа 1 3. Ввод ключа 1 | Вывод результата расчёта и вывода формулы. | Выводит формулу и результат математической операции и дочерних элементов в формате json |
| 5 | Ввод ключа | 1. Запустить приложение 2. Ввод числа 1 3. Ввод ключа -1 | Вывод ошибки о том что данный узел не найден | Выводит ошибку “Данный узел не существует!” |
| 6 | Вывод результата | 1. Запустить приложение 2. Ввод числа 1 3. Ввод ключа 1 4. Ввод ключа 2 5. Ввод числа 0 | Вывод результата | Вывод результата подсчёта и математических формул |
| 7 | Добавление узла | 1. Запустить приложение 2. Ввод “add 1=>2” 3. Ввод “$+2” | Добавление узла | Вывод сообщения о том что узел создан |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 8 | Удаление узла | 1. Запустить приложение 2. Ввод “remove 1=>2” | Удаление узла | Вывод сообщения о том что узел удалён |
| 9 | Удаление узла | 1. Запустить приложение 2. Ввод “remove 1=>hello” | Вывод ошибки | Вывод сообщения о том что не найден |
| 10 | Добавление узла | 1. Запустить приложение 2. Ввод “add 1=>2” 3. Ввод “$+ркере” | Вывод ошибки | Вывод сообщения о том что формула не верна |

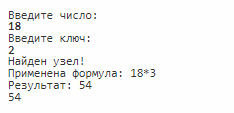


Рисунок 1 - вывод значения



Рисунок 2 - проверка ввода числа

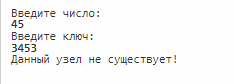


Рисунок 3 - проверка поиска узла

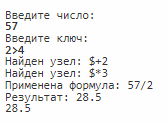


Рисунок 4 - проверка поиска вложенного узла

# **ВЫВОДЫ**

В данном курсовом проекте был реализован и дополнен алгоритм бинарного дерева с поиском математических формул. Был изучен принцип работы данного алгоритма и представлен в виде мультикластерной структуры бинарного дерева.

Данная программа была испытанна. проверена и готова к эксплуатации.