ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

доцент		В. А. Кузнецов
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 3.1

РЕКУРСИВНЫЙ АЛГОРИТМ

по курсу:

ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4326		Г. С. Томчук
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

СОДЕРЖАНИЕ

1 Постановка задачи

Задача: реализовать программную функцию на языке C/C++, выполняющую поставленную задачу. Вариант задания, пример входных и выходных данных представлен в таблице 1.

- Написать код функции, принимающей в качестве аргументов и возвращающей все необходимые параметры, без использования глобальных переменных. Допустимо использование дополнительных функций.
- Протестировать функцию для всех возможных исключительных ситуаций, особое значение придается тестам на возникновение ошибок в ходе работы программы.
- Из наименования функции и принимаемых аргументов должно быть ясно их назначение.
- В ходе тестирования функции при каждом вызове рекурсивной функции необходимо вывести отладочную информацию: порядковый номер вызова рекурсивной функции, значения изменяющегося аргумента и возвращаемого значения, если они присутствуют. Привести глубину рекурсии для каждого тестового примера.

Таблица 1 – Вариант

N	Текст задания	Вход	Выход
7	Реализовать рекурсивную функцию вычисления (ax + b)((a - 1) x + b - 1)...((a - k) x + b - k)...(x + 1) по заданным a,b,x. При условии, что если a-k< 1,то a-k = 1, если b-k<1,то b-k = 1.	2, 3, 1	30

2 Схема алгоритма решения

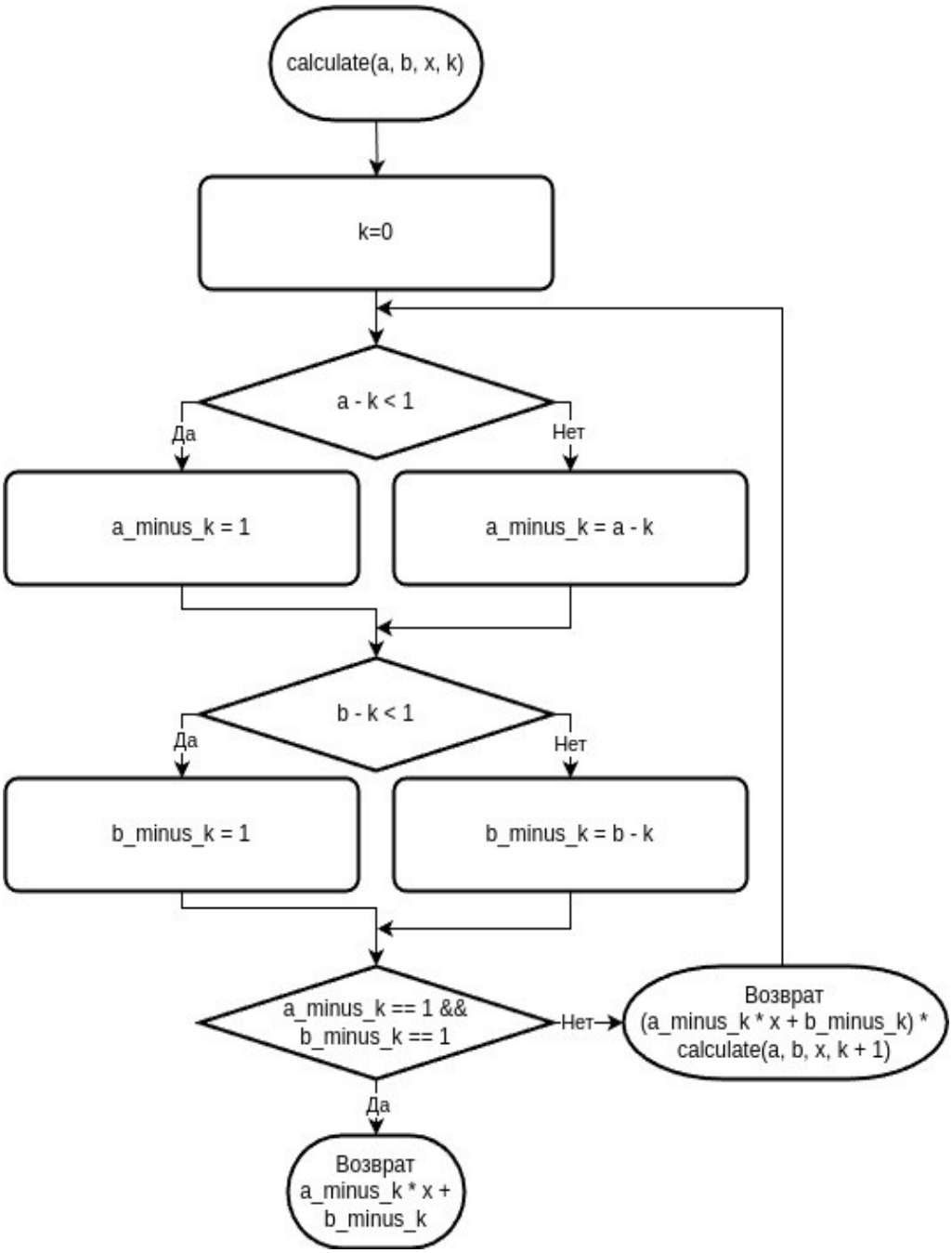

Рисунок 1 – Блок-схема алгоритма

3 Полное описание реализованной функции

Функция calculate_recursion_with_debug рекурсивно вычисляет значение выражения (ax + b)((a - 1) x + b - 1)...((a - k) x + b - k)...(x + 1) с учетом заданных значений a, b, и x. При этом, если значение выражения (a−k) или (b−k) становится меньше 1, то оно заменяется на 1. Функция также выводит отладочную информацию о каждом вызове. Принимает следующие аргументы:

1. int a — начальное значение параметра a.
2. int b — начальное значение параметра b.
3. int x — начальное значение параметра x.
4. int &call_count — ссылка на переменную, отслеживающую количество вызовов функции.
5. int k — текущий шаг рекурсии (по умолчанию 0).

Возвращает int64_t — результат вычисления выражения. Работа функции происходит следующим образом:

1. Увеличиваем call_count и сохраняем его значение в current_call_count.
2. Вычисляем текущие значения (a - k) и (b - k). Если одно из значений меньше 1, заменяем его на 1.
3. Выводим отладочную информацию, включающую текущий номер вызова, значение k, и текущие значения (a - k) и (b - k).
4. Проверяем базовый случай: если (a - k) == 1 и (b - k) == 1, вычисляем результат как 1 * x + 1, выводим отладочную информацию и возвращаем результат.
5. Если базовый случай не достигнут, вычисляем рекурсивно выражение для следующего шага k + 1 и умножаем его на текущее выражение (a - k) * x + (b - k).
6. Выводим отладочную информацию о возвращаемом значении и возвращаем результат.

4 Листинг программы

Листинг 1

```
#include <iostream>

int64_t calculate_recursion_with_debug(int a, int b, int x, int &call_count,
int k = 0) {
    // Увеличиваем номер вызова и сохраняем значение
    int current_call_count = ++call_count;

    // Вычисляем текущие значения (a - k) и (b - k)
    int a_minus_k = (a - k < 1) ? 1 : (a - k);
    int b_minus_k = (b - k < 1) ? 1 : (b - k);

    // Вывод отладочной информации
    std::cout << "Вызов #" << current_call_count << ": k=" << k << ", a-k=" <<
a_minus_k << ", b-k="
              << b_minus_k << std::endl;

    // Базовый случай: прекращаем рекурсию
    if (a_minus_k == 1 && b_minus_k == 1) {
        int64_t result = a_minus_k * x + b_minus_k;
        std::cout << "Возврат из вызова #" << current_call_count << ": " <<
result << std::endl;
        return result;
    }

    // Рекурсивный случай: продолжаем умножение
    int64_t result = (a_minus_k * x + b_minus_k) *
calculate_recursion_with_debug(a, b, x, call_count, k + 1);
    std::cout << "Возврат из вызова #" << current_call_count << ": " << result
<< std::endl;

    return result;
}

int main() {
    int a, b, x;

    std::cout << "a: ";
    std::cin >> a;

    std::cout << "b: ";
    std::cin >> b;

    std::cout << "x: ";
    std::cin >> x;

    int call_count = 0;

    int64_t result = calculate_recursion_with_debug(a, b, x, call_count);
    std::cout << "Результат: " << result << ", глубина: " << call_count;

    return 0;
}
```

5 Результаты тестирования программы и функции

```
a: 2
b: 3
x: 1
Вызов #1: k=0, a-k=2, b-k=3
Вызов #2: k=1, a-k=1, b-k=2
Вызов #3: k=2, a-k=1, b-k=1
Возврат из вызова #3: 2
Возврат из вызова #2: 6
Возврат из вызова #1: 30
Результат: 30, глубина: 3
Process finished with exit code 0
```

Рисунок 2

```
a: 4
b: 3
x: 4
Вызов #1: k=0, a-k=4, b-k=3
Вызов #2: k=1, a-k=3, b-k=2
Вызов #3: k=2, a-k=2, b-k=1
Вызов #4: k=3, a-k=1, b-k=1
Возврат из вызова #4: 5
Возврат из вызова #3: 45
Возврат из вызова #2: 630
Возврат из вызова #1: 11970
Результат: 11970, глубина: 4
Process finished with exit code 0
```

Рисунок 3

```
a: -1000
b: 5
x: 2
Вызов #1: k=0, a-k=1, b-k=5
Вызов #2: k=1, a-k=1, b-k=4
Вызов #3: k=2, a-k=1, b-k=3
Вызов #4: k=3, a-k=1, b-k=2
Вызов #5: k=4, a-k=1, b-k=1
Возврат из вызова #5: 3
Возврат из вызова #4: 12
Возврат из вызова #3: 60
Возврат из вызова #2: 360
Возврат из вызова #1: 2520
Результат: 2520, глубина: 5
Process finished with exit code 0
```

Рисунок 4

```
a: 4
b: 0
x: 3
Вызов #1: k=0, a-k=4, b-k=1
Вызов #2: k=1, a-k=3, b-k=1
Вызов #3: k=2, a-k=2, b-k=1
Вызов #4: k=3, a-k=1, b-k=1
Возврат из вызова #4: 4
Возврат из вызова #3: 28
Возврат из вызова #2: 280
Возврат из вызова #1: 3640
Результат: 3640, глубина: 4
Process finished with exit code 0
```

Рисунок 5

```
a: 100000
b: 100000
x: 1
Вызов #87245: k=87244, a-k=12756, b-k=12756
Вызов #87246: k=87245, a-k=12755, b-k=12755
Вызов #87247: k=87246, a-k=12754, b-k=12754
Вызов #87248: k=87247, a-k=12753, b-k=12753
Вызов #87249: k=87248, a-k=12752, b-k=12752
Вызов #87250: k=87249, a-k=12751, b-k=12751
Вызов #87251: k=87250, a-k=12750, b-k=12750
Вызов #87252: k=87251, a-k=12749, b-k=12749
Вызов #87253: k=87252, a-k=12748, b-k=12748
Вызов #87254: k=87253, a-k=12747, b-k=12747
Вызов #87255: k=87254, a-k=12746, b-k=12746
Вызов #87256: k=87255, a-k=12745, b-k=12745
Вызов #87257: k=87256, a-k=12744, b-k=12744

Process finished with exit code 139 (interrupted by signal 11: SIGSEGV)
```

Рисунок 6

```
a: 40
b: 26
x: 3
Возврат из вызова #13: -5260380434000248832
Возврат из вызова #12: -1603226130448384000
Возврат из вызова #11: -3921273164142739456
Возврат из вызова #10: -7064934360381652992
Возврат из вызова #9: 6254222159711830016
Возврат из вызова #8: 1284518976138772 48
Возврат из вызова #7: -2775612564816527360
Возврат из вызова #6: 760954233599033344
Возврат из вызова #5: 6690329999326576640
Возврат из вызова #4: -7386239702006759424
Возврат из вызова #3: -4730154822907461632
Возврат из вызова #2: -7599198199315693568
Возврат из вызова #1: -2678292677518163968
Результат: -2678292677518163968, глубина: 40
Process finished with exit code 0
```

Рисунок 7

6 Вывод по результату тестирования

Нормальные значения. Входные данные: a = 2, b = 3, x = 1. Ожидаемый результат: 30. Фактический результат: 30. Глубина рекурсии: 3. Функция корректно вычисляет результат для нормальных значений (рис. 2, 3).

Отрицательные и нулевые значения. Входные данные: a = -1000, b = 5, x = 2. Фактический результат: 2520. Глубина рекурсии: 5. При значениях a, b < 2 значение (a-k) или (b-k) сразу становится ≤ 1, соответственно приравниваются 1 и вычисления происходят обычным образом (рис. 4, 5).

Большие значения. Функция сталкивается с переполнением стека (рис. 6) или переполнением целочисленных типов (рис. 7) при больших значениях. Чтобы избежать переполнения стека, стоит использовать итеративный метод для решения задачи, или же искусственно ограничить глубину рекурсии. Для работы с большими значениями потребуется использование специальных библиотек для длинной арифметики.

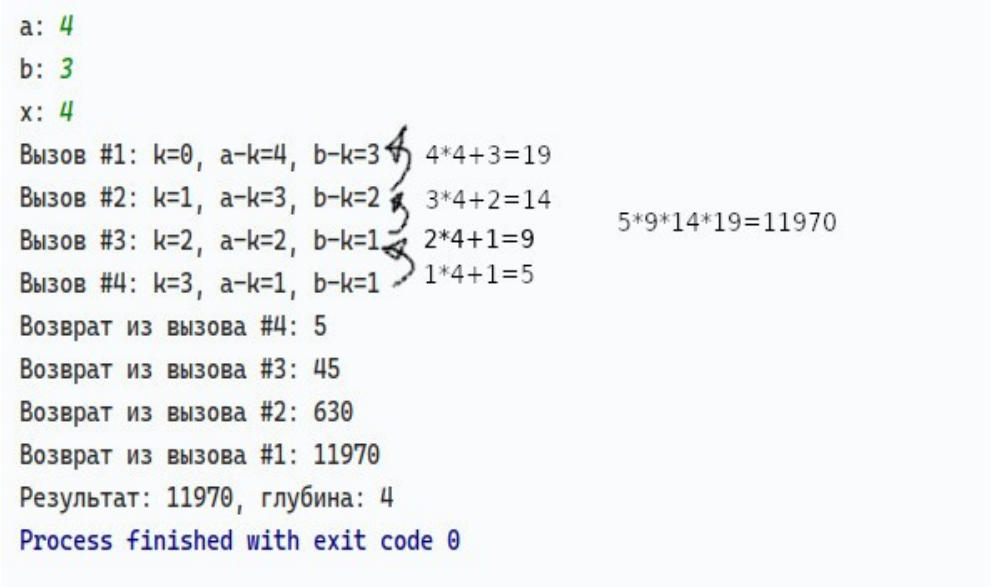

Рисунок 8

Из рис. 8 видно, что функция алгоритмически корректно решает задачу (при условии, что входные значения не слишком велики).