МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»
КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Старший преподаватель		С.А.Рогачёв
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

Элементы теории рекурсивных функций

по дисциплине: Теория вычислительных процессов

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР.	4134к		Столяров Н.С.
		подпись, дата	инициалы, фамилия

Санкт-Петербург
2024

Цель работы:

Целью данной работы является изучение и применение принципов рекурсии в программировании. Задача заключается в преобразовании заданной арифметической функции в рекурсивную форму с использованием операторов примитивной рекурсии. В рамках работы также предполагается создание программы на языке высокого уровня, включающей две функции: одна для вычисления функции итеративным способом, другая — рекурсивным. Это позволит сравнить и проанализировать различия в подходах к решению задач с использованием рекурсии и итерации, а также оценить эффективность и применимость каждого из методов.

Основные сведенья из теории:

Рекурсия (от лат. recurso – бегу назад, возвращаюсь) — это метод задания вычислимой функции, при котором каждое значение функции определяется через значения этой (или другой) вычислимой функции для меньших (или ранее определённых) значений аргументов. Функции, задаваемые таким образом, называются рекурсивными. Рекурсивные функции широко используются в программировании для решения задач, которые могут быть разделены на подзадачи того же типа.

Наиболее известным примером рекурсивной функции является ряд Фибоначчи, в котором каждое число является суммой двух предыдущих чисел. Ещё одним классическим примером является вычисление факториала числа, где значение факториала n определяется как произведение всех натуральных чисел от 1 до n.

Рекурсия позволяет существенно упростить код и логику задачи, особенно в случаях, когда решение задачи может быть представлено в виде повторяющейся подзадачи. Однако важно учитывать, что неправильное использование рекурсии может привести к увеличению времени выполнения программы и превышению предела глубины рекурсии.

Постановка задачи:

1) Преобразовать заданную арифметическую функцию в рекурсивную форму, используя операторы примитивной рекурсии. Примитивная рекурсия предполагает использование базового случая, который завершает рекурсивные вызовы, и рекурсивного случая, который определяет, как текущее значение функции вычисляется на основе значений функции для меньших аргументов.

2) Разработать программу на языке высокого уровня, содержащую две функции:

- Первая функция должна вычислять заданную арифметическую функцию итеративным способом, используя циклы и другие итеративные конструкции.
- Вторая функция должна вычислять ту же арифметическую функцию рекурсивным способом, применяя подходы рекурсии.

3) Программа должна предусматривать возможность ввода аргументов арифметической функции пользователем во время выполнения программы, что обеспечит гибкость и универсальность решения.

4) Провести тестирование обеих функций и сравнить результаты их работы, чтобы выявить различия в поведении и эффективности рекурсивного и итеративного подходов.

5) На основе выполненной работы сделать выводы о преимуществах и недостатках использования рекурсии и итерации в программировании, а также о применимости каждого подхода в различных задачах.

Преобразование арифметической функции в рекурсивную с использованием операторов примитивной рекурсии:

Заданная арифметическая функция: f(x,y)=x−y+1, где x и y — целые числа. Для этой функции требуется реализовать вычисление как итеративным, так и рекурсивным способом. Преобразование функции в рекурсивную форму предполагает использование примитивной рекурсии, при которой каждый вызов функции основывается на значении функции для более простых (меньших) аргументов.

Итеративный подход:

В итеративной реализации функции используется цикл для последовательного уменьшения значения y, с одновременным изменением результата. Начальное значение результата задаётся равным x+1. На каждой итерации y уменьшается на 1, и результат также корректируется до тех пор, пока y>0. Пример реализации итеративной функции:

```
def iterative_function(x, y):
    result = x
    while y > 0:
        result -= 1
        y -= 1
    return result + 1
```

Рекурсивный подход:

Для рекурсивного преобразования функции используются операторы примитивной рекурсии: базовый случай и рекурсивный вызов. В базовом случае, когда y=0, функция возвращает x + 1. В рекурсивном случае функция вызывает саму себя, постепенно уменьшая y на единицу и вычитая 1 из результата на каждом уровне рекурсии. Это соответствует примитивной рекурсии, где каждый вызов основывается на значении функции для меньших аргументов y−1.

Функция f(x,y)=x – y + 1 может быть представлена с помощью примитивной рекурсии. Примитивная рекурсия основывается на двух основных компонентах: базовом случае и рекурсивном вызове.

Базовый случай и рекурсивный случай можно описать следующим образом:

- **Базовый случай:** Если y=0, возвращается x + 1.
- **Рекурсивный случай:** Если y>0, функция вызывает саму себя с аргументами (x,y−1) и уменьшает результат на 1.

Обоснование вычислимости:
Функция f(x,y)=x−y+1 реализуется с использованием **примитивной рекурсии**, что гарантирует её вычислимость. Это происходит по следующим причинам:

1. **Базовый случай**: Для y=0 результат вычисляется за конечное время.

2. **Рекурсивный случай**: С каждым вызовом значение y уменьшается на единицу, что приводит к конечному числу рекурсивных вызовов.

Так как функция определяется с использованием примитивной рекурсии и её реализация возможна в конечное время, она является вычислимой.

Согласно теории вычислимости, функция f(x,y), определяемая с помощью примитивной рекурсии, вычисляется за конечное время, так как:

- Базовая функция g(x)=x+1g(x) = x + 1g(x)=x+1 является вычислимой.
- Рекурсивная функция h(x,y,z)=z−1, также вычислима.

Таким образом, f(x,y)=x−y+1 является вычислимой функцией.

Разложение арифметических функций с помощью примитивной рекурсии

Арифметические функции могут быть выражены через примитивную рекурсию следующим образом:

1. **Функции базовых операций**:
   - **Константная функция**: f(x)=c, где c — константа. Это базовая функция, не зависящая от x.
   - **Функция сложения**: f(x,y)=x+y. Эта функция может быть реализована с помощью примитивной рекурсии: f(x,0)=x; f(x,y+1)=f(x,y)+1
   - **Функция умножения**: f(x,y)=x·y. Эта функция может быть определена как: f(x,0)=0; f(x,y+1)=f(x,y)+x
2. **Обобщение на произвольные арифметические функции**:

Рассмотрим функцию f(x,y)=3x−y. Эта функция может быть выражена через примитивную рекурсию следующим образом:

- **Базовый случай**: Когда y=0, функция возвращает 3x: f(x,0)=3x
- **Рекурсивный случай**: Когда y>0, функция вызывает саму себя с аргументами (x,y−1) и вычитает 1: f(x,y)=f(x,y−1)−1

В данном случае, функция f(x,y) определяется через базовый случай и рекурсивное определение, что обеспечивает её вычислимость, так как каждый рекурсивный шаг уменьшает значение y, гарантируя конечное число шагов.

Обоснование вычислимости

Функции, определяемые с помощью примитивной рекурсии, являются вычислимыми, так как:

- **Базовый случай** всегда вычисляется за конечное время. В данном случае, при y=0, результат вычисляется мгновенно как x+1.
- **Рекурсивный случай** приводит к конечному числу шагов благодаря уменьшению значения аргумента на каждом уровне рекурсии.

Таким образом, любой арифметический функционал, который может быть представлен в виде примитивной рекурсии, также является вычислимым.

Код рекурсивной функции выглядит следующим образом:

```
def recursive_function(x, y):
```

```
    if y == 0:
        return x + 1
    else:
        return recursive_function(x, y - 1) - 1
```

Обоснование выбора примитивной рекурсии:

Использование примитивной рекурсии для данной задачи обосновано тем, что вычисление текущего значения зависит только от предыдущего значения, что соответствует принципам примитивной рекурсии. Базовый случай определяет конечное условие рекурсии, предотвращая бесконечный цикл вызовов. Примитивная рекурсия подходит для таких арифметических преобразований, так как она обеспечивает понятный и логически корректный способ перехода от одного состояния функции к следующему, уменьшая сложность задачи на каждом шаге.

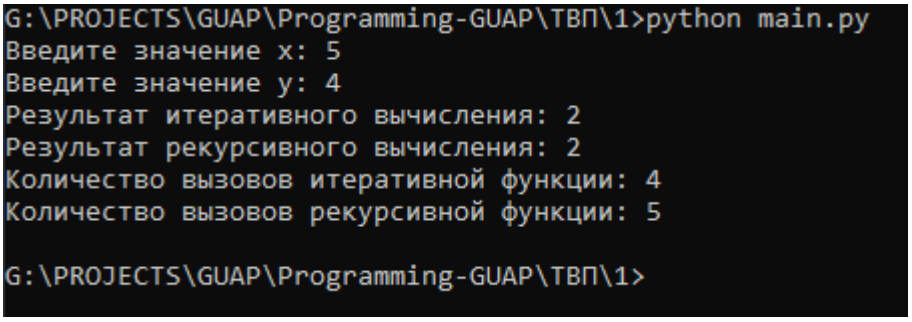
Таким образом, данное преобразование функции в рекурсивную форму позволяет последовательно вычитать единицу, реализуя необходимую арифметическую операцию через рекурсивные вызовы.

Листинг программы на языке высокого уровня с комментариями:

```
# Глобальные переменные для подсчета числа вызовов функций
iter_count = 0
rec_count = 0


# Итеративная функция для вычисления x - y + 1
def iterative_function(x, y):
    global iter_count
    result = x
    for _ in range(y):
        iter_count += 1
        result -= 1
    return result + 1


# Рекурсивная функция для вычисления x - y + 1
def recursive_function(x, y):
    global rec_count
    rec_count += 1
    if y == 0:
        return x + 1
    else:  # Рекурсивный случай
        return recursive_function(x, y - 1) - 1


x = int(input("Введите значение x: "))
y = int(input("Введите значение y: "))


# Вычисление итеративным способом
iterative_result = iterative_function(x, y)
print(f"Результат итеративного вычисления: {iterative_result}")


# Вычисление рекурсивным способом
recursive_result = recursive_function(x, y)
print(f"Результат рекурсивного вычисления: {recursive_result}")


# Вывод количества вызовов функций
print(f"Количество вызовов итеративной функции: {iter_count}")
```

```
print(f"Количество вызовов рекурсивной функции: {rec_count}")
```

Пример результата выполнения:



Вывод:

В результате выполнения лабораторной работы были реализованы две функции для вычисления арифметической операции f(x,y)=x−y+1 итеративным и рекурсивным способами. Оба подхода продемонстрировали правильность выполнения вычислений, выдавая одинаковый результат для заданных входных данных.

Сравнение результатов: В обоих случаях (итеративный и рекурсивный подходы) функция правильно вычислила значение 5-4+1=2. Это подтверждает корректность обеих реализаций.

Итеративная функция была вызвана 4 раз, что соответствует количеству итераций, необходимых для выполнения вычитания 1 из результата на каждом шаге. Рекурсивная функция была вызвана 5 раз, включая 1 базовый вызов и 4 рекурсивных вызовов, что подтверждает правильность работы рекурсивного алгоритма.

Примитивная рекурсия подходит для задач, которые могут быть разбиты на более мелкие аналогичные подзадачи. Однако следует учитывать ограничения и возможности рекурсивного подхода, особенно в задачах с высокой вычислительной сложностью или глубокими уровнями рекурсии.

В реальных сценариях итеративные алгоритмы зачастую предпочтительнее для задач с большими объемами данных, так как они эффективнее используют память.