НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ

Отчет к лабораторной работе №6

Дисциплины «Архитектура вычислительных систем»

Выполнил студент группы А-13-22

Кокляева Мария

Преподаватель: Шамаева О. Ю.

Москва 2024

# 1 Цель работы:

Освоение методов и алгоритмов перевода чисел из одной позиционной СС в другую позиционную систему и разработка соответствующего программного продукта.

## Постановка задачи:

Разработать алгоритм и создать программу, реализующую перевод чисел из 10 СС в 4 СС на языке Bash.

# 2.Данные

Входные данные

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип** | **Имя** | **Смысл** | **Структура** |
| Текстовый файл | Input.txt | Текстовый файл содержащий числа в 10 СС | С/П |

Промежуточные данные

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип** | **Имя** | **Смысл** | **Структура** |
| строка | num | Переменная, в которой хранится каждое число, прочитанное из input.txt | С/П |
| строка | integer\_part | Целая часть в 10 СС | С/П |
| строка | fractional\_part | Дробная часть в 10 СС | С/П |
| строка | integer\_result | Целая часть в 4 СС | С/П |
| строка | fractional\_result | Дробная часть в 4 СС | С/П |
| строка | base4\_num | Переведенное число из 10 СС в 4 СС | С/П |
| Цел. | precision | Точность | П/П |

Выходные данные

Вывод производится в консоль.

# 3.Входная форма.

Input.txt

123

-45.67

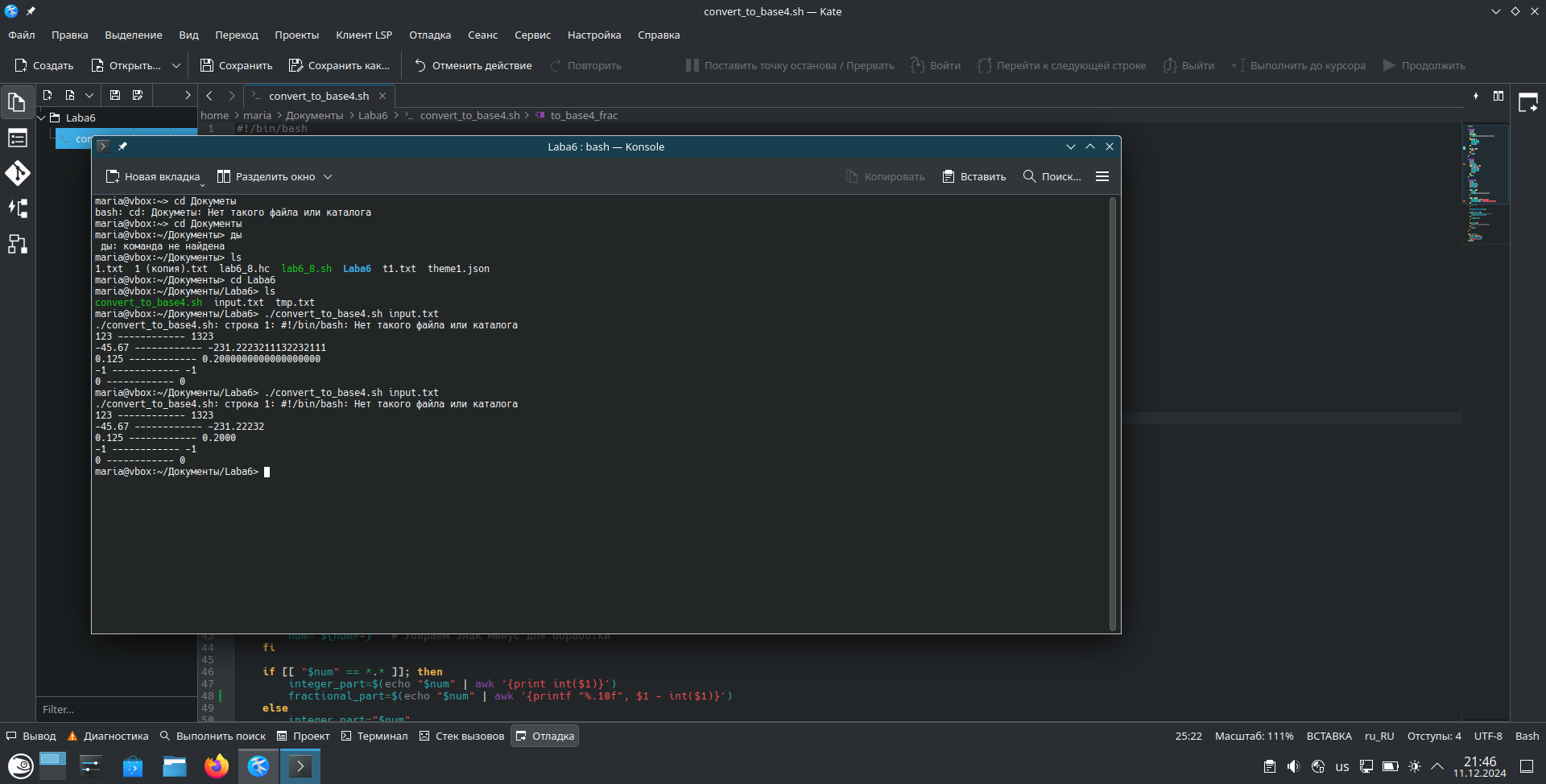
0.125

-1

0

# 4.Выходная форма.

Консоль



# 5.Метод

* Программа считывает числа из файла input.txt построчно. Пустые строки пропускаются.
* Перед обработкой каждое число проверяется на наличие недопустимых символов (всех символов кроме цифр, точки и минуса). Если обнаружены недопустимые символы, выводится сообщение об ошибке, и обработка числа пропускается.
* Если число вещественное (содержит точку), оно разделяется на целую и дробную части. Если число целое, дробная часть отсутствует.
* Целая часть преобразуется в четверичную систему счисления с помощью функции to\_base4\_int. Преобразование производится путем многократного деления на 4 и записи остатков в обратном порядке. Если целая часть равна 0, то функция возвращает строку “0”.
* Дробная часть преобразуется в четверичную систему счисления с помощью функции to\_base4\_frac. Эта функция использует алгоритм умножения на 4 и взятия целой части. Результат формируется путем конкатенации целых частей результатов умножения. Точность результата ограничивается переменной precision.
  + Конкатенация — операция склеивания объектов линейной структуры, обычно строк.
* Результаты преобразования целой и дробной частей (если дробная часть присутствует) объединяются с точкой в качестве разделителя, формируя четверичное представление исходного числа.
* Если исходное число было отрицательным, добавляется знак минус к результату.
* Преобразованное число выводится на консоль вместе с исходным числом.

# 6.Аномалии

Программа обрабатывает ошибки, связанные с некорректным форматом входных данных. Если обнаружены недопустимые символы, выводится сообщение об ошибке «Error», и обработка строки пропускается.

# 7.Тесты

|  |  |  |
| --- | --- | --- |
| № | Входные данные | Результат |
| 1 | 123  -45.67  0.125  -1  0 |  |
| 2 | -7fsgsj8.346g |  |

**#!/bin/bash**to\_base4\_int() {  
 local num="$1" #объявляет локальную переменную и присваивает ей значение первого аргумента, переданного функции.  
 local base4=""  
 if (( num < 0 )); then  
 num=$(( -num )) # Преобразуем в положительное для конвертации  
 fi  
 while (( num > 0 )); do  
 remainder=$(( num % 4 ))  
 base4="$remainder$base4"  
 num=$(( num / 4 ))  
 done  
  
 if [[ -z "$base4" ]]; then #если строка пустая записываем вначале 0  
 echo "0"  
 else  
 echo "$base4"  
 fi  
}  
  
to\_base4\_frac() {  
 local num="$1"  
 local base4=""  
 local precision=5  
 for (( i = 0; i < precision; i++ )); do  
 num=$(echo "$num \* 4" | bc) # (Basic Calculator).  
 local int\_part="${num%%.\*}" #из num удаляется все после точки, остаётся только целая часть.  
 base4="$base4$int\_part" #Конкатенация  
 num=".${num#\*.}" #Обновляем значение num, оставляя только дробную часть результата умножения.  
 done  
 echo "$base4"  
}  
  
convert\_to\_base4() {  
 local num="$1"  
 local integer\_part=""  
 local fractional\_part=""  
 local is\_negative=0  
  
 if [[ "$num" =~ [^0-9.-] ]]; then  
 echo "Error"  
 return 1 # Возвращаем код ошибки  
 fi  
  
 if [[ "$num" == -\* ]]; then  
 is\_negative=1  
 num="${num#-}" # Убираем знак минус для обработки  
 fi  
  
 if [[ "$num" == \*.\* ]]; then  
 integer\_part=$(echo "$num" | awk '{print int($1)}')  
 fractional\_part=$(echo "$num" | awk '{printf "%.10f", $1 - int($1)}')  
 else  
 integer\_part="$num"  
 fi  
  
 integer\_result=$(to\_base4\_int "$integer\_part")  
  
 if [[ -n "$fractional\_part" ]]; then # -n истинно если строка содержит хотя бы 1 символ  
 fractional\_result=$(to\_base4\_frac "$fractional\_part")  
 result="$integer\_result.$fractional\_result"  
 else  
 result="$integer\_result"  
 fi  
  
 if (( is\_negative )); then  
 echo "-$result" # Добавляем знак минус обратно  
 else  
 echo "$result"  
 fi  
}  
  
while IFS= read -r num; do #чтения одной строки   
 if [[ -z "$num" ]]; then continue; fi # -z возвращает true (истина) или false  
 base4\_num=$(convert\_to\_base4 "$num")  
 echo "$num ------------ $base4\_num"  
done < input.txt #Конец цикла while input.txt указан в качестве источника данных для цикла. Каждая строка из этого файла будет читаться и обрабатываться в цикле.

IFS=  Устанавливает внутренний разделитель полей (Internal Field Separator) в пустую строку. Это предотвращает разбор пробелов в строке.