Министерство образования и науки Российской Федерации

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

федеральное государственное бюджетное образовательное учреждение

высшего образования

"Санкт-Петербургский государственный технологический институт

(технический университет)"

Кафедра САПР и У

Факультетинформатики и вычислительной техники

Курс 3

Группа 4291

Учебная дисциплина: «Разработка программных систем»

КОНТРОЛЬНАЯ РАБОТА №2

Вариант 3

Студент Борисова Д.С. Личная подпись Расшифровка подписи

Руководитель Личная подпись Расшифровка подписи должность

Г. Санкт-Петербург

2025 г.

**Контрольная работа №2**

Вариант 3

**Задание:**

Разработать алгоритм распознающий те числа, которые удовлетворяют заданным условиям:

Числа, сумма цифр которых является квадратным числом и разность между соседними цифрами равна *k*.

Например:

1357→1+3+5+7=16=42, при *k*=2,…

Программа должна позволять по выбору пользователя либо печатать числа, обладающих свойством, для заданного пользователем диапазона [*n*,*m*], либо вводить, печатать информацию и подсчитывать числа, обладающих свойством, до тех пор, пока пользователь не введет два нуля подряд (количество чисел неизвестно). Опционально программа должна распечатывать ряд чисел согласно вариантам.

**Математическая модель:**

Число N, сумма цифр которого является квадратом целого числа, и разность между соседними цифрами равна k, может быть найдено, следуя следующим шагам:

1. Выбираем начальную цифру d1∈{0,1,2,…,9}.
2. Для каждой следующей цифры di+1​, вычисляем её как: di+1=di±k, di+1​ ∈ {0,1,2,…,9}.
3. После построения всех цифр, проверяем, что сумма цифр S(N)=m2 для некоторого m ∈ Z, m ≥ 0.

**Тестовые планы:**

Тестирование программы предлагается разбить на части:

1. **Тестирование печати чисел в заданном диапазоне:**

Вызвать функцию печати чисел, обладающих свойством, для заданного диапазона и использовать следующие данные:

Таблица 5 – План для тестирования печати чисел

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тест | От | До | K | Результат |
| 1 | 10 | 100 | 2 | 1 4 9 13 31 79 97 |
| 2 | 10 | 100 | 4 | 1 4 9 40 |
| 3 | 10 | 100 | 6 | 1 4 9 |
| 4 | 10 | 100 | 9 | 1 4 9 90 |
| 5 | 10 | 100 | –1 | Incorrect digit difference value |

**2. Тестирование последовательности ввода чисел**

В качестве тестовых последовательностей можно использовать следующие данные (жирным выделены числа, обладающие свойством):

Таблица 6 – план тестирования ввода чисел

|  |  |  |  |
| --- | --- | --- | --- |
| Тест | Описание | Ввод | Результат |
| 1 | |  | | --- | |  |  |  | | --- | | Ввод одного числа, которое удовлетворяет условиям. | | 1357 | Программа увеличивает счётчик на 1 и выводит "Количество чисел, удовлетворяющих условиям: 1". |
| 2 | Ввод одного числа, которое не удовлетворяет условиям (сумма цифр не квадрат). | 1234 | Программа не увеличивает счётчик и выводит "Количество чисел, удовлетворяющих условиям: 0". |
| 3 | Ввод числа, которое не удовлетворяет разности между цифрами. | 1235 | Программа не увеличивает счётчик и выводит "Количество чисел, удовлетворяющих условиям: 0". |
| 4 | |  | | --- | |  |  |  | | --- | | Ввод чисел, где разность между соседними цифрами равна 2 и сумма цифр является квадратом. | | 1357, 3579 | Программа увеличивает счётчик для чисел, где разность между цифрами равна 2 и сумма цифр — квадрат. |
| 5 | Ввод чисел, где разность между соседними цифрами равна 3 и сумма цифр является квадратом. | 147, 258 | Программа увеличивает счётчик для чисел, где разность между цифрами равна 3 и сумма цифр — квадрат. |
| 6 | Ввод чисел в диапазоне от 10 до 100, включая 10 и 100. | 10, 99 , 100 | Программа проверяет все числа в диапазоне и выводит те, которые удовлетворяют условиям. |
| 7 | Ввод отрицательных чисел. | -12, -34 | Программа игнорирует отрицательные числа или выводит ошибку. |
| 8 | Ввод чисел с нецелыми значениями (дроби). | 12.5, 3.7 | Программа игнорирует дробные числа. |
| 9 | Ввод строки или символа вместо числа. | abc, @ | Программа обрабатывает ошибку ввода (например, запрашивает повторный ввод). |
| 10 | |  | | --- | |  |  |  | | --- | | Ввод нескольких чисел, завершающихся числом 0. | | 1357, 1234, 90, 0 | Программа подсчитывает количество чисел, удовлетворяющих условиям, и завершает ввод. |
| 11 | Ввод только числа 0 сразу. | 0 | Программа выводит "Количество чисел, удовлетворяющих условиям: 0". |
| 12 | |  | | --- | |  |  |  | | --- | | Ввод чисел с более чем двумя цифрами (например, 100, 123). | | 123, 147, 100 | Программа проверяет все числа с более чем двумя цифрами. |
| 13 | |  | | --- | |  |  |  | | --- | | Ввод большого диапазона чисел (например, от 1 до 1000). | | от 1 до 1000 | Программа корректно обрабатывает все числа в пределах диапазона и предоставляет результаты. |

**Декомпозиция задач:**

Исходя из условий задания, главную задачу main можно разбить на две основные задачи:

Проверка свойств чисел: printNumbersInRange

Ввод данных печати результатов: countNumbersWithConditions

Задача printNumbersInRange выполняет проверку введенных чисел satisfiesConditions. satisfiesConditions в своу очередь проверяет, является ли сумма цифр числа квадратом числа (isSumOfDigitsSquare) и одновременно есть ли постоянная разность между соседними цифрами (hasConstantDifference).

Задача countNumbersWithConditions состоит из проверки числа, удовлетворяющего заданию satisfiesConditions и считывания последнего числа = 0 для завершения подсчета.

main

countNumbersWithConditions

printNumbersInRange

satisfiesConditions

satisfiesConditions

hasConstantDifference

isSumOfDigitsSquare

Рисунок 7 – дерево декомпозиции задач

**Блок-схемы алгоритмов:**

Алгоритм работы isSumOfDigitsSquare (рис 8):

1. **Инициализация переменной** sum — обнуляем переменную для хранения суммы цифр.
2. **Цикл по цифрам числа** — извлекаем цифры числа, добавляем их к сумме, уменьшаем число (делим на 10).
3. **Вычисление корня из суммы** — находим квадратный корень из суммы цифр.
4. **Проверка, является ли сумма квадратом числа** — если квадрат корня равен сумме, возвращаем true, иначе false.
5. **Конец** — функция завершает выполнение.

Этот алгоритм проверяет, является ли сумма цифр числа квадратом целого числа.

Num, k

Начало

Начало

num

Prev\_digit = num % 10

нет

Num > 0

нет

Num > 0

да

да

Sum += num % 10

curr\_digit = num % 10

Num /= 10

нет

да

Abs !=k

Int root = statis\_cast<int>(sqrt(sum))

prev\_digit =curr\_digit

Return true

Root\*root == sum

num /= 10

нет

да

Return true

Return false

Return true

Конец

Конец

Рисунок 8 – isSumOfDigitsSquare Рисунок 9 - hasConstantDifference

Алгоритм функции hasConstantDifference (рис 9):

1. **Инициализация** — сохраняем последнюю цифру числа в prev\_digit и сокращаем число.
2. **Цикл по цифрам числа** — для каждой цифры проверяем, что разница между текущей и предыдущей цифрой равна k.
3. **Возврат результата** — если разница не равна k, возвращаем false. Если все проверки прошли, возвращаем true.

Функция satisfiesConditions(int num, int k) (рис 10) вызывает две другие функции:

- isSumOfDigitsSquare(num) — проверяет, является ли сумма цифр числа квадратом.

- hasConstantDifference(num, k) — проверяет, является ли разница между соседними цифрами постоянной и равной k.

Если обе проверки возвращают true, то функция возвращает true, иначе — false. Таким образом, функция проверяет, удовлетворяет ли число обоим условиям.

Начало

Num, k

isSumOfDigitsSquare = true

нет

да

Return true

hasConstantDifference

Конец

Рисунок 10 – алгоритм satisfiesConditions

Функция printNumbersInRange (рис. 11) перебирает все числа от n до m с шагом 1.

**- Проверка условий:** Для каждого числа i проверяется, удовлетворяет ли оно условиям с помощью функции satisfiesConditions(i, k).

**- Печать числа:** Если число удовлетворяет условиям, оно выводится на экран.

**- Завершение:** После завершения цикла выводится новая строка, и функция заканчивает выполнение.

N, m, k

Начало

нет

да

Int I = n;

I<=m; ++i

satisfiesConditions(i, k)

да

Return true

std::cout << i << " "

std::cout << std::endl;

Конец

Рисунок 11 – алгоритм printNumbersInRange

Функция countNumbersWithConditions (рис 12) начинает с инициализации переменной num для ввода и переменной count для подсчёта чисел, которые удовлетворяют условиям.

**- Ввод числа:** В цикле программа запрашивает ввод числа от пользователя.

**- Проверка на завершение:** Если пользователь вводит 0, цикл прерывается и программа выводит результат. В противном случае продолжается проверка числа.

**- Проверка условий:** Для каждого введённого числа вызывается функция satisfiesConditions(num, 0). Если число удовлетворяет условиям, увеличивается счётчик count.

**- Завершение работы:** После завершения цикла выводится количество чисел, которые удовлетворяют условиям.

Начало

Num, count

нет

да

Num == 0 ?

satisfiesConditions(num, 0)

Прерывание цикла break

нет

да

Return true

Повторить ввод числа

count++;

Вывести количество чисел std::cout << count

Конец

Рисунок 12 – алгоритм countNumbersWithConditions

**Листинг программы:**

Interface.cpp

#include <iostream>

#include "Task.h"

void printMenu() {

std::cout << "Выберите опцию:\n";

std::cout << "1. Печать чисел в диапазоне [n, m], которые удовлетворяют условиям\n";

std::cout << "2. Ввод чисел для подсчёта (ввод завершится, когда введёте 0)\n";

std::cout << "3. Выход\n";

}

//основная функция запуска программы

int main() {

setlocale(LC\_ALL, "Russian");

int choice;

while (true) {

printMenu();

std::cin >> choice;

if (choice == 1) {

int n, m, k;

std::cout << "Введите диапазон [n, m] и разницу k между соседними цифрами: ";

std::cin >> n >> m >> k;

printNumbersInRange(n, m, k);

}

else if (choice == 2) {

countNumbersWithConditions();

}

else if (choice == 3) {

std::cout << "Выход из программы.\n";

break;

}

else {

std::cout << "Некорректный выбор. Пожалуйста, попробуйте снова.\n";

}

}

return 0;

}

Task.h

#pragma once

#ifndef TASK\_H

#define TASK\_H

// Функция проверки, является ли сумма цифр числа квадратом числа

bool isSumOfDigitsSquare(int num);

// Функция проверки разности между соседними цифрами

bool hasConstantDifference(int num, int k);

// Функция, которая проверяет, удовлетворяет ли число всем условиям

bool satisfiesConditions(int num, int k);

// Функция для печати чисел в диапазоне [n, m]

void printNumbersInRange(int n, int m, int k);

// Функция для ввода чисел с подсчётом

void countNumbersWithConditions();

#endif

Solve.cpp

#include "Task.h"

#include <iostream>

#include <cmath>

// Проверка, является ли сумма цифр числа квадратом числа

bool isSumOfDigitsSquare(int num) {

int sum = 0;

while (num > 0) {

sum += num % 10;

num /= 10;

}

// Проверяем, является ли сумма цифр квадратом числа

int root = static\_cast<int>(sqrt(sum));

return root \* root == sum;

}

// Проверка, есть ли постоянная разность между соседними цифрами

bool hasConstantDifference(int num, int k) {

int prev\_digit = num % 10;

num /= 10;

while (num > 0) {

int curr\_digit = num % 10;

if (std::abs(curr\_digit - prev\_digit) != k) {

return false;

}

prev\_digit = curr\_digit;

num /= 10;

}

return true;

}

// Проверка, удовлетворяет ли число всем условиям

bool satisfiesConditions(int num, int k) {

return isSumOfDigitsSquare(num) && hasConstantDifference(num, k);

}

// Печать чисел в диапазоне [n, m] с проверкой условий

void printNumbersInRange(int n, int m, int k) {

for (int i = n; i <= m; ++i) {

if (satisfiesConditions(i, k)) {

std::cout << i << " ";

}

}

std::cout << std::endl;

}

// Функция для ввода чисел с подсчётом

void countNumbersWithConditions() {

int num;

int count = 0;

while (true) {

std::cout << "Введите число (для завершения введите 0): ";

std::cin >> num;

if (num == 0) {

break;

}

// Проверка и подсчёт числа

if (satisfiesConditions(num, 0)) { // Для подсчёта проверяем без разницы k

++count;

}

}

std::cout << "Количество чисел, удовлетворяющих условиям: " << count << std::endl;

}