Государственное образовательное учреждение высшего образования

«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина»

090302 200008 006

**«Технологии обработки информации»**

Семестр 4

**ОТЧЁТ**

по лабораторным работам

Лабораторная работа №3



Преподаватель: Плотников В.Ю.

Студент : Киселев В.А.

Группа : ФТ — 200008

Екатеринбург 2022 г.

Оглавление

[Цель работы 3](#_Toc105290886)

[Постановка задачи 3](#_Toc105290887)

[Средства разработки 3](#_Toc105290888)

[Описание алгоритма 3](#_Toc105290889)

[Реализация 3](#_Toc105290890)

[С++ 3](#_Toc105290891)

[Python 4](#_Toc105290892)

[Результаты 5](#_Toc105290893)

[C++ 5](#_Toc105290894)

[Python 6](#_Toc105290895)

[Анализ результатов 6](#_Toc105290896)

[Выводы 7](#_Toc105290897)

# Цель работы

Исследовать поведение различных представлений вещественных чисел при округлении на примере натуральных дробей вида 1/n. Для различных диапазонов значений n определить долю таких дробей, которые при умножении на n дают неверный результат. Исследовать зависимость этой доли от величины диапазона. Оформить отчёт о проделанной работе.

# Постановка задачи

1) Разработать алгоритм

2) Реализация алгоритма

3) Провести эксперимент

4) Анализ результатов

5) Выводы

# Средства разработки

1) CLion 2021.3.3

2) PyCharm Professional 2021.3.2

3) WebStorm 2021.3.2

4) Google Chrome 98.0.4758.102

# Описание алгоритма

Мы делим единицу на счётчик, который каждую итерацию растёт. Если умножение частного на счётчик не вернуло единицу, значит произошёл сбой. Производим расчёт доли неправильных дробей.

# Реализация

## С++

#include <fstream>  
#include <iostream>  
#include <iomanip>  
  
#define numberOfExaminationPasses 250000  
  
std::ofstream myFile;  
  
template <typename T> T processing (T variable) {  
  
 std::string file\_name = typeid(variable).name();  
 file\_name += "\_research.csv";  
  
 myFile.open(file\_name);  
 if(!myFile.is\_open()){  
 std::cout << "ERROR: The program was unable to create an output file." << std::endl  
 << "The program ended with an error";  
 return -1;  
 }  
  
 T divideResult = T (0.0);  
 T multiplyResult = T (0.0);  
 int accumulatedErrors = 0;  
 long double errorAccumulationFactor = 0;  
  
 myFile << "number;accumulated errors;" << std::endl;  
 for (int i = 1; i < numberOfExaminationPasses; ++i) {  
 divideResult = T(1.0) / T(i);  
 multiplyResult = divideResult \* T(i);  
 if (multiplyResult != T(1.0)) {  
 accumulatedErrors++;  
 errorAccumulationFactor = (long double) accumulatedErrors / (long double) i;  
 std::cout << i << ";" << errorAccumulationFactor << ";" << std::endl;  
 myFile << i << ";" << errorAccumulationFactor << ";" << std::endl;  
 }  
 }  
  
 myFile.close();  
 return 0;  
}  
  
  
int main() {  
  
 float a = 1.0;  
 double b = 1.0;  
 long double c = 1.0;  
  
 processing(a);  
 processing(b);  
 processing(c);  
  
 return 0;  
}

## Python

def Processing(i):  
 accumulated\_errors = 0  
 numberOfExaminationPasses = 10000000  
  
 myFile = open("graph.csv", "w")  
  
 for i in range(i, numberOfExaminationPasses + 1):  
  
 divide\_result = 1 / i  
 multiply\_result = divide\_result \* i  
  
 if multiply\_result != 1:  
 accumulated\_errors += 1  
 errorAccumulationFactor = accumulated\_errors / i  
 string = str(i) + ";" + str(errorAccumulationFactor).replace('.', ',') + "\n"  
 myFile.writelines(string)  
  
 return accumulated\_errors  
  
  
print(Processing(1))

# Результаты

## C++

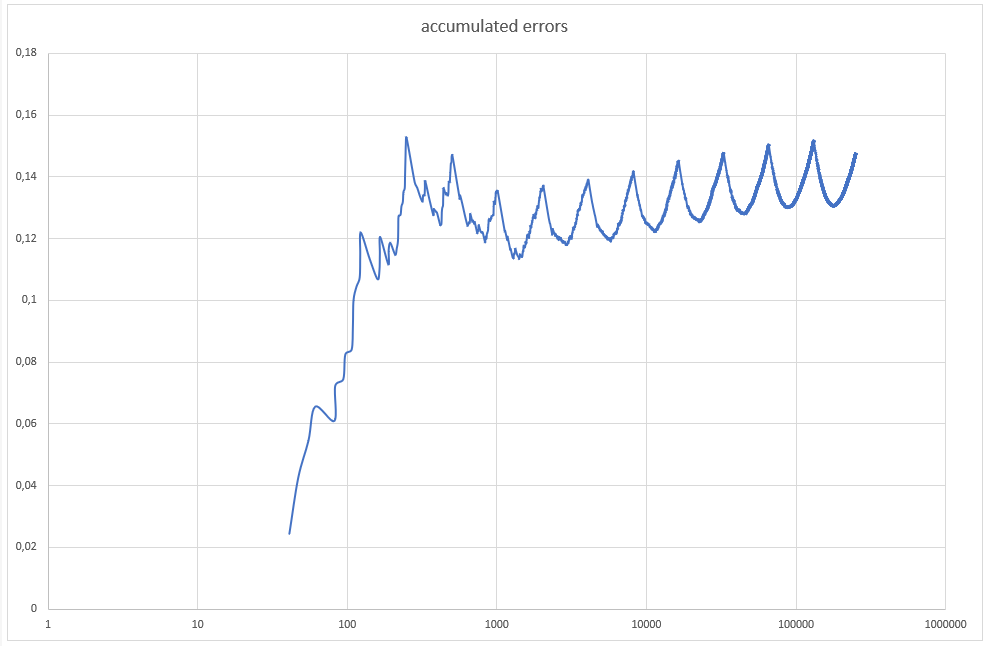


Рис 1.1 Float

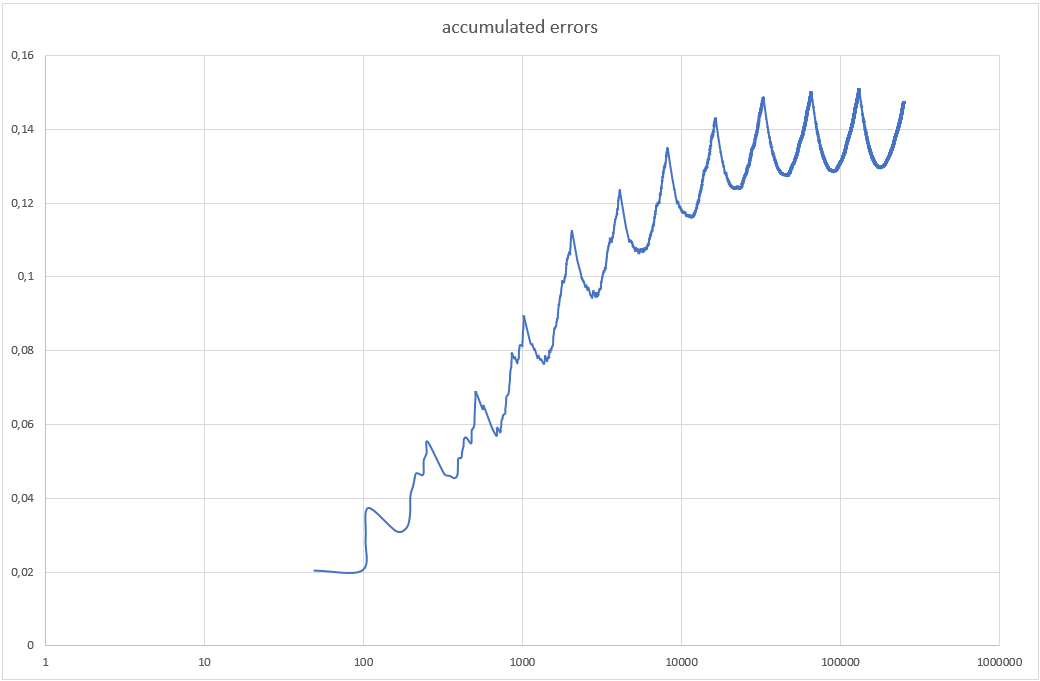


Рис 1.2 Double

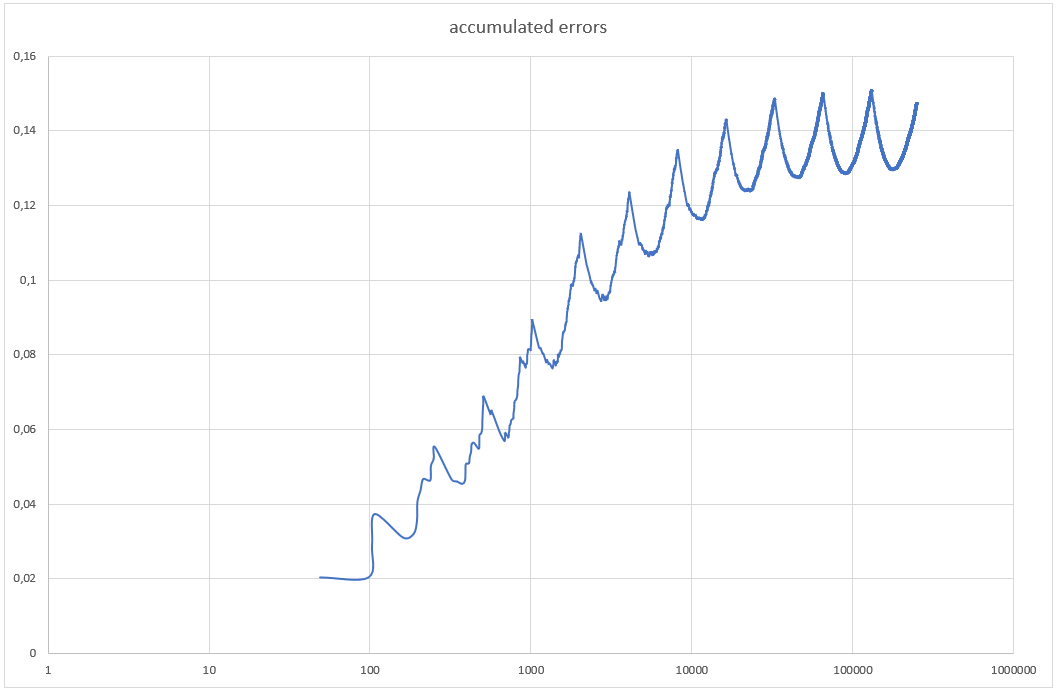


Рис 1.3 Long Double

## Python

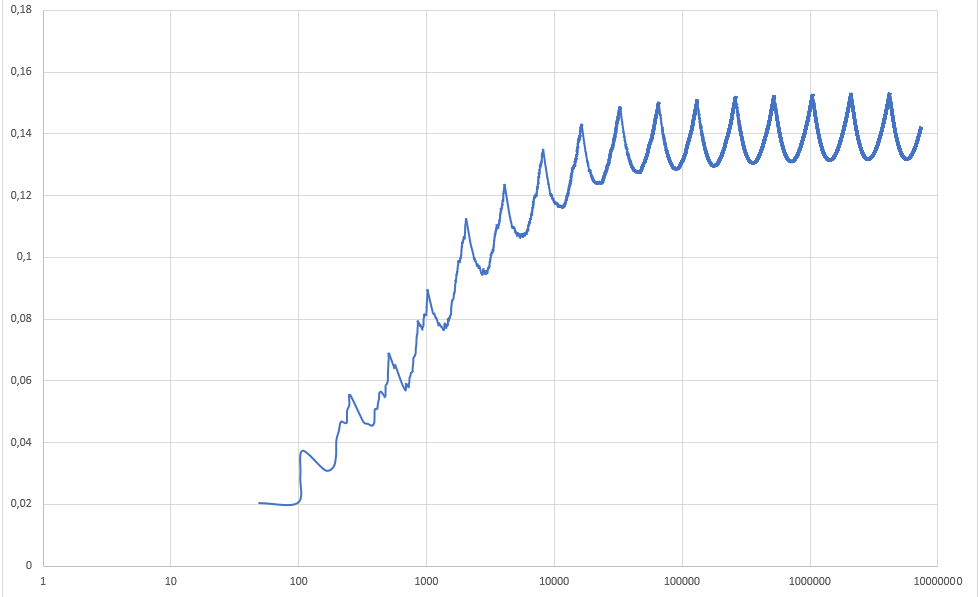


Рис 2.1 График вычислений на Python

# Анализ результатов

Графики представлены с логарифмической шкалой по горизонтальной шкале. Как мы можем заметить то с точки n = 41 начинается возрастание графиков вне зависимости от языка, далее заметны колебания. На графиках с логарифмической шкалой расстояние одинаковое между пиками на графике. Это значит, что пики находятся на расстоянии кратном 2. Данный эффект объясним тем, что представление чисел с плавающей запятой состоит из порядка и мантиссы. Один период соответствует изменению порядка на единицу. И именно один порядок проходит между двумя пиками.

Так же мы можем наблюдать, что каждое седьмое округление проходит некорректно. Это объясняется тем, что при округлении используется 3 дополнительных бита. В них используется цифры до 1112, то есть 710.

# Выводы

Значение максимумов на графиках

Для Float: 0,15261.

Для Double: 0,150877.

Для Long double: 0,150877.

Double и Long double совпали.

На графиках видно, что после достижения максимального значения они стабилизируются. Так же кроме роста, графики показывают периодическое понижение и повышение. Обратив внимание на расстояние между максимумами этих периодических колебаний, то заметим, что они равно удалены, то есть каждый максимум удален от другого на расстояние кратное 2.

1/n при умножении на n не всегда дает результат равный 1, это связано с тем, что мы работаем с двоичным форматом, где запись производится в экспоненциальном виде.

Что интересно, в Phyton и JavaScript значения совпадают с double и long double на c++, равны 0,150877. Можно сделать вывод, что в этих языках поведение при делении одинаково и соответствует представлению double.