Лабораторная работа №1

Представление чисел

Инструментарий и требования к работе

Работа выполняется на C/C++ (C11 и новее / C++20), Python (3.11.5) или Java (Temurin-17.0.8.1+1). Требования для всех работ: Правила оформления и написания работ.

Задание

Необходимо написать программу, которая позволяет выполнять арифметические действия с дробными числами в форматах с фиксированной и плавающей точкой. Программа должна использовать только целочисленные вычисления и типы данных.

Аргументы программе передаются через командную строку в одном из двух вариантов:

1. **<формат> <округление> <число>**
2. **<формат> <округление> <число1> <операция> <число2>**

где:

- формат – задаёт формат представления входных чисел. Возможные варианты:

формат	пояснение
A.B	числа с фиксированной точкой, где A и B – неотрицательные целые числа, обозначающие целую и дробную часть. Гарантируется, что A+B<=32 и A>=1.
h	числа с плавающей точкой половинной точности (half precision floating point IEEE-754), 16 бит.
f	формат числа с плавающей точкой одинарной

	точности (single precision floating point IEEE-754), 32 бита.

Числа с фикс. точкой – числа со знаком в дополнении до 2.

- округление – задаёт тип округления:

тип округления	пояснение
0	к нулю (toward_zero)
1	к ближайшему чётному (nearest_even)
2	к +∞ (toward_infinity)
3	к -∞ (toward_neg_infinity)

- операция – символ арифметической операции: +, -, *, /.
- число – одно или два числа, записанные в 16-ричной побитовой форме с префиксом ‘0x’. Пример: 0xC4D

В случае присутствия операции, её результат должен быть в том же формате, что и входные числа. Для чисел с плавающей точкой результат должен быть в соответствии с IEEE-754. Для чисел с фиксированной точкой в случае деления на 0 вывести "error" и завершиться с 0 кодом возврата.

Результат операции или единственное входное число необходимо вывести в стандартный поток вывода в следующем формате:

1. для чисел с фиксированной точкой – десятичная запись с 3 десятичными цифрами после точки. Пример: 0.120
2. для чисел с плавающей точкой – шестнадцатеричная показательная форма, степень в десятичном представлении, знак экспоненты выводится всегда, перед точкой всегда 1 (кроме нуля). Буквы в нижнем регистре. Для single 6 цифр после точки, для half – 3 цифры.

0 выводится с экспонентой +0. NaN выводится как nan, -∞ как -inf, +∞ как inf.

Примеры:

Входные аргументы	Результат
16.12 0 0x17360	23.210
8.8 1 0xdc9f + 0xd736	-76.168
f 0 0xB9CD542	0x1.39aa84p-104
f 0 0x414587dd * 0x42ebf110	0x1.6c1b72p+10
h 0 0x4145 * 0x42eb	0x1.238p+3
f 0 0x1 / 0x0	inf

Модификация ППА

Без модификации.

Содержание миниотчёта

1. Инструментарий (с указанием языка и версии компилятора/интерпретатора).
2. Результат работы написанной программы на тестовых данных из репозитория.
3. Описание работы написанного кода коротко: как вы храните числа (какие типы данных используются), что нужно было сделать с числами, чтобы арифметические операции работали, как работает округление и как оно реализовано в коде. Может быть полезно приложить формулы или небольшие рисунки для иллюстрации пояснений, чтобы не рисовать их на защите.

Порядок сдачи работы

1. Выполнить работу.

2. Оформить отчет в формате pdf.
3. Загрузить файл отчета и файлы с исходным кодом (расширения *.c/*.h/*.cpp/*.hpp или *.py или *.java) в выданный вам репозиторий в корень (не надо плодить папки, это неудобно для автотестов).
4. Запустить автотесты. Подробнее: Автотесты на GitHub.
5. Отправить на проверку работу (в открытом PR отмечаем Assignee Викторию и ставим label submit) и ждём ответа.

В репозиторий необходимо загружать файл с миниотчётом в формате pdf, названным в формате **“Фамилия_Имя_Группа_НомерРаботы.pdf”**. Номер группы – только 2 последние цифры группы, номер работы – порядковый номер лабораторной работы: 1, 2 и т.д.

Общие комментарии по работе

Для поднятия баллов по тестам можно исправлять те пункты, по которым стоит не 0 и всякие специальные случаи (вывод NaN, inf и подобное). В комментарии PR при отправке на повторную проверку необходимо написать, что было исправлено (кратко).

На исправления даётся +4 дня (если ответ получен после 12:00 дня) и +3 дня (если ответ получен до 12:00 дня) включая день получения ответа по проверке. Срок на исправление указывается включительно до 23:59.

Всем нужно просмотреть пункты ниже и проверить, есть у вас такие проблемы или нет. Если есть – исправить. Индивидуальные комментарии будут описаны в комментариях в PR.

Общие проблемы:

1. Нет ни намёка на его существование - нужно сделать.
2. Отчёт должен называться в формате (см. выше), а не "отчёт.pdf"...

Общие проблемы по тестам:

3. Если что-то пошло не так (неверное число аргументов, проблемы при парсинге) - необходимо писать сообщение об ошибке в поток вывода ОШИБОК (stderr, cerr) И завершаться с ненулевым кодом возврата, а не продолжать исполнение программы …
4. Если в аргументах программе передано значение округление, которое соответствует вашему варианту – продолжаем работу. Если нет – то пишем сообщение об ошибке в поток вывода ОШИБОК (stderr, cerr) И завершаться с ненулевым кодом возврата.
5. Если вы не поддерживаете какие-то операции, то необходимо сообщать ошибку (например, "Operation / is not supported") и завершаться с ненулевым кодом возврата, а не пытаться писать 0 или NaN. В любом случае баллов за это вы не получите, а только увеличиваете время проверки. Если же в таблице появятся баллы за

такое, то это всё равно будет видно на защите и баллы обнулятся уже на ней.

6. /0 в фиксированной точке постулировано по ТЗ, см. выше.
7. Ещё раз смотрим, как по ТЗ надо печатать NaN и бесконечности …
8. Ещё раз смотрим, как по ТЗ надо печатать экспоненту плавающих чисел …

Запуск автотестов

https://skkv-itmo.gitbook.io/comp-arch-cource/avtotesty-na-github

Число попыток = 15.

из Web-интерфейса

4 варианта округления. Нужно выбрать подходящий (по умолчанию выбран округление к 0).

Вывод. Если вы поддерживаете только фиксированную точку, то выбираете из выпадающего списка этот вариант. Этот выбор влияет на запуск тестов, когда на вход подаётся 1 число.

Тесты с операциями. + с фиксированной и +* с плавающей запускает тесты с этими операциями. Тест с половинной точностью позволяет проверить работу half. Если ничего не выбрано, то будут запущены тесты только с выводом чисел.

Специальные случаи - вывод бесконечностей, nan и подобное.

через CLI интерфейс:

Запуск скрипта

```
gh workflow run ci.yaml --ref main -f <field>=<value>
```

Пример

```
gh workflow run ci.yaml --ref main -f rounding="0" -f fixed_op=true
-f floating_op=false
```

Ключи

1. rounding ("Округление") default: "0"
   options:
   - "0"
   - "1"
   - "2"
   - "3"
2. print ("Тест на вывод") default: "Фиксированная и плавающая точка"
   options:
   - "Только фиксированная точка"
   - "Только плавающая точка"
   - "Фиксированная и плавающая точка"
3. fixed_op ("Тест с + (фиксированная точка)") type: boolean
4. floating_op ("Тест с +* (плавающая точка)") type: boolean
5. half_op ("Тест с * и выводом (плавающая точка с половинной точностью)") type: boolean
6. floating_special ("Тест на специальные случаи с плавающей точкой") type: boolean