Лабораторная работа №2

В рамках данной лабораторной работы вам предстоит поработать с целочисленной арифметикой в C++. При выполнении лабораторной работы старайтесь **не использовать дополнительную память** (т.е. постарайтесь не использовать массивы).

1.1 Определения и понятия

Определение 1.1. Натуральное число k называется **простым**, если оно делится только на самого себя и на единицу.

Пример 1.1. Числа $2, 3, 5, 7, 11 \dots$

Определение 1.2. Натуральное число k называется **совершенным**, если оно равно сумме всех своих делителей за исключением себя самого.

Пример 1.2. Числа $6, 28, 496, 8128 \dots$

Определение 1.3. Натуральное число k называется **палиндромом**, если его прямая и обратная записи совпадают.

Пример 1.3. Числа $4, 33, 121, 141, 8338, 13331 \dots$

Определение 1.4. Натуральное число k называется **двойным палиндромом**, если оно само и его квадрат являются палиндромами.

Пример 1.4. Числа $11, 22, 101, 111, 121 \dots$

Определение 1.5. Натуральное число k называется **автоморфным**, если десятичная запись его квадрата оканчивается числом k.

Пример 1.5. Числа $1, 5(\mathbf{5}^2 = 2\mathbf{5}), 6(\mathbf{6}^2 = 3\mathbf{6}), 25(\mathbf{25}^2 = 6\mathbf{25}), 76(\mathbf{76}^2 = 57\mathbf{76})$, $376, 625, 9376 \dots$

Определение 1.6. Натуральное число k называется **числом Армстронга**, если оно равно сумме n-х степеней своих цифр, где n — количество цифр в десятичной записи числа k.

Пример 1.6. Числа $153(153 = 1^3 + 5^3 + 3^3), 370(370 = 3^3 + 7^3 + 0^3), 371, 407$, $1634, 8208, 9474, 54748 \dots$

Определение 1.7. Натуральное число k называется **числом Мерсенна**, если оно представимо в виде: $k = 2^n - 1, n \in \mathbb{N}$.

Пример 1.7. Числа $1, 3, 7, 15, 31, 63, 127, 255, 511, 1023\ldots$

Определение 1.8. Простое число k называется **простым числом Мерсенна**, если оно представимо в виде: $k = 2^p - 1, p$ — простое число.

Пример 1.8. Числа $3, 7, 31, 127, 8191, 131071, 524287\ldots$

Определение 1.9. Простое число k называется **сверхпростым**, если оно остаётся простым при любой перестановке своих цифр.

Пример 1.9. Числа $11, 13, 17, 31, 37, 71, 73, 79, 97, 113, 131, 199, 311, 337\ldots$

Определение 1.10. Два простых числа a и $b(b > a)$ называются **близнецами**, если $b - a = 2$.

Пример 1.10. Числа $(3, 5); (5, 7); (11, 13); (17, 19)\ldots$

Определение 1.11. Два натуральных числа a и $b(b > a)$ называются **дружественными**, если сумма делителей первого (кроме его самого) равна второму, а сумма делителей второго (опять же кроме его самого) равна первому.

Пример 1.11. Числа 220 и $284(220 = 1 + 2 + 4 + 71 + 142;$
$284 = 1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110); 2620$ и $2924\ldots$

Определение 1.12. Три натуральных числа a, b, c образуют **Пифагорову тройку**, если $a^2 + b^2 = c^2$.

Пример 1.12. Числа $(3, 4, 5); (5, 12, 13); (8, 15, 17)\ldots$

1.2 Задания

1. С клавиатуры вводится натуральное число n. Удалить из десятичной записи этого числа все цифры, совпадающие с минимальной цифрой.

2. С клавиатуры вводится натуральное число n. Удалить из десятичной записи этого числа все цифры, повторяющиеся четное количество раз.

3. С клавиатуры вводится натуральное число n. Добавить слева и справа от этого числа наименьшую отличную от нуля цифру.

4. С клавиатуры вводится натуральное число n. Найти все совершенные числа, не превосходящие n.

5. С клавиатуры вводится натуральное число n. Найти все двойные палиндромы, не превосходящие n.

6. С клавиатуры вводится натуральное число n. Найти все **простые числа Мерсенна**, не превосходящие n.
7. С клавиатуры вводится натуральное число n. Найти все Пифагоровы тройки чисел, каждое из которых не превосходит n.
8. С клавиатуры вводятся числа a и b. Найти все простые числа, лежащие на отрезке $[a, b]$.
9. С клавиатуры вводятся числа a и b. Найти все палиндромы, лежащие на отрезке $[a, b]$.
10. С клавиатуры вводятся числа a и b. Найти все **числа Мерсенна**, лежащие на отрезке $[a, b]$.
11. С клавиатуры вводятся числа a и b. Найти все сверхпростые числа, лежащие на отрезке $[a, b]$.
12. С клавиатуры вводятся числа a и b. Найти все дружественные числа, лежащие на отрезке $[a, b]$.
13. С клавиатуры вводятся числа a и b. Найти все числа на отрезке $[a, b]$, у которых в десятичной записи все цифры различны.
14. С клавиатуры вводятся числа a и b. Найти все числа Армстронга на отрезке $[a, b]$.

1.3 Дополнительные задания

Данные задания выполнять не обязательно. Здесь вам будут предоставлены фрагменты кода, который нужно будет запустить и попытаться найти в нём уязвимости и ошибки (т.е. сделать так, чтобы программа завершилась с ненулевым кодом).

1.3.1 Определение неотрицательности числа

Ниже будет предоставлен код, который проверяет на неотрицательность введённое число двумя разными способами: через реализованную функцию ***IsNotNegative(int number)*** и через обычный оператор сравнения в C++. Далее сравнивается работа этих двух способов. Может ли что-то пойти не так?

```
#include <iostream>

void TryRead(int& number) {
  if (!(std::cin >> number)) {
    std::cout << "Fail on reading the number." << std::endl;
    exit(0);
  }
}

bool IsNotNegative(int number) {
    return abs(number) == number;
}

int main() {
  int number;

  std::cout << "Enter the number: ";
  TryRead(number);

  bool std_non_negative_check = (number >= 0);

  if (IsNotNegative(number) == std_non_negative_check) {
    std::cout << "Try again...." << std::endl;
  } else {
    std::cout << "Shit, you broke my program :(" << std::endl;
    exit(1);
  }

  return 0;
}
```

Листинг 1.1: Определение неотрицательности числа

1.3.2 Нахождение наибольшего общего делителя

Для запуска данной задачи нужен стандарт языка C++17. Ниже будет предоставлен код, который находит наибольший общий делитель двух натуральных чисел a и b разными способами: через реализованную функцию ***Gcd(int a, int b)*** и через стандартную функцию ***std::gcd()*** из библиотеки <***numeric***>. Затем сравнивается работа этих двух функций. Может ли что-то пойти не так?

```
#include <iostream>
#include <numeric>

int Gcd(int a, int b) {
  while (a * b) {
    int rem = a % b;
    a = b;
    b = rem;
  }
  return std::max(a, b);
}

void TryRead(int& number) {
  if (!(std::cin >> number)) {
    std::cout << "Fail on reading the number" << std::endl;
    exit(0);
  }
}

int main() {
  int a, b;

  std::cout << "Enter a: ";
  TryRead(a);

  std::cout << "Enter b: ";
  TryRead(b);

  if (a <= 0 || b <= 0) {
    std::cout << "Numbers should be positive" << std::endl;
    return 0;
  }
```

Листинг 1.2: Определение НОДа чисел

Продолжение см. на следующей странице:

```
  int gcd1 = Gcd(a, b);
  int gcd2 = std::gcd(a, b);
  if (gcd1 != gcd2) {
    std::cout << "Shit, you broke my program :(" << std::endl;
    return 1;
  } else {
    std::cout << "Try again..." << std::endl;
  }

  return 0;
}
```

Листинг 1.3: Определение НОДа чисел(продолжение)

Теоретические вопросы к задаче 1.3.2

На первый взгляд может показаться, что ответ к задаче абсолютно случаен, и найти его можно только случайно подбирая входные данные. Но на самом деле можно установить закономерность между входными данными, которые могут поломать программу. То есть можно описать множество ответов формально, а не перечислением.

Соответственно, к этой задаче есть несколько теоретических вопроса:

1. Найти самую первую пару решений. Отношение порядка элементов в множестве определим следующим образом:

   $$(x_1, y_1) < (x_2, y_2) \Rightarrow (x_1 < x_2) \vee (x_1 = x_2 \wedge y_1 < y_2),$$

   где «$\vee$» — логическое ИЛИ, «$\wedge$» — логическое И.

2. Найти формальное описание множества ответов.

3. Доказать полноту этого решения, то есть показать, что не существует ответов, которые не входят в представленное множество.