**Лабораторная работа № 34-35**

**«Программная реализация рекурсивных алгоритмов и их создание»**

**Цель работы:** получение навыков составления и отладки программ с рекурсивных алгоритмов в С#.

**Задания для лабораторной работы:**

### 3: Функция Аккермана

В теории вычислимости важную роль играет функция Аккермана A(m,n), определенная следующим образом:

A(m,n)=⎧⎩⎨n+1A(m−1,1)A(m−1,A(m,n−1))m=0m>0,n=0m>0,n>0

Даны два целых неотрицательных числа m и n, каждое в отдельной строке.

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введите значение m:");

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

Console.WriteLine("Введите значение n:");

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

long result = Ackermann(m, n);

Console.WriteLine($"A({m}, {n}) = {result}");

}

static long Ackermann(int m, int n)

{

if (m == 0)

return n + 1;

else if (m == 1)

return n + 2;

else if (m == 2)

return 2 \* n + 3;

else if (m == 3)

return (1 << (n + 3)) - 3;

else

throw new ArgumentOutOfRangeException("Значение m слишком большое");

}

}

### 

Контрольные вопросы:

1. **Какое определение называется рекурсивным? Приведите собственные примеры рекурсивных определений.**

В C# допускается, чтобы метод вызывал самого себя. Этот процесс называется ***рекурсией***, а метод, вызывающий самого себя, — рекурсивным.

1. **Какой вспомогательный алгоритм (подпрограмма) называются рекурсивными? Приведите собственные примеры содержательных задач, где для решения может быть использован рекурсивный вспомогательный алгоритм.**

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введите номер элемента Фибоначчи:");

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

if (n < 0)

{

Console.WriteLine("Номер элемента не может быть отрицательным.");

}

else

{

long result = Fibonacci(n);

Console.WriteLine($"Число Фибоначчи для {n} = {result}");

}

}

static long Fibonacci(int n)

{

if (n == 0) // базовый случай

return 0;

if (n == 1) // базовый случай

return 1;

return Fibonacci(n - 1) + Fibonacci(n - 2);

}

}

**3.Что следует непременно указать в соответствующем месте  при написании рекурсивных методов, чтобы организовать возврат из метода без рекурсии.**

При написании рекурсивных методов следует непременно указать в соответствующем месте условный оператор, например if, чтобы организовать возврат из метода без рекурсии.

**4. Главное преимущество рекурсии?**

Главное преимущество рекурсии заключается в том, что она позволяет

реализовать некоторые алгоритмы яснее и проще, чем итерационным способом.

Вывод: получил навыки составления и отладки программ с рекурсивных алгоритмов в С#.