**v n1. Что такое рекурсия? Что такое рекурсивный метод (функция)?**

*Рекурсия* – это разработка метода таким образом, чтобы он вызывал сам себя. Рекурсивные вызовы метода должны завершаться при достижении некоторого условия. В противном случае произойдет переполнение памяти и программа «зависнет» не достигнув вычисления необходимого результата.

*Рекурсивный метод* – это метод, который вызывает сам себя. В рекурсивном методе помещается вызов этого же метода по его имени.

Последовательный процесс рекурсивных вызовов метода подобен циклическому процессу.

**2. Как работает рекурсивный вызов метода?**

Если метод вызывает сам себя, то в памяти происходят следующие процессы:

* в системном стеке выделяется память для новых локальных переменных и параметров;
* программный код метода выполняется сначала с новыми локальными переменными и параметрами. Эти локальные переменные и параметры получают новые начальные значения. Сам программный код метода не копируется;
* при возврате из рекурсивного метода (оператор return) происходит восстановление старых локальных переменных и параметров а также их значений в точке вызова этого метода.

**3. Какие преимущества дает использование рекурсии в программах?**

Рекурсию всегда сравнивают с итерацией. Для многих задач рекурсия дает следующие взаимосвязанные преимущества:

* при вызове рекурсивной функции не нужно дополнительно сохранять временные значения локальных переменных. Компилятор строит код рекурсивной функции таким образом, что временные значения локальных переменных автоматически сохраняются при каждом рекурсивном вызове;
* в некоторых случаях рекурсивные алгоритмы выглядят более упрощенно и более наглядно чем итерационные. Это связано с тем, что в итерационных алгоритмах для запоминания промежуточных результатов нужно вводить дополнительные переменные, которые могут усложнить восприятие хода выполнения самого алгоритма.

**4. Какие недостатки использования рекурсии в программах?**

В сравнении с итерационными вызовами, построение рекурсивных вызовов имеет следующие недостатки:

* для заданного алгоритма рекурсивные вызовы работают медленнее чем итерационные. Это связано с дополнительными затратами системных ресурсов на неоднократные вызовы методов;
* многоразовый вызов методов может привести к переполнению системного стека. В этом случае среда CLR сгенерирует соответствующую исключительную ситуацию. Поэтому, важно, чтобы рекурсивный метод был разработан таким образом, чтобы в нем объявлялось минимально возможное количество параметров и локальных переменных.

Мой код (C#)

задание: Проверить, является ли фрагмент строки с i-го по j-ый символ палиндромом.

using System.Text;

class MyProg

{

static bool IsPalindrom(string str)

{

if (str == null) throw new ArgumentNullException("str");

str = str.ToLower().Replace(" ", string.Empty);

return IsPalindromInternal(str);

}

static bool IsPalindromInternal(string str)

{

if (str.Length == 1 || string.IsNullOrEmpty(str)) return true;

if (!str[0].Equals(str[str.Length - 1])) return false;

return IsPalindromInternal(str.Substring(1, str.Length - 2));

}

static void Main(string[] args)

{

string str, str2 = "";

int \_i, j;

Console.WriteLine("Input string");

str = Console.ReadLine();

Console.WriteLine("Input i");

int.TryParse(Console.ReadLine(), out \_i);

if (\_i < 0) throw new Exception("Wrong i");

Console.WriteLine("Input j");

Int32.TryParse(Console.ReadLine(), out j);

if (\_i > str.Length) throw new Exception("Wrong j");

for (int i = \_i - 1; i < j; i++)

{

str2 += str[i];

}

Console.WriteLine(IsPalindrom(str2));

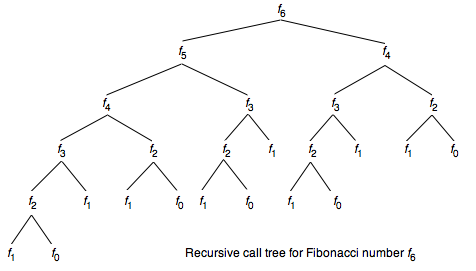
StreamWriter sw = new StreamWriter("../../../out.txt");

sw.WriteLine("Original string : " + str + "\nIs Palindrom " + IsPalindrom(str2));

sw.Close();

}

}



источник: <https://www.bestprog.net/ru/2019/01/21/recursion-examples-of-recursive-methods-functions-in-c-ru/>

https://habr.com/ru/post/319790/