**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ образовательное учреждение высшего образования**

**«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Кафедра вычислительной техники**

**КОНТРОЛЬНАЯ РАБОТА**

**по дисциплине «ИНФОРМАТИКА»**

|  |  |
| --- | --- |
| Факультет: АВТ  Группа: ДТ-460а  Студент: Пантюхин А.Е. | Преподаватель:  Копылова О.А. |

Новосибирск, 2025 г.

Содержание

[ЗАДАНИЕ №1. Анализ программ 3](#_Toc1)

[Решение   3](#_Toc2)

[Пояснения к коду: 4](#_Toc3)

[Пример работы программы 4](#_Toc4)

[ЗАДАНИЕ №2. Строки 4](#_Toc5)

[Решение 5](#_Toc6)

[Пояснения к коду: 5](#_Toc7)

[Пример работы программы 6](#_Toc8)

[ЗАДАНИЕ №3. Итерационные циклы 6](#_Toc9)

[Решение: 6](#_Toc10)

[Пример работы программы 8](#_Toc11)

[Выводы 8](#_Toc12)

## ЗАДАНИЕ №1. Анализ программ

Содержательно сформулировать результат выполнения функции, определить «смысл» отдельных переменных, найти стандартные контексты, их определяющие, написать вызов функции.

В тексте программы в комментариях указать назначение основных переменных. Для функций в комментариях указать их назначение, а также назначения входных и выходного параметров.

Вариант задания 8:

int F8(int a) {

int n,k,s;

for (n=a, s=0; n!=0; n=n/10)

{ k=n%10; s=s+k;}

return s;

### Решение

Исходный код программы с комментариями и вызовом функции:

#include <stdio.h>

int F8(int a) { //на вход принимаем целое число

int n,k,s; //объявляем три переменных, n - бесполезная

//k тоже бесполезная, но чуть менее.

/\*

делим число нацело на 10, пока оно не станет равным нулю

\*/

for (n = a,s=0; n!=0;n=n/10) {

k = n%10; //на каждой итерации деления записываем остаток в k

s=s+k; //прибавляя к s - в итоге получим сумму остатков от деления на 10.

}

//возвращаем сумму остатков

return s;

}

int main(void) {

fprintf(stdout, "F8(18745)=%d\n", F8(18745));

fprintf(stdout, "F8(-648)=%d\n", F8(-648));

return 0;

}

### Пояснения к коду:

Функция возвращает сумму цифр полученного на вход числа. Переменная n бесполезна по причине того, что переменная a – уже копия исходного значения, и мы могли использовать её безо всяких проблем. Аналогично, вычисление остатка можно было сократить до s+=n%10.

Данная функция при работе с отрицательными числами выдаст отрицательную же сумму цифр.

### Пример работы программы

Пример работы программы представлен на рисунке:



Рисунок 1 – Пример работы программы

## ЗАДАНИЕ №2. Строки

Содержательно определите действие, производимое над строкой. Напишите вызов функции (входные неизменяемые строки могут быть представлены фактическими параметрами – строковыми константами). В тексте программы в комментариях указать назначение основных переменных. Для функций в комментариях указать их назначение, а также назначения входных и выходного параметров.

Вариант задания 7:

int F7(char c[])

{ int i,s;

for (i=0; c[i] !='\0'; i++)

if (c[i] >='0' && c[i]<='7') break;

for (s=0; c[i] >='0' && c[i] <='7'; i++)

s = s \* 8 + c[i] - '0';

return s;

}

### Решение

#include <stdio.h>

int F7(char c[]) { //на вход получаем строку

int i,s; //объявляем счётчики

/\*

движемся по строке, пока не встретим

цифру в восьмеричной системе счисления

\*/

for (i=0; c[i] !='\0'; i++)

if (c[i] >='0' && c[i]<='7') break;

/\*

с места встречи считаем данный участок строки

числом в восьмеричной с.с., до тех пор, пока

не встретим неподходящий символ

\*/

for (s=0; c[i] >='0' && c[i] <='7'; i++)

//в s формируется равное число в десятичной с.с.

s = s \* 8 + c[i] - '0';

//возвращаем число в десятичной с.с.

return s;

}

int main(void) {

printf("F7(\"fd44\")==%d\n", F7("fd44"));

printf("F7(\"fd22sd884\")==%d\n", F7("fd22sd884"));

return 0;

}

### Пояснения к коду:

Функция возвращает число в десятичной системе счисления, равное первому найденному во входной строке числу в восьмеричной системе счисления.

Функция воспринимает любое число как положительное.

### Пример работы программы



Рисунок 2 – Пример работы программы

## ЗАДАНИЕ №3. Итерационные циклы

Для заданного варианта написать функцию вычисления суммы ряда. Стандартный диапазон значений x =0.1...1.0, шаг 0.1. Убедиться, что на этом диапазоне элемент суммы ряда стремится к 0, при необходимости скорректировать диапазон. Вычислить значения суммы ряда и контрольной функции, к которой он сходится, с точностью до 2, 4 и 5 знаков после запятой.

В тексте программы в комментариях указать назначение основных переменных. Для функций в комментариях указать их назначение, а также назначения входных и выходного параметров.

Вариант задания 4:

Функция: ln(x)

Ряд:



### Решение:

#include <stdio.h>

#include <math.h>

#include <locale.h>

double getValue(double x, double n) {

return (double)(pow(x-1.0f, n)/(n\*pow(x, n))); //вычисляем значение n-го члена ряда

}

double sumSequence(double x, double epsilon) {

double result = 0.0f; //изначально сумма равна нулю

int idx = 1; //ряд начинается с первого элемента, n==1

double elem = getValue(x, idx); //вычисляем первый элемент

while (fabs(elem) > epsilon) { //пока n-й элемент по модулю меньше нашего "эпсилон"

result += elem; //суммируем элементы

elem = getValue(x, (double)++idx); //вычисляем следующий (++idx обеспечивает увеличение счётчика ДО копирования значения в функцию)

}

return result;//возвращаем результат

}

int main(void) {

setlocale(LC\_ALL, "RUSSIAN"); //установили локаль

const double startX = 0.5f; //начало диапазона значений X

const double increment = 0.1f; //приращение X

const double target\_epsilon[3] = {1e-2, 1e-4, 1e-5}; //целевые значения точности

double referenceResult[3][10] = {}; //массив под результаты ln(x)

double sumSequenceResult[3][10] = {}; //массив под результаты sumSequence(x, epsilon)

for (int i = 0; i < 3; i++) { //вычисляем сразу три точности

for (int j = 0; j < 10; j++) {// для каждой - по 10 экземпляров значения функции и суммы ряда

referenceResult[i][j] = log(startX+increment\*j); //несмотря на то, что i,j - int, их можно так использовать

sumSequenceResult[i][j] = sumSequence(startX+increment\*j, target\_epsilon[i]);

}

}

/\*

Форматированный вывод

\*/

printf("Точность сходимости: %.e\t\t\t", target\_epsilon[0]);

printf("Точность сходимости: %.e\t\t\t", target\_epsilon[1]);

printf("Точность сходимости: %.e\n", target\_epsilon[2]);

printf("%8s\t%10s\t%8s\t", "X", "Сумма ряда", "ln(x)");

printf("%8s\t%10s\t%8s\t", "X", "Сумма ряда", "ln(x)");

printf("%8s\t%10s\t%8s\n", "X", "Сумма ряда", "ln(x)");

for (int row = 0; row < 10; row++) { //выводим значения по рядам.

for (int i = 0; i < 3; i++) {

printf("%8.1lf\t%10.5lf\t%8.5lf\t", startX+0.1f\*(double)(row), sumSequenceResult[i][row], referenceResult[i][row]);

}

printf("\n");

}

return 0;

}

Получение значения n-го элемента ряда было решено вынести в отдельную функцию для улучшения читабельности кода. Было обнаружено, что стандартный диапазон значений показывает расходимость на промежутке [0.1;0.4], диапазон скорректирован до [0.5;1.4].

### Пример работы программы

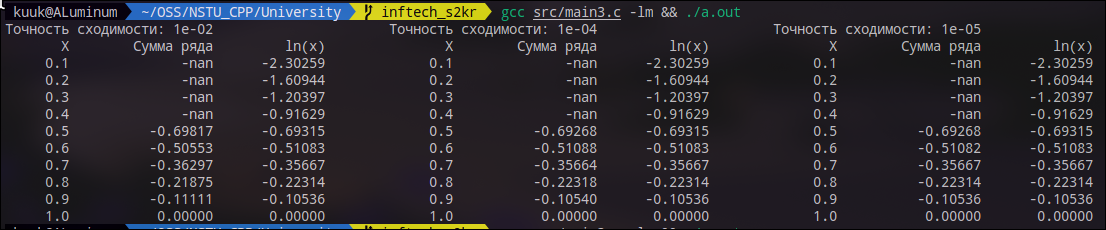


Рисунок 3 – результаты без корректировки диапазона.

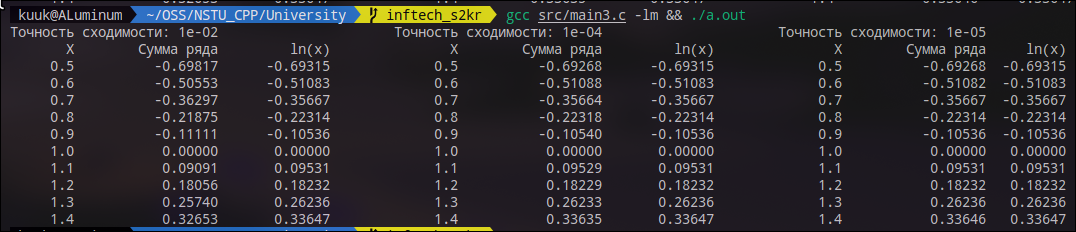


Рисунок 4 – результаты на скорректированном диапазоне.

## Выводы

В рамках данной контрольной работы были закреплены навыки работы с языком C, я получил возможность продемонстрировать знание и понимание языковых конструкций и базовых алгоритмов.