ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

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

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 1.2

АЛГОРИТМ С ИСПОЛЬЗОВАНИЕМ ЦИКЛОВ

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. Глобальные параметры использовать запрещено; допустимо использование дополнительных функций.

Таблица 1 – Вариант

N	Текст задания	Вход	Выход
7	Реализовать функцию нахождения всех простых множителей числа A на экран таких, что в них нет цифры B в младшем разряде десятичной записи.	47740, 1	2,5,7

2 Схема алгоритма решения

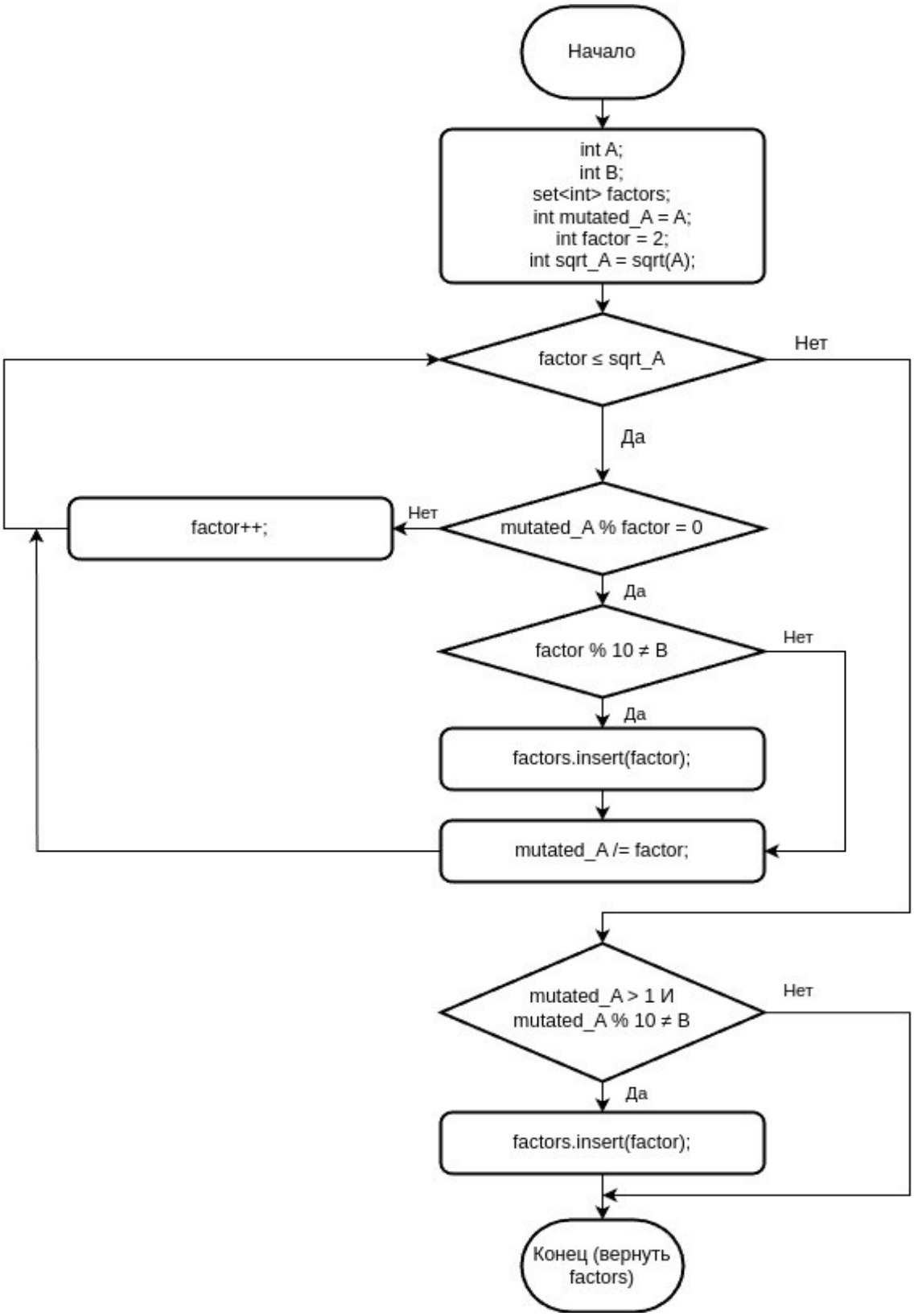

Рисунок 1 – Блок-схема алгоритма

3 Полное описание реализованной функции

Функция find_prime_factors принимает следующие аргументы:

1. int A – число, простые множители которого необходимо найти.
2. int B – цифра, которой не должно быть в младшем разряде десятичной записи каждого выводимого простого множителя.

Возвращает функция множество (set) factors, в котором в отсортированном в порядке возрастания виде хранятся значения простых множителей числа A без цифры B в младшем разряде, каждое в единственном экземпляре.

4 Листинг программы

Листинг 1

```cpp
#include <iostream>
#include <cmath>
#include <set>

std::set<int> find_prime_factors(int A, int B) {
    std::set<int> factors;
    int mutated_A = A;
    int factor = 2;
    int sqrt_A = (int) sqrt(A);

    // Раскладываем число A на простые множители путем деления на простые
    // числа, начиная с наименьшего - 2 (prime factorization)
    while (factor ≤ sqrt_A) {
        if (mutated_A % factor == 0) {
            if (factor % 10 ≠ B)
                factors.insert(factor);
            mutated_A /= factor;
        } else
            factor++;
    }

    // Если mutated_A больше 1, значит mutated_A - последний
    // простой множитель A
    if (mutated_A > 1 && mutated_A % 10 ≠ B)
        factors.insert(mutated_A);

    return factors;
}

int main() {
    int A, B;

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

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

    std::set<int> result = find_prime_factors(A, B);

    for (int i : result) {
        std::cout << i << " ";
    }

    return 0;
}
```

5 Несколько тестов работы программы

Рисунок 2

Рисунок 3

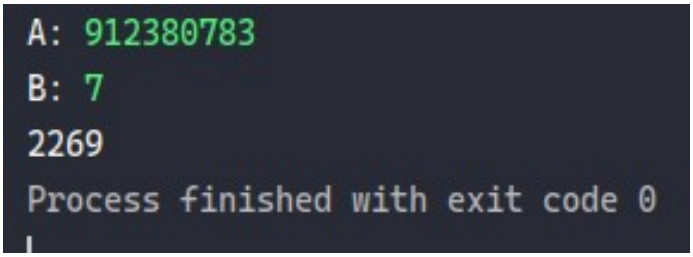

Рисунок 4

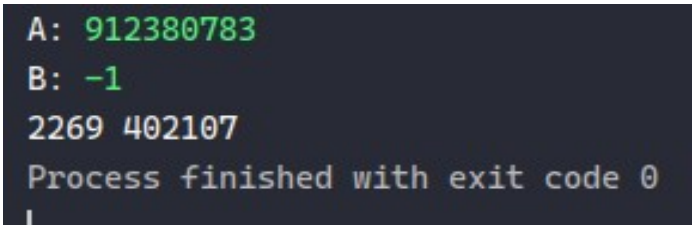

Рисунок 5