**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**
**РОССИЙСКОЙ ФЕДЕРАЦИИ**
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ**
**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**
**ВЫСШЕГО ОБРАЗОВАНИЯ**
**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**
**ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

# Отчет по лабораторной работе №12 по дисциплине:

# основы программной инженерии

Выполнила:

студент группы ПИЖ-б-о-20-1

Лазарева Дарья Олеговна

Проверил:

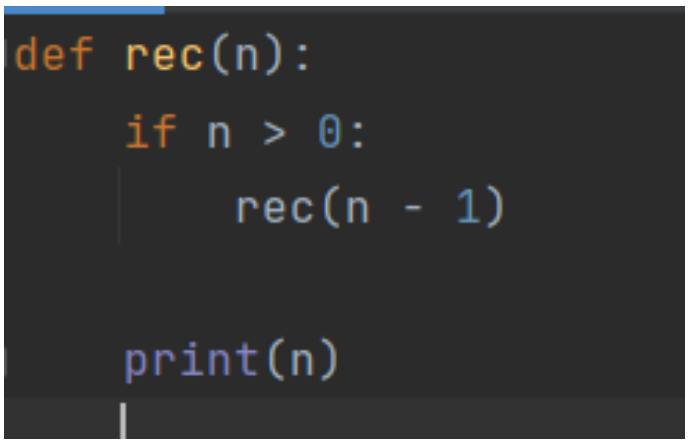
доцент кафедры инфокоммуникаций

Романкин Р.А.

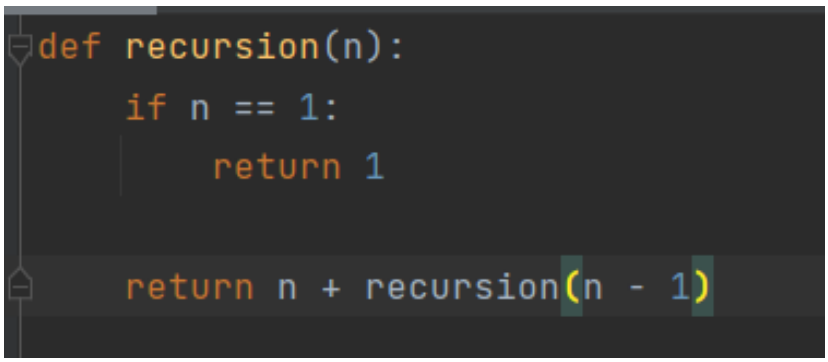
Ставрополь, 2021 г.

Ход работы:

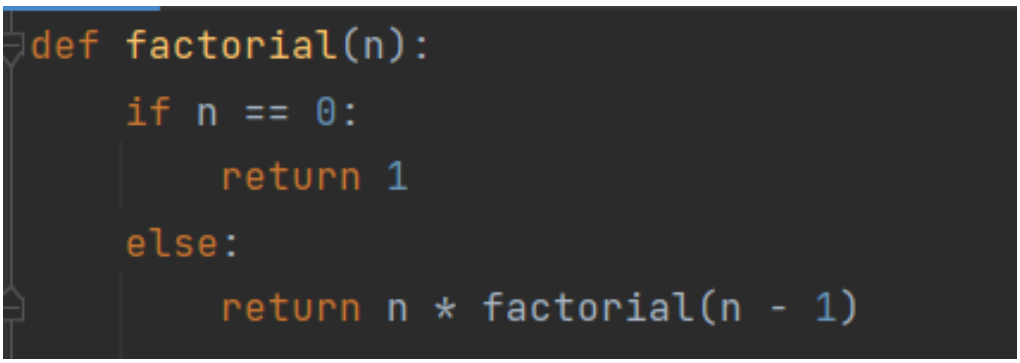
1. Пример рекурсивной функции




2. Выполнение примера №1




3. Использование факториала



4. Выполнение базового случая

```python
def factorial(n):
    if n == 0:
        return 1
    elif n == 1:
        return 1
    else:
        return n * factorial(n - 1)
```

5. «Параллельный» рекурсивный вызов функции

```python
def fib(n):
    if n == 0 or n == 1:
        return n
    else:
        return fib(n - 2) + fib(n - 1)
```

6. Вариант обхода ограничения TCO

```python
def factorial(n):
    product = 1
    while n > 1:
        product *= n
        n -= 1
    return product

def fib(n):
    a, b = 0, 1
    while n > 0:
        a, b = b, a + b
        n -= 1
    return a
```

7. Использование декоратора lru_cache

```python
from functools import lru_cache
. . .
@lru_cache
def fib(n):
    if n == 0 or n == 1:
        return n
    else:
        return fib(n - 2) + fib(n - 1)
```

8. Пример линейной рекурсии

```python
def fib(n):
    if n <= 1:
        return (n, 0)
    else:
        (a, b) = fib(n - 1)
        return (a + b, a)
```

9. Увеличение максимальной глубины рекурсии

```python
def cursing(depth):
    try:
        cursing(depth + 1)
    except RuntimeError as RE:
        print('I recursed {} times!'.format(depth))


if __name__ == '__main__':
    cursing(0)
```

```
modul
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/12/modul.py
I recursed 998 times!
```

Выполнение индивидуального задания (вариант 13)

Напишите программу вычисления функции Аккермана для всех неотрицательных целых аргументов m и n:

$$A(m,n) = \begin{cases} A(0,n) = n+1 \\ A(m,0) = A(m-1,1), & m \\ A(m,n) = A(m-1,A(m,n-1)), & m,n > 0. \end{cases}$$

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys
sys.setrecursionlimit(5000)

m = int(input("Введите значение m: "))
n = int(input("Введите значение n: "))

def A(m, n):
    if m == 0:
        return n + 1
    if m != 0 and n == 0:
        return A(m - 1, 1)
    if m > 0 and n > 0:
        return A(m - 1, A(m, n - 1))

print(A(m, n))
```

```
Введите значение m: 3
Введите значение n: 1
13
```

```
Введите значение m: 2
Введите значение n: 8
19
```

```
 A  | n=0    1    2    3    4    5    6    7    8    9   10
----+-------------------------------------------------------
m=0 |   1    2    3    4    5    6    7    8    9   10   11
  1 |   2    3    4    5    6    7    8    9   10   11   12
  2 |   3    5    7    9   11   13   15   17   19   21   23
  3 |   5   13   29   61  125  253  509 1021 2045 4093 8189
  4 |  13 65533  big  really big...
```

Контрольные вопросы:

1. Для чего нужна рекурсия? Рекурсия подразумевает более компактный вид записи выражения. Обычно это зависимость процедур (функций, членов прогресс и т.д.) соседних порядковых номеров. Некоторые зависимости очень сложно выразить какой- либо формулой, кроме как рекурсивной. Рекурсия незаменима в ряде случаев при программировании замкнутых циклов

2. Что называется базой рекурсии? База рекурсии – это такие аргументы функции, которые делают задачу настолько простой, что решение не требует дальнейших вложенных вызовов.

3. Самостоятельно изучите что является стеком программы. Как используется стек программы при вызове функций? Стек вызовов – в теории вычислительных систем, LIFO-стек, хранящий информацию для возврата управления из подпрограмм в программу и для возврата в программу из обработчика прерывания.

При вызове подпрограммы или возникновении прерываний, в стек заносится адрес возврата – адрес в памяти следующей инструкции приостановленной программы и управление передаётся подпрограмме или подпрограмме обработчику.

4. Как получить текущее значение максимальной глубины рекурсии в языке Python? Чтобы проверить текущие параметры лимита нужно запустить: sys.getrecursionlimit()

5. Что произойдет если число рекурсивных вызовов превысит максимальную глубину рекурсии в языке Python? Программа выдаст ошибку: RuntimeError: Maximum Recursion Depth Exceeded

6. Как изменить максимальную глубину рекурсии в языке Python? Изменить максимальную глубины рекурсии можно с помощью sys.setrecursionlimit(limit). Чтобы проверить параметры лимита, нужно запустить sys.getrecursionlimit().

7. Каково назначение декоратора lru_cache ? Декоратор можно использовать для уменьшения количества лишних вычислений.

8. Что такое хвостовая рекурсия? Как проводится оптимизация хвостовых вызовов? Хвостовая рекурсия – частный случай рекурсии, при котором любой рекурсивный вызов является последней операцией перед возвратом из функции. Подобный вид рекурсии примечателен тем, что может быть легко заменён на итерацию путём формальной и гарантированно корректной перестройки кода функции.

Оптимизация хвостовой рекурсии путём преобразования её в плоскую итерацию реализована во многих оптимизирующих компиляторах. В некоторых функциональных языках программирования спецификация гарантирует обязательную оптимизацию хвостовой рекурсии.