Федеральное государственное автономное образовательное учреждение

высшего образования «Санкт-Петербургский государственный университет аэрокосмического приборостроения» (ГУАП)

ОТЧЕТ ЗАЩИЩЕН

Ст. преподаватель Рогачев С.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Должность Ф.И.О дата, подпись

**Отчет**

**о лабораторной работе №1**

**«Рекурсия»**

по дисциплине «Теория вычислительных процессов»

ОТЧЕТ ВЫПОЛНИЛ:

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ф.И.О. группа дата, подпись

Санкт-Петербург

2024

**Цель работы:**

Целью данной работы является преобразование заданной арифметической функции в рекурсивную с использованием примитивной рекурсии, а также реализация программы, вычисляющей эту функцию как итеративным, так и рекурсивным способами.

**Основные сведения из теории:**

Рекурсия (от лат. recurso – бегу назад, возвращаюсь) – есть такой способ задания вычислимой функции, когда каждое ее значение задается через значение этой (или другой) вычислимой функции для меньших (или ранее определенных) значений аргументов, а функции задаваемые таким образом называются рекурсивными. Наиболее ярким примером является ряд Фибоначчи.

**Постановка задачи:**

Необходимо преобразовать заданную арифметическую функцию в рекурсивную, используя операторы примитивной рекурсии.

Во второй части лабораторной работы требуется создать программу на языке высокого уровня содержащую две функции. Первая функция должна вычислять заданную арифметическую функцию итеративным способом. Вторая функция должна вычислять заданную арифметическую функцию рекурсивным способом.

Аргументы арифметической функции должны задаваться пользователем в процессе выполнения программы.

**Вариант:**



**Преобразование функции 2x1 + x2 в рекурсивную с использованием примитивной рекурсии:**

Функция f(x1, x2) = 2x1 + x2 может быть преобразована в рекурсивную с использованием примитивной рекурсии.   
В процессе преобразования мы разложим функцию на несколько простых шагов, каждый из которых будет выражен через рекурсивные функции.

**1. Разложение функции на простые шаги**

Функция f(x1, x2) = 2x1 + x2 состоит из двух частей:

1. Умножение: g(x1) = 2x1, где удвоение числа можно свести к последовательному сложению.

2. Сложение: h(z, x2) = z + x2, где к результату удвоения добавляется x2.

Таким образом, полная функция f(x1, x2) = h(g(x1), x2).

**2. Примитивно-рекурсивное определение удвоения**

Удвоение числа x1 можно определить через примитивную рекурсию с использованием сложения:

- Основание рекурсии: g(0) = 0

- Рекурсивный шаг: g(x1 + 1) = g(x1) + 2

Это определение означает, что для каждого увеличения x1 на единицу, к результату добавляется 2.

**3. Примитивно-рекурсивное определение сложения**

Сложение z + x2 можно определить через примитивную рекурсию следующим образом:

- Основание рекурсии: h(z, 0) = z

- Рекурсивный шаг: h(z, x2 + 1) = h(z, x2) +

На каждом шаге к результату прибавляется 1, пока не достигнем значения x2.

**4. Полная функция**

Теперь полную функцию f(x1, x2) = 2x1 + x2 можно выразить через комбинацию двух примитивно-рекурсивных функций:

1. g(x1) = 2x1 — примитивно-рекурсивная функция удвоения.

2. h(z, x2) = z + x2 — примитивно-рекурсивная функция сложения.

Итоговое определение:

f(x1, x2) = h(g(x1), x2)

где g(x1) — рекурсивная функция удвоения, а h(z, x2) — рекурсивная функция сложения.

**Вывод**

Таким образом, функция f(x1, x2) = 2x1 + x2 является примитивно-рекурсивной. Она строится на основе примитивно-рекурсивных функций удвоения и сложения, что подтверждает её принадлежность к классу рекурсивных функций.

**Вторая часть:**

Эта программа вычисляет выражение 2 \* x1 + x2 двумя методами: итеративным и рекурсивным. Программа позволяет пользователю ввести два числа (x1 и x2) и затем вычисляет результат с помощью двух различных методов, проверяя их результаты.

**1. Итеративная версия функции 2 \* x1 + x2**

Функция итеративного вычисления использует цикл для умножения x1 на 2 с помощью повторяющегося сложения. После этого добавляется значение x2.

Код функции:

|  |
| --- |
| def iterative\_function(x1, x2):  product = 0  for \_ in range(2): # Умножаем x1 на 2 как повторение сложения  product += x1  result = product + x2 # Добавляем x2  return result |

**2. Рекурсивная версия функции 2 \* x1 + x2**

Рекурсивная версия функции использует рекурсию для умножения x1 на 2. Рекурсивная функция вызывает себя дважды, уменьшая количество оставшихся операций, пока не достигнет базового случая.

Код функции:

|  |
| --- |
| def recursive\_function(x1, x2):  # Рекурсивная версия умножения x1 на 2  def recursive\_double(x1, times=2):  if times == 0:  return 0  return x1 + recursive\_double(x1, times - 1)    product = recursive\_double(x1) # Вычисляем 2 \* x1  return product + x2 # Добавляем x2 |

3. Функция для безопасного ввода значений

Функция input\_integer обеспечивает безопасный ввод целых чисел. Она продолжает запрашивать ввод до тех пор, пока пользователь не введет корректное число.

Код функции:

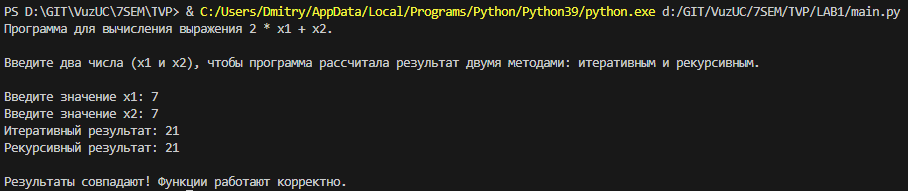
|  |
| --- |
| def input\_integer(prompt):  while True:  try:  value = int(input(prompt))  return value  except ValueError:  print("Ошибка: пожалуйста, введите корректное число.") |

4. Главная программа

Главная программа управляет процессом ввода и вывода. Она запрашивает у пользователя значения x1 и x2, вычисляет результат обоими методами и выводит их, а также проверяет их совпадение.

Код главной программы:

|  |
| --- |
| def main():  print("Программа для вычисления выражения 2 \* x1 + x2.  ")  print("Введите два числа (x1 и x2), чтобы программа рассчитала результат двумя методами: итеративным и рекурсивным.  ")    # Ввод значений x1 и x2  x1 = input\_integer("Введите значение x1: ")  x2 = input\_integer("Введите значение x2: ")  # Вычисление итеративным способом  result\_iterative = iterative\_function(x1, x2)  print(f"Итеративный результат: {result\_iterative}")  # Вычисление рекурсивным способом  result\_recursive = recursive\_function(x1, x2)  print(f"Рекурсивный результат: {result\_recursive}")  # Проверка совпадения результатов  if result\_iterative == result\_recursive:  print("  Результаты совпадают! Функции работают корректно.")  else:  print("  Результаты не совпадают! Проверьте реализацию функций.") |



Программа иллюстрирует два подхода к вычислению одного и того же выражения: итеративный и рекурсивный методы. Она демонстрирует использование цикла и рекурсии для решения математической задачи и обеспечивает безопасный ввод данных.

Листинг:

|  |
| --- |
| # Итеративная версия функции 2 \* x1 + x2  def iterative\_function(x1, x2):      product = 0      for \_ in range(2):  # Умножаем x1 на 2 как повторение сложения          product += x1      result = product + x2  # Добавляем x2      return result  # Рекурсивная версия функции 2 \* x1 + x2  def recursive\_function(x1, x2):      # Рекурсивная версия умножения x1 на 2      def recursive\_double(x1, times=2):          if times == 0:              return 0          return x1 + recursive\_double(x1, times - 1)        product = recursive\_double(x1)  # Вычисляем 2 \* x1      return product + x2  # Добавляем x2  # Функция для безопасного ввода значений  def input\_integer(prompt):      while True:          try:              value = int(input(prompt))              return value          except ValueError:              print("Ошибка: пожалуйста, введите корректное число.")  # Главная программа  def main():      print("Программа для вычисления выражения 2 \* x1 + x2.\n")      print("Введите два числа (x1 и x2), чтобы программа рассчитала результат двумя методами: итеративным и рекурсивным.\n")        # Ввод значений x1 и x2      x1 = input\_integer("Введите значение x1: ")      x2 = input\_integer("Введите значение x2: ")      # Вычисление итеративным способом      result\_iterative = iterative\_function(x1, x2)      print(f"Итеративный результат: {result\_iterative}")      # Вычисление рекурсивным способом      result\_recursive = recursive\_function(x1, x2)      print(f"Рекурсивный результат: {result\_recursive}")      # Проверка совпадения результатов      if result\_iterative == result\_recursive:          print("\nРезультаты совпадают! Функции работают корректно.")      else:          print("\nРезультаты не совпадают! Проверьте реализацию функций.")  # Запуск программы  if \_\_name\_\_ == "\_\_main\_\_":      main() |

Вывод:

Выполнив лабораторную работу, я научился преобразованию заданной арифметической функции в рекурсивную с использованием примитивной рекурсии, а также реализовал программу, вычисляющую эту функцию как итеративным, так и рекурсивным способами.