ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дожность |  |  |  | [Рогачев С.A](https://pro.guap.ru/inside/profile/911) |
| [старший преподаватель](https://guap.ru/rasp/?p=317) |  | подпись, дата |  | инициалы, фамилия |

|  |  |
| --- | --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1 |  |
| [Рекурсия](https://pro.guap.ru/inside/student/tasks/168447) |  |
| по дисциплине: [Теория вычислительных процессов](https://pro.guap.ru/inside/students/subjects/3391377) |  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4236 |  |  |  | Л. Мвале |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург

2025

1. **Цель работы:** Освоить понятие рекурсивных функций, научиться преобразовывать арифметические функции в рекурсивную форму с использованием примитивной рекурсии. Реализовать функцию умножения итеративным и рекурсивным способом на языке высокого уровня
2. **Основные сведения из теории**

Рекурсия – это метод определения функций, при котором значение функции в текущий момент выражается через её значение на предыдущих шагах (для меньших аргументов). Иными словами, функция вызывает сама себя до тех пор, пока не достигнет базового случая.

Примитивная рекурсия – это особый вид рекурсии, где функция определяется через:

- базис (значение функции для минимального аргумента),

- рекурсивное правило (вычисление для большего аргумента через результат для меньшего).

Примитивная рекурсия всегда заканчивается, потому что каждый рекурсивный шаг уменьшает аргумент и в итоге приводит к базовому случаю.

Примеры рекурсивных функций:

Факториал:

* n! = 1, если n = 0
* n! = n × (n – 1)!, если n > 0

Последовательность Фибоначчи:

* F(0) = 0, F(1) = 1
* F(n) = F(n – 1) + F(n – 2), если n > 1

Умножение через сложение (наш случай):

* x1 \* 0 = 0 (база рекурсии)
* x1 \* x2 = x1 + (x1 \* (x2 – 1)) (рекурсивное определение)

Таким образом, операция умножения может быть сведена к многократному сложению, а её рекурсивное определение соответствует принципу примитивной рекурсии.

1. **Постановка задачи**  
    Необходимо реализовать программу, которая вычисляет произведение двух целых чисел двумя способами: итеративным (с помощью сложения) и рекурсивным (с помощью примитивной рекурсии). Пользователь вводит числа, программа выводит результаты обоими методами и проверяет их корректность.
2. **Преобразование арифметической функции**

Шаг преобразования:

x1 \* 0 = 0 (база рекурсии).

x1 \* x2 = x1 + (x1 \* (x2 - 1)) (рекурсивное определение).

1. **Листинг программы**

def iterative\_multiply(x1, x2):

    """

    Итеративное умножение двух целых чисел (с помощью сложения).

    Обрабатывает отрицательные числа.

    """

    negative = False  # Флаг для определения знака результата

    # Случай, когда оба числа отрицательные -> результат положительный

    if x1 < 0 and x2 < 0:

        x1, x2 = -x1, -x2

    # Случай, когда только первое отрицательное -> результат отрицательный

    elif x1 < 0:

        x1 = -x1

        negative = True

    # Случай, когда только второе отрицательное -> результат отрицательный

    elif x2 < 0:

        x2 = -x2

        negative = True

    result = 0

    # Суммируем x1 ровно x2 раз

    for \_ in range(x2):

        result += x1

    # Возвращаем с учётом знака

    return -result if negative else result

def recursive\_multiply(x1, x2):

    """

    Рекурсивное умножение двух целых чисел.

    Основано на примитивной рекурсии: x1 \* x2 = x1 + (x1 \* (x2 - 1))

    Обрабатывает отрицательные числа.

    """

    # Базовый случай: умножение на ноль всегда даёт ноль

    if x1 == 0 or x2 == 0:

        return 0

    # Обработка отрицательных чисел

    if x1 < 0 and x2 < 0:       # (-a) \* (-b) = a \* b

        return recursive\_multiply(-x1, -x2)

    elif x1 < 0:                # (-a) \* b = -(a \* b)

        return -recursive\_multiply(-x1, x2)

    elif x2 < 0:                # a \* (-b) = -(a \* b)

        return -recursive\_multiply(x1, -x2)

    # Рекурсивный шаг:

    # уменьшаем второй аргумент на 1 и прибавляем x1

    return x1 + recursive\_multiply(x1, x2 - 1)

def main():

    print("=== Лабораторная работа №1: Рекурсивные функции ===")

    print("Задача: Умножение итеративным и рекурсивным способом")

    print("=" \* 55)

    try:

        # Ввод данных от пользователя

        x1 = int(input("Введите первое число (x1): "))

        x2 = int(input("Введите второе число (x2): "))

        # Вычисление результатов

        iterative\_result = iterative\_multiply(x1, x2)

        recursive\_result = recursive\_multiply(x1, x2)

        # Вывод результатов

        print("\nРезультаты:")

        print(f"Итеративное умножение: {x1} \* {x2} = {iterative\_result}")

        print(f"Рекурсивное умножение: {x1} \* {x2} = {recursive\_result}")

        print(f"Встроенное умножение:  {x1} \* {x2} = {x1 \* x2}")

        # Проверка совпадения всех способов

        if iterative\_result == recursive\_result == x1 \* x2:

            print("\n✓ Все методы дали одинаковый результат!")

        else:

            print("\n✗ Результаты не совпали!")

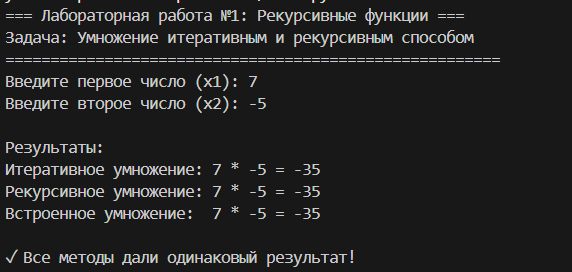
    except ValueError:

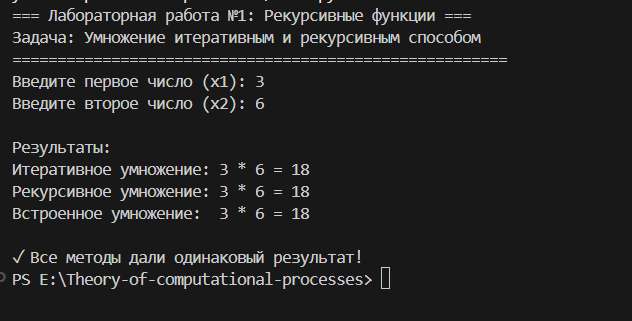
        print("Ошибка: нужно ввести целые числа!")

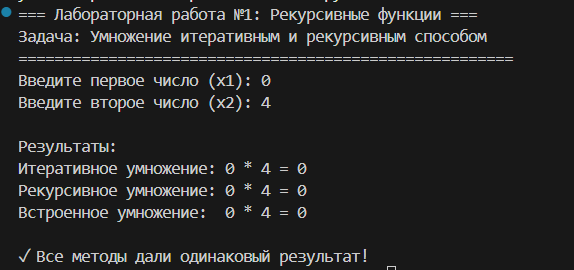
if \_\_name\_\_ == "\_\_main\_\_":

    main()

1. **Пример выполнения**







1. **Выводы**

В ходе работы были изучены рекурсивные функции, реализованы алгоритмы итеративного и рекурсивного умножения. Оба метода дают одинаковые результаты, что подтверждает правильность реализации примитивной рекурсии.