МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ образовательное учреждение высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра вычислительной техники

ОТЧЁТ по лабораторной работе №1

по дисциплине «ИНФОРМАТИКА»

«Арифметические задачи»

|  |  |
| --- | --- |
| Факультет: АВТ  Группа: ДТ-460а  Студент: Пантюхин Артём Евгеньевич | Преподаватель:  Копылова Оксана Андреевна |

Новосибирск, 2025 г.

Содержание

[Задание 2](#_Toc1)

[Теоретические сведения 2](#_Toc2)

[Программа №43 3](#_Toc3)

[1. Основные «идеи» алгоритма: 3](#_Toc4)

[2. Образная модель: 3](#_Toc5)

[3. Внутреннее представление данных в программе: 4](#_Toc6)

[4. Стандартные фрагменты и необходимые переменные 4](#_Toc7)

[5. Исходный код – см. приложение 1. 4](#_Toc8)

[Пример работы программы 5](#_Toc9)

[Выводы 6](#_Toc10)

[Список литературы 7](#_Toc11)

[Приложение 1. Исходный код программы 8](#_Toc12)

# Задание

43. Дополнение до квадрата.

«Какое число, — спросил полковник Крэкхэм, — обладает тем свойством, что если его прибавить к числам 100 и 164 в отдельности, то каждый раз получатся точные квадраты?»

Обобщённое понимание задачи:

Реализовать программу, которая, получив два целых числа на вход, найдёт, если это возможно, такое число, которое, будучи прибавленным к любому из входных значений, в результате операции даст точный квадрат.

# Теоретические сведения

Точный квадрат – квадрат некоего целого числа. Число, квадратный корень из которого извлекается нацело.

Способы проверки, является ли число точным квадратом:

1. Квадрат натурального числа n можно представить в виде суммы первых n нечётных чисел:
2. В десятичной записи:
   1. Последние две цифры квадрата должны выглядеть следующим образом:

|  |  |
| --- | --- |
| Последняя цифра | Предпоследняя цифра |
| 0 | 0 |
| 5 | 2 |
| 1, 4, 9 | Чётная |
| 6 | Нечётная |

* 1. Квадрат не может оканчиваться нечётным количеством нулей.
  2. Квадрат либо кратен 4, либо даёт остаток 1 при делении на 8.

1. Бинарный поиск (подстановки больше/меньше).

# Программа №43

Основные «идеи» алгоритма:

* 1. Логика проверки, является ли число точным квадратом, описана выше. Используем первый вариант.
  2. Если оба входных числа уже являются квадратами, ответ - 0
  3. Если входные числа равны, ответом является любое число, приводящее к полному квадрату. Чтобы избежать перебора, просто скажем, что ответ – входное число, умноженное на -1, что гарантированно даст 0 при сложении.
  4. Существование решения зависит от существования в целых числах двух уравнений:  
     Пусть входные числа будут x, y, решение – z, квадраты в результате суммы – a^2 и b^2 соответственно.  
     x + z = a^2  
     y + z = b^2  
     Тогда,  
     x – y = a^2 – b^2 = (a-b)(a+b)  
     Зная, что a и b – целые числа, отметим следующую закономерность:  
     Если как a, так и b – чётные или нечётные, то:  
     x – y = c, где c – кратно четырём (в скобках получим чётные числа, у числа c гарантированно будет множитель 4).  
     Если чётность чисел a, b отличается, то:  
     x – y = c , где c – нечётное.  
     Таким образом, решение в целых числах существует тогда и только тогда, когда (x – y) кратно четырём или нечётно.
  5. Если решение существует, производим полный перебор значений от меньшего из входных, умноженного на -1 (от квадрата суммы, равного нулю), до переполнения переменной или нахождения подходящего числа.
  6. Если по выходу из цикла подходящее число не вызывает переполнения при сложении с большим из входных – решение найдено.

Образная модель:

* 1. Возьмём два числа из условия: 100 и 164
  2. Числа не равны, лишь одно из них является квадратом.
  3. Разность чисел по модулю равна 64, это число кратно четырём, значит, решение существует.
  4. Перебираем варианты, начиная с меньшего, умноженного на -1.
  5. Первое же число (-100) оказалось подходящим:  
     100+(-100) = 0, точный квадрат  
     164+(-100) = 64, точный квадрат (8\*8).

Внутреннее представление данных в программе:

* 1. Логика буферизации и конвертации ввода пользователя описана в комментариях к исходному коду и не является частью решения.
  2. В дальнейшем, данные используются «как есть», без конвертаций в другие форматы и/или разбиения.
  3. В ходе определения, является ли полученное число квадратом, эталонное число n (ближайший точный квадрат, больший или равный входному числу) представлено суммой ряда из первых n элементов этого числа.

Стандартные фрагменты и необходимые переменные

* 1. Логика буферизации и конвертации ввода пользователя описана в комментариях к исходному коду и не является частью решения.
  2. Входные числа представим в виде массива из двух элементов.  
     Если решение не будет найдено моментально (по равенству элементов или если оба элемента - квадраты), по индексу 0 расположен меньший элемент, по индексу 1 – больший.
  3. Решение(если существует) находится методом полного перебора значений, переменная-счётчик solution ограничена диапазоном от меньшего-входного \* -1 до предельного значения для типа данных long long int.
  4. Проверка, является ли число квадратом, выполняется дважды на каждой итерации цикла, вынесена в отдельную функцию is\_square.
  5. Решение представлено переменной-счётчиком и переменной-флагом. В случае, если флаг покажет наличие решения, выведем переменную-счётчик в качестве ответа.

Исходный код – см. [приложение 1.](#_Приложение)

# Пример работы программы

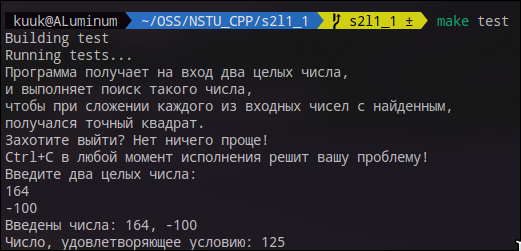


Рисунок 1. Пример работы программы. Корректный ввод.

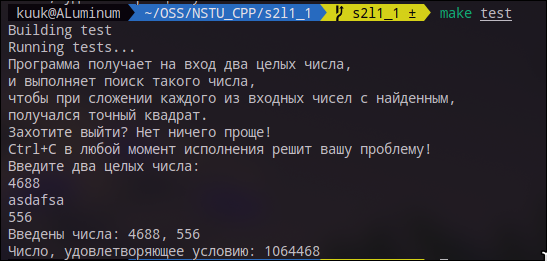


Рисунок 2. Пример работы программы. Пропуск некорректного ввода.

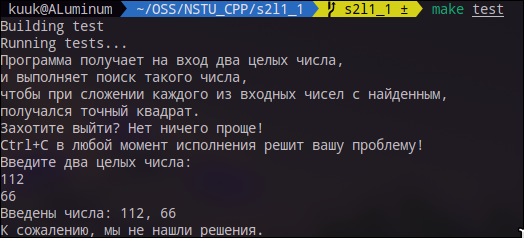


Рисунок 3. Пример работы программы. Корректный ввод, нет решения.

# Выводы

В процессе выполнения данной лабораторной работы были использованы алгоритмы полного перебора, переменные-флаги и –счётчики, в рамках решения задействованы свойства целых чисел и точного квадрата.

Были изучены и применены принципы структурного программирования, принципы проектирования программ, углублены знания в области стандартных программных контекстов.

# Список литературы

1. Подбельский В.В., Фомин С.С. Программирование на языке Си: Учеб.пособие. – 2-е доп. Изд. – М.: Финансы и статистика, 2004. – 600 с.

2. Романов Е. Л. Си/Си++. От дилетанта до профессионала. Электронное учебное пособие по дисциплинам "Информатика", "Программирование", "Технология программирования" для студентов 1–2 курсов направления 230100 : учеб. пособие / Е. Л. Романов. – Новосибирский государственный технический университет, № гос регистрации 0321000528, 2010. - 581 с.

3. Си/Си++ от дилетанта до профессионала [Электронный ресурс]. URL: http://ermak.cs.nstu.ru/cprog/HTML/index.htm (дата обращения 01.10.2022).

# Приложение 1. Исходный код программы

#include <locale.h>

#include <stdio.h>

#include <limits.h>

//Вынесем в #define для удобства, и чтобы не засорять main

#define GREETINGS\_STRING \

"Программа получает на вход два целых числа,\n\

и выполняет поиск такого числа,\n\

чтобы при сложении каждого из входных чисел с найденным,\n\

получался точный квадрат.\n"

#define EXIT\_STRING \

"Захотите выйти? Нет ничего проще!\n\

Ctrl+C в любой момент исполнения решит вашу проблему!\n"

//Функция для определения, является ли число точным квадратом.

int is\_square(long long int value) {

/\*

Проверим, не подкинули ли нам

отрицательный квадрат

\*/

if (value < 0) {

return 0;

}

/\*

Квадрат натурального числа n всегда можно представить как сумму первых n нечётных чисел

\*/

long long int odd = 1;

//Спокойно меняем value, ведь это лишь копия оригинального значения.

while(value > 0) {

value -= odd;

odd += 2;

}

/\*

Если в ходе наших вычитаний мы получили 0 -

значит, сумма нечётных оказалась равна числу, и это точный квадрат.

\*/

return value == 0;

}

int main(void) {

setlocale(LC\_ALL, "Russian");

/\*

Почему long long int?

Чтобы нам совершенно точно хватило памяти

\*/

long long int numbers[2] = {};

int numbers\_entered = 0;

printf(GREETINGS\_STRING);

printf(EXIT\_STRING);

printf("Введите два целых числа:\n");

while (numbers\_entered < 2) {

//Входной буфер того же размера, что и буфер stdin, чтобы ничего не потерять

char inbuffer[BUFSIZ] = {};

//Получаем из потока stdin весь буфер

fgets(inbuffer, sizeof(inbuffer), stdin);

long long int num\_buffer[2] = {};

/\*

1) Считываем из буфера char числа в буфер long long int

2) sscanf возвращает количество успешно считанных значений, запоминаем их в consumed.

\*/

int consumed = sscanf(inbuffer, "%lli%lli", &num\_buffer[0], &num\_buffer[1]);

/\*

Пробегаемся по полученным значениям, и добавляем их

в итоговый массив, если в нём есть свободное место

\*/

for (int i = 0; i < consumed && numbers\_entered < 2; i++) {

numbers[numbers\_entered++] = num\_buffer[i];

}

}

printf("Введены числа: %lli, %lli\n", numbers[0], numbers[1]);

/\*

Видя подобный ввод, появляется логичный вопрос:

Зачем всё это?

Можно ведь просто получать значения сразу

scanf("%lli%lli", &numbers[0], &numbers[1]) решит проблему!1!1!

Нет, не решит. Если мы вдруг получим в stdin

невалидное значение, к примеру, букву,

то scanf просто оставит её в потоке, после чего,

видя что поток не пустой, начнёт считывать и отбрасывать

эту букву раз за разом.

fgets же скушает весь поток и не подавится.

\*/

long long int solution = 0;

int solved = 0;

if (is\_square(numbers[0]) && is\_square(numbers[1])) {

solved = 1;

} else if (numbers[0] == numbers[1]) {

solution = numbers[0]\*-1;

solved = 1;

} else {

/\*

Немножко (множко) подумав, мы понимаем, что решение задачки выглядит как:

numbers[0] + solution = a^2;

numbers[1] + solution = b^2;

a, b, solution - должны существовать в целых числах, numbers[] целые и так.

Попробуем вычесть одно из другого, получим:

numbers[1] - numbers[0] = b^2 - a^2 = (b-a)(b+a)

Работаем с целыми числами, значит, можем пользоваться суммой/разностью чётных/нечётных.

Итак, представим:

b % 2 == 1, a % 2 == 0

b-a % 2 == 1

b+a % 2 == 1

b % 2 == 1, a % 2 == 1

b-a % 2 == 0

b+a % 2 == 0

b % 2 == 0, a % 2 == 0

b-a % 2 == 0

b+a % 2 == 0

Получается, что numbers[1] - numbers[0] всегда будет == нечёт\*нечёт или чёт\*чёт

Это мы уже можем проверить

Если говорим о чёт\*чёт - значит каждое число кратно двум, значит, результат кратен 4

Если говорим о нечёт\*нечёт - значит, результат кратен двум никогда не будет.

Учитывая, что всё это мы выяснили, исходя из предположения, что test\_value существует,

рискну сказать, что верно и обратное.

\*/

if ((numbers[1] - numbers[0]) % 4 == 0 || (numbers[1] - numbers[0]) % 2 != 0) {

/\*

Определим меньшее и большее из введённых значений

запишем меньшее в numbers[0], большее в numbers[1]

\*/

if (numbers[0] > numbers[1]) {

long long int buff = numbers[0];

numbers[0] = numbers[1];

numbers[1] = buff;

}

/\*

Что нам даёт такое присвоение?

Мы знаем, что квадрат находится в пределах [0;+inf)

А значит, первый кандидат на прибавление к числам - тот, который сведёт

наименьшее из них в 0.

\*/

solution = numbers[0]\*-1;

for ( ; (!is\_square(numbers[0]+solution) || !is\_square(numbers[1]+solution))

&& numbers[1]+solution < LLONG\_MAX; solution++);

solved = numbers[1]+solution == LLONG\_MAX ? 0 : 1;

}

}

if (solved) {

printf("Число, удовлетворяющее условию: %lli\n", solution);

} else {

printf("К сожалению, мы не нашли решения.\n");

}

return 0;

}