**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И**

**МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Лабораторная работа №2

по дисциплине:

«Системное программирование»

Выполнил:

Студент группы БФИ2102

Шаинян С.А.

Проверил:

Шананин В.А.

Москва, 2023

**Оглавление**

[Постановка задачи 3](#_Toc145522194)

[Контрольные значения 4](#_Toc145522195)

[Задание 1 5](#_Toc145522196)

[Задание 2 7](#_Toc145522197)

[Задание 3 9](#_Toc145522198)

[Задание 4 11](#_Toc145522199)

[Задание 5 13](#_Toc145522200)

# Постановка задачи

1. Напишите программу, которая вводит с клавиатуры натуральное число N и определяет его факториал, то есть произведение натуральных чисел от 1 до N. N! = 1 \* 2 \* 3 \* … \* N.
2. Ряд чисел Фибоначчи задаётся след. образом: первые два числа равны 1 (F1 = F2 = 1), а каждое следующее равно сумме двух предыдущих: Fn = Fn-1+Fn-2. Напишите программу, которая вводит натуральное число N и выводит на экран первые N чисел Фибоначчи.
3. Автоморфные числа. Натуральное число называется автоморфным, если оно равно последним цифрам своего квадрата. Например, 252=625. Напишите программу, которая вводит натуральное число N и выводит на экран все автоморфные числа, не превосходящие N.
4. Числа Армстронга. Натуральное число называется числом Армстронга, если сумма цифр числа, возведённых в N-ную степень (где N – кол-во цифр в числе) равна самому числу. Например, 153 = 13 + 53 + 33. Найдите все трёхзначные и четырёхзначные числа армстронга.
5. Алгоритм Евклида для вычисления наибольшего общего делителя двух натуральных чисел, формулируется так: нужно заменять большее число на разность большего и меньшего до тех пор, пока одно из них не станет равным нулю; тогда второе и есть НОД. Напишите программу, которая реализует этот алгоритм.

# Контрольные значения

Задание 1: 13

Задание 2: 42

Задание 3: 100

Задание 4: Задание не требует ввода данных

Задание 5: 36, 48

# Задание 1

C++

#include <iostream>

using namespace std;

int factorial(int n);

int main() {

    cout << (factorial(13)) << endl;

}

// Тот же подход, как и с числами Фибоначчи

int factorial(int n) {

    if (n == 0 || n == 1) {

        return 1;

    }

    int\* cache = new int[n + 1];

    cache[0] = 1;

    cache[1] = 1;

    for (int i = 2; i <= n; i++) {

        cache[i] = cache[i-1] \* i;

    }

    return cache[n];

}

C

#include <stdio.h>

#include <stdlib.h>

int factorial(int n);

int main() {

    printf("%d\n", factorial(13));

    return 0;

}

int factorial(int n) {

    if (n == 0 || n == 1) {

        return 1;

    }

    int\* cache = (int\*)malloc((n + 1) \* sizeof(int));

    cache[0] = 1;

    cache[1] = 1;

    for (int i = 2; i <= n; i++) {

        cache[i] = cache[i - 1] \* i;

    }

    int result = cache[n];

    free(cache);

    return result;

}

}

Результат



# Задание 2

C++

#include <iostream>

using namespace std;

int fibonacci(int n);

int main() {

    int n;

    cout << "Введите число n: ";

    cin >> n;

    cout << (fibonacci(n)) << endl;

}

int fibonacci(int n) {

    if (n == 0) {return 0;} // Фибоначчи от 0 -> 0

    if (n == 1 || n == 2) {return 1;} // Фибоначчи от 1, 2 -> 1

    int\* cache = new int[n + 1];

    // массив для прошлых чисел, ибо получение элемента через индекс на порядок быстрее рекурсии

    cache[0] = 0;

    cache[1] = 1;

    // начальные значения чисел Фибоначчи

    for (int i = 2; i <= n; i++) {

        cache[i] = cache[i - 1] + cache[i - 2]; // Функция Фибоначчи для вычисления числа

    }

    return cache[n];

}

C

#include <stdio.h>

#include <stdlib.h>

int fibonacci(int n);

int main() {

    int n;

    printf("Введите число n: ");

    scanf("%d", &n);

    printf("%d\n", fibonacci(n));

    return 0;

}

int fibonacci(int n) {

    if (n == 0) {

        return 0;

    }

    if (n == 1 || n == 2) {

        return 1;

    }

    int\* cache = (int\*)malloc((n + 1) \* sizeof(int));

    // malloc - возвращает указатель на начало выделенной области; (n + 1) \* sizeof(int) - размер массива

    cache[0] = 0;

    cache[1] = 1;

    for (int i = 2; i <= n; i++) {

        cache[i] = cache[i - 1] + cache[i - 2];

    }

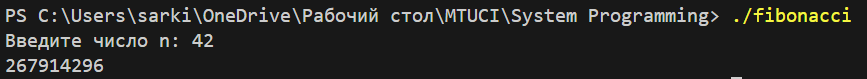
    int result = cache[n];

    free(cache); // освобождаем память

    return result;

}

Результат



# Задание 3

C++

#include <iostream>

#include <cmath>

using namespace std;

bool isAutomorphic(int num);

int main() {

    int n;

    cout << "Введите число n: ";

    cin >> n;

    for (int i = 0; i <= n; i++) {

        if (isAutomorphic(i)) {

            cout << i << '|' << i \* i << endl;

        }

    }

}

bool isAutomorphic(int num) {

    int square = num \* num; // вычисляем квадрат числа

    /\*

        Суть цикла -> сравнивать концовку числа, до тех пор пока она не отличается или число больше 0

    \*/

    while (num > 0) {

        if (num % 10 != square % 10) { // сравниваем концовки

            return false;

        }

        num /= 10;

        square /= 10;

    }

    return true;

}

C

#include <stdio.h>

#include <stdbool.h>

#include <math.h>

bool isAutomorphic(int num);

int main() {

    int n;

    printf("Введите число n: ");

    scanf("%d", &n);

    for (int i = 0; i <= n; i++) {

        if (isAutomorphic(i)) {

            printf("%d|%d\n", i, i \* i);

        }

    }

    return 0;

}

bool isAutomorphic(int num) {

    int square = num \* num;

    while (num > 0) {

        if (num % 10 != square % 10) {

            return false;

        }

        num /= 10;

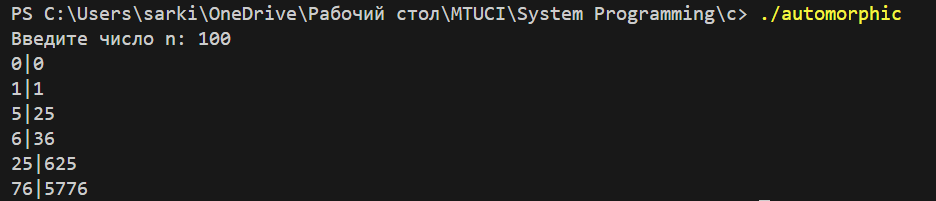
        square /= 10;

    }

    return true;

}

Результат



# Задание 4

C++

#include <iostream>

#include <string>

#include <cmath>

using namespace std;

bool armstrong(int num);

int main() {

    for (int i = 100; i <= 9999; i++) {

        if (armstrong(i)) {

            cout << i << " число Армстронга" << endl;

        }

    }

}

bool armstrong(int num) {

    string numAsString = to\_string(num); // представление числа в виде строки

    int len = numAsString.length(); // получаем длину числа

    int sum = 0;

    for (char digit : numAsString) { // проходим по "цифрам"

        int numericValue = digit - '0'; // приводим char к типу int ('9' - '0' == 9)

        sum += pow(numericValue, len) ;

    }

    return sum == num;

}

C

#include <stdio.h>

#include <stdbool.h>

#include <math.h>

bool armstrong(int num);

int main() {

    for (int i = 100; i <= 9999; i++) {

        if (armstrong(i)) {

            printf("%d число Армстронга\n", i);

        }

    }

    return 0;

}

bool armstrong(int num) {

    char numAsString[10]; // представление числа в виде строки (пусть будет до 10 цифр)

    sprintf(numAsString, "%d", num); // преобразуем число в строку (%d - digit)

    int len = 0;

    while (numAsString[len] != '\0') { // в с строки заканчиваются нулевым символом

        len++; // получаем длину числа

    }

    int sum = 0;

    for (int i = 0; i < len; i++) {

        int numericValue = numAsString[i] - '0'; // приводим char к типу int ('9' - '0' == 9)

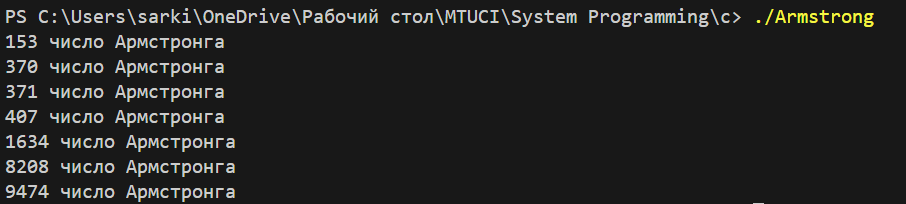
        sum += pow(numericValue, len);

    }

    return sum == num;

}

Результат



# Задание 5

C++

#include <iostream>

using namespace std;

int euclid(int dividerFirst, int dividerSecond);

int main() {

    cout << (euclid(36, 48)) << endl;

}

int euclid(int dividerFirst, int dividerSecond) {

    if (dividerFirst != dividerSecond) { // если числа равны, то это и является НОД

        if (dividerFirst > dividerSecond) {

            dividerFirst = dividerFirst - dividerSecond; // отнимаем у бОльшего делителя меньший

        } else {

            dividerSecond = dividerSecond - dividerFirst; // аналогично

        }

        return euclid(dividerFirst, dividerSecond); // рекурсия

    }

    return dividerSecond;

}

C

#include <stdio.h>

int euclid(int dividerFirst, int dividerSecond);

int main() {

    printf("%d\n", euclid(36, 48));

    return 0;

}

int euclid(int dividerFirst, int dividerSecond) {

    if (dividerFirst != dividerSecond) {

        if (dividerFirst > dividerSecond) {

            dividerFirst = dividerFirst - dividerSecond;

        } else {

            dividerSecond = dividerSecond - dividerFirst;

        }

        return euclid(dividerFirst, dividerSecond);

    }

    return dividerSecond;

}

Результат

