**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра інтелектуальних технологій**

Лабораторна робота № 3

з дисципліни «Алгоритмізація та програмування»

Тема роботи: «Програмування циклічного обчислювального процесу. Цикли з перед- та пост- умовою»

**Виконав студент**

**групи КН-12**

**Сімак О. С.**

**Перевірив**

**Круглов О. І.**

**Київ – 2020**

**Завдання 1**

Дано цілі додатні числа N і K. Використовуючи тільки операції додавання і віднімання, знайти частку від ділення без остачі N на K, а також остачу від цього ділення:

1. **Математична постановка задачі (МПЗ).**

*Вхідні дані*: K, N –змінні дійсного типу.

*Вихідні дані*: remainder, quotient – результати розрахунку дійсного типу.

*Математична модель задачі*: remainder = N, до тих пір, поки remainder >= K виконувати дію remainder = remainder – K. dividend = N – remainder, до тих пір, поки dividend != 0 виконувати дії dividend -= K; quotient++; Вивести quotient, remainder.

Подання МПЗ у вигляді таблиці:

|  |  |  |
| --- | --- | --- |
| Вхідні дані | Дії | Вихідні дані |
| K, N –змінні дійсного типу | 1) remainder = N, до тих пір, поки remainder >= K виконувати дію remainder = remainder – K.  2) dividend = N – remainder, до тих пір, поки dividend != 0 виконувати дії dividend -= K; quotient++;  3) Вивести quotient, remainder | remainder, quotient – результати розрахунку дійсного типу |

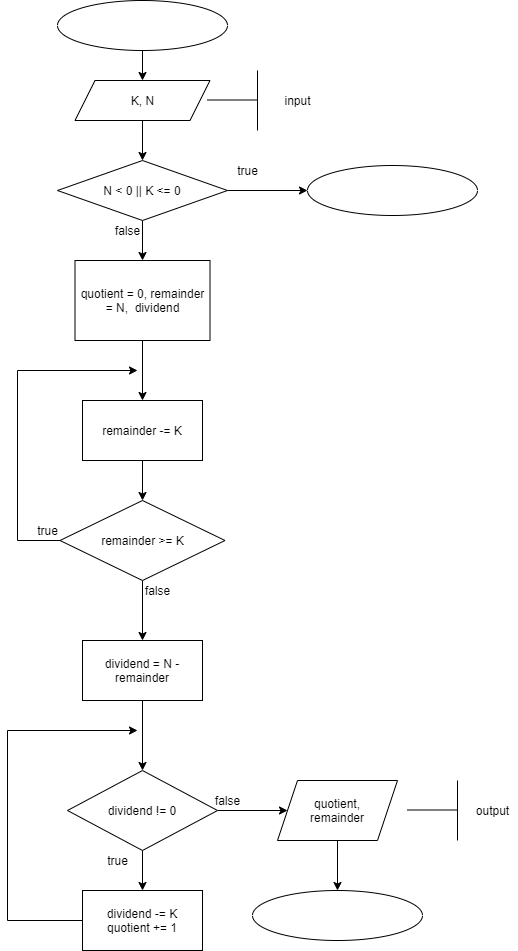
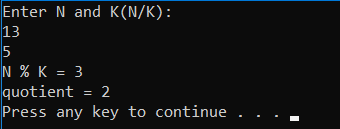


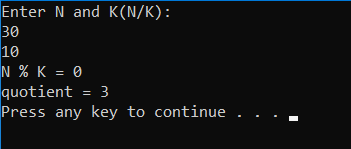
Рисунок 1 - Схема алгоритму задачі 1.

1. **Тестові приклади.**

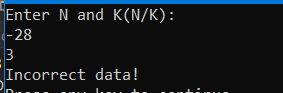
* N = 13; K = 5;



* N = 30; K = 10;



* N = -28; K = 3;



**3. Текст програми:**

#include <iostream>

#include <cmath>

using namespace std;

int main()

{

int K, N, quotient = 0, remainder, dividend;

cout << "Enter N and K(N/K): " << endl;

cin >> N >> K;

if (N < 0 || K <= 0) {

cout << "Incorrect data!" << endl;

return 0;

}

remainder = N;

do {

remainder -= K;

} while (remainder >= K);

dividend = N - remainder;

while (dividend != 0) {

dividend -= K;

quotient++;

}

cout << "N % K = " << remainder << endl;

cout << "quotient = " << quotient << endl;

return 0;

}

**Завдання 2**

Дано ціле число N (N > 1). Знайти найменше ціле число K, при якому виконується нерівність 3^𝐾 > 𝑁.

1. **Математична постановка задачі (МПЗ).**

*Вхідні дані*: N –змінна дійсного типу.

*Вихідні дані*: k – степінь дійсного типу.

*Математична модель задачі*: k=0; Доки 3^k <= N: k++; Вивести k

Подання МПЗ у вигляді таблиці:

|  |  |  |
| --- | --- | --- |
| Вхідні дані | Дії | Вихідні дані |
| N –змінна дійсного типу. | 1) k=0; Доки 3^k <= N: k++;  2) Вивести k | k – степінь дійсного типу. |

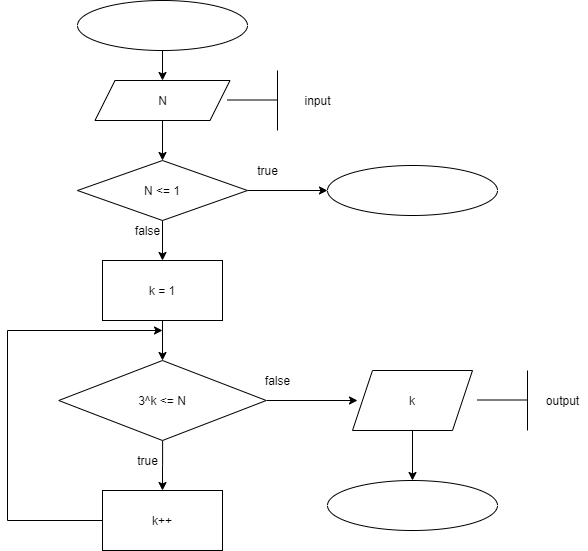
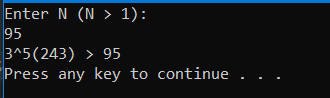


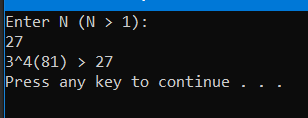
Рисунок 2 - Схема алгоритму задачі 2.

1. **Тестові приклади.**

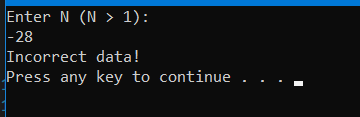
* N = 95;



* N = 27;



* N = -28;



1. **Текст програми:**

#include <iostream>

#include <cmath>

using namespace std;

int main()

{

int N, k = 1;

cout << "Enter N (N > 1): " << endl;

cin >> N;

if (N <= 1) {

cout << "Incorrect data!" << endl;

return 0;

}

while (pow(3, k) <= N) {

k++;

}

cout << "3^" << k << "(" << pow(3, k) << ") > " << N << endl;

return 0;

}

**Завдання 3**

Дано число A (А > 1). Вивести найбільше із цілих чисел K, для яких сума 1 + 1/2 + ... + 1 / K буде менше A, і саму цю суму.

1. **Математична постановка задачі (МПЗ).**

*Вхідні дані*: A –змінна дійсного типу.

*Вихідні дані*: k – максимальне значення дільника дробового типу, sum – сума всіх коефіцієнтів дробового типу.

*Математична модель задачі*: k = 1.0, sum = 0; Доки A > sum: k++;

sum += (1 / k); Якщо sum + (1 / k) >= A, то перейти на наступну ітерацію. Вивести k, sum.

Подання МПЗ у вигляді таблиці:

|  |  |  |
| --- | --- | --- |
| Вхідні дані | Дії | Вихідні дані |
| A –змінна дійсного типу. | 1) k = 1.0, sum = 0;  Доки A > sum: k++;  sum += (1 / k);  Якщо sum + (1 / k) >= A, то перейти на наступну ітерацію.  2) Вивести k, sum. | k – максимальне значення дільника дробового типу, sum – сума всіх коефіцієнтів дробового типу. |

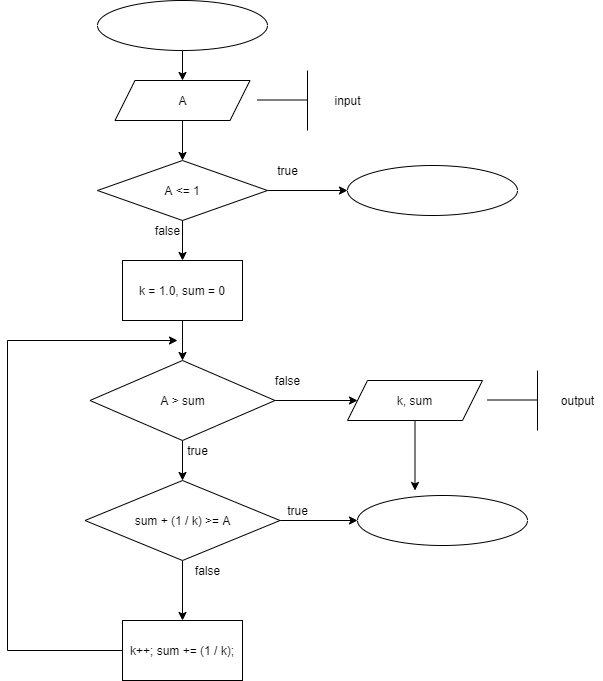
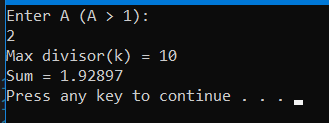


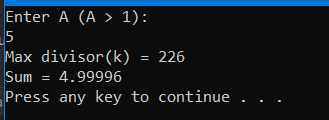
Рисунок 3 - Схема алгоритму задачі 3.

1. **Тестові приклади.**

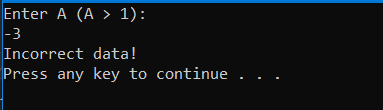
* N = 2;



* N = 5;



* N = -3;



1. **Текст програми:**

#include <iostream>

#include <cmath>

using namespace std;

int main()

{

int A;

double k = 1.0, sum = 0; // k - divisor

cout << "Enter A (A > 1): " << endl;

cin >> A;

if (A <= 1) {

cout << "Incorrect data!" << endl;

return 0;

}

while (A > sum) {

if (sum + (1 / k) >= A) break;

k++;

sum += (1 / k);

}

cout << "Max divisor(k) = " << k << endl;

cout << "Sum = " << sum << endl;

return 0;

}

**Завдання 4**

Дано ціле число N (N > 0). За допомогою операцій ділення і остачі від ділення поміняти місцями сусідні цифри числа, що стоять у парних та непарних розрадах (наприклад, для 12345 отримуємо 13254).

1. **Математична постановка задачі (МПЗ).**

*Вхідні дані*: N –змінна дійсного типу.

*Вихідні дані*: acc – нове число дійсного типу.

*Математична модель задачі*: nCopy = N, rank = 0; Доки nCopy >= 1: nCopy /= 10; rank++; Якщо rank == 2, то acc = N; В іншому випадку: nexRankN = 0, reverseRankCounter = 0; nCopy = 0; acc = (N / pow(10, rank - 1)); Доки rank >= 1: reverseRankCounter++; nCopy = (N / pow(10, rank - 1));nexRankN = (N / pow(10, rank - 2));nCopy = nCopy % 100 % 10;nexRankN = nexRankN % 100 % 10; rank--; Якщо !(reverseRankCounter % 2), то acc = acc \* 10 + nCopy; Якщо nexRankN == 0, то acc = acc, в іншому випадку - acc = acc \* 10 + nexRankN; Вивести acc;

Подання МПЗ у вигляді таблиці:

|  |  |  |
| --- | --- | --- |
| Вхідні дані | Дії | Вихідні дані |
| N –змінна дійсного типу. | 1) nCopy = N, rank = 0; Доки nCopy >= 1: nCopy /= 10; rank++; Якщо rank == 2, то acc = N; В іншому випадку: nexRankN = 0, reverseRankCounter = 0; nCopy = 0; acc = (N / pow(10, rank - 1)); Доки rank >= 1: reverseRankCounter++; nCopy = (N / pow(10, rank - 1));nexRankN = (N / pow(10, rank - 2));nCopy = nCopy % 100 % 10;nexRankN = nexRankN % 100 % 10; rank--; Якщо !(reverseRankCounter % 2), то acc = acc \* 10 + nCopy; Якщо nexRankN == 0, то acc = acc, в іншому випадку - acc = acc \* 10 + nexRankN;  2) Вивести acc; | acc – нове число дійсного типу. |

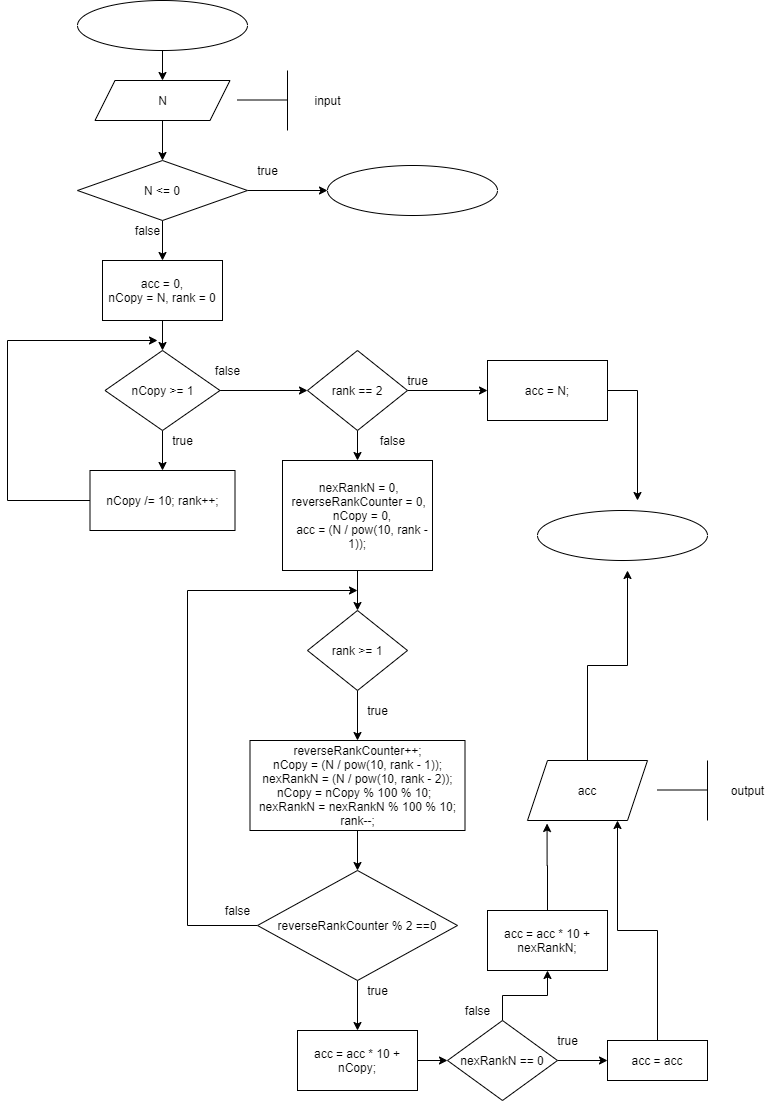
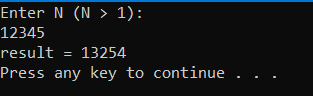


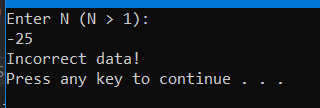
Рисунок 4 - Схема алгоритму задачі 4.

1. **Тестові приклади.**

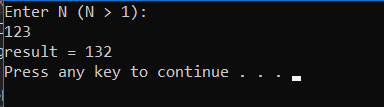
* N = 12345;



* N = -25;



* N = -3;



1. **Текст програми:**

#include <iostream>

#include <cmath>

using namespace std;

int main()

{

// поменять - парный-непарный разряд 213 -> 231

int N, acc = 0;//acc - accumulator, накопитель для нового числа

cout << "Enter N (N > 1): " << endl;

cin >> N;

int nCopy = N, rank = 0; //копия числа и счетчик кол-ва его разрядов начиная с последнего

if (N <= 0) {

cout << "Incorrect data!" << endl;

return 0;

}

// накапливаем количество разрядов числа

while (nCopy >= 1) {

nCopy /= 10;

rank++;

}

if (rank == 2) { // проверка базового случая, если число из двух цифр - менять нечего, соответственно, нужно вернуть само число

acc = N;

}

else {

int nexRankN = 0, reverseRankCounter = 0; //nexRankN - значение следующей цифры | reverseRankCounter - счетчик кол-ва разрядов с начала

nCopy = 0; // обнуляем копию числа, потому что будем считать его по новому

acc = (N / pow(10, rank - 1)); // устанавливаем начальное значение накопителю(то есть цифру, к которой будут добавляться остальные) - первую цифру оригиального числа, оно не парное и от него ничего не зависит

while (rank >= 1) {

reverseRankCounter++; //увеличиваем количество обратных разрядов(rank начинает с последнего, reverseRankCounter - с первого)

nCopy = (N / pow(10, rank - 1)); // текущий разряд

nexRankN = (N / pow(10, rank - 2)); // следующий разряд

nCopy = nCopy % 100 % 10; // N % 100 % 10 - получить последнюю цифру числа текущеко разряда

nexRankN = nexRankN % 100 % 10; // последняя цифра числа следующего разряда

if (!(reverseRankCounter % 2)) { // если разряд парный

if (nexRankN == 0) acc = acc; // если следующего разряда не существует, ничего не меняем

else acc = acc \* 10 + nexRankN; // накапливаем новое число

acc = acc \* 10 + nCopy;

}

rank--; // уменьшаем кол-во разрядов числа, чтобы выйти из цикла, когда они закончатся

}

}

cout << "result = " << acc << endl;

return 0;

}

**Завдання 5**

Дано ціле число N (N > 1), що є числом Фібоначчі: 𝑁 = Послідовність чисел Фібоначчі 𝐹𝑘 визначається наступним чином: = 1, = 1, = + , де k = 3, 4, .... Знайти цілі числа = + - попереднє і наступне числа Фібоначчі.

1. **Математична постановка задачі (МПЗ).**

*Вхідні дані*: N –змінна дійсного типу.

*Вихідні дані*: prev, next– попереднє та наступне число фібоначчі дійсного типу.

*Математична модель задачі*: prev = 0, next = 1, Робити:

currentF = prev + next; prev = next; next = currentF; Якщо currentF == N, то next += prev;, Поки: currentF != N. Вивести next та prev.

Подання МПЗ у вигляді таблиці:

|  |  |  |
| --- | --- | --- |
| Вхідні дані | Дії | Вихідні дані |
| N –змінна дійсного типу. | 1) prev = 0, next = 1, Робити:  currentF = prev + next; prev = next; next = currentF; Якщо currentF == N, то next += prev;, Поки: currentF != N.  2) Вивести next та prev. | prev, next– попереднє та наступне число фібоначчі дійсного типу. |

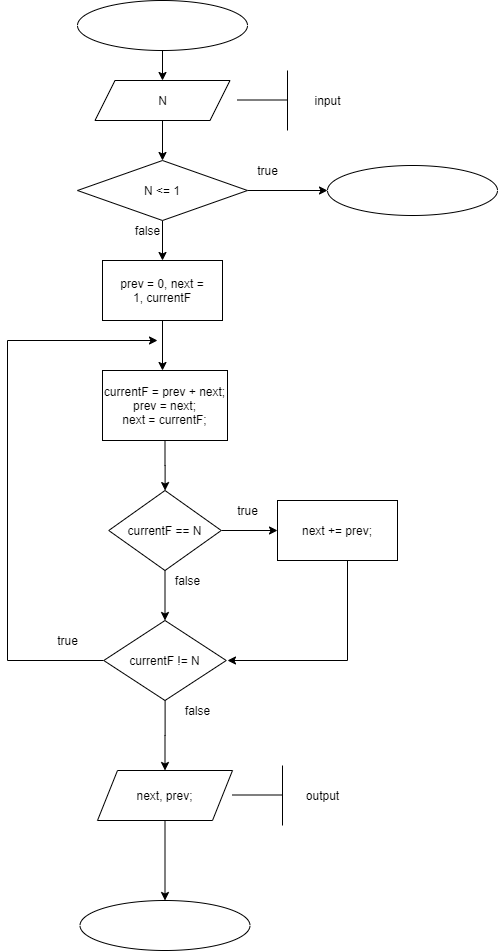
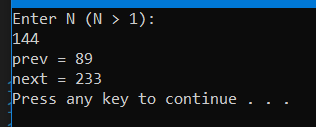


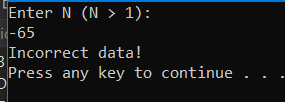
Рисунок 5 - Схема алгоритму задачі 5.

1. **Тестові приклади.**

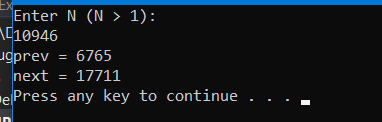
* N = 144;



* N = -65;



* N = 10946;



1. **Текст програми:**

#include <iostream>

#include <cmath>

using namespace std;

int main()

{

int N, prev = 0, next = 1;

cout << "Enter N (N > 1): " << endl;

cin >> N;

if (N <= 1) {

cout << "Incorrect data!" << endl;

return 0;

}

int currentF; // для своей последовательности фибоначчи

do {

currentF = prev + next;

prev = next;

next = currentF;

if (currentF == N) {

next += prev;

}

} while (currentF != N);

cout << "prev = " << prev << endl << "next = " << next << endl;

return 0;

}