

Лабораторная работа № 02

Тема: Операторы, литералы

Цель:

- Изучение механизмов перегрузки операторов;
- Изучение механизмов работы с пользовательскими литералами;

Порядок выполнения работы

1. Ознакомиться с теоретическим материалом.
2. Получить у преподавателя вариант задания.
3. Реализовать задание своего варианта в соответствии с поставленными требованиями.
4. Подготовить тестовые наборы данных.
5. Создать репозиторий на GitHub.
6. Отправить файлы лабораторной работы в репозиторий.
7. Отчитаться по выполненной работе путём демонстрации работающей программы на тестовых наборах данных (как подготовленных самостоятельно, так и предложенных преподавателем) и ответов на вопросы преподавателя (как из числа контрольных, так и по реализации программы).

Требования к программе

Разработать программу на языке C++ согласно варианту задания. Программа на C++ должна собираться с помощью системы сборки CMake. Программа должна получать данные из стандартного ввода и выводить данные в стандартный вывод.

Необходимо настроить сборку лабораторной работы с помощью CMake. Собранная программа должна называться **oop_exercise_02** (в случае использования Windows **oop_exercise_02.exe**)

Необходимо зарегистрироваться на GitHub (если студент уже имеет регистрацию на GitHub то можно использовать ее) и создать репозиторий для задания лабораторной работы.

Преподавателю необходимо предъявить ссылку на публичный репозиторий на Github. Имя репозитория должно быть https://github.com/login/oop_exercise_02

Где login – логин, выбранный студентом для своего репозитория на Github.

Репозиторий должен содержать файлы:

- main.cpp //файл с заданием работы

- CMakeLists.txt // файл с конфигураций CMake
- test_xx.txt // файл с тестовыми данными. Где xx – номер тестового набора 01, 02, ... Тестовых наборов должно быть больше 1.
- report.doc // отчет о лабораторной работе

Варианты заданий

№ ЗАДАНИЕ

1	<p>Комплексное число в алгебраической форме представляются парой действительных чисел (a, b), где a – действительная часть, b – мнимая часть. Реализовать класс Complex для работы с комплексными числами. Обязательно должны быть присутствовать операции</p> <ul style="list-style-type: none"> - сложения add, $(a, b) + (c, d) = (a + c, b + d)$; - вычитания sub, $(a, b) - (c, d) = (a - c, b - d)$; - умножения mul, $(a, b) \cdot (c, d) = (ac - bd, ad + bc)$; - деления div, $(a, b) / (c, d) = (ac + bd, bc - ad) / (c^2 + d^2)$; - сравнение equ, $(a, b) = (c, d)$, если $(a = c)$ и $(b = d)$; - сопряженное число conj, $\text{conj}(a, b) = (a, -b)$. Реализовать операции сравнения модулей. <p>Операции сложения, вычитания, умножения, деления, сравнения на равенство должны быть выполнены в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа Complex.</p>
2	<p>Комплексное число в тригонометрической форме представляются парой действительных чисел (r, j), где r – радиус (модуль), j – угол. Реализовать класс Complex для работы с комплексными числами. Обязательно должны быть присутствовать операции</p> <ul style="list-style-type: none"> - сложения add, $(r_1, j_1) + (r_2, j_2)$; - вычитания sub, $(r_1, j_1) - (r_2, j_2)$; - умножения mul, $(r_1, j_1) \cdot (r_2, j_2)$; - деления div, $(r_1, j_1) / (r_2, j_2)$; - сравнение equ, $(r_1, j_1) = (r_2, j_2)$, если $(r_1 = r_2)$ и $(j_1 = j_2)$; - сопряженное число conj, $\text{conj}(r, j) = (r, -j)$. <p>Реализовать операции сравнения по действительной части. Операции сложения, вычитания, умножения, деления, сравнения на равенство должны быть выполнены в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа Complex.</p>

3	<p>Рациональная (несократимая) дробь представляется парой целых чисел (a, b), где a — числитель, b — знаменатель. Создать класс Rational для работы с рациональными дробями. Обязательно должны быть реализованы операции:</p> <ul style="list-style-type: none"> - сложения add, $(a, b) + (c, d) = (ad + bc, bd)$; - вычитания sub, $(a, b) - (c, d) = (ad - bc, bd)$; - умножения mul, $(a, b) \cdot (c, d) = (ac, bd)$; - деления div, $(a, b) / (c, d) = (ad, bc)$; - операции сравнения. <p>Должна быть реализована функция сокращения дроби reduce(), которая обязательно вызывается при выполнении арифметических операций. Операции сложения, вычитания, умножения, деления, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа Rational.</p>
4	<p>Реализовать класс FuzzyNumber для работы с нечеткими числами, которые представляются тройками чисел $(x - e_l, x, x + e_r)$. Для чисел $A = (A - a_l, A, A + a_r)$ и $B = (B - b_l, B, B + b_r)$ арифметические операции выполняются по следующим формулам:</p> <ul style="list-style-type: none"> - сложение $A + B = (A + B - a_l - b_l, A + B, A + B + a_r + b_r)$; - вычитание $A - B = (A - B - a_l - b_l, A - B, A - B + a_r + b_r)$; - умножение $A \cdot B = (A \cdot B - B \cdot a_l - A \cdot b_l + a_l \cdot b_l, A \cdot B, A \cdot B + B \cdot a_l + A \cdot b_l + a_l \cdot b_l)$; - обратное число $A = (1 / (A + a_r), 1 / A, 1 / (A - a_l))$, $A > 0$; - деление $A / B = ((A - a_l) / (B + b_r), A / B, (A + a_r) / (B - b_l))$, $B > 0$; <p>Считать $e_l = e_r$, то есть число представлено парой $\langle x, e \rangle$.</p> <p>Реализовать операции сравнения по x. Операции сложения, вычитания, умножения, деления, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа FuzzyNumber.</p>
5	<p>Создать класс Modulo для работы с целыми числами по модулю N. В классе должно быть два поля: число и N. Реализовать все арифметические операции. Реализовать операции сравнения. Операции сложения, вычитания, умножения, деления, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа Modulo.</p>

- 6 **Создать класс `BitString`** для работы с 96-битовыми строками. Битовая строка должна быть представлена двумя полями: старшая часть `unsigned long long`, младшая часть `unsigned int`. Должны быть реализованы все традиционные операции для работы с битами: `and`, `or`, `xor`, `not`. Реализовать сдвиг влево `shiftLeft` и сдвиг вправо `shiftRight` на заданное количество битов. Реализовать операцию вычисления количества единичных битов, операции сравнения по количеству единичных битов. Реализовать операцию проверки включения. Операции `and`, `or`, `xor`, `not`, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа **`BitString`**.
- 7 **Создать класс `BitString`** для работы с 128-битовыми строками. Битовая строка должна быть представлена двумя полями типа `unsigned long long`. Должны быть реализованы все традиционные операции для работы с битами: `and`, `or`, `xor`, `not`. Реализовать сдвиг влево `shiftLeft` и сдвиг вправо `shiftRight` на заданное количество битов. Реализовать операцию вычисления количества единичных битов, операции сравнения по количеству единичных битов. Реализовать операцию проверки включения. Операции `and`, `or`, `xor`, `not`, `>>`, `<<`, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов.
- 8 Необходимо реализовать пользовательский литерал для работы с константами типа **`BitString`**.
- Создать класс `Money`** для работы с денежными суммами. Сумма денег должна быть представлено двумя полями: типа `unsigned long long` для рублей и типа `unsigned char` – для копеек. Дробная часть (копейки) при выводе на экран должна быть отделена от целой части запятой. Реализовать сложение сумм, вычитание, деление сумм, деление суммы на дробное число, умножение на дробное число и операции сравнения. Операции сложения, вычитания, умножения, деления, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа **`Money`**.
- 9 **Создать класс `BritishMoney`** для работы с денежными суммами в старой британской системе. Сумма денег должна быть представлено тремя полями: типа `unsigned long long` для фунтов стерлингов, типа `unsigned char` – для шиллингов, `unsigned char` – для пенсов (пенни). Реализовать сложение сумм, вычитание, деление сумм, деление суммы на дробное число, умножение на дробное число и операции сравнения. 1 фунт = 20 шиллингов, 1 шиллинг = 12 пенни. Операции сложения, вычитания, умножения, деления, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа **`BritishMoney`**.

- 10 **Создать класс `Angle`** для работы с углами на плоскости, задаваемыми величиной в градусах и минутах. Обязательно должны быть реализованы: перевод в радианы, приведение к диапазону 0–360, сложение и вычитание углов, деление углов, получение значений тригонометрических функций, сравнение углов. Операции сложения, вычитания, деления, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа `Angle`.
- 11 **Создать класс `vector3D`**, задаваемый тройкой координат. Обязательно должны быть реализованы: операции сложения и вычитания векторов, векторное произведение векторов, скалярное произведение векторов, умножения на скаляр, сравнение векторов на совпадение, вычисление длины вектора, сравнение длины векторов, вычисление угла между векторами. Операции сложения, вычитания, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа `vector3D`.
- 12 **Разработать класс `Rectangle`**, представляющий собой прямоугольник со сторонами, параллельными осям координат. Поля – координаты левого нижнего и правого верхнего угла. Требуется реализовать следующие методы: вычисление площади и периметра, перемещения вдоль осей, изменение размеров, сравнение по площади и по периметру. Реализовать метод получения прямоугольника, представляющего общую часть (пересечение) двух прямоугольников. Реализовать метод объединения двух прямоугольников: наименьший прямоугольник, включающего оба заданных прямоугольника.
- Необходимо реализовать:
- операцию приведения к типу `double`, вычисляющую площадь прямоугольника;
 - операцию префиксного и постфиксного инкремента, увеличивающую одновременно размеры сторон прямоугольника.
 - операции сравнения (больше, меньше, равно);
- Необходимо реализовать пользовательский литерал для работы с константами типа `Rectangle`.
- 13 **Создать класс `Long`** для работы с целыми беззнаковыми числами из 64 бит. Число должно быть представлено двумя полями `unsigned int`. Должны быть реализованы арифметические операции, присутствующие в C++, и сравнения. Операции необходимо реализовать в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа `Long`.

14 **Создать класс `TimePoint`** для работы с моментами времени в формате «час:минута:секунда». Обязательными операциями являются: вычисление разницы между двумя моментами времени, сумма моментов времени, сложение момента времени и заданного количества секунд, вычитание из момента времени заданного количества секунд, вычисление во раз сколько один момент времени больше (меньше) другого, сравнение моментов времени, перевод в секунды и обратно, перевод в минуты (с округлением до минуты) и обратно. Операции сложения и вычитания `TimePoint`, а так же сравнения (больше, меньше и равно) необходимо реализовать в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа **`TimePoint`**.

15 **Создать класс `TransNumber`** для работы с трансцендентными числами. Трансцендентное число представлено парой (a, b) , где a – действительная часть, b – трансцендентная часть. Трансцендентная часть представляет собой действительное число b , умноженное на константу 2^2 . Реализовать арифметические операции (по аналогии с операциями над комплексными числами в алгебраической форме), и операции сравнения по значению $(a + b \cdot 2^2)$.

16 Арифметические операции, а также сравнения (больше, меньше и равно) необходимо реализовать в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа **`TransNumber`**.

Создать класс `Position` для работы с географическими координатами. Координаты задаются двумя числами широта и долгота. Долгота находится в диапазоне от -180 до 180 градусов. Широта находится в диапазоне от -90 до 90 градусов. Реализовать арифметические операции сложения, вычитания, умножения и деления, а также операции сравнения в виде перегрузки операторов.

Необходимо реализовать пользовательский литерал для работы с константами типа **`Position`**.

17 **Создать класс `Budget`** для работы с бюджетом. Класс состоит из двух вещественных чисел (a, b) . Где a – собственная часть средств бюджета в рублях, b – заемная часть средств бюджета в рублях. Оба числа должны округляться до второго знака после запятой. Реализовать арифметические операции сложения, вычитания, умножения и деления, а также операции сравнения в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа **`Budget`**.

- 18 **Создать класс `IPAddress`** для работы с адресом в интернет. Класс состоит из четырех чисел `unsigned char` (`a,b,c,d`). Реализовать арифметические операции сложения, вычитания, а также операции сравнения (для сравнение на больше/меньше считать что левые байты главнее т.е. вначале сравниваются первые байты, потом вторые и т.д.). Так же реализовать функцию, которая будет определять принадлежность адреса к подсети по адресу подсети (`a1,b1,c1,d1`) и битовой маске подсети (`a2,b2,c2,d2`). Например, адрес 192.168.1.30 принадлежит подсети 192.168.0.0 с маской 255.255.0.0.

Операции сложения, вычитания и сравнения необходимо реализовать в виде перегрузки операторов.

Необходимо реализовать пользовательский литерал для работы с константами типа **`IPAddress`**.

- 19 **Создать класс `Address`** для работы с адресами домов. Адрес должен состоять из строк с названием города и улицы и чисел с номером дома и квартиры. Реализовать операции сравнения адресов, а также операции проверки принадлежности адреса к улице и городу. В операциях не должен учитываться регистр строки. Так же необходимо сделать операцию, которая возвращает истину если два адреса находятся по соседству (на одной улице в одном городе и дома стоят подряд).

Операцию сравнения равенства реализовать в виде перегрузки оператора. Операцию нахождения «по соседству» реализовать в виде перегрузки оператора `&`.

Необходимо реализовать пользовательский литерал для работы с константами типа **`Address`**.

- 20 **Создать класс `Bottle`** для работы с емкостями. Класс должен состоять из двух вещественных чисел: `a`- объем емкости в литрах и `b` – процент наполнения емкости (0 – пустая, 1 – полная). Реализовать операции сложения и вычитания, а также сравнения объектов класса бутылка в виде перегрузки операторов. При сложении должен складываться фактический объем заполнения бутылок.

Необходимо реализовать пользовательский литерал для работы с константами типа **`Bottle`**.

Отчет

1. Код программы на языке C++.
2. Ссылка на репозиторий на GitHub.
3. Набор testcases.
4. Результаты выполнения тестов.
5. Объяснение результатов работы программы.