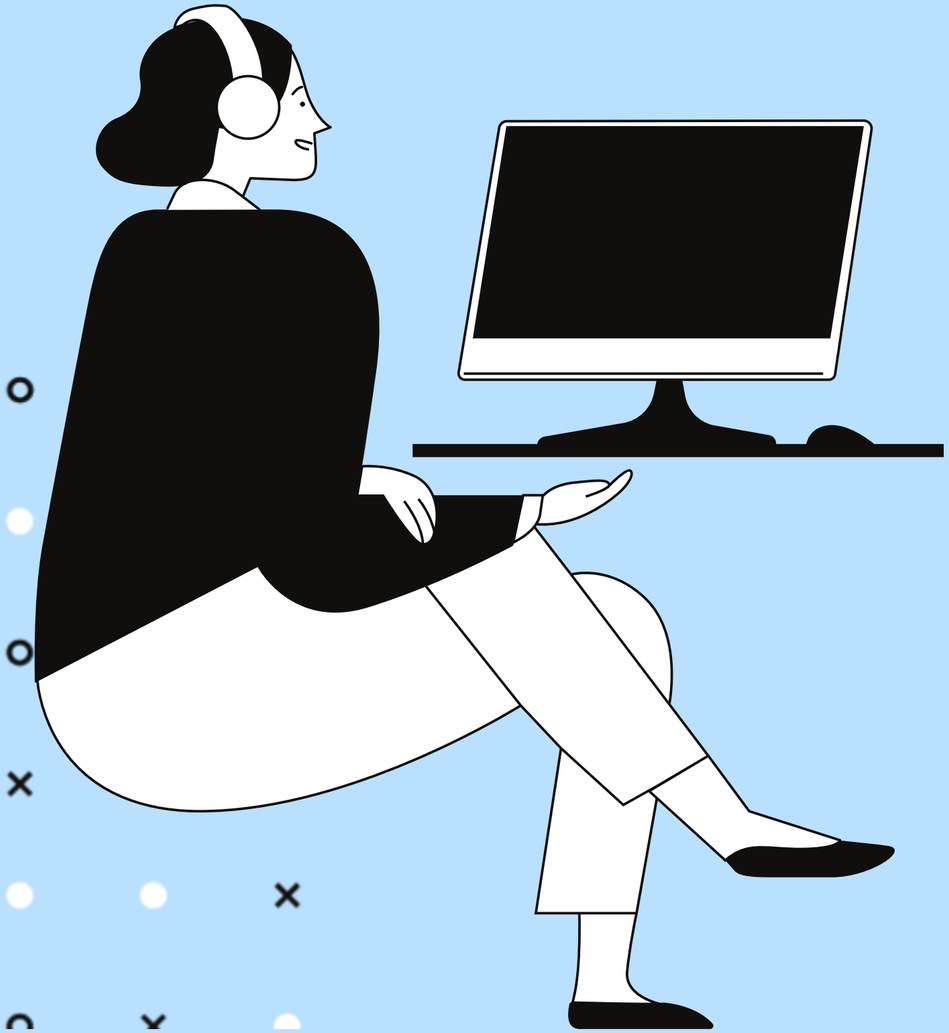
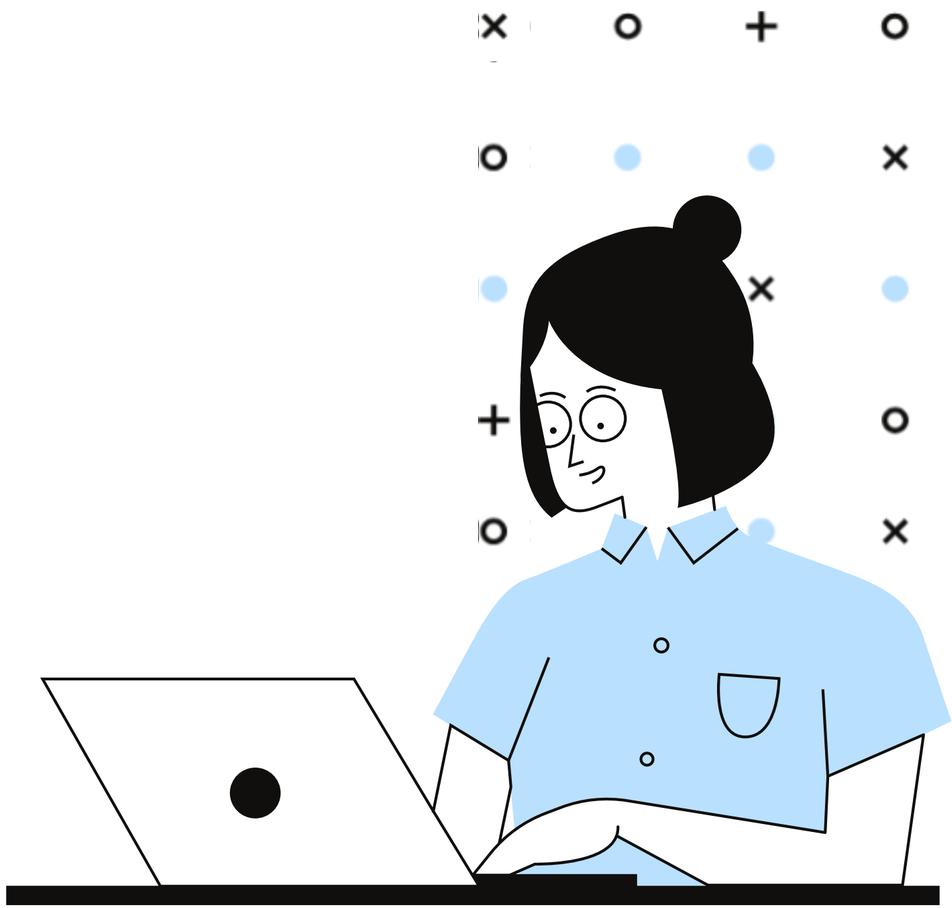


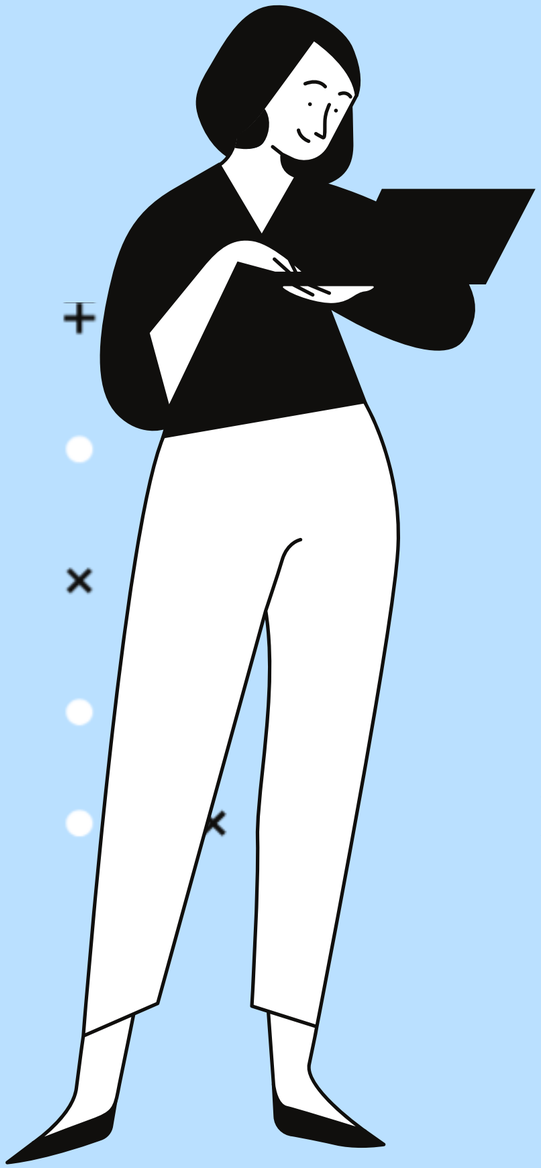
КРИТИЧЕСКИЕ СИТУАЦИИ В РЕКУРСИИ

Крушители идей

Критические ситуации в рекурсии возникают, когда рекурсивная функция вызывает саму себя бесконечное число раз, что приводит к переполнению стека и аварийному завершению программы. Это может произойти, если функция не имеет условия выхода или если условие выхода не выполняется.



Рассмотрим пример



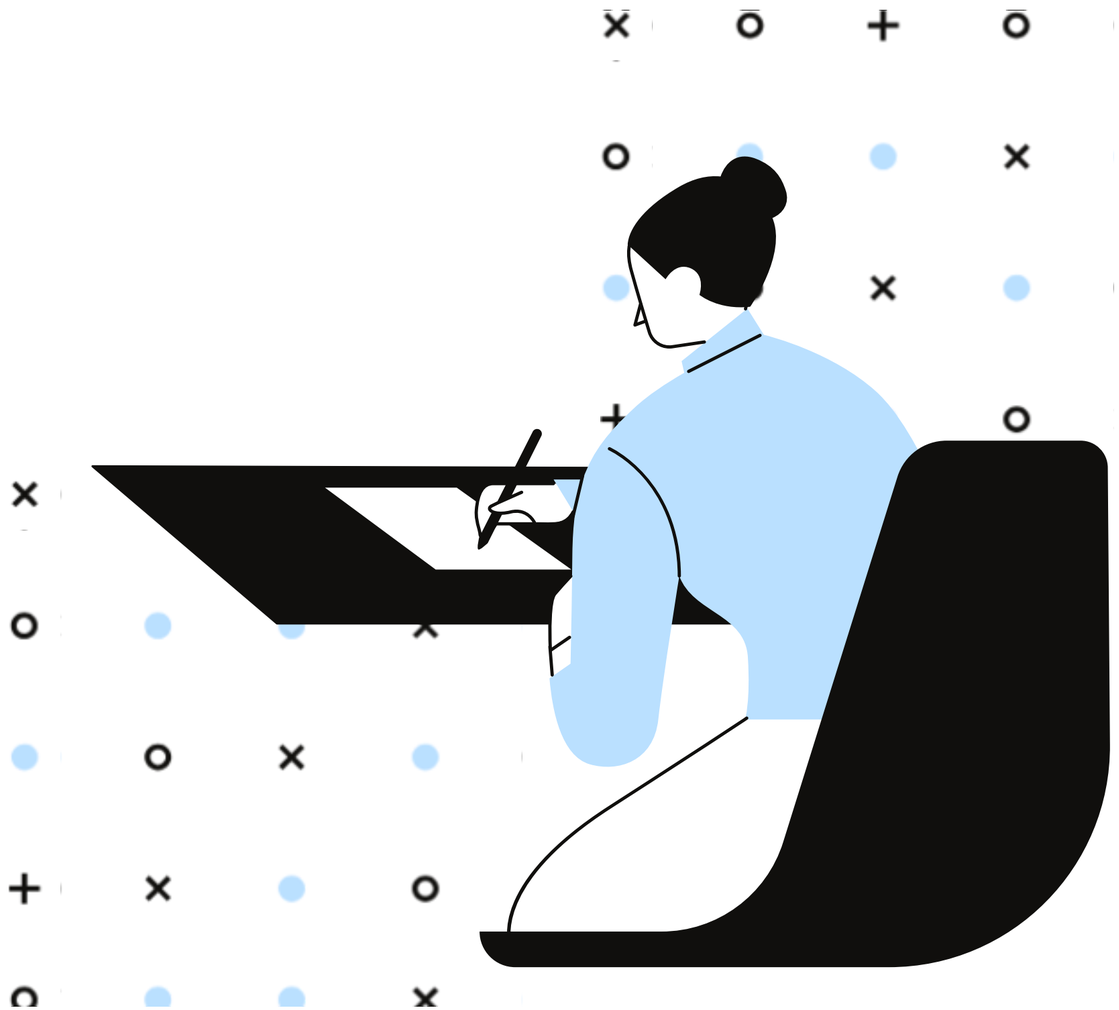
```
#INCLUDE <IOSTREAM>
USING NAMESPACE STD;

INT ISPRIME(INT N, INT I = 2) {
    IF (N <= 2) {
        IF (N == 2) {
            RETURN 1;
        } ELSE {
            RETURN 0;
        }
    }
    IF (N % I == 0) {
        RETURN 0;
    }
    IF (I * I > N) {
        RETURN 1;
    }
    RETURN ISPRIME(N, I + 1);
}

INT COUNTPRIMES(INT N) {
    INT COUNT = 0;
    FOR (INT I = 2; I <= N; I++) {
        IF (ISPRIME(I)) {
            COUNT++;
        }
    }
    RETURN COUNT;
}

INT MAIN() {
    INT NUM;
    COUT << "ВВЕДИТЕ
НАТУРАЛЬНОЕ ЧИСЛО: ";
    CIN >> NUM;

    INT RESULT =
COUNTPRIMES(NUM);

    IF (RESULT == 0) {
        COUT << "ПРОСТЫХ ЧИСЕЛ, НЕ
ПРЕВЫШАЮЩИХ " << NUM << ",
НЕТ." << ENDL;
    } ELSE {
        COUT << "КОЛИЧЕСТВО
ПРОСТЫХ ЧИСЕЛ, НЕ
ПРЕВЫШАЮЩИХ " << NUM << ",
СОСТАВЛЯЕТ: " << RESULT << ENDL;
    }

    RETURN 0;
}
```

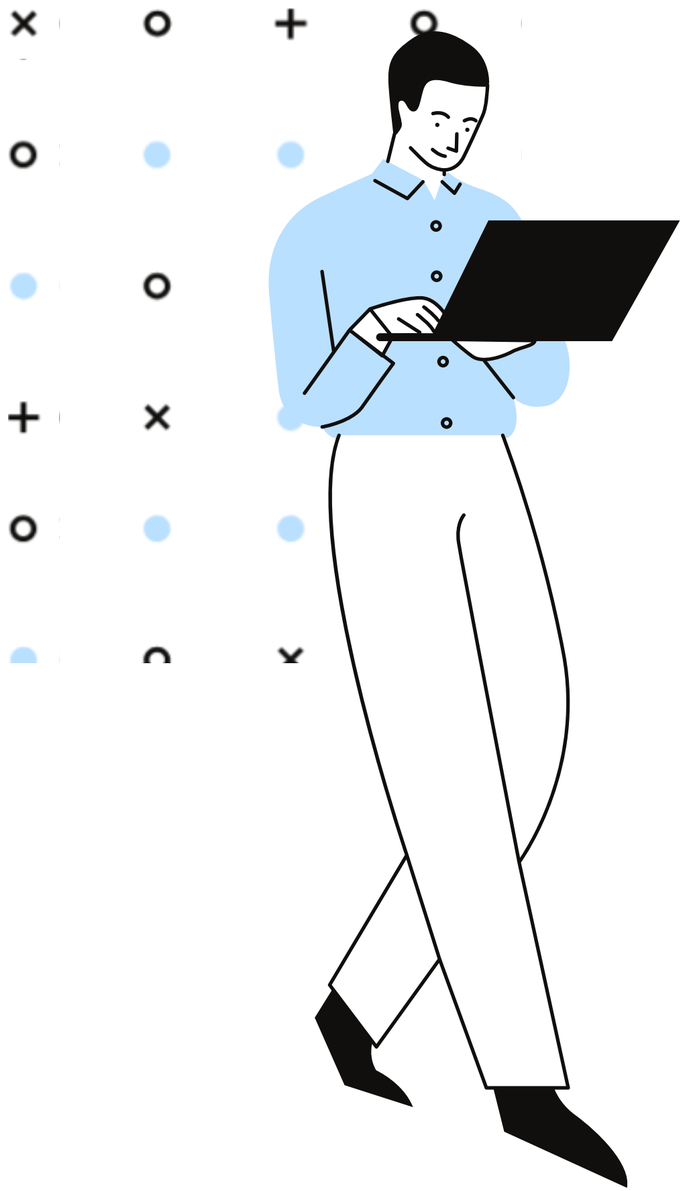
Критическая ситуация

Критической ситуацией в данном коде может быть бесконечная рекурсия, если значение переменной n будет отрицательным. Поскольку функция isPrime вызывает саму себя с аргументом n, равным n-1, если n будет отрицательным, это приведёт к бесконечным рекурсивным вызовам, что приведёт к переполнению стека и аварийному завершению программы.

Путь устранения

Чтобы избежать этой критической ситуации, можно добавить проверку входного значения n в функцию isPrime, чтобы убедиться, что оно положительное. Например, можно добавить условие if (n < 2) return 0; в начало функции isPrime, чтобы вернуть 0, если n меньше.





Спасибо за внимание!