Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Рязанский государственный радиотехнический
университет имени В.Ф. Уткина»
Рязанский станкостроительный колледж

Отчёт о практической работе
Оптимизация
по дисциплине
«Основы алгоритмизации и программирования»

Выполнил:
Студент группы ИСП-22
Стуканов М.О.
Проверил:
Родин Е.Н.

Рязань 2020

Основная часть

Цель работы: получение навыков оптимизации кода

Ход выполнения работы:

- **Постановка задачи**

1. Проведите испытания программ из примера 2. Оцените время их работы. Для проведения испытаний в папке с лекциями приложено 2 файла: «Лекция ОАиП
2. №9. Пример 2. Вариант1» и «Лекция ОАиП №9. Пример2. Вариант1». 2. Изучите пример 2 и найдите еще варианты для оптимизации программы.

Решение

Код программы вариант №1

```
const int n = 10; //Размер массива
int i, j;   //Счетчики
int sum;    //Сумма
int[,] matr = new int[n, n]; //Массив

// Инициализация случайными значениями в диапазоне -100-100
Random rnd = new Random();
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++) matr[i, j] = rnd.Next(10);
//Ищем сумму положительных чисел
sum = 0;
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
        if (i == j) sum = sum + matr[i, j];
Console.WriteLine("Сумма пложительных элементов= " + sum);
Console.ReadKey();//Пауза
```

Определим сложность работы алгоритма?

Код программы вариант №2

```
const int n = 10; //Размер массива
int i, j,   //Счетчики
    sum;    //Сумма
int[,] matr = new int[n, n]; //Массив

// Инициализация случайными значениями в диапазоне -100-100
Random rnd = new Random();
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++) matr[i, j] = rnd.Next(-100, 100);
//Ищем сумму положительных чисел
sum = 0;
for (i = 0; i < n; i++)
    sum = sum + matr[i, i];
Console.WriteLine("Сумма пложительных элементов= " + sum);
Console.ReadKey();//Пауза
```

Определим сложность работы алгоритма?

Рисунок 1 - Пример кода вариант №1

Пример 2.
Подсчитать сумму делителей элементов главной диагонали матрицы.

Код программы вариант №1

```
int n; //Размер массива
int i, j, k;   //Счетчики
int sum = 0,    //Сумма
    kol; // Количество

Console.Write("Введите размер массива (100/1000/10000) - ");
n = Convert.ToInt32(Console.ReadLine());
int[,] matr = new int[n, n]; //Массив

// Инициализация случайными значениями в диапазоне 0 - n
Random rnd = new Random();
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++) matr[i, j] = rnd.Next(n) + 1;

//Ищем сумму делителей чисел
for (i = 0; i < n; i++)
{
    Console.WriteLine("Шаг - " + i);
    for (j = 0; j < n; j++)
    {
        kol = 0;
        for (k = 1; k <= matr[i, j]; k++)
            if (matr[i, j] % k == 0) kol++;
        if (i == j) sum = sum + kol;
    }
}
Console.WriteLine("Сумма делителей = " + sum);
Console.ReadKey();//Пауза
```

Определим сложность работы алгоритма?

Код программы вариант №2

```
 //Ищем сумму делителей чисел
for (i = 0; i < n; i++)
{
    Console.WriteLine("Шаг - " + i);
        kol = 0;
        for (k = 1; k <= matr[i, i]; k++)
            if (matr[i,i] % k == 0) kol++;
        sum = sum + kol;
}
```

Определим сложность работы алгоритма?

Рисунок 2 - Пример кода вариант №2

Заключение

В ходе запуска обеих программ и подсчета времени их выполнения, можно сделать вывод, что код из 2 примера более оптимизирован, чем из 1, потому что второй использует один цикл инициализации массива и поиска суммы элементов данного массива, т.е. второй алгоритм выполняет меньше операций, чем первый. 1 код можно также оптимизировать с помощью использования только 1 цикла инициализации массива и поиска суммы элементов данного массива.