

# Задания к работе №6 по Фундаментальным Алгоритмам.

Все задания реализуются на языке программирования C++ (стандарт C++20 и выше).

Реализованные в заданиях приложения не должны завершаться аварийно; все возникающие исключительные ситуации должны быть перехвачены и обработаны.

Во всех заданиях запрещено использование: глобальных переменных (включая `errno`), 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 системных ресурсов уровня ОС (файлы, средства синхронизации, etc.), должны быть обработаны; все открытые системные ресурсы должны быть возвращены ОС.

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 объекта, в случае метода.

Для реализованных компонентов должны быть переопределены (либо перекрыты - при обосновании) следующие механизмы классов C++: конструктор копирования, деструктор, оператор присваивания, конструктор перемещения, присваивание перемещением.

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 Си, если существует их аналог в стандартной библиотеке языка C++.

Для задач, каталоги которых в репозитории содержат папку tests, требуется демонстрация прохождения всех описанных тестов для реализованных компонентов. Модификация кода тестов запрещена.

1. Реализуйте класс длинного целого числа (repo path: */arithmetic/big\_integer*). Распределение памяти под вложенные в объект длинного целого числа данные организуйте через объект аллокатора, подаваемый объекту длинного целого числа через конструктор. Данными объекта числа является: информация о знаке числа (хранится как значение типа *bool*); динамический массив цифр в системе счисления с основанием  $2^{8 \times \text{sizeof}(\text{unsigned int})}$ ; аллокатор, используемый для выделения памяти под динамический массив цифр. Порядок хранения цифр числа - little endian.

Также для класса реализуйте операторы для: сложения длинных целых чисел ( $+=$ ,  $+$ ); вычитания длинных целых чисел ( $-=$ ,  $-$ ); отношения эквивалентности на множестве длинных целых чисел ( $==$ ,  $!=$ ); отношения порядка на множестве длинных целых чисел ( $<=>$ ); поразрядные ( $\sim$ ,  $\&$ ,  $|$ ,  $\wedge$ ); 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. Реализуйте функционал умножения длинных целых чисел (repo path: */arithmetic/big\_integer*) согласно алгоритму умножения чисел в столбик. Также для класса длинного целого числа реализуйте операторы умножения (\*, \*=), делегирующие выполнение операции умножения реализованному алгоритму умножения.

Продемонстрируйте работу реализованного функционала.

3. Реализуйте функционал умножения длинных целых чисел (repo path: */arithmetic/big\_integer*) согласно алгоритму Карацубы умножения чисел.

Продемонстрируйте работу реализованного функционала.

4. Реализуйте функционал умножения длинных целых чисел (repo path: */arithmetic/big\_integer*) согласно алгоритму Шёнхаге-Штрассена умножения чисел.

Продемонстрируйте работу реализованного функционала.

5. Реализуйте функционал целочисленного деления длинных целых чисел (repo path: */arithmetic/big\_integer*) согласно алгоритму деления чисел в столбик. Также для класса длинного целого числа реализуйте операторы взятия целой части от деления ( $/$ ,  $/=$ ) и операторы взятия остатка от деления ( $\%$ ,  $\%=$ ), делегирующие выполнение операции деления реализованному алгоритму деления.

Продемонстрируйте работу реализованного функционала.

6. Реализуйте класс целочисленного деления длинных целых чисел (repo path: */arithmetic/big\_integer*) на основе контракта *bigint::division* (repo path: */arithmetic/big\_integer*) согласно алгоритму Ньютона деления чисел.

Продемонстрируйте работу реализованного функционала.



7. Реализуйте функционал целочисленного деления длинных целых чисел (repo path: */arithmetic/big\_integer*) согласно алгоритму Бурникеля-Циглера деления чисел.

Продемонстрируйте работу реализованного функционала.

8. На основе реализованного в заданиях 2-7 функционала реализуйте класс дроби (геро path: */arithmetic/fraction*), 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: 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 ( $\sin$ ,  $\cos$ ,  $\text{tg}$ ,  $\text{ctg}$ ,  $\sec$ ,  $\text{cosec}$ ,  $\arcsin$ ,  $\arccos$ ,  $\text{arctg}$ ,  $\text{arcctg}$ ,  $\text{arcsec}$ ,  $\text{arccosec}$ ) над объектом дроби с заданной точностью  $\varepsilon$  (задаётся как параметр метода в виде объекта дроби);
  - возведения дроби в целую неотрицательную степень;
  - вычисления корня натуральной степени из дроби с заданной точностью  $\varepsilon$  (задаётся как параметр метода в виде объекта дроби);
  - вычисления двоичного, натурального и десятичного логарифмов из дроби с заданной точностью  $\varepsilon$  (задаётся как параметр метода в виде объекта дроби).

Продемонстрируйте работу реализованного функционала.

9. На основе реализованного в задании 8 класса дроби реализуйте компонент (repo path: */arithmetic/continued\_fraction*), 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.