**Лабораторная работа №4**

**Полный перебор**

Вариант 1,9:

Скобочные выражения

Будем считать, что скобочное выражение состоит только из круглых

открывающихся и закрывающихся скобок.

Примеры:

(())()

()()

Скобочное выражение может быть правильным (см. примеры выше) и

неправильным, например,

)(()

(()))

())(

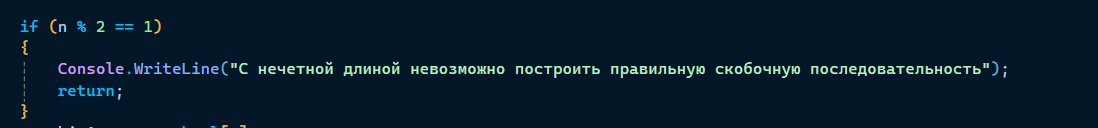
Написать программу, выводящую все правильные скобочные выражения

данной длины N (N<=20) и подсчитывающей их количество. При N>6 результаты

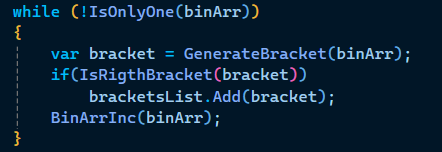
выводить в текстовый файл.

**Алгоритм**

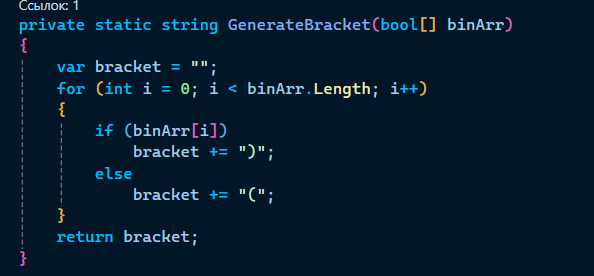
1. Перед выполнением алгоритма проверяем является ли введенное число от пользователя четным



Если число не четное, то построить правильную скобочную последовательность невозможно, так как всегда будет оставаться одна открытая или закрытая скобка.

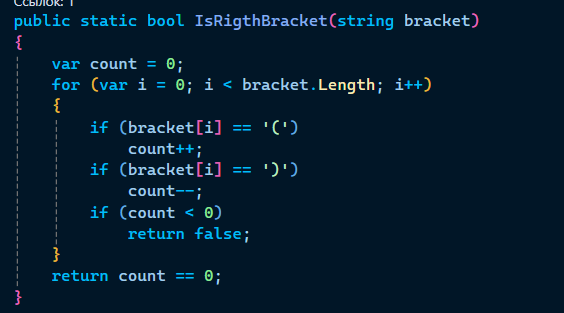
1. Создаем булевый массив через который будем генерировать скобочную последовательность. Так же создаем список для хранения правильных скобочных последовательностей
2. 

Пока в бинарном массиве все элементы не равны true выполняем тело цикла, в котором вызываем метод GenerateBracket, который вернет сгенерированную скобочную последовательность

1. 

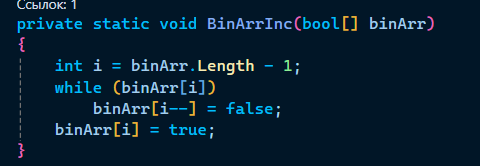
В данном методе создаем строковую переменную bracket, в которую будем добавлять открывающую и закрывающую скобки. Проходим через цикл по бинарному массиву и если элемент равен true, то добавляем в переменную bracket закрывающую скобку, а иначе открывающую скобку. В итоге возвращаем скобочную последовательность

1. Дальше в цикле проверяем сгенерированную скобочную последовательность на правильность вызывая метод IsRightBracket

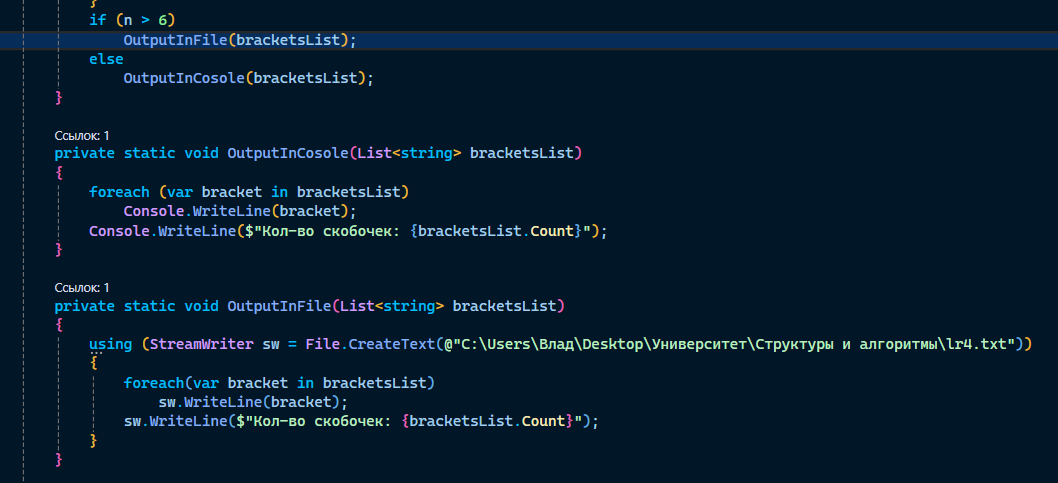


В данном методе мы создаем переменную count, которая будет для нас балансировщиком. Через цикл проходим по скобочной последовательности. Если в i-том элементе строки равен открывающей скобке, то увеличиваем count на 1. Если в i-тый элемент строки равен закрывающей скобке, то уменьшаем count на 1. Если во время прохождения цикла count становится меньше 0, то скобочная последовательность неправильная. Если после цикла count равняется 0, то скобочная последовательность правильная.

1. Если скобочная последовательность правильная, то добавляем данную последовательность в список.
2. Увеличиваем бинарный массив на 1.



1. Если N было больше 6, то записываем результат в файл lr4.txt. А если нет, то выводим в консоль



**Код программы**

Program.cs

using Lr1;

using Lr1.Stack;

using Lr1.Queue;

using Lr2;

using Lr4;

namespace Lr

{

internal class Program

{

static void Main()

{

Console.WriteLine("Введите натуральное число:");

var input = int.Parse(Console.ReadLine());

Lab4.var1(input);

}

}

}

Lab.4.cs

namespace Lr4

{

public static class Lab4

{

public static void var1(int n)

{

if (n % 2 == 1)

{

Console.WriteLine("С нечетной длиной невозможно построить правильную скобочную последовательность");

return;

}

var binArr = new bool[n];

var bracketsList = new List<string>();

while (!IsOnlyOne(binArr))

{

var bracket = GenerateBracket(binArr);

if(IsRigthBracket(bracket))

bracketsList.Add(bracket);

BinArrInc(binArr);

}

if (n > 6)

OutputInFile(bracketsList);

else

OutputInCosole(bracketsList);

}

private static void OutputInCosole(List<string> bracketsList)

{

foreach (var bracket in bracketsList)

Console.WriteLine(bracket);

Console.WriteLine($"Кол-во скобочек: {bracketsList.Count}");

}

private static void OutputInFile(List<string> bracketsList)

{

using (StreamWriter sw = File.CreateText(@"C:\Users\Влад\Desktop\Университет\Структуры и алгоритмы\lr4.txt"))

{

foreach(var bracket in bracketsList)

sw.WriteLine(bracket);

sw.WriteLine($"Кол-во скобочек: {bracketsList.Count}");

}

}

public static bool IsRigthBracket(string bracket)

{

var count = 0;

for (var i = 0; i < bracket.Length; i++)

{

if (bracket[i] == '(')

count++;

if (bracket[i] == ')')

count--;

if (count < 0)

return false;

}

return count == 0;

}

private static string GenerateBracket(bool[] binArr)

{

var bracket = "";

for (int i = 0; i < binArr.Length; i++)

{

if (binArr[i])

bracket += ")";

else

bracket += "(";

}

return bracket;

}

private static void BinArrInc(bool[] binArr)

{

int i = binArr.Length - 1;

while (binArr[i])

binArr[i--] = false;

binArr[i] = true;

}

private static bool IsOnlyOne(bool[] binArr)

{

return binArr.All(el => el == true);

}

}

}

**Тесты**

