

using System;

class RecursiveNumberSystemConverter

{

static void Main()

{

Console.WriteLine("Рекурсивный перевод числа в p-ичную систему счисления (p < 10)");

// Ввод данных

Console.Write("Введите целое число N: ");

int N = int.Parse(Console.ReadLine());

Console.Write("Введите основание системы счисления p (2 ≤ p < 10): ");

int p = int.Parse(Console.ReadLine());

// Проверка корректности ввода

if (p < 2 || p >= 10)

{

Console.WriteLine("Ошибка: основание системы должно быть 2 ≤ p < 10");

return;

}

// Обработка нуля

if (N == 0)

{

Console.WriteLine($"Число 0 в {p}-ичной системе: 0");

return;

}

// Для отрицательных чисел

bool isNegative = N < 0;

N = Math.Abs(N);

// Вызов рекурсивной функции

Console.Write($"Число {(isNegative ? "-" : "")}{Math.Abs(N)} в {p}-ичной системе: ");

if (isNegative) Console.Write("-");

ConvertToBase(N, p);

Console.WriteLine();

}

// Рекурсивная функция перевода в p-ичную систему

static void ConvertToBase(int number, int baseNum)

{

if (number == 0)

return;

// Рекурсивный вызов для старших разрядов

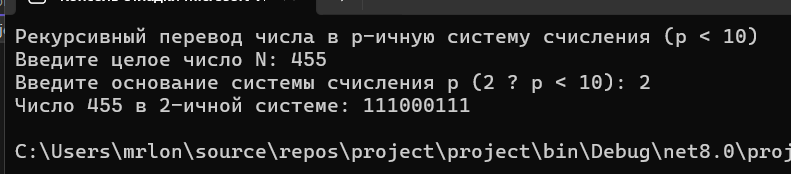
ConvertToBase(number / baseNum, baseNum);

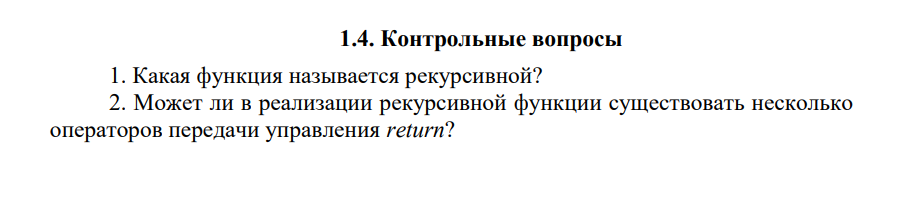
// Вывод текущего разряда

Console.Write(number % baseNum);

}

}





**1. Какая функция называется рекурсивной?**

Функция называется рекурсивной, если в своем теле она вызывает саму себя (прямо или косвенно через другие функции). Рекурсивная функция должна содержать:

* **Базовый случай** (условие завершения) - простейший вариант задачи, который решается без рекурсии
* **Рекурсивный случай** - вызов той же функции с измененными параметрами, приближающими к базовому случаю

**2. Может ли в реализации рекурсивной функции существовать несколько операторов передачи управления return?**

Да, в рекурсивной функции может быть несколько операторов return. Это часто используется для:

* Возврата разных значений в базовом случае
* Возврата разных значений в зависимости от условий в рекурсивном случае