Практическое занятие № 21

Организация функций. Применение рекурсивных функций.

Цель: Получение практических навыков по организации функций при построении алгоритмов; решение задач с использованием рекурсивных алгоритмов.

Норма времени: 2 часа.

Оборудование: Компьютер, среда программирования Visual Studio.

Порядок выполнения работы

Рекурсия - это одна из фундаментальных концепций в математике и программировании и является мощным средством, позволяющим строить элегантные и выразительные алгоритмы. ***Рекурсия*** — это такой способ организации вспомогательного алгоритма (подпрограммы), при котором эта подпрограмма (процедура или функция) в ходе выполнения ее операторов обращается сама к себе. Если процедура **p** содержит явное обращение к самой себе, то она называется явно рекурсивной. Если процедура **p** содержит обращение к некоторой процедуре **q**, которая в свою очередь содержит прямое или косвенное обращение к **p**, то **p** - называется косвенно рекурсивной.

Но рекурсивная программа не может вызывать себя бесконечно, иначе она никогда не остановится, таким образом в программе (функции) должен присутствовать еще один важный элемент - так называемое терминальное (граничное) условие, то есть условие при котором программа прекращает рекурсивный процесс.

Рекурсивная функция Фибоначчи

Другим распространенным показательным примером рекурсивной функции служит функция, вычисляющая числа Фибоначчи. n-й член последовательности Фибоначчи определяется по формуле: f(n)=f(n-1) + f(n-2), причем f(0)=0, а f(1)=1. То есть последовательность Фибоначчи будет выглядеть так 0 (0-й член), 1 (1-й член), 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ... Для определения чисел этой последовательности определим следующий метод:

```
int Fibonachi(int n)
{
    if (n == 0 || n == 1) return n;

    return Fibonachi(n - 1) + Fibonachi(n - 2);
}

int fib4 = Fibonachi(4);
int fib5 = Fibonachi(5);
int fib6 = Fibonachi(6);

Console.WriteLine($"4 число Фибоначчи = {fib4}");
Console.WriteLine($"5 число Фибоначчи = {fib5}");
Console.WriteLine($"6 число Фибоначчи = {fib6}");
```

Здесь базовый вариант выглядит следующий образом:

```
if (n == 0 || n == 1) return n;
```

То есть, если мы ищем нулевой или первый элемент последовательности, то возвращается это же число - 0 или 1. Иначе возвращается результат выражения `Fibonachi(n - 1) + Fibonachi(n - 2);`

Рекурсии и циклы

Это простейшие пример рекурсивных функций, которые призваны дать понимание работы рекурсии. В то же время для обоих функций вместо рекурсий можно использовать

циклические конструкции. И, как правило, альтернативы на основе циклов работают быстрее и более эффективны, чем рекурсия. Например, вычисление чисел Фибоначчи с помощью циклов:

```
    static int Fibonachi2(int n)
    {
        int result = 0;
        int b = 1;
        int tmp;

        for (int i = 0; i < n; i++)
        {
            tmp = result;
            result = b;
            b += tmp;
        }

        return result;
    }
```

В то же время в некоторых ситуациях рекурсия предоставляет элегантное решение, например, при обходе различных древовидных представлений, к примеру, дерева каталогов и файлов.

Пример 1: Написать программу для вывода n первых членов арифметической прогрессии 1, 2, 3… с использованием рекурсивного метода.

```
using System;

class Program
{
    static uint ArithmeticProgression(uint n)
    {
        if (n == 0)
        {
            return 1;
        }
        return ArithmeticProgression(n - 1) + 1;
    }

    static void Main(string[] args)
    {
        Console.Write("N = ");
        var n = Convert.ToUInt32(Console.ReadLine());
        for (uint i = 0; i < n; i++)
        {
            Console.WriteLine(ArithmeticProgression(i));
        }

        Console.ReadLine();
    }
}
```

Пример 2: Напишите программу для поиска индекса максимального элемента массива с использованием рекурсии.

```
using System;

class Program
{
    static int IndexOfMax(int[] array, int len)
    {
```

```
        if (len == 0)
        {
            return len;
        }

        var i = IndexOfMax(array, len - 1);
        return array[len] > array[i] ? len : i;
    }

    static void Main(string[] args)
    {
        var a = new[] { 0, 5, 6, 1, 9, 7, 8, 3, 2, 4 };
        Console.WriteLine("Индекс максимального элемента массива: " +
IndexOfMax(a, a.Length - 1));
        Console.ReadLine();
    }
}
```

Задание 1. Напишите программу для получения суммы n первых членов арифметической прогрессии. Разность прогрессии d задается в качестве параметра.

Задание 2. Напишите программу для переворота строки с использованием рекурсии. Строка “abc” переворачивается в “cba”.

Контрольные вопросы

1. Что такое рекурсия?
2. Приведите примеры рекурсивных процессов из жизни.
3. На чем основан рекурсивный метод программирования?
4. В чем заключается суть рекурсивного процесса?
5. Перечислите основные достоинства и недостатки рекурсивных определений.