Задания к работе 2 по алгоритмам и структурам данных.

Все задания реализуются на языке программирования C++ (стандарт C++14 и выше). Реализованные в заданиях приложения не должны завершаться аварийно.

Во всех заданиях запрещено использование глобальных переменных (включая errno).

Во всех заданиях запрещено использование оператора безусловного перехода (goto).

Во всех заданиях запрещено пользоваться функциями, позволяющими завершить выполнение приложения из произвольной точки выполнения, вне контекста исполнения функции main.

Во всех заданиях при реализации необходимо разделять контексты работы с данными (поиск, сортировка, добавление/удаление, модификация и т. п.) и отправка данных в поток вывода / выгрузка данных из потока ввода.

Во всех заданиях все параметры функций и вводимые (с консоли, файла, командной строки) пользователем данные должны подвергаться валидации в соответствии с типом валидируемых данных, если не сказано обратное; валидация должна зависеть от типа данных и логики применения этих данных для выполнения целевой подзадачи. При передаче аргументов приложению в командную строку, их количество также должно валидироваться.

Во всех заданиях необходимо контролировать ситуации с невозможностью [пере]выделения памяти; во всех заданиях необходимо корректно освобождать всю выделенную динамическую память.

Все ошибки, связанные с операциями открытия файла, должны быть обработаны; все открытые файлы должны быть закрыты.

Во всех заданиях запрещено использование глобальных переменных. Во всех заданиях при реализации функций необходимо обеспечить возможность обработки ошибок различных типов на уровне вызывающего кода.

Во всех заданиях сравнение (на предмет эквивалентности или отношения порядка) вещественных чисел на уровне функции должно использовать значение эпсилон, которое является параметром этой функции.

Во всех заданиях при реализации функций необходимо максимально ограничивать возможность модификации (если она не подразумевается) передаваемых в функцию параметров (используйте ключевое слово const), а также вызывающего объекта, в случае вызова его методов.

Для реализованных компонентов должны быть переопределены (либо перекрыты / оставлены реализации по умолчанию - при обосновании) следующие механизмы классов C++: конструктор копирования, деструктор, оператор присваивания.

Во всех заданиях необходимо уменьшать количество копирований нетривиально копируемых объектов.

Во всех заданиях необходимо проектировать компоненты с учетом SOLID принципов. Компонент не должен управлять ресурсом, если это не является его единственной задачей.

Запрещается пользоваться элементами стандартной библиотеки языка C, если существует их аналог в стандартной библиотеке языка C++.

Запрещается использование STL.

1. Реализуйте класс длинного целого числа. Данными объекта числа является: информация о знаке числа (хранится как бит значения типа *int*); динамический массив цифр в системе счисления с основанием $2^{8\times sizeof(int)}$. Обеспечьте эффективность по памяти для чисел в диапазоне

$$[-2^{8*sizeof(int)-1} \ldots 2^{8*sizeof(int)-1} - 1]$$

засчёт хранения значения в знаковом поле. Порядок хранения цифр числа - little endian. Формат хранения - знаковый (для отрицательных чисел хранение числа необходимо организовать при помощи дополнительного кода). Для класса реализуйте два конструктора: от динамического массива цифр (*int* *) в формате little endian и размера этого массива; от строкового представления числа (значение типа *char const* *) в формате big endian и основания системы счисления (значение типа *size_t*). Также для класса реализуйте операторы для: сложения длинных целых чисел (+=, +); вычитания длинных целых чисел (-=, -); умножения в столбик длинных целых чисел (*=, *); целочисленного деления в столбик длинных целых чисел (/=, /); взятия остатка от деления длинных целых чисел (%=, %); отношения эквивалентности на множестве длинных целых чисел (==, !=); отношения порядка на множестве длинных целых чисел (<, <=, >, >=); поразрядные (~, &, &=, |, |=, ^, ^=); битового сдвига (<<, <<=, >>, >>=); вставки в поток (friend <<, вывод значения числа в системе счисления с основанием 10); выгрузки из потока (friend >>, ввод значения числа в системе счисления с основанием 10).

   Продемонстрируйте работу реализованного функционала.

2. На основе реализованного в задании 1 функционала реализуйте класс дроби, хранящей в себе значения числителя (длинное целое число) и знаменателя (длинное целое число). Знак дроби должен располагаться в знаменателе дроби; в произвольный момент времени модули числителя и знаменателя любого объекта дроби должны быть взаимно простыми числами. Для класса реализуйте операторы для: сложения дробей (+=, +); вычитания дробей (-=, -); умножения дробей (*=, *); деления дробей (/=, /); отношения эквивалентности на множестве дробей (==, !=); отношения порядка на множестве дробей (<, <=, >, >=); вставки в поток (friend <<, вывод значения в формате: “<знак><числитель>/<модуль знаменателя>”); выгрузки из потока (friend >>, ввод значения в формате “<знак><числитель>/<модуль знаменателя>” или в формате строкового представления числа в системе счисления с основанием 10). Также реализуйте функционал для:
   - вычисления тригонометрических функций (sin, cos, tg, ctg, sec, cosec, arcsin, arccos, arctg, arcctg, arcsec, arccosec) над объектом дроби с заданной точностью ε (задаётся как параметр метода в виде объекта дроби);
   - возведения дроби в целую неотрицательную степень;
   - вычисления корня натуральной степени из дроби с заданной точностью ε (задаётся как параметр метода в виде объекта дроби);
   - вычисления двоичного, натурального и десятичного логарифмов из дроби с заданной точностью ε (задаётся как параметр метода в виде объекта дроби).

   Продемонстрируйте работу реализованного функционала.

3. На основе реализованного в задании 2 функционала реализуйте класс полинома от одной переменной. Полином должен быть реализован на базе структуры данных вида односвязный список (тип списка для хранения данных реализуйте самостоятельно) структур, хранящих степень переменной (целое неотрицательное число типа *unsigned int*) и коэффициент при степени переменной (объект дроби, тип которой реализован в задании 2); в произвольный момент времени список не должен содержать две одинаковых степени переменной, а также список должен быть отсортирован по значению степени переменной по убыванию. Для класса реализуйте операторы для: сложения полиномов (+=, +); вычитания полиномов (-=, -); умножения полиномов (*=, *); деления полиномов (/=, /); отношения эквивалентности на множестве полиномов (==, !=); вставки в поток (friend <<, вывод значения в формате: “({[<знак>][<коэффициент]x^[<степень>]} [...])”); выгрузки из потока (friend >>, ввод значения в формате представления, определяемом самостоятельно). Также реализуйте методы для: дифференцирования полинома по переменной; интегрирования полинома по переменной (для свободного коэффициента $c = 0$).

   Продемонстрируйте работу реализованного функционала, реализовав приложение, принимающее на вход путь к текстовому файлу, каждая строка которого описывает примеры в форматах:

   <unary operation> <polynomial>

   <polynomial1> <binary operation> <polynomial2>

   , где polynomial[1][2] - строковое представление полинома; <unary operation> - одна из операций: “diff”, “intgr”, соответствующая реализованным для типа полинома операторам/методам; <binary operation> - одна из операций: “+”, “-”, “*”, “/”, “%”, “==”, “!=”, соответствующая реализованным для типа полинома операторам. Приложение должно для каждого примера вывести результат вычисления выражения в консольный поток вывода.

4. На основе реализованного в задании 2 функционала реализуйте класс комплексного числа, в котором вещественная и мнимая части должны являться числами, репрезентируемыми объектами дробей. Для класса реализуйте операторы: сложения комплексных чисел (+=, +); вычитания комплексных чисел (-=, -); умножения комплексных чисел (*=, *); деления комплексных чисел (/=, /); отношения эквивалентности на множестве комплексных чисел (==, !=); вставки в поток (friend <<, вывод значения в формате “<вещественная часть> +/- <мнимая часть>i”); выгрузки из потока (friend >>, ввод значения в формате “<вещественная часть> +/- <мнимая часть>i”). Также на уровне типа комплексного числа реализуйте объектный функционал для вычисления аргумента комплексного числа, модуля комплексного числа, корня n -й степени из комплексного числа (все вычисления выполнять с заданной точностью ε (задаётся как параметр метода в виде объекта дроби).

   Продемонстрируйте работу реализованного функционала.

5. На основе реализованного в задании 2 класса дроби реализуйте приложение, решающее задачи линейной алгебры.

Программа на вход принимает файл с заданиями трёх типов:

- задачи из раздела векторной алгебры:
  - скалярное произведение двух векторов в n-мерном пространстве;
  - векторное произведение двух векторов в n-мерном пространстве;
  - смешанное произведение трёх векторов в n-мерном пространстве;
  - стандартные арифметические операции над векторами в n-мерном пространстве;
  - нахождение модуля вектора;
  - процесс ортогонализации переданного множества векторов;
- задачи из раздела матричной алгебры:
  - стандартные арифметические операции над матрицами (сложение матриц, умножение матриц, умножение матрицы на число);
  - вычисление определителя матрицы;
  - нахождение обратной матрицы;
  - решение СЛАУ методами: Гаусса, Жордана-Гаусса;
  - нахождение собственных чисел матрицы;
  - нахождение собственных векторов матрицы;
- задачи из раздела прямых и плоскостей в пространстве:
  - уравнением задается прямая на плоскости - необходимо вывести остальные уравнения этой этой прямой на плоскости;
  - уравнениями задаются две прямые на плоскости - необходимо найти точку пересечения прямых;
  - уравнением задаётся прямая в n-мерном пространстве, а также на вход подаётся точка - необходимо найти расстояние от точки до прямой;
  - уравнением задаётся прямая в n-мерном пространстве, а также на вход подаётся точка, не лежащая на прямой - необходимо найти симметричную относительно прямой точку для данной;
  - уравнением задаётся плоскость в трёхмерном пространстве - необходимо вывести остальные уравнения этой плоскости в трёхмерном пространстве;
  - уравнением задана прямая в n-мерном пространстве - необходимо вывести остальные уравнения этой прямой в трёхмерном пространстве;
  - уравнениями задаются две плоскости в трёхмерном пространстве - вывести все уравнения прямой в трёхмерном пространстве, являющейся их пересечением;
  - уравнением задана плоскость в трёхмерном пространстве, а также уравнением задана не параллельная заданной плоскости прямая - необходимо найти проекцию прямой на плоскость.

Входные данные поступают из файла. Файл считается корректным и валидация содержимого файла не требуется. Для обработки входного файла реализуйте класс. Под каждую задачу должен быть реализован отдельный метод класса, который её решает.

6. Реализуйте приложение, генерирующее файлы со входными данными для интерпретатора задач линейной алгебры, реализованного в задании 5, по предоставленным ответам к этим задачам (пример: если требуется составить 10 задач на тему “вычисление скалярного произведения двух векторов в n-мерном пространстве” и в качестве ответа указано значение $\frac{125}{3}$, генератор должен составить 10 различных задач с такими векторами, чтобы их скалярное произведение равнялось $\frac{125}{3}$). Формат представления ответа и типа задачи, для которой предоставлен ответ, предусмотрите самостоятельно. Генерируемые задачи аналогичны задачам, представленным в задании 5. Ответы на задачи считаются согласованными.

   Продемонстрируйте работу вашей программы, сгенерировав для каждого из 20 типов задач, для каждой задачи для 10 различных ответов, по 25 задач, и решите их с помощью вашей реализации интерпретатора из задания 5, сравните указанные при генерации и полученные при решении ответы.

7. На основе реализованного в задании 2 класса дроби реализуйте сервис, предоставляющий функционал для:
   - вычисления коэффициентов цепной дроби по значению обыкновенной дроби и наоборот;
   - построения коллекции подходящих дробей для цепной дроби, для обыкновенной дроби;
   - построения представления значения обыкновенной дроби в виде пути (значения типа $std::vector < bool >$) в дереве Штерна-Броко и наоборот;
   - построения представления значения обыкновенной дроби в виде пути (значения типа $std::vector < bool >$) в дереве Калкина-Уилфа и наоборот.

   Продемонстрируйте работу реализованного функционала.

8. При помощи класса дроби из задания 2 реализуйте методы для вычисления числа π с точностью 500000 знаков после запятой в системе счисления с основанием 10. Вычисление организовать четырьмя различными способами:
   - по формуле Бэйли-Боруэйна-Плаффа:

$$\pi = \sum_{n=0}^{\infty} \frac{1}{16^n}\left(\frac{4}{8n+1} - \frac{2}{8n+4} - \frac{1}{8n+5} - \frac{1}{8n+6}\right);$$

   - по формуле Беллара:

$$\pi = \frac{1}{2^6}\sum_{n=0}^{\infty} \frac{(-1)^n}{2^{10n}}\left(-\frac{2^5}{4n+1} - \frac{1}{4n+3} + \frac{2^8}{10n+1} - \frac{2^6}{10n+3} - \frac{2^2}{10n+5} - \frac{2^2}{10n+7} + \frac{1}{10n+9}\right);$$

   - по формуле Чудновских:

$$\frac{1}{\pi} = \frac{1}{426880\sqrt{10005}}\sum_{n=0}^{\infty} \frac{(6n)!(13591409+545140134n)}{(3n)!(n!)^3(-640320)^{3n}};$$

   - по формуле Рамануджана:

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801}\sum_{n=0}^{\infty} \frac{(4n)!(1103+26390n)}{(n!)^4 396^{4n}}.$$

   Результатами работы методов должны являться вычисленные значения и количество затраченных на нахождение итераций.