МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ В.Ф. УТКИНА»

КАФЕДРА

«ВЫЧИСЛИТЕЛЬНАЯ И ПРИКЛАДНАЯ МАТЕМАТИКА»

МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ Б1.В «ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ»

Направление подготовки – 09.03.04 «Программная инженерия»

Направленность (профиль) подготовки
«Программная инженерия»

ОПОП академического бакалавриата

Квалификация выпускника – бакалавр

Формы обучения – очная

Рязань 2022 г

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Рекомендуется следующим образом организовать время, необходимое для изучения дисциплины:

Для освоения лекционного материала следует: изучить конспект лекции в тот же день, после лекции: 10 – 15 минут, повторно прочитать конспект лекции за день перед следующей лекцией: 10 – 15 минут. Также следует изучить теоретический лекционный материал по рекомендуемому учебнику/ учебному пособию: 1 час в неделю.

Следует максимально использовать лекционное время для изучения дисциплины, понимания лекционного материала и написания конспекта лекций. В процессе лекционного занятия студент должен уметь выделять важные моменты и основные положения. При написании конспекта лекций следует придерживаться следующих правил и рекомендаций.

1. При ведении конспекта рекомендуется структурировать материал по разделам, главам, темам. Вести нумерацию формул. Выделять по каждой теме постановку задачи, основные положения, выводы. Кратко записывать те пояснения лектора, которые показались особенно важными. Это позволит при подготовке к сдаче зачёта и экзамена не запутаться в структуре лекционного материала.

2. Лекционный материал следует записывать в конспект лишь после того, как излагаемый лектором тезис будет вами дослушан до конца и понят.

3. При конспектировании следует отмечать непонятные, на данном этапе, положения, доказательства и пр.

4. Рекомендуется по каждой теме выразить свое мнение, комментарий, вывод.

Подготовка к практическим занятиям:

Практические занятия по дисциплине существенно дополняют лекции. В процессе анализа теоретических положений и решения практических задач студенты расширяют и углубляют свои знания, полученные из лекционного курса и учебников, приобретают умение применять общие закономерности к конкретным случаям. В процессе решения задач развивается логическое мышление и вырабатываются навыки вычислений, работы со справочной литературой. Практические занятия способствуют закреплению знаний и практических навыков, формированию конструктивного стиля мышления, расширению кругозора.

При подготовке к практическому занятию необходимо внимательно ознакомиться с соответствующим теоретическим материалом по конспекту

лекций и рекомендуемому учебнику, затем изучить конспект или материалы предыдущего практического занятия и выполнить заданное расчетное задание: 1 – 2 часа в неделю.

Следует максимально использовать аудиторное время практических занятий. В процессе занятия студент должен активно участвовать в дискуссиях, обсуждениях и решениях практических задач и вести конспект практических занятий отдельно от конспекта лекций.

Дополнительно в часы самостоятельной работы студенты могут повторно решить задачи, с которыми они плохо освоились во время аудиторных занятий, и обязательно те задачи, которые не получились дома при предыдущей подготовке к практическим занятиям.

Подготовка к лабораторным работам.

Перед началом проведения лабораторной работы необходимо ознакомится с методическими указаниями к данной лабораторной работе, внимательно ознакомиться с заданием и желательно заранее выполнить подготовку проекта в используемой инструментальной среде, чтобы время лабораторного занятия использовать для исправления ошибок, модификации проекта и защиты данной работы.

Выполнение каждой из запланированных работ заканчивается предоставлением отчета. Требования к форме и содержанию отчета приведены в методических указаниях к лабораторным работам или определяются преподавателем на первом занятии. Отчет по лабораторной работе студент должен начать оформлять еще на этапе подготовки к ее выполнению. Допускаясь к лабораторной работе, каждый студент должен представить преподавателю «заготовку» отчета, содержащую: оформленный титульный лист или название и номер работы при ведении общего конспекта, цель работы, задание, проект решения, полученные результаты, выводы.

Изучение методических указаний к лабораторной работе – 2 часа перед выполнением лабораторной работы и в ходе разработки проекта и 2 часа для оформления отчета, отладки проекта и подготовки к сдаче работы.

После выполнения лабораторной работы необходимо согласовать полученные результаты с преподавателем. Важным этапом является защита лабораторной работы. В процессе защиты студент отвечает на вопросы преподавателя, касающиеся теоретического материала, относящегося к данной работе, и проекта, реализующего его задание, комментирует полученные в ходе работы результаты. При подготовке к защите лабораторной работы рекомендуется ознакомиться со списком вопросов по изучаемой теме и попытаться самостоятельно на них ответить, используя конспект лекций и рекомендуемую

литературу. Кроме чтения учебной литературы рекомендуется активно использовать информационные ресурсы сети Интернет по изучаемой теме.

Подготовка к сдаче экзамена или зачета:

Экзамен/зачет – форма промежуточной проверки знаний, умений, навыков, степени освоения дисциплины. Главная задача экзамена/зачета состоит в том, чтобы у студента по окончанию изучения данной дисциплины сформировались определенное представление об общем содержании дисциплины, определенные теоретические знания и практические навыки, определенный кругозор. Готовясь к экзамену/зачету, студент приводит в систему знания, полученные на лекциях, на практических и лабораторных занятиях, разбирается в том, что осталось непонятным, и тогда изучаемая им дисциплина может быть воспринята в полном объеме с присущей ей строгостью и логичностью, ее практической направленностью.

Экзамены/зачеты дают возможность преподавателю определить теоретические знания студента и его практические навыки при решении определенных прикладных задач. Оцениваются: понимание и степень усвоения теоретического материала; степень знакомства с основной и дополнительно литературой, а также с современными публикациями; умение применить теорию к практике, решать определенные практические задачи данной предметной области, правильно проводить расчеты и т. д.; знакомство с историей данной науки; логика, структура и стиль ответа, умение защищать выдвигаемые положения.

Значение экзаменов/зачетов не ограничивается проверкой знаний, являясь естественным завершением обучения студента по данной дисциплине, они способствуют обобщению и закреплению знаний и умений, приведению их в стройную систему, а также устранению возникших в процессе обучения пробелов.

Подготовка к экзамену/зачету – это тщательное изучение и систематизация учебного материала, осмысление и запоминание теоретических положений, формулировок, формул, установление и осмысление внутрипредметных связей между различными темами и разделами дисциплины, закрепление теоретических знаний путем решения определенных задач.

Перед экзаменом назначается консультация, ее цель – дать ответы на вопросы, возникшие в ходе самостоятельной подготовки студента, студент имеет возможность получить ответ на все неясные ему вопросы, кроме того, преподаватель будет отвечать на вопросы других студентов, что будет способствовать повторению и закреплению знаний всех присутствующих. Преподаватель на консультации, как правило, обращает внимание на те разделы, по которым на предыдущих экзаменах ответы были

неудовлетворительными, а также фиксирует внимание на наиболее трудных разделах курса.

На непосредственную подготовку к экзамену обычно дается 3 – 5 дней. Этого времени достаточно для углубления, расширения и систематизации знаний, полученных в ходе обучения, на устранение пробелов в знании отдельных вопросов, для определения объема ответов на каждый из вопросов рабочей программы дисциплины.

Планируйте подготовку к зачету/экзамену, учитывая сразу несколько факторов: неоднородность в сложности учебного материала и степени его проработки в ходе обучения, свои индивидуальные способности. Рекомендуется делать перерывы в занятиях через каждые 50-60 минут на 10 минут. После 3-4 часов занятий следует сделать часовой перерыв. Чрезмерное утомление приведет к снижению тонуса интеллектуальной деятельности. Целесообразно разделять весь рабочий день на три рабочих периода – с утра до обеда, с обеда до ужина и с ужина до сна. Каждый рабочий период дня должен заканчиваться отдыхом не менее 1 часа. Работая в сессионном режиме, студент имеет возможность увеличить время занятий с 10 (как требовалось в семестре) до 12 часов в сутки.

Подготовку к экзаменам или зачетам следует начинать с общего планирования своей деятельности. С определения объема материала, подлежащего проработке, необходимо внимательно сверить свои конспекты с программой дисциплины, чтобы убедиться, все ли разделы отражены в лекциях, отсутствующие темы изучить по учебнику. Второй этап предусматривает системное изучение материала по данному предмету с обязательной записью всех выкладок, выводов, формул. На третьем этапе – этапе закрепления – полезно чередовать углубленное повторение особенно сложных вопросов с беглым повторением всего материала.

Рекомендации по работе с литературой:

Теоретический материал курса становится более понятным, когда дополнительно к прослушиванию лекции и изучению конспекта изучаются и книги по данному предмету. Литературу по дисциплине рекомендуется читать как в бумажном, так и в электронном виде (если отсутствует бумажный аналог). Полезно использовать несколько учебников и пособий по дисциплине. Рекомендуется после изучения очередного параграфа ответить на несколько вопросов по данной теме. Кроме того, полезно мысленно задать себе следующие вопросы (и попробовать ответить на них): «о чем этот параграф?», «какие новые понятия введены, каков их смысл?», «зачем мне это нужно по специальности?».

Рекомендуется самостоятельно изучать материал, который еще не прочитан на лекции и не применялся на лабораторном или практическом занятии, тогда занятия будут гораздо понятнее. В течение недели рекомендуется выбрать время (1 час) для работы с литературой.

Лабораторная работа №1
Структуры и модули

В данной лабораторной работе необходимо создать модуль, содержащий описание структуры данных, представляющее собой некоторую сущность: вектор, дробь, фигура и т.д. (по вариантам ниже), а также 8 операций (функций) над этой структурой. Основная программа должна запрашивать у пользователя какую операцию необходимо протестировать, далее запрашивать данные для операндов и показывать результат операции с помощью функции модуля «вывод в консоль». В качестве упрощения, задание не подразумевает использование динамической памяти, следовательно, во всех вариантах используются статические массивы (если это обусловлено задачей).

Важно: структуры создаются в модуле с помощью отдельной функции, возвращающей структуру по значению. Это необходимо, например, для того, чтобы рассчитать некоторые поля структуры на основе переданных данных.

Вариант 1. «Комплексное число» – Complex.

Разработать структуру данных Complex, представляющую комплексные числа в виде значений вещественной и мнимой части (a,b), а также логического значения real, показывающего что комплексное число имеет только вещественную часть (мнимая часть равна нулю). В модуле для работы с комплексными числами должны быть следующие функции:

1) Создание структуры комплексного числа из значений действительной и мнимой части. Как результат возвращается новое комплексное число.

2) Сложение двух комплексных чисел, как результат возвращается новое комплексное число.

3) Вычитание двух комплексных чисел, как результат возвращается новое комплексное число.

4) Умножение двух комплексных чисел, как результат возвращается новое комплексное число.

5) Деление двух комплексных чисел, как результат возвращается новое комплексное число.

6) Преобразование переданного комплексного числа в сопряжённое.

7) Преобразование переданного комплексного числа в обратное.

8) Вывод в консоль переданного комплексного числа в виде:

(*a + b * i) [real],*

где real имеет значение либо «*комплексное*», либо «*вещественное*».

Вариант 2. «**Дробь**» – **Fraction.**

Разработать структуру данных Fraction, представляющую простую дробь в виде пары целых положительных чисел (*m,n*) а также отдельно логического значения, указывающего на знак дроби. В модуле для работы с дробями должны быть следующие функции:

1) Создание структуры дроби из значений числителя, знаменателя, а также знака. Как результат возвращается новая дробь.

2) Сложение двух дробей, как результат возвращается новая дробь.

3) Вычитание двух дробей, как результат возвращается новая дробь.

4) Умножение двух дробей, как результат возвращается новая дробь.

5) Деление двух дробей, как результат возвращается новая дробь.

6) Приведение двух переданных дробей к общему знаменателю.

7) Сравнение двух дробей, как результат возвращается целое значение: -1 «меньше», 1 «больше», 0 «равны».

8) Вывод в консоль переданной дроби в виде:

[sign] a / b,

где sign имеет значение либо «+», либо «-».

Вариант 3. «**Двумерный вектор**» – **Vector2d.**

Разработать структуру данных **Vector2d**, представляющую двумерный вектор в виде двух компонент (x,y), а также логического значения norm, показывающего нормализован ли данный вектор. В модуле для работы с двумерными векторами должны быть следующие функции:

1) Создание структуры двумерного вектора из значений его компонент. Как результат возвращается новый вектор.

2) Сложение двух векторов, как результат возвращается новый вектор.

3) Вычитание двух векторов, как результат возвращается новый вектор.

4) Умножение переданного вектора на заданный скаляр.

5) Нормализация переданного вектора.

6) Получение значения скалярного произведения двух векторов.

7) Установка заданной длины для переданного вектора.

8) Вывод в консоль переданного вектора в виде:

(x,y) [norm],

где norm имеет значение либо «*нормализован*», либо «*не нормализован*».

Вариант 4. «**Трёхмерный вектор**» – **Vector3d.**

Разработать структуру данных **Vector3d**, представляющую трёхмерный вектор в виде трех компонент (x,y,z), а также логического значения norm, показывающего нормализован ли данный вектор. В модуле для работы с трёхмерными векторами должны быть следующие функции:

1) Создание структуры трёхмерного вектора из значений его компонент. Как результат возвращается новый вектор.

2) Сложение двух векторов, как результат возвращается новый вектор.

3) Вычитание двух векторов, как результат возвращается новый вектор.

4) Получение значения угла между двумя переданными векторами.

5) Нормализация переданного вектора.

6) Векторное произведение двух векторов, как результат возвращается новый вектор.

7) Проекция вектора на вектор, как результат возвращается новый вектор.

8) Вывод в консоль переданного вектора в виде:

(x,y,z) [norm],

где norm имеет значение либо «*нормализован*», либо «*не нормализован*».

Вариант 5. «Квадратная матрица 3x3» – Matrix3x3.

Разработать структуру данных **Matrix3x3**, представляющую квадратную матрицу 3 на 3 в виде массива из 9 её элементов, а также логического значения identity, показывающего является ли данная матрица единичной. В модуле для работы с матрицами должны быть следующие функции:

1) Создание структуры матрицы из массива значений её элементов. Как результат возвращается новая матрица.

2) Сложение двух матриц, как результат возвращается новая матрица.

3) Умножение двух матриц, как результат возвращается новая матрица.

4) Умножение переданной матрицы на скаляр.

5) Преобразование переданной матрицы в транспонированную.

6) Нахождение определителя переданной матрицы.

7) Преобразование переданной матрицы в обратную.

8) Вывод в консоль переданной матрицы в виде:

-1.4 3 2.5

3.2 1 -0.4

3.2 4.3 4

[identity]

где identity имеет значение либо «единичная», либо «не единичная»

Вариант 6. «Квадратная матрица 4x4» – Matrix4x4.

Разработать структуру данных **Matrix4x4**, представляющую квадратную матрицу 4 на 4 в виде массива из 16 её элементов, а также логического значения zero, показывающего является ли данная матрица нулевой. В модуле для работы с матрицами должны быть следующие функции:

1) Создание структуры матрицы из массива значений её элементов. Как результат возвращается новая матрица.

2) Сложение двух матриц, как результат возвращается новая матрица.

3) Умножение двух матриц, как результат возвращается новая матрица.

4) Умножение переданной матрицы на скаляр.

5) Нахождение тройки значений соответственно минимального, максимального и среднего значений элементов переданной матрицы.

6) Нахождение определителя переданной матрицы.

7) Преобразование переданной матрицы в обратную.

8) Вывод в консоль переданной матрицы в виде:

-1.4 3 2.5 3.2

3.2 1 -0.4 -3

3.2 4.3 4 1.2

-4 3.1 -3.4 2

[zero]

где zero имеет значение либо «нулевая», либо «не нулевая»

Вариант 7. «Прямоугольник» – Rectangle.

Разработать структуру данных Rectangle, представляющую фигуру выровненного по осям прямоугольника, заданный двумя координатами: соответственно верхней левой и нижней правой вершины, а также логическим значением square, показывающим является ли данный прямоугольник квадратом. В модуле для работы с прямоугольниками должны быть следующие функции:

1) Создание структуры прямоугольника по координатам его двух вершин. Как результат возвращается новый прямоугольник.

2) Перемещение переданного прямоугольника на заданное смещение по оси X и Y.

3) Изменение ширины и высоты для переданного прямоугольника.

4) Вычисление пары значений – площади и периметра – для переданного прямоугольника.

5) Нахождение прямоугольника минимальной площади, внутрь которого попадают все точки, переданные в качестве массива. Как результат возвращается новый прямоугольник.

6) Нахождение прямоугольника, как пересечение двух других переданных прямоугольников. Как результат возвращается новый прямоугольник.

7) Нахождение прямоугольника, вписанного в заданную окружность с координатами центра x,y и радиусом R. Как результат возвращается новый прямоугольник.

8) Вывод в консоль переданного прямоугольника в виде:

(0, 5) (10.5, 5)

(0,0) (10.5,0)

[square]

где square имеет значение либо «квадрат», либо «прямоугольник».

Вариант 8. «Круг» – Circle.

Разработать структуру данных Circle, представляющую фигуру круг, заданный координатами его центра и радиусом. В модуле для работы с кругами должны быть следующие функции:

1) Создание структуры круга по координатам его центра и радиусу. Как результат возвращается новый круг.

2) Определение попадания заданной точки в переданный круг.

3) Определение факта пересечения двух переданных кругов.

4) Определение факта пересечения переданного круга с прямоугольной областью с параметрами (x,y,width,height).

5) Рассчитать площадь пересечения двух переданных кругов.

6) Рассчитать круг, описанный вокруг треугольника, заданный тремя координатами вершин. Как результат возвращается новый круг.

7) Рассчитать круг минимального радиуса, внутрь которого попадают все круги, переданные в качестве массива. Как результат возвращается новый круг.

8) Вывод в консоль переданного прямоугольника в виде:

(x,y) [R]

Вариант 9. «Треугольник» - Triangle.

Разработать структуру данных Triangle, представляющую фигуру треугольник, заданный координатами его вершин, а также логическим значением equilateral, показывающим является ли данный треугольник

равносторонним. В модуле для работы с треугольниками должны быть следующие функции:

1) Создание структуры треугольника по координатам его вершин. Как результат возвращается новый треугольник.

2) Перемещение переданного треугольника на заданное смещение относительно осей X и Y.

3) Расчёт координат центроида переданного треугольника.

4) Поворот переданного треугольника на заданный угол вокруг его центроида.

5) Изменение размера переданного треугольника на заданный коэффициент масштаба относительно его центроида.

6) Создание равностороннего треугольника по заданному размеру стороны, центроид которого совпадает с точкой (0,0). Как результат возвращается новый треугольник.

7) Создание прямоугольного треугольника по двум заданным длинам катетов, совпадающих с положительными полуосями X и Y. Как результат возвращается новый треугольник.

8) Вывод в консоль переданного треугольника в виде:

(3, 9.5)

(2.3, 3.4) (8.6, 2.5)

[equilateral]

где equilateral имеет значение либо «равносторонний», либо «не равносторонний».

Вариант 10. «Многочлен» – Polynom.

Разработать структуру данных **Polynom**, представляющую многочлен от одной переменной вида $ax^n+bx^{n-1}+\ldots dx + e$ степени n (максимальная степень 10), заданный статическим массивом коэффициентов (e, d … b, a) и без знаковым целым числом, выражающее степень n. В модуле для работы с многочленами должны быть следующие функции:

1) Создание структуры многочлена по переданному массиву коэффициентов. Как результат возвращается новый многочлен.

2) Сложение двух многочленов, как результат возвращается новый многочлен.

3) Вычитание двух многочленов, как результат возвращается новый многочлен.

4) Умножение двух многочленов, как результат возвращается новый многочлен.

5) Деление двух многочленов, как результат возвращается новый многочлен (остаток отбрасывается).

6) Умножение переданного многочлена на число.

7) Расчёт значения переданного многочлена для заданного значения переменной.

8) Вывод в консоль переданного многочлена в виде:

a * x^n + b * x^n–1 + … + d * x + e

Вариант 11. «Множество целых чисел» – Set.

Разработать структуру данных Set, представляющую множество целочисленных элементов (с максимальным количеством элементов 50), представленной статическим массивом элементов, а также целочисленной переменной size, задающее текущий размер множества. В модуле для работы с множествами должны быть следующие функции:

1) Создание структуры множества по переданному массиву элементов. Как результат возвращается новое множество.

2) Принадлежность заданного элемента переданному множеству.

3) Добавление элемента к переданному множеству.

4) Удаление элемента из переданного множества.

5) Создать множество, являющееся пересечением двух переданных множеств. Как результат возвращается новое множество.

6) Создать множество, являющееся объединением двух переданных множеств. Как результат возвращается новое множество.

7) Создать множество, являющееся вычитанием двух переданных множеств. Как результат возвращается новое множество.

8) Вывод в консоль переданного множества в виде:

(a,b,c,d,e…) [size]

где size – размер множества.

Вариант 12. «Битовая матрица» – BitMatrix.

Разработать структуру данных **BitMatrix**, представляющую собой матрицу битовых значений представленной статическим массивом *целочисленных значений типа long long (8 байт), задающий строки данной матрицы, а также две целочисленные переменные, задающие текущий размер данной матрицы sizeX* и *sizeY (максимальный размер матрицы 64 на 64 бит).* В модуле для работы с матрицами должны быть следующие функции:

1) Создание структуры битовой матрицы по переданному двумерному массиву элементов типа bool. Как результат возвращается новая матрица.

2) Получение значения бита с заданными индексами в переданной матрице как значение типа bool.

3) Логическое сложение двух равных по размеру матриц. Как результат возвращается новая матрица.

4) Логическое умножение двух равных по размеру матриц. Как результат возвращается новая матрица.

5) Инверсия переданной матрицы.

6) установка для переданной матрицы определённого логического значения для сегмента, заданного как (x,y,width,height).

7) Вычисление количества 0 и 1 в переданной матрице.

8) Вывод в консоль переданной матрицы в виде:

1, 0, 1, 1, 0, 1

1, 1, 1, 0, 0, 1

0, 0, 0, 1, 1, 1

[sizeX, sizeY]

где sizeX и sizeY – текущий размер матрицы.

Вариант 13. «Массив бит» – BitArray.

Разработать структуру данных **BitArray**, представляющую собой массив битовых значений представленный без знаковым long long значением (т.е. максимальной длины 64 бит), хранящим биты и целочисленной переменной size, хранящий текущий размер массива. В модуле для работы с массивом должны быть следующие функции:

1) Создание структуры битового массива по переданному массиву элементов типа bool. Как результат возвращается новый массив.

2) Установка значения бита на заданной позиции переданного массива.

3) Получение значения бита на заданной позиции переданного массива как логическое значение типа bool.

4) Выделение подмассива из переданного массива начиная с i-го бита и длиной n бит. Как результат возвращается новый массив.

5) Циклический сдвиг битов переданного массива вправо\влево на заданное число позиций (если значение сдвига положительное – то вправо, отрицательное – то влево).

6) Логическое сложение двух равных по размеру массивов. Как результат возвращается новый массив.

7) Логическое умножение двух равных по размеру массивов. Как результат возвращается новый массив.

8) Вывод в консоль переданного массива в виде:

1, 0, 1, 1, 0, 1 [size]

где size – текущий размер массива.

Вариант 14. «Комплексное число» – Complex.

Разработать структуру данных Complex, представляющую комплексные числа в виде значений вещественной и мнимой части (a,b), а также логического значения real, показывающего что комплексное число имеет только вещественную часть (мнимая часть равна нулю). В модуле для работы с комплексными числами должны быть следующие функции:

1) Создание структуры комплексного числа из значений действительной и мнимой части. Как результат возвращается новое комплексное число.

2) Сложение двух комплексных чисел, как результат возвращается новое комплексное число.

3) Вычитание двух комплексных чисел, как результат возвращается новое комплексное число.

4) Умножение двух комплексных чисел, как результат возвращается новое комплексное число.

5) Деление двух комплексных чисел, как результат возвращается новое комплексное число.

6) Преобразование переданного комплексного числа в сопряжённое.

7) Преобразование переданного комплексного числа в обратное.

8) Вывод в консоль переданного комплексного числа в виде:

(*a + b * i) [real],*

где real имеет значение либо «*комплексное*», либо «*вещественное*».

Вариант 15. «**Дробь**» – **Fraction.**

Разработать структуру данных Fraction, представляющую простую дробь в виде пары целых положительных чисел (*m*,*n*) а также отдельно логического значения, указывающего на знак дроби. В модуле для работы с дробями должны быть следующие функции:

1) Создание структуры дроби из значений числителя, знаменателя, а также знака. Как результат возвращается новая дробь.

2) Сложение двух дробей, как результат возвращается новая дробь.

3) Вычитание двух дробей, как результат возвращается новая дробь.

4) Умножение двух дробей, как результат возвращается новая дробь.

5) Деление двух дробей, как результат возвращается новая дробь.

6) Приведение двух переданных дробей к общему знаменателю.

7) Сравнение двух дробей, как результат возвращается целое значение: -1 «меньше», 1 «больше», 0 «равны».

8) Вывод в консоль переданной дроби в виде:

[sign] a / b,

где sign имеет значение либо «+», либо «-».

Вариант 16. «**Двумерный вектор**» – **Vector2d.**

Разработать структуру данных **Vector2d**, представляющую двумерный вектор в виде двух компонент (x,y), а также логического значения norm, показывающего нормализован ли данный вектор. В модуле для работы с двумерными векторами должны быть следующие функции:

1) Создание структуры двумерного вектора из значений его компонент. Как результат возвращается новый вектор.

2) Сложение двух векторов, как результат возвращается новый вектор.

3) Вычитание двух векторов, как результат возвращается новый вектор.

4) Умножение переданного вектора на заданный скаляр.

5) Нормализация переданного вектора.

6) Получение значения скалярного произведения двух векторов.

7) Установка заданной длины для переданного вектора.

8) Вывод в консоль переданного вектора в виде:

(x,y) [norm],

где norm имеет значение либо «*нормализован*», либо «*не нормализован*».

Вариант 17. «**Трёхмерный вектор**» – **Vector3d.**

Разработать структуру данных **Vector3d**, представляющую трёхмерный вектор в виде трех компонент (x,y,z), а также логического значения norm, показывающего нормализован ли данный вектор. В модуле для работы с трёхмерными векторами должны быть следующие функции:

1) Создание структуры трёхмерного вектора из значений его компонент. Как результат возвращается новый вектор.

2) Сложение двух векторов, как результат возвращается новый вектор.

3) Вычитание двух векторов, как результат возвращается новый вектор.

4) Получение значения угла между двумя переданными векторами.

5) Нормализация переданного вектора.

6) Векторное произведение двух векторов, как результат возвращается новый вектор.

7) Проекция вектора на вектор, как результат возвращается новый вектор.

8) Вывод в консоль переданного вектора в виде:

(x,y,z) [norm],

где norm имеет значение либо «*нормализован*», либо «*не нормализован*».

Вариант 18. «**Квадратная матрица 3x3**» – **Matrix3x3**.

Разработать структуру данных **Matrix3x3**, представляющую квадратную матрицу 3 на 3 в виде массива из 9 её элементов, а также логического значения identity, показывающего является ли данная матрица единичной. В модуле для работы с матрицами должны быть следующие функции:

1) Создание структуры матрицы из массива значений её элементов. Как результат возвращается новая матрица.

2) Сложение двух матриц, как результат возвращается новая матрица.

3) Умножение двух матриц, как результат возвращается новая матрица.

4) Умножение переданной матрицы на скаляр.

5) Преобразование переданной матрицы в транспонированную.

6) Нахождение определителя переданной матрицы.

7) Преобразование переданной матрицы в обратную.

8) Вывод в консоль переданной матрицы в виде:

-1.4 3 2.5

3.2 1 -0.4

3.2 4.3 4

[identity]

где identity имеет значение либо «единичная», либо «не единичная»

Вариант 19. «**Квадратная матрица 4x4**» – **Matrix4x4**.

Разработать структуру данных **Matrix4x4**, представляющую квадратную матрицу 4 на 4 в виде массива из 16 её элементов, а также логического значения zero, показывающего является ли данная матрица нулевой. В модуле для работы с матрицами должны быть следующие функции:

1) Создание структуры матрицы из массива значений её элементов. Как результат возвращается новая матрица.

2) Сложение двух матриц, как результат возвращается новая матрица.

3) Умножение двух матриц, как результат возвращается новая матрица.

4) Умножение переданной матрицы на скаляр.

5) Нахождение тройки значений соответственно минимального, максимального и среднего значений элементов переданной матрицы.

6) Нахождение определителя переданной матрицы.

7) Преобразование переданной матрицы в обратную.

8) Вывод в консоль переданной матрицы в виде:

```
-1.4    3    2.5   3.2
 3.2    1   -0.4   -3
 3.2  4.3      4   1.2
  -4  3.1   -3.4    2
```

[zero]

где zero имеет значение либо «нулевая», либо «не нулевая»

Вариант 20. «Прямоугольник» – Rectangle.

Разработать структуру данных Rectangle, представляющую фигуру выровненного по осям прямоугольника, заданный двумя координатами: соответственно верхней левой и нижней правой вершины, а также логическим значением square, показывающим является ли данный прямоугольник квадратом. В модуле для работы с прямоугольниками должны быть следующие функции:

1) Создание структуры прямоугольника по координатам его двух вершин. Как результат возвращается новый прямоугольник.

2) Перемещение переданного прямоугольника на заданное смещение по оси X и Y.

3) Изменение ширины и высоты для переданного прямоугольника.

4) Вычисление пары значений – площади и периметра – для переданного прямоугольника.

5) Нахождение прямоугольника минимальной площади, внутрь которого попадают все точки, переданные в качестве массива. Как результат возвращается новый прямоугольник.

6) Нахождение прямоугольника, как пересечение двух других переданных прямоугольников. Как результат возвращается новый прямоугольник.

7) Нахождение прямоугольника, вписанного в заданную окружность с координатами центра x,y и радиусом R. Как результат возвращается новый прямоугольник.

8) Вывод в консоль переданного прямоугольника в виде:

(0, 5) (10.5, 5)

(0,0) (10.5,0)

[square]

где square имеет значение либо «квадрат», либо «прямоугольник».

Вариант 21. «Круг» – Circle.

Разработать структуру данных Circle, представляющую фигуру круг, заданный координатами его центра и радиусом. В модуле для работы с кругами должны быть следующие функции:

1) Создание структуры круга по координатам его центра и радиусу. Как результат возвращается новый круг.

2) Определение попадания заданной точки в переданный круг.

3) Определение факта пересечения двух переданных кругов.

4) Определение факта пересечения переданного круга с прямоугольной областью с параметрами (x,y,width,height).

5) Рассчитать площадь пересечения двух переданных кругов.

6) Рассчитать круг, описанный вокруг треугольника, заданный тремя координатами вершин. Как результат возвращается новый круг.

7) Рассчитать круг минимального радиуса, внутрь которого попадают все круги, переданные в качестве массива. Как результат возвращается новый круг.

8) Вывод в консоль переданного прямоугольника в виде:

(x,y) [R]

Вариант 22. «Треугольник» - Triangle.

Разработать структуру данных Triangle, представляющую фигуру треугольник, заданный координатами его вершин, а также логическим значением equilateral, показывающим является ли данный треугольник равносторонним. В модуле для работы с треугольниками должны быть следующие функции:

1) Создание структуры треугольника по координатам его вершин. Как результат возвращается новый треугольник.

2) Перемещение переданного треугольника на заданное смещение относительно осей X и Y.

3) Расчёт координат центроида переданного треугольника.

4) Поворот переданного треугольника на заданный угол вокруг его центроида.

5) Изменение размера переданного треугольника на заданный коэффициент масштаба относительно его центроида.

6) Создание равностороннего треугольника по заданному размеру стороны, центроид которого совпадает с точкой (0,0). Как результат возвращается новый треугольник.

7) Создание прямоугольного треугольника по двум заданным длинам катетов, совпадающих с положительными полуосями X и Y. Как результат возвращается новый треугольник.

8) Вывод в консоль переданного треугольника в виде:

(3, 9.5)

(2.3, 3.4) (8.6, 2.5)

[equilateral]

где equilateral имеет значение либо «равносторонний», либо «не равносторонний».

Вариант 23. «Многочлен» – Polynom.

Разработать структуру данных **Polynom**, представляющую многочлен от одной переменной вида $ax^n+bx^{n-1}+\ldots dx + e$ степени n (максимальная степень 10), заданный статическим массивом коэффициентов (e, d … b, a) и без

знаковым целым числом, выражающее степень n. В модуле для работы с многочленами должны быть следующие функции:

1) Создание структуры многочлена по переданному массиву коэффициентов. Как результат возвращается новый многочлен.

2) Сложение двух многочленов, как результат возвращается новый многочлен.

3) Вычитание двух многочленов, как результат возвращается новый многочлен.

4) Умножение двух многочленов, как результат возвращается новый многочлен.

5) Деление двух многочленов, как результат возвращается новый многочлен (остаток отбрасывается).

6) Умножение переданного многочлена на число.

7) Расчёт значения переданного многочлена для заданного значения переменной.

8) Вывод в консоль переданного многочлена в виде:

a * x^n + b * x^n–1 + … + d * x + e

Вариант 24. «Множество целых чисел» – Set.

Разработать структуру данных Set, представляющую множество целочисленных элементов (с максимальным количеством элементов 50), представленной статическим массивом элементов, а также целочисленной переменной size, задающее текущий размер множества. В модуле для работы с множествами должны быть следующие функции:

1) Создание структуры множества по переданному массиву элементов. Как результат возвращается новое множество.

2) Принадлежность заданного элемента переданному множеству.

3) Добавление элемента к переданному множеству.

4) Удаление элемента из переданного множества.

5) Создать множество, являющееся пересечением двух переданных множеств. Как результат возвращается новое множество.

6) Создать множество, являющееся объединением двух переданных множеств. Как результат возвращается новое множество.

7) Создать множество, являющееся вычитанием двух переданных множеств. Как результат возвращается новое множество.

8) Вывод в консоль переданного множества в виде:

(a,b,c,d,e…) [size]

где size – размер множества.

Вариант 25. «Битовая матрица» – BitMatrix.

Разработать структуру данных **BitMatrix**, представляющую собой матрицу битовых значений представленной статическим массивом *целочисленных значений типа long long (8 байт), задающий строки данной матрицы, а также две целочисленные переменные, задающие текущий размер данной матрицы sizeX* и *sizeY (максимальный размер матрицы 64 на 64 бит).* В модуле для работы с матрицами должны быть следующие функции:

1) Создание структуры битовой матрицы по переданному двумерному массиву элементов типа bool. Как результат возвращается новая матрица.

2) Получение значения бита с заданными индексами в переданной матрице как значение типа bool.

3) Логическое сложение двух равных по размеру матриц. Как результат возвращается новая матрица.

4) Логическое умножение двух равных по размеру матриц. Как результат возвращается новая матрица.

5) Инверсия переданной матрицы.

6) установка для переданной матрицы определённого логического значения для сегмента, заданного как (x,y,width,height).

7) Вычисление количества 0 и 1 в переданной матрице.

8) Вывод в консоль переданной матрицы в виде:

1, 0, 1, 1, 0, 1

1, 1, 1, 0, 0, 1

0, 0, 0, 1, 1, 1

[sizeX, sizeY]

где sizeX и sizeY – текущий размер матрицы.

Вариант 26. «**Массив бит**» – **BitArray.**

Разработать структуру данных **BitArray**, представляющую собой массив битовых значений представленный без знаковым long long значением (т.е. максимальной длины 64 бит), хранящим биты и целочисленной переменной size, хранящий текущий размер массива. В модуле для работы с массивом должны быть следующие функции:

1) Создание структуры битового массива по переданному массиву элементов типа bool. Как результат возвращается новый массив.

2) Установка значения бита на заданной позиции переданного массива.

3) Получение значения бита на заданной позиции переданного массива как логическое значение типа bool.

4) Выделение подмассива из переданного массива начиная с i-го бита и длиной n бит. Как результат возвращается новый массив.

5) Циклический сдвиг битов переданного массива вправо\влево на заданное число позиций (если значение сдвига положительное – то вправо, отрицательное – то влево).

6) Логическое сложение двух равных по размеру массивов. Как результат возвращается новый массив.

7) Логическое умножение двух равных по размеру массивов. Как результат возвращается новый массив.

8) Вывод в консоль переданного массива в виде:

1, 0, 1, 1, 0, 1 [size]

где size – текущий размер массива.

Вариант 27. «**Комплексное число**» – **Complex.**

Разработать структуру данных Complex, представляющую комплексные числа в виде значений вещественной и мнимой части (a,b), а также логического значения real, показывающего что комплексное число имеет только вещественную часть (мнимая часть равна нулю). В модуле для работы с комплексными числами должны быть следующие функции:

1) Создание структуры комплексного числа из значений действительной и мнимой части. Как результат возвращается новое комплексное число.

2) Сложение двух комплексных чисел, как результат возвращается новое комплексное число.

3) Вычитание двух комплексных чисел, как результат возвращается новое комплексное число.

4) Умножение двух комплексных чисел, как результат возвращается новое комплексное число.

5) Деление двух комплексных чисел, как результат возвращается новое комплексное число.

6) Преобразование переданного комплексного числа в сопряжённое.

7) Преобразование переданного комплексного числа в обратное.

8) Вывод в консоль переданного комплексного числа в виде:

(*a + b * i) [real],*

где real имеет значение либо «*комплексное*», либо «*вещественное*».

Вариант 28. «**Дробь**» – **Fraction.**

Разработать структуру данных Fraction, представляющую простую дробь в виде пары целых положительных чисел (*m,n*) а также отдельно логического значения, указывающего на знак дроби. В модуле для работы с дробями должны быть следующие функции:

1) Создание структуры дроби из значений числителя, знаменателя, а также знака. Как результат возвращается новая дробь.

2) Сложение двух дробей, как результат возвращается новая дробь.

3) Вычитание двух дробей, как результат возвращается новая дробь.

4) Умножение двух дробей, как результат возвращается новая дробь.

5) Деление двух дробей, как результат возвращается новая дробь.

6) Приведение двух переданных дробей к общему знаменателю.

7) Сравнение двух дробей, как результат возвращается целое значение: -1 «меньше», 1 «больше», 0 «равны».

8) Вывод в консоль переданной дроби в виде:

[sign] a / b,

где sign имеет значение либо «+», либо «-».

Вариант 29. «**Двумерный вектор**» – **Vector2d.**

Разработать структуру данных **Vector2d**, представляющую двумерный вектор в виде двух компонент (x,y), а также логического значения norm, показывающего нормализован ли данный вектор. В модуле для работы с двумерными векторами должны быть следующие функции:

1) Создание структуры двумерного вектора из значений его компонент. Как результат возвращается новый вектор.

2) Сложение двух векторов, как результат возвращается новый вектор.

3) Вычитание двух векторов, как результат возвращается новый вектор.

4) Умножение переданного вектора на заданный скаляр.

5) Нормализация переданного вектора.

6) Получение значения скалярного произведения двух векторов.

7) Установка заданной длины для переданного вектора.

8) Вывод в консоль переданного вектора в виде:

(x,y) [norm],

где norm имеет значение либо «*нормализован*», либо «*не нормализован*».

Вариант 30. «**Трёхмерный вектор**» – **Vector3d.**

Разработать структуру данных **Vector3d**, представляющую трёхмерный вектор в виде трех компонент (x,y,z), а также логического значения norm, показывающего нормализован ли данный вектор. В модуле для работы с трёхмерными векторами должны быть следующие функции:

1) Создание структуры трёхмерного вектора из значений его компонент. Как результат возвращается новый вектор.

2) Сложение двух векторов, как результат возвращается новый вектор.

3) Вычитание двух векторов, как результат возвращается новый вектор.

4) Получение значения угла между двумя переданными векторами.

5) Нормализация переданного вектора.

6) Векторное произведение двух векторов, как результат возвращается новый вектор.

7) Проекция вектора на вектор, как результат возвращается новый вектор.

8) Вывод в консоль переданного вектора в виде:

(x,y,z) [norm],

где norm имеет значение либо «*нормализован*», либо «*не нормализован*».

Вариант 31. «**Квадратная матрица 3x3**» – **Matrix3x3**.

Разработать структуру данных **Matrix3x3**, представляющую квадратную матрицу 3 на 3 в виде массива из 9 её элементов, а также логического значения identity, показывающего является ли данная матрица единичной. В модуле для работы с матрицами должны быть следующие функции:

1) Создание структуры матрицы из массива значений её элементов. Как результат возвращается новая матрица.

2) Сложение двух матриц, как результат возвращается новая матрица.

3) Умножение двух матриц, как результат возвращается новая матрица.

4) Умножение переданной матрицы на скаляр.

5) Преобразование переданной матрицы в транспонированную.

6) Нахождение определителя переданной матрицы.

7) Преобразование переданной матрицы в обратную.

8) Вывод в консоль переданной матрицы в виде:

-1.4 3 2.5

3.2 1 -0.4

3.2 4.3 4

[identity]

где identity имеет значение либо «единичная», либо «не единичная»

Вариант 32. «**Квадратная матрица 4x4**» – **Matrix4x4**.

Разработать структуру данных **Matrix4x4**, представляющую квадратную матрицу 4 на 4 в виде массива из 16 её элементов, а также логического значения zero, показывающего является ли данная матрица нулевой. В модуле для работы с матрицами должны быть следующие функции:

1) Создание структуры матрицы из массива значений её элементов. Как результат возвращается новая матрица.

2) Сложение двух матриц, как результат возвращается новая матрица.

3) Умножение двух матриц, как результат возвращается новая матрица.

4) Умножение переданной матрицы на скаляр.

5) Нахождение тройки значений соответственно минимального, максимального и среднего значений элементов переданной матрицы.

6) Нахождение определителя переданной матрицы.

7) Преобразование переданной матрицы в обратную.

8) Вывод в консоль переданной матрицы в виде:

-1.4 3 2.5 3.2

3.2 1 -0.4 -3

3.2 4.3 4 1.2

-4 3.1 -3.4 2

[zero]

где zero имеет значение либо «нулевая», либо «не нулевая»

Вариант 33. «Прямоугольник» – Rectangle.

Разработать структуру данных Rectangle, представляющую фигуру выровненного по осям прямоугольника, заданный двумя координатами: соответственно верхней левой и нижней правой вершины, а также логическим значением square, показывающим является ли данный прямоугольник квадратом. В модуле для работы с прямоугольниками должны быть следующие функции:

1) Создание структуры прямоугольника по координатам его двух вершин. Как результат возвращается новый прямоугольник.

2) Перемещение переданного прямоугольника на заданное смещение по оси X и Y.

3) Изменение ширины и высоты для переданного прямоугольника.

4) Вычисление пары значений – площади и периметра – для переданного прямоугольника.

5) Нахождение прямоугольника минимальной площади, внутрь которого попадают все точки, переданные в качестве массива. Как результат возвращается новый прямоугольник.

6) Нахождение прямоугольника, как пересечение двух других переданных прямоугольников. Как результат возвращается новый прямоугольник.

7) Нахождение прямоугольника, вписанного в заданную окружность с координатами центра x,y и радиусом R. Как результат возвращается новый прямоугольник.

8) Вывод в консоль переданного прямоугольника в виде:

(0, 5) (10.5, 5)

(0,0) (10.5,0)

[square]

где square имеет значение либо «квадрат», либо «прямоугольник».

Лабораторная работа №2
Простые классы

В данной лабораторной работе необходимо преобразовать лабораторную работу №1 таким образом, чтобы заданная сущность была представлена в виде класса. Класс должен быть обязательно оформлен отдельными файлами описания (.h) и реализации (.cpp).

Данный класс должен иметь конструктор с параметрами, необходимыми для создания сущности, методы для реализации всех нужных операций над сущностью. Доступ к полям класса должен быть приватным, извне доступ к ним осуществляется исключительно через геттеры и сеттеры класса.

Также в данном задании необходимо написать класс юнит-тестов для каждого нетривиального метода созданного класса (см. ниже «Юнит-тестирование»). Количество тестов должно быть не менее 10. Результаты работы юнит-тестов необходимо выводить в консоль. Функция main должна содержать только запуск всех юнит-тестов.

Каждый тест должен быть написан по принципу arrange -> act -> assert и осуществлять вывод в консоль следующей информации:

- Название теста (какая функция тестируется)
- Исходные данные
- Ожидаемый результат операции

- Фактический результат операции
- Тест пройден \ Тест провален

В конце класс юнит-тестов должен вывести количество пройдённых и проваленных тестов.

Юнит-тестирование

Модульное тестирование или юнит-тестирование (unit testing) — процесс позволяющий проверить на корректность отдельные модули исходного кода программы вместе с соответствующими управляющими данными, процедурами использования и обработки.

Идея юнит-тестирования состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок. Например, обновить используемую в проекте библиотеку до актуальной версии можно в любой момент, прогнав тесты и выявив несовместимости.

Юнит-тестирование – это первый рубеж на борьбе с ошибками в коде программ. За ним следует еще интеграционное, приемочное и ручное тестирование (в том числе «свободный поиск»). Любой долгосрочный проект без надлежащего покрытия тестами обречен рано или поздно быть переписанным с нуля.

Каждый юнит-тест должен быть атомарным, т.е. один тест проверяет только одну функцию. Юнит-тест является спецификацией метода класса, контрактом: какие входные параметры ожидает этот метод, и что остальные компоненты системы ждут от него на выходе.

Написание любого юнит-теста начинается с выбора его имени. Один из рекомендуемых подходов – формировать его имя из трех частей:

- имя тестируемой рабочей единицы

- сценарий теста
- ожидаемый результат

Таким образом название читается как завершенное предложение, а это повышает простоту работы с тестами. Чтобы понять, что тестируется, достаточно, без вникания в логику работы кода, просто прочитать названия тестов.

Но в ряде случаев с логикой работы тестового кода все-таки нужно разбираться. Чтобы упростить эту работу, структуру юнит-теста можно формировать из трех частей:

- Arrange — здесь производится создание и инициализация требуемых для проведения теста объектов
- Act — собственно проведение тестируемого действия
- Assert — здесь производится сравнение полученного результата с эталонным

Для того чтобы повысить читабельность тестов рекомендуется эти части отделять друг от друга пустой строкой. Это сориентирует тех, кто читает ваш код, и поможет быстрее найти ту часть теста, которая их интересует больше всего.

Приведём пример класса для юнит-тестирования: см. проект UnitTests.dev

Лабораторная работа №3
Наследование, статические члены, перегрузка операторов

В данной лабораторной работе необходимо изменить класс, созданный в лабораторной работе №2 следующим образом:

1. Добавить для методов параметры-ссылки там, где это необходимо.
2. Добавить константные методы и параметры там, где это оправданно.
3. Заменить методы, реализующие операции над сущностью, перегруженными операторами (соответствующими смыслу операции). Например операция add() заменяется оператором +.
4. Добавить статический метод для класса, возвращающий новый объект, заполненный случайными значениями из заданного диапазона

(который задаётся через параметры данного метода). Для бинарных значений диапазон не указывается.

5. Добавить наследование от базового класса Object, в котором должна быть реализация следующего функционала:
    a. Подсчёт количества созданных объектов и количества активных объектов на данный момент через статические поля класса. Статический метод printTotalInfo для вывода этой информации в консоль.
    b. Функционал для создания **динамического** массива (через malloc или через new) – списка всех выполненных операций с объектом. При выполнении каждой операции её обозначение заносится в данный массив через метод addOp. Метод clearOp – очищает массив операций. Метод printOp – выводит полный список всех выполненных операций с объектом.
    c. Конструктор копии и оператор присваивания (=) для класса Object.
6. Дополнить класс юнит-тестирования **минимум 5 тестами** для проверки работы базового класса Object и статического метода для случайной генерации класса сущности.

Важно: при перегрузке оператора присваивания (=) необходимо возвращать ссылку на текущий объект (*this), чтобы можно было их использовать в цепочке вызовов. Также стоит всегда проверять на самоприсваивание (особенно при наличии выделения динамической памяти), и не выполнять лишних действий, сразу возвращая ссылку на текущий объект.

Лабораторная работа №4
Полиморфизм, абстрактные классы

В данной лабораторной работе необходимо разработать классы в соответствии со своей предметной областью (по вариантам ниже). Все задания состоят из набора сущностей, которые будет необходимо оформить в правильные отношения между объектами. В вариантах заданий описаны только обязательные сущности, можно добавлять свои по необходимости.

В каждом задании есть сущность базового абстрактного класса, которую реализуют классы-наследники. Также есть композитная сущность – это класс, который владеет односвязным списком указателей на базовый абстрактный класс, может добавлять и удалять экземпляры, а также выполнять другие операции над ними (через полиморфный вызов методов). При добавлении\удалении объектов используется динамическое выделение

памяти через new\delete. Список необходимо реализовать вручную, через указатели.

В программе необходимо предусмотреть возможность случайной генерации **N** объектов-сущностей из варианта задания и их добавления в композитную сущность, возможность удаления объектов по указанному номеру из списка, а также отображения результатов выполнения действий над объектами.

Вариант 1.

Сущности: преподаватель, персона, университет, студент, завкафедрой, научный сотрудник.

Действия: суммарный расчёт выплачиваемых денежных средств; средняя зарплата по каждому типу сущностей; самый младший и самый старший человек по каждому типу сущностей; вывод информации по каждому человеку включая список аффилированных с ним предметов.

Описание сущностей:
Студент получает стипендию из расчёта «базовое значение» * «коэффициент успеваемости». Коэффициент рассчитывается исходя из оценок сданных предметов: все пятёрки – коэффициент 1.5, есть четвёрки – 1.0, есть тройки – 0.5, есть двойки – 0.0. Кроме того студент может быть практикантом у научного сотрудника.

Преподаватель получает зарплату в зависимости от кол-ва часов, которые он ведёт по каждому предмету. Рассчитывается зарплата как «базовое значение» * «коэффициент ставки» (суммарно 60 часов = коэффициент ставки 1.0).

Зав. кафедры – получает к зарплате надбавку к зарплате 60%. Кроме того, имеет нескольких преподавателей – заместителей.

Научный сотрудник получает зарплату исходя из часов, потраченных на исследование предметов. Каждый час работы научного сотрудника по каждому предмету оплачивается отдельно как «базовое значение» * 0.05. Также имеет нескольких студентов – практикантов.

Зав. кафедры и научные сотрудники обязаны быть преподавателями.

Каждый из типов людей должен иметь своё ФИО и возраст.

Вариант 2.

Сущности: инженер, завод, персонал, рабочий, завхоз, начальник цеха.

Действия: суммарный расчёт выплачиваемых денежных средств; минимальная и максимальная зарплата по каждой должности; общее

количество людей, занятых на каждом типе должности; вывод информации по каждому человеку включая список аффилированных с ним выпуска деталей.

Описание сущностей:

Рабочий создаёт разного типа деталей, для каждой детали указывается норма выработки и фактическая выработка. Коэффициент выполнения нормы по каждой детали рассчитывается как «фактическая выработка» / «норма выработки». Рабочему выплачивают зарплату как «базовое значение» * «коэффициент производительности». Коэффициент производительности – это среднее арифметическое всех коэффициентов выполнения норм по каждой детали.

У инженера есть подчинённые - несколько рабочих. Инженеру выплачивают зарплату как «базовое значение» * «коэффициент производительности». Для инженера коэффициент производительности – это среднее арифметическое коэффициентов производительности его подчиненных. Кроме того, ему выплачивают надбавку 10% за перевыполнение нормы подчинёнными.

У начальника цеха есть подчинённые - несколько инженеров. Зарплату ему выплачивают фиксированную «базовое значение» + «премия». Премия рассчитывается как 50% от «базового значения» если средний коэффициент производительности подчинённых был выше на 30% от нормы и 25% от «базового значения» если средний коэффициент выше на 10%. Кроме того, содержит информацию о закреплённом за его цехом завхозе.

Завхозом может быть любой работник, к его зарплате доплачивается 60%. Также завхоз имеет информацию о нескольких начальников цехов, для которых он поставляет материалы со склада.

Каждый из типов людей должен иметь своё ФИО и возраст.

Вариант 3.

Сущности: юридическое лицо, страховая компания, автомобильный завод, налоговая инспекция, завод микроэлектроники, завод;

Действия: суммарный расчёт выплачиваемых налогов; общая сумма налоговых льгот; средний доход по каждому заводу; вывод информации по каждому субъекту включая список аффилированного с ним выпуска продукции, услуг либо кол-ва клиентов.

Описание сущностей:

Автомобильный завод и завод микроэлектроники выпускают продукцию. На каждый тип выпускаемой продукции установлена своя ставка налога, таким образом общий налог рассчитывается как сумма по всем типам продукций: «стоимость продукции» * «ставка налога на продукцию» * «кол-во выпущенной продукции».

Автомобильный завод также может оказывать несколько типов сервисных услуг, налог суммарно с которых составляет 12% с каждой оказанной услуги. Кроме того, автомобильный завод имеет информацию о своих поставщиках – других заводах.

Завод микроэлектроники может получать налоговую льготу, в зависимости от значимости направления для государства. Если завод высокой значимости – льгота составляет 35% от уплачиваемого налога, средней значимости - 15% от налога.

Страховая компания имеет список клиентов – заводов, каждый из которых характеризуется страховой премией выплаченной компании, и суммой страховых выплат (если 0 – страховых выплат не было). Налог вычисляется только с той суммы премий, которая была получена от клиентов, у которых не было страховых выплат. Налог составляет 25% если клиентов меньше 100, в другом случае 10%.

Каждое предприятие имеет название и год основания.

Вариант 4.

Сущности: республика, мир, государство, монархия, президентская республика, конфедерация;

Действия: суммарный расчёт количества людей, занятых в управлении в мире; расчёт минимального и максимального возраста управленца по каждому государству; средний возраст управленцев в каждом государстве; вывод информации по каждому субъекту включая список аффилированных с ним управленцев, с указанием их должностной характеристики.

Описание сущностей:

В республике управляет парламент, список состоит из парламентариев. Каждый парламентарий может быть обычным либо спикером. Также парламентарий может состоять в какой-либо партии либо быть беспартийным.

В президентской республике помимо парламента присутствует президент, а также кабинет министров. Министры могут быть главой какого-либо

министерства, либо быть премьер-министром. Все министры являются парламентариями.

В монархии присутствует верховный монарх, а также список членов его семьи. У каждого члена семьи есть приоритет по престолонаследию.

Конфедерация – является по совокупностью разных государств (т.е. может состоять из нескольких государств со своими формами правления). Имеет палату представителей, для решения общих вопросов.

Каждый управленец имеет ФИО и возраст.

Вариант 5.

Сущности: деталь, механизм, изделие, узел, устройство, датчик.

Действия: расчёт стоимости и веса всего устройства; расчёт среднего количества деталей на каждое устройство; расчёт общего количества датчиков и их суммарную стоимость по каждому узлу; вывод информации по каждому объекту, включая его состав.

Описание сущностей:

Любое изделие имеет вес и стоимость. Стоимость складывается из стоимости материалов и работы по изготовлению.

Деталь является частью механизма. Механизм состоит из нескольких деталей. Также механизм содержит информацию о расчётном времени износа. Вес и стоимость механизма складывается из соответственно веса и стоимости его деталей. Может включать в себя несколько датчиков. Каждый датчик повышает стоимость устройства.

Узел состоит из нескольких механизмов. Вес, стоимость, расчётное время износа узла складывается из соответственно веса, стоимости и износа его механизмов. Кроме того, есть коэффициент сложности транспортировки: если вес превышает 20 кг. – стоимость повышается на 10%, если превышает 50 кг. – то на 20%.

Датчик имеет стоимость, вес не учитывается. Имеет тип замеряемой характеристики (температура, вибрация и т.д.)

Все объекты имеют названия.

Вариант 6.

Сущности: преподаватель, персона, университет, студент, завкафедрой, научный сотрудник.

Действия: суммарный расчёт выплачиваемых денежных средств; средняя зарплата по каждому типу сущностей; самый младший и самый старший человек по каждому типу сущностей; вывод информации по каждому человеку включая список аффилированных с ним предметов.

Описание сущностей:
Студент получает стипендию из расчёта «базовое значение» * «коэффициент успеваемости». Коэффициент рассчитывается исходя из оценок сданных предметов: все пятёрки – коэффициент 1.5, есть четвёрки – 1.0, есть тройки – 0.5, есть двойки – 0.0. Кроме того студент может быть практикантом у научного сотрудника.

Преподаватель получает зарплату в зависимости от кол-ва часов, которые он ведёт по каждому предмету. Рассчитывается зарплата как «базовое значение» * «коэффициент ставки» (суммарно 60 часов = коэффициент ставки 1.0).

Зав. кафедры – получает к зарплате надбавку к зарплате 60%. Кроме того, имеет нескольких преподавателей – заместителей.

Научный сотрудник получает зарплату исходя из часов, потраченных на исследование предметов. Каждый час работы научного сотрудника по каждому предмету оплачивается отдельно как «базовое значение» * 0.05. Также имеет нескольких студентов – практикантов.

Зав. кафедры и научные сотрудники обязаны быть преподавателями.

Каждый из типов людей должен иметь своё ФИО и возраст.

Вариант 7.

Сущности: инженер, завод, персонал, рабочий, завхоз, начальник цеха.

Действия: суммарный расчёт выплачиваемых денежных средств; минимальная и максимальная зарплата по каждой должности; общее количество людей, занятых на каждом типе должности; вывод информации по каждому человеку включая список аффилированных с ним выпуска деталей.

Описание сущностей:

Рабочий создаёт разного типа деталей, для каждой детали указывается норма выработки и фактическая выработка. Коэффициент выполнения нормы по каждой детали рассчитывается как «фактическая выработка» / «норма выработки». Рабочему выплачивают зарплату как «базовое значение» * «коэффициент производительности». Коэффициент производительности –

это среднее арифметическое всех коэффициентов выполнения норм по каждой детали.

У инженера есть подчинённые - несколько рабочих. Инженеру выплачивают зарплату как «базовое значение» * «коэффициент производительности». Для инженера коэффициент производительности – это среднее арифметическое коэффициентов производительности его подчиненных. Кроме того, ему выплачивают надбавку 10% за перевыполнение нормы подчинёнными.

У начальника цеха есть подчинённые - несколько инженеров. Зарплату ему выплачивают фиксированную «базовое значение» + «премия». Премия рассчитывается как 50% от «базового значения» если средний коэффициент производительности подчинённых был выше на 30% от нормы и 25% от «базового значения» если средний коэффициент выше на 10%. Кроме того, содержит информацию о закреплённом за его цехом завхозе.

Завхозом может быть любой работник, к его зарплате доплачивается 60%. Также завхоз имеет информацию о нескольких начальников цехов, для которых он поставляет материалы со склада.

Каждый из типов людей должен иметь своё ФИО и возраст.

Вариант 8.

Сущности: юридическое лицо, страховая компания, автомобильный завод, налоговая инспекция, завод микроэлектроники, завод;

Действия: суммарный расчёт выплачиваемых налогов; общая сумма налоговых льгот; средний доход по каждому заводу; вывод информации по каждому субъекту включая список аффилированного с ним выпуска продукции, услуг либо кол-ва клиентов.

Описание сущностей:

Автомобильный завод и завод микроэлектроники выпускают продукцию. На каждый тип выпускаемой продукции установлена своя ставка налога, таким образом общий налог рассчитывается как сумма по всем типам продукций: «стоимость продукции» * «ставка налога на продукцию» * «кол-во выпущенной продукции».

Автомобильный завод также может оказывать несколько типов сервисных услуг, налог суммарно с которых составляет 12% с каждой оказанной услуги. Кроме того, автомобильный завод имеет информацию о своих поставщиках – других заводах.

Завод микроэлектроники может получать налоговую льготу, в зависимости от значимости направления для государства. Если завод высокой значимости – льгота составляет 35% от уплачиваемого налога, средней значимости - 15% от налога.

Страховая компания имеет список клиентов – заводов, каждый из которых характеризуется страховой премией выплаченной компании, и суммой страховых выплат (если 0 – страховых выплат не было). Налог вычисляется только с той суммы премий, которая была получена от клиентов, у которых не было страховых выплат. Налог составляет 25% если клиентов меньше 100, в другом случае 10%.

Каждое предприятие имеет название и год основания.

Вариант 9.

Сущности: республика, мир, государство, монархия, президентская республика, конфедерация;

Действия: суммарный расчёт количества людей, занятых в управлении в мире; расчёт минимального и максимального возраста управленца по каждому государству; средний возраст управленцев в каждом государстве; вывод информации по каждому субъекту включая список аффилированных с ним управленцев, с указанием их должностной характеристики.

Описание сущностей:

В республике управляет парламент, список состоит из парламентариев. Каждый парламентарий может быть обычным либо спикером. Также парламентарий может состоять в какой-либо партии либо быть беспартийным.

В президентской республике помимо парламента присутствует президент, а также кабинет министров. Министры могут быть главой какого-либо министерства, либо быть премьер-министром. Все министры являются парламентариями.

В монархии присутствует верховный монарх, а также список членов его семьи. У каждого члена семьи есть приоритет по престолонаследию.

Конфедерация – является по совокупностью разных государств (т.е. может состоять из нескольких государств со своими формами правления). Имеет палату представителей, для решения общих вопросов.

Каждый управленец имеет ФИО и возраст.

Вариант 10.

Сущности: деталь, механизм, изделие, узел, устройство, датчик.

Действия: расчёт стоимости и веса всего устройства; расчёт среднего количества деталей на каждое устройство; расчёт общего количества датчиков и их суммарную стоимость по каждому узлу; вывод информации по каждому объекту, включая его состав.

Описание сущностей:

Любое изделие имеет вес и стоимость. Стоимость складывается из стоимости материалов и работы по изготовлению.

Деталь является частью механизма. Механизм состоит из нескольких деталей. Также механизм содержит информацию о расчётном времени износа. Вес и стоимость механизма складывается из соответственно веса и стоимости его деталей. Может включать в себя несколько датчиков. Каждый датчик повышает стоимость устройства.

Узел состоит из нескольких механизмов. Вес, стоимость, расчётное время износа узла складывается из соответственно веса, стоимости и износа его механизмов. Кроме того, есть коэффициент сложности транспортировки: если вес превышает 20 кг. – стоимость повышается на 10%, если превышает 50 кг. – то на 20%.

Датчик имеет стоимость, вес не учитывается. Имеет тип замеряемой характеристики (температура, вибрация и т.д.)

Все объекты имеют названия.

Вариант 11.

Сущности: преподаватель, персона, университет, студент, завкафедрой, научный сотрудник.

Действия: суммарный расчёт выплачиваемых денежных средств; средняя зарплата по каждому типу сущностей; самый младший и самый старший человек по каждому типу сущностей; вывод информации по каждому человеку включая список аффилированных с ним предметов.

Описание сущностей:
Студент получает стипендию из расчёта «базовое значение» * «коэффициент успеваемости». Коэффициент рассчитывается исходя из оценок сданных предметов: все пятёрки – коэффициент 1.5, есть четвёрки – 1.0, есть тройки – 0.5, есть двойки – 0.0. Кроме того студент может быть практикантом у научного сотрудника.

Преподаватель получает зарплату в зависимости от кол-ва часов, которые он ведёт по каждому предмету. Рассчитывается зарплата как «базовое значение» * «коэффициент ставки» (суммарно 60 часов = коэффициент ставки 1.0).

Зав. кафедры – получает к зарплате надбавку к зарплате 60%. Кроме того, имеет нескольких преподавателей – заместителей.

Научный сотрудник получает зарплату исходя из часов, потраченных на исследование предметов. Каждый час работы научного сотрудника по каждому предмету оплачивается отдельно как «базовое значение» * 0.05. Также имеет нескольких студентов – практикантов.

Зав. кафедры и научные сотрудники обязаны быть преподавателями.

Каждый из типов людей должен иметь своё ФИО и возраст.

Вариант 12.

Сущности: инженер, завод, персонал, рабочий, завхоз, начальник цеха.

Действия: суммарный расчёт выплачиваемых денежных средств; минимальная и максимальная зарплата по каждой должности; общее количество людей, занятых на каждом типе должности; вывод информации по каждому человеку включая список аффилированных с ним выпуска деталей.

Описание сущностей:

Рабочий создаёт разного типа деталей, для каждой детали указывается норма выработки и фактическая выработка. Коэффициент выполнения нормы по каждой детали рассчитывается как «фактическая выработка» / «норма выработки». Рабочему выплачивают зарплату как «базовое значение» * «коэффициент производительности». Коэффициент производительности – это среднее арифметическое всех коэффициентов выполнения норм по каждой детали.

У инженера есть подчинённые - несколько рабочих. Инженеру выплачивают зарплату как «базовое значение» * «коэффициент производительности». Для инженера коэффициент производительности – это среднее арифметическое коэффициентов производительности его подчиненных. Кроме того, ему выплачивают надбавку 10% за перевыполнение нормы подчинёнными.

У начальника цеха есть подчинённые - несколько инженеров. Зарплату ему выплачивают фиксированную «базовое значение» + «премия». Премия рассчитывается как 50% от «базового значения» если средний коэффициент производительности подчинённых был выше на 30% от нормы и 25% от

«базового значения» если средний коэффициент выше на 10%. Кроме того, содержит информацию о закреплённом за его цехом завхозе.

Завхозом может быть любой работник, к его зарплате доплачивается 60%. Также завхоз имеет информацию о нескольких начальников цехов, для которых он поставляет материалы со склада.

Каждый из типов людей должен иметь своё ФИО и возраст.

Вариант 13.

Сущности: юридическое лицо, страховая компания, автомобильный завод, налоговая инспекция, завод микроэлектроники, завод;

Действия: суммарный расчёт выплачиваемых налогов; общая сумма налоговых льгот; средний доход по каждому заводу; вывод информации по каждому субъекту включая список аффилированного с ним выпуска продукции, услуг либо кол-ва клиентов.

Описание сущностей:

Автомобильный завод и завод микроэлектроники выпускают продукцию. На каждый тип выпускаемой продукции установлена своя ставка налога, таким образом общий налог рассчитывается как сумма по всем типам продукций: «стоимость продукции» * «ставка налога на продукцию» * «кол-во выпущенной продукции».

Автомобильный завод также может оказывать несколько типов сервисных услуг, налог суммарно с которых составляет 12% с каждой оказанной услуги. Кроме того, автомобильный завод имеет информацию о своих поставщиках – других заводах.

Завод микроэлектроники может получать налоговую льготу, в зависимости от значимости направления для государства. Если завод высокой значимости – льгота составляет 35% от уплачиваемого налога, средней значимости - 15% от налога.

Страховая компания имеет список клиентов – заводов, каждый из которых характеризуется страховой премией выплаченной компании, и суммой страховых выплат (если 0 – страховых выплат не было). Налог вычисляется только с той суммы премий, которая была получена от клиентов, у которых не было страховых выплат. Налог составляет 25% если клиентов меньше 100, в другом случае 10%.

Каждое предприятие имеет название и год основания.

Вариант 14.

Сущности: республика, мир, государство, монархия, президентская республика, конфедерация;

Действия: суммарный расчёт количества людей, занятых в управлении в мире; расчёт минимального и максимального возраста управленца по каждому государству; средний возраст управленцев в каждом государстве; вывод информации по каждому субъекту включая список аффилированных с ним управленцев, с указанием их должностной характеристики.

Описание сущностей:

В республике управляет парламент, список состоит из парламентариев. Каждый парламентарий может быть обычным либо спикером. Также парламентарий может состоять в какой-либо партии либо быть беспартийным.

В президентской республике помимо парламента присутствует президент, а также кабинет министров. Министры могут быть главой какого-либо министерства, либо быть премьер-министром. Все министры являются парламентариями.

В монархии присутствует верховный монарх, а также список членов его семьи. У каждого члена семьи есть приоритет по престолонаследию.

Конфедерация – является по совокупностью разных государств (т.е. может состоять из нескольких государств со своими формами правления). Имеет палату представителей, для решения общих вопросов.

Каждый управленец имеет ФИО и возраст.

Вариант 15.

Сущности: деталь, механизм, изделие, узел, устройство, датчик.

Действия: расчёт стоимости и веса всего устройства; расчёт среднего количества деталей на каждое устройство; расчёт общего количества датчиков и их суммарную стоимость по каждому узлу; вывод информации по каждому объекту, включая его состав.

Описание сущностей:

Любое изделие имеет вес и стоимость. Стоимость складывается из стоимости материалов и работы по изготовлению.

Деталь является частью механизма. Механизм состоит из нескольких деталей. Также механизм содержит информацию о расчётном времени износа. Вес и стоимость механизма складывается из соответственно веса и стоимости его деталей. Может включать в себя несколько датчиков. Каждый датчик повышает стоимость устройства.

Узел состоит из нескольких механизмов. Вес, стоимость, расчётное время износа узла складывается из соответственно веса, стоимости и износа его механизмов. Кроме того, есть коэффициент сложности транспортировки: если вес превышает 20 кг. – стоимость повышается на 10%, если превышает 50 кг. – то на 20%.

Датчик имеет стоимость, вес не учитывается. Имеет тип замеряемой характеристики (температура, вибрация и т.д.)

Все объекты имеют названия.

Вариант 16.

Сущности: преподаватель, персона, университет, студент, завкафедрой, научный сотрудник.

Действия: суммарный расчёт выплачиваемых денежных средств; средняя зарплата по каждому типу сущностей; самый младший и самый старший человек по каждому типу сущностей; вывод информации по каждому человеку включая список аффилированных с ним предметов.

Описание сущностей:
Студент получает стипендию из расчёта «базовое значение» * «коэффициент успеваемости». Коэффициент рассчитывается исходя из оценок сданных предметов: все пятёрки – коэффициент 1.5, есть четвёрки – 1.0, есть тройки – 0.5, есть двойки – 0.0. Кроме того студент может быть практикантом у научного сотрудника.

Преподаватель получает зарплату в зависимости от кол-ва часов, которые он ведёт по каждому предмету. Рассчитывается зарплата как «базовое значение» * «коэффициент ставки» (суммарно 60 часов = коэффициент ставки 1.0).

Зав. кафедры – получает к зарплате надбавку к зарплате 60%. Кроме того, имеет нескольких преподавателей – заместителей.

Научный сотрудник получает зарплату исходя из часов, потраченных на исследование предметов. Каждый час работы научного сотрудника по

каждому предмету оплачивается отдельно как «базовое значение» * 0.05. Также имеет нескольких студентов – практикантов.

Зав. кафедры и научные сотрудники обязаны быть преподавателями.

Каждый из типов людей должен иметь своё ФИО и возраст.

Вариант 17.

Сущности: инженер, завод, персонал, рабочий, завхоз, начальник цеха.

Действия: суммарный расчёт выплачиваемых денежных средств; минимальная и максимальная зарплата по каждой должности; общее количество людей, занятых на каждом типе должности; вывод информации по каждому человеку включая список аффилированных с ним выпуска деталей.

Описание сущностей:

Рабочий создаёт разного типа деталей, для каждой детали указывается норма выработки и фактическая выработка. Коэффициент выполнения нормы по каждой детали рассчитывается как «фактическая выработка» / «норма выработки». Рабочему выплачивают зарплату как «базовое значение» * «коэффициент производительности». Коэффициент производительности – это среднее арифметическое всех коэффициентов выполнения норм по каждой детали.

У инженера есть подчинённые - несколько рабочих. Инженеру выплачивают зарплату как «базовое значение» * «коэффициент производительности». Для инженера коэффициент производительности – это среднее арифметическое коэффициентов производительности его подчиненных. Кроме того, ему выплачивают надбавку 10% за перевыполнение нормы подчинёнными.

У начальника цеха есть подчинённые - несколько инженеров. Зарплату ему выплачивают фиксированную «базовое значение» + «премия». Премия рассчитывается как 50% от «базового значения» если средний коэффициент производительности подчинённых был выше на 30% от нормы и 25% от «базового значения» если средний коэффициент выше на 10%. Кроме того, содержит информацию о закреплённом за его цехом завхозе.

Завхозом может быть любой работник, к его зарплате доплачивается 60%. Также завхоз имеет информацию о нескольких начальников цехов, для которых он поставляет материалы со склада.

Каждый из типов людей должен иметь своё ФИО и возраст.

Вариант 18.

Сущности: юридическое лицо, страховая компания, автомобильный завод, налоговая инспекция, завод микроэлектроники, завод;

Действия: суммарный расчёт выплачиваемых налогов; общая сумма налоговых льгот; средний доход по каждому заводу; вывод информации по каждому субъекту включая список аффилированного с ним выпуска продукции, услуг либо кол-ва клиентов.

Описание сущностей:

Автомобильный завод и завод микроэлектроники выпускают продукцию. На каждый тип выпускаемой продукции установлена своя ставка налога, таким образом общий налог рассчитывается как сумма по всем типам продукций: «стоимость продукции» * «ставка налога на продукцию» * «кол-во выпущенной продукции».

Автомобильный завод также может оказывать несколько типов сервисных услуг, налог суммарно с которых составляет 12% с каждой оказанной услуги. Кроме того, автомобильный завод имеет информацию о своих поставщиках – других заводах.

Завод микроэлектроники может получать налоговую льготу, в зависимости от значимости направления для государства. Если завод высокой значимости – льгота составляет 35% от уплачиваемого налога, средней значимости - 15% от налога.

Страховая компания имеет список клиентов – заводов, каждый из которых характеризуется страховой премией выплаченной компании, и суммой страховых выплат (если 0 – страховых выплат не было). Налог вычисляется только с той суммы премий, которая была получена от клиентов, у которых не было страховых выплат. Налог составляет 25% если клиентов меньше 100, в другом случае 10%.

Каждое предприятие имеет название и год основания.

Вариант 19.

Сущности: республика, мир, государство, монархия, президентская республика, конфедерация;

Действия: суммарный расчёт количества людей, занятых в управлении в мире; расчёт минимального и максимального возраста управленца по каждому государству; средний возраст управленцев в каждом государстве;

вывод информации по каждому субъекту включая список аффилированных с ним управленцев, с указанием их должностной характеристики.

Описание сущностей:

В республике управляет парламент, список состоит из парламентариев. Каждый парламентарий может быть обычным либо спикером. Также парламентарий может состоять в какой-либо партии либо быть беспартийным.

В президентской республике помимо парламента присутствует президент, а также кабинет министров. Министры могут быть главой какого-либо министерства, либо быть премьер-министром. Все министры являются парламентариями.

В монархии присутствует верховный монарх, а также список членов его семьи. У каждого члена семьи есть приоритет по престолонаследию.

Конфедерация – является по совокупностью разных государств (т.е. может состоять из нескольких государств со своими формами правления). Имеет палату представителей, для решения общих вопросов.

Каждый управленец имеет ФИО и возраст.

Вариант 20.

Сущности: деталь, механизм, изделие, узел, устройство, датчик.

Действия: расчёт стоимости и веса всего устройства; расчёт среднего количества деталей на каждое устройство; расчёт общего количества датчиков и их суммарную стоимость по каждому узлу; вывод информации по каждому объекту, включая его состав.

Описание сущностей:

Любое изделие имеет вес и стоимость. Стоимость складывается из стоимости материалов и работы по изготовлению.

Деталь является частью механизма. Механизм состоит из нескольких деталей. Также механизм содержит информацию о расчётном времени износа. Вес и стоимость механизма складывается из соответственно веса и стоимости его деталей. Может включать в себя несколько датчиков. Каждый датчик повышает стоимость устройства.

Узел состоит из нескольких механизмов. Вес, стоимость, расчётное время износа узла складывается из соответственно веса, стоимости и износа его механизмов. Кроме того, есть коэффициент сложности транспортировки:

если вес превышает 20 кг. – стоимость повышается на 10%, если превышает 50 кг. – то на 20%.

Датчик имеет стоимость, вес не учитывается. Имеет тип замеряемой характеристики (температура, вибрация и т.д.)

Все объекты имеют названия.

Вариант 21.

Сущности: преподаватель, персона, университет, студент, завкафедрой, научный сотрудник.

Действия: суммарный расчёт выплачиваемых денежных средств; средняя зарплата по каждому типу сущностей; самый младший и самый старший человек по каждому типу сущностей; вывод информации по каждому человеку включая список аффилированных с ним предметов.

Описание сущностей:
Студент получает стипендию из расчёта «базовое значение» * «коэффициент успеваемости». Коэффициент рассчитывается исходя из оценок сданных предметов: все пятёрки – коэффициент 1.5, есть четвёрки – 1.0, есть тройки – 0.5, есть двойки – 0.0. Кроме того студент может быть практикантом у научного сотрудника.

Преподаватель получает зарплату в зависимости от кол-ва часов, которые он ведёт по каждому предмету. Рассчитывается зарплата как «базовое значение» * «коэффициент ставки» (суммарно 60 часов = коэффициент ставки 1.0).

Зав. кафедры – получает к зарплате надбавку к зарплате 60%. Кроме того, имеет нескольких преподавателей – заместителей.

Научный сотрудник получает зарплату исходя из часов, потраченных на исследование предметов. Каждый час работы научного сотрудника по каждому предмету оплачивается отдельно как «базовое значение» * 0.05. Также имеет нескольких студентов – практикантов.

Зав. кафедры и научные сотрудники обязаны быть преподавателями.

Каждый из типов людей должен иметь своё ФИО и возраст.

Вариант 22.

Сущности: инженер, завод, персонал, рабочий, завхоз, начальник цеха.

Действия: суммарный расчёт выплачиваемых денежных средств; минимальная и максимальная зарплата по каждой должности; общее количество людей, занятых на каждом типе должности; вывод информации

по каждому человеку включая список аффилированных с ним выпуска деталей.

Описание сущностей:

Рабочий создаёт разного типа деталей, для каждой детали указывается норма выработки и фактическая выработка. Коэффициент выполнения нормы по каждой детали рассчитывается как «фактическая выработка» / «норма выработки». Рабочему выплачивают зарплату как «базовое значение» * «коэффициент производительности». Коэффициент производительности – это среднее арифметическое всех коэффициентов выполнения норм по каждой детали.

У инженера есть подчинённые - несколько рабочих. Инженеру выплачивают зарплату как «базовое значение» * «коэффициент производительности». Для инженера коэффициент производительности – это среднее арифметическое коэффициентов производительности его подчиненных. Кроме того, ему выплачивают надбавку 10% за перевыполнение нормы подчинёнными.

У начальника цеха есть подчинённые - несколько инженеров. Зарплату ему выплачивают фиксированную «базовое значение» + «премия». Премия рассчитывается как 50% от «базового значения» если средний коэффициент производительности подчинённых был выше на 30% от нормы и 25% от «базового значения» если средний коэффициент выше на 10%. Кроме того, содержит информацию о закреплённом за его цехом завхозе.

Завхозом может быть любой работник, к его зарплате доплачивается 60%. Также завхоз имеет информацию о нескольких начальников цехов, для которых он поставляет материалы со склада.

Каждый из типов людей должен иметь своё ФИО и возраст.

Вариант 23.

Сущности: юридическое лицо, страховая компания, автомобильный завод, налоговая инспекция, завод микроэлектроники, завод;

Действия: суммарный расчёт выплачиваемых налогов; общая сумма налоговых льгот; средний доход по каждому заводу; вывод информации по каждому субъекту включая список аффилированного с ним выпуска продукции, услуг либо кол-ва клиентов.

Описание сущностей:

Автомобильный завод и завод микроэлектроники выпускают продукцию. На каждый тип выпускаемой продукции установлена своя ставка налога, таким

образом общий налог рассчитывается как сумма по всем типам продукций: «стоимость продукции» * «ставка налога на продукцию» * «кол-во выпущенной продукции».

Автомобильный завод также может оказывать несколько типов сервисных услуг, налог суммарно с которых составляет 12% с каждой оказанной услуги. Кроме того, автомобильный завод имеет информацию о своих поставщиках – других заводах.

Завод микроэлектроники может получать налоговую льготу, в зависимости от значимости направления для государства. Если завод высокой значимости – льгота составляет 35% от уплачиваемого налога, средней значимости - 15% от налога.

Страховая компания имеет список клиентов – заводов, каждый из которых характеризуется страховой премией выплаченной компании, и суммой страховых выплат (если 0 – страховых выплат не было). Налог вычисляется только с той суммы премий, которая была получена от клиентов, у которых не было страховых выплат. Налог составляет 25% если клиентов меньше 100, в другом случае 10%.

Каждое предприятие имеет название и год основания.

Вариант 24.

Сущности: республика, мир, государство, монархия, президентская республика, конфедерация;

Действия: суммарный расчёт количества людей, занятых в управлении в мире; расчёт минимального и максимального возраста управленца по каждому государству; средний возраст управленцев в каждом государстве; вывод информации по каждому субъекту включая список аффилированных с ним управленцев, с указанием их должностной характеристики.

Описание сущностей:

В республике управляет парламент, список состоит из парламентариев. Каждый парламентарий может быть обычным либо спикером. Также парламентарий может состоять в какой-либо партии либо быть беспартийным.

В президентской республике помимо парламента присутствует президент, а также кабинет министров. Министры могут быть главой какого-либо министерства, либо быть премьер-министром. Все министры являются парламентариями.

В монархии присутствует верховный монарх, а также список членов его семьи. У каждого члена семьи есть приоритет по престолонаследию.

Конфедерация – является по совокупностью разных государств (т.е. может состоять из нескольких государств со своими формами правления). Имеет палату представителей, для решения общих вопросов.

Каждый управленец имеет ФИО и возраст.

Вариант 25.

Сущности: деталь, механизм, изделие, узел, устройство, датчик.

Действия: расчёт стоимости и веса всего устройства; расчёт среднего количества деталей на каждое устройство; расчёт общего количества датчиков и их суммарную стоимость по каждому узлу; вывод информации по каждому объекту, включая его состав.

Описание сущностей:

Любое изделие имеет вес и стоимость. Стоимость складывается из стоимости материалов и работы по изготовлению.

Деталь является частью механизма. Механизм состоит из нескольких деталей. Также механизм содержит информацию о расчётном времени износа. Вес и стоимость механизма складывается из соответственно веса и стоимости его деталей. Может включать в себя несколько датчиков. Каждый датчик повышает стоимость устройства.

Узел состоит из нескольких механизмов. Вес, стоимость, расчётное время износа узла складывается из соответственно веса, стоимости и износа его механизмов. Кроме того, есть коэффициент сложности транспортировки: если вес превышает 20 кг. – стоимость повышается на 10%, если превышает 50 кг. – то на 20%.

Датчик имеет стоимость, вес не учитывается. Имеет тип замеряемой характеристики (температура, вибрация и т.д.)

Все объекты имеют названия.

Лабораторная работа №5
Шаблонные классы

В данной лабораторной работе необходимо разработать шаблонный класс в соответствии с вариантом задания. Любой класс должен иметь конструктор

копии и перегруженный оператор =. Также класс должен содержать перегрузку оператора << для класса iostream, чтобы его можно было выводить в консоль через стандартный поток вывода cout с указанием количества элементов и их значений. В главной функции main обеспечить консольный интерфейс для тестирование всех функций шаблонного класса с типами **int, float, char*, struct Vec2 {float x; float y;}** (аналогично лаб 1)

Варианты

1. **Динамический массив**, использующий избыточное резервирование памяти под элементы. Увеличивает ёмкость на количество элементов, заданных в non-type параметре.
   Необходимые методы:
   reserve(n) – зарезервировать ещё n элементов
   getCapacity() – получить текущую ёмкость
   getLength() – получить текущую длину
   insert(elem,n) – добавить элемент на позицию n
   remove(elem,n) – удалить элемент на позиции n

   Перегрузить следующие операции:
   + – добавить элемент в конец массива
   - – удалить первое вхождение элемента в массиве (для типа char* должна быть своя специализация шаблона)
   [] – доступ к элементу по индексу
   == и != – проверка на идентичность и не идентичность массивов (для типа char* должна быть своя специализация шаблона)

2. **Динамически-инициализированный массив,** количество элементов задаётся через non-type параметр, выделение памяти происходит в конструкторе при создании объекта.
   Необходимые методы:
   sort(bAscending) – сортировать элементы по возрастанию (**bAscending=true**), либо по убыванию. Для типа char* должна быть своя специализация шаблона, сортирующая по длине строк.
   getMaxSize() – получить максимальный размер массива
   getLength() – получить текущее кол-во элементов
   insert(elem,n) – добавить элемент на позицию n
   remove(elem,n) – удалить элемент на позиции n

   Перегрузить следующие операции:
   + – добавить элемент в конец массива
   - – удалить первое вхождение элемента в массиве (для типа char* должна быть своя специализация шаблона)

[] – доступ к элементу по индексу
< и > – проверка, что в двух одинакового размера массивов все элементы больше или меньше элементов другого массива (для типа char* и Vec2 должна быть своя специализация шаблона: char* исходя из длины строки, для Vec2 – исходя из длины вектора).

3. **Множество**, динамическая структура данных, где через non-type параметр типа enum задаётся будет ли использоваться мультимножество (множество с повторениями элементов), либо простое множество (все элементы уникальны).
   Необходимые методы:
   getLength() – получить текущее кол-во элементов
   getSizeInBytes() – получить текущий размер всей структуры в памяти в байтах (для типа char* должна быть своя специализация шаблона исходя из длины строки)
   getRepetitions(elem) - получить текущее кол-во повторений элемента для мультимножества (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)

   Перегрузить следующие операции:
   + – добавить элемент в множество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
   - – удалить элемент из множества (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
   [] – проверить вхождение элемента в множество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
   > и < – проверить на подмножество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).

4. **Множество**, динамическая структура данных, где через non-type параметр задаётся максимально допустимый размер множества (больше этого кол-ва нельзя увеличить множество).
   Необходимые методы:
   getLength() – получить текущее кол-во элементов
   isExist(elem) – получить вхождение элемента в множество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
   unite(set) – объединить с другим множеством.

   Перегрузить следующие операции:
   + – добавить элемент в множество
   - – удалить элемент из множества (для типа char* должна быть своя

специализация шаблона исходя из содержимого строки)
***** – получить пересечение двух множеств (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).
== и != – проверка множеств на тождественность \ не тождественность (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).

5. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимая характеристика добавляемого элемента (для типов int и float это модуль значения, для char* - длина строки, для Vec2 – длина вектора). Т.е. элементы с характеристикой, превышающей заданное значение нельзя добавить в список.
   Необходимые методы:
   getLength() – получить текущее кол-во элементов
   isExist(elem) – получить вхождение элемента в список (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
   insert(elem,n) – добавить элемент на позицию n
   remove(elem,n) – удалить элемент на позиции n

   Перегрузить следующие операции:
   + – добавить элемент в начало списка.
   - – удалить все вхождения элемента из списка (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
   -- – удалить первый элемент из начала списка.
   [] – доступ к элементу на заданной позиции.

6. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимое кол-во элементов в списке.
   Необходимые методы:
   getLength() – получить текущее кол-во элементов
   insert(elem,n) – добавить элемент на позицию n
   remove(elem,n) – удалить элемент на позиции n

   Перегрузить следующие операции:
   + – добавить элемент в конец списка.
   + – добавить все элементы из другого списка.
   - – удалить первое и последнее вхождение элемента из списка (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)

[] – доступ к элементу на заданной позиции.
== и != – проверка списков на идентичность и не идентичность (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).

7. **Динамический массив**, использующий избыточное резервирование памяти под элементы. Увеличивает ёмкость на количество элементов, заданных в non-type параметре.
   Необходимые методы:
   reserve(n) – зарезервировать ещё n элементов
   getCapacity() – получить текущую ёмкость
   getLength() – получить текущую длину
   insert(elem,n) – добавить элемент на позицию n
   remove(elem,n) – удалить элемент на позиции n

   Перегрузить следующие операции:
   + – добавить элемент в конец массива
   - – удалить первое вхождение элемента в массиве (для типа char* должна быть своя специализация шаблона)
   [] – доступ к элементу по индексу
   == и != – проверка на идентичность и не идентичность массивов (для типа char* должна быть своя специализация шаблона)

8. **Динамически-инициализированный массив,** количество элементов задаётся через non-type параметр, выделение памяти происходит в конструкторе при создании объекта.
   Необходимые методы:
   sort(bAscending) – сортировать элементы по возрастанию (**bAscending=true**), либо по убыванию. Для типа char* должна быть своя специализация шаблона, сортирующая по длине строк.
   getMaxSize() – получить максимальный размер массива
   getLength() – получить текущее кол-во элементов
   insert(elem,n) – добавить элемент на позицию n
   remove(elem,n) – удалить элемент на позиции n

   Перегрузить следующие операции:
   + – добавить элемент в конец массива
   - – удалить первое вхождение элемента в массиве (для типа char* должна быть своя специализация шаблона)
   [] – доступ к элементу по индексу
   < и > – проверка, что в двух одинакового размера массивов все элементы больше или меньше элементов другого массива (для типа

char* и Vec2 должна быть своя специализация шаблона: char* исходя из длины строки, для Vec2 – исходя из длины вектора).

9. **Множество**, динамическая структура данных, где через non-type параметр типа enum задаётся будет ли использоваться мультимножество (множество с повторениями элементов), либо простое множество (все элементы уникальны).
   Необходимые методы:
   getLength() – получить текущее кол-во элементов
   getSizeInBytes() – получить текущий размер всей структуры в памяти в байтах (для типа char* должна быть своя специализация шаблона исходя из длины строки)
   getRepetitions(elem) - получить текущее кол-во повторений элемента для мультимножества (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)

   Перегрузить следующие операции:
   + – добавить элемент в множество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
   - – удалить элемент из множества (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
   [] – проверить вхождение элемента в множество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
   > и < – проверить на подмножество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).

10. **Множество**, динамическая структура данных, где через non-type параметр задаётся максимально допустимый размер множества (больше этого кол-ва нельзя увеличить множество).
    Необходимые методы:
    getLength() – получить текущее кол-во элементов
    isExist(elem) – получить вхождение элемента в множество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
    unite(set) – объединить с другим множеством.

    Перегрузить следующие операции:
    + – добавить элемент в множество
    - – удалить элемент из множества (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
    ***** – получить пересечение двух множеств (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).

== и != – проверка множеств на тождественность \ не тождественность (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).

11. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимая характеристика добавляемого элемента (для типов int и float это модуль значения, для char* - длина строки, для Vec2 – длина вектора). Т.е. элементы с характеристикой, превышающей заданное значение нельзя добавить в список.
    Необходимые методы:
    getLength() – получить текущее кол-во элементов
    isExist(elem) – получить вхождение элемента в список (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
    insert(elem,n) – добавить элемент на позицию n
    remove(elem,n) – удалить элемент на позиции n

    Перегрузить следующие операции:
    + – добавить элемент в начало списка.
    - – удалить все вхождения элемента из списка (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
    -- – удалить первый элемент из начала списка.
    [] – доступ к элементу на заданной позиции.

12. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимое кол-во элементов в списке.
    Необходимые методы:
    getLength() – получить текущее кол-во элементов
    insert(elem,n) – добавить элемент на позицию n
    remove(elem,n) – удалить элемент на позиции n

    Перегрузить следующие операции:
    + – добавить элемент в конец списка.
    + – добавить все элементы из другого списка.
    - – удалить первое и последнее вхождение элемента из списка (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
    [] – доступ к элементу на заданной позиции.
    == и != – проверка списков на идентичность и не идентичность (для типа char* должна быть своя специализация шаблона исходя из

содержимого строки).

13. **Динамический массив**, использующий избыточное резервирование памяти под элементы. Увеличивает ёмкость на количество элементов, заданных в non-type параметре.
    Необходимые методы:
    reserve(n) – зарезервировать ещё n элементов
    getCapacity() – получить текущую ёмкость
    getLength() – получить текущую длину
    insert(elem,n) – добавить элемент на позицию n
    remove(elem,n) – удалить элемент на позиции n

    Перегрузить следующие операции:
    + – добавить элемент в конец массива
    - – удалить первое вхождение элемента в массиве (для типа char* должна быть своя специализация шаблона)
    [] – доступ к элементу по индексу
    == и != – проверка на идентичность и не идентичность массивов (для типа char* должна быть своя специализация шаблона)

14. **Динамически-инициализированный массив,** количество элементов задаётся через non-type параметр, выделение памяти происходит в конструкторе при создании объекта.
    Необходимые методы:
    sort(bAscending) – сортировать элементы по возрастанию (**bAscending=true**), либо по убыванию. Для типа char* должна быть своя специализация шаблона, сортирующая по длине строк.
    getMaxSize() – получить максимальный размер массива
    getLength() – получить текущее кол-во элементов
    insert(elem,n) – добавить элемент на позицию n
    remove(elem,n) – удалить элемент на позиции n

    Перегрузить следующие операции:
    + – добавить элемент в конец массива
    - – удалить первое вхождение элемента в массиве (для типа char* должна быть своя специализация шаблона)
    [] – доступ к элементу по индексу
    < и > – проверка, что в двух одинакового размера массивов все элементы больше или меньше элементов другого массива (для типа char* и Vec2 должна быть своя специализация шаблона: char* исходя из длины строки, для Vec2 – исходя из длины вектора).

15. **Множество**, динамическая структура данных, где через non-type параметр типа enum задаётся будет ли использоваться мультимножество (множество с повторениями элементов), либо простое множество (все элементы уникальны).
Необходимые методы:
getLength() – получить текущее кол-во элементов
getSizeInBytes() – получить текущий размер всей структуры в памяти в байтах (для типа char* должна быть своя специализация шаблона исходя из длины строки)
getRepetitions(elem) - получить текущее кол-во повторений элемента для мультимножества (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)

Перегрузить следующие операции:
+ – добавить элемент в множество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
- – удалить элемент из множества (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
[] – проверить вхождение элемента в множество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
> и < – проверить на подмножество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).

16. **Множество**, динамическая структура данных, где через non-type параметр задаётся максимально допустимый размер множества (больше этого кол-ва нельзя увеличить множество).
Необходимые методы:
getLength() – получить текущее кол-во элементов
isExist(elem) – получить вхождение элемента в множество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
unite(set) – объединить с другим множеством.

Перегрузить следующие операции:
+ – добавить элемент в множество
- – удалить элемент из множества (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
***** – получить пересечение двух множеств (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).
== и != – проверка множеств на тождественность \ не тождественность (для типа char* должна быть своя специализация шаблона исходя из

содержимого строки).

17. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимая характеристика добавляемого элемента (для типов int и float это модуль значения, для char* - длина строки, для Vec2 – длина вектора). Т.е. элементы с характеристикой, превышающей заданное значение нельзя добавить в список.
    Необходимые методы:
    getLength() – получить текущее кол-во элементов
    isExist(elem) – получить вхождение элемента в список (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
    insert(elem,n) – добавить элемент на позицию n
    remove(elem,n) – удалить элемент на позиции n

    Перегрузить следующие операции:
    + – добавить элемент в начало списка.
    - – удалить все вхождения элемента из списка (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
    -- – удалить первый элемент из начала списка.
    [] – доступ к элементу на заданной позиции.

18. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимое кол-во элементов в списке.
    Необходимые методы:
    getLength() – получить текущее кол-во элементов
    insert(elem,n) – добавить элемент на позицию n
    remove(elem,n) – удалить элемент на позиции n

    Перегрузить следующие операции:
    + – добавить элемент в конец списка.
    + – добавить все элементы из другого списка.
    - – удалить первое и последнее вхождение элемента из списка (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
    [] – доступ к элементу на заданной позиции.
    == и != – проверка списков на идентичность и не идентичность (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).

19. **Динамический массив**, использующий избыточное резервирование памяти под элементы. Увеличивает ёмкость на количество элементов, заданных в non-type параметре.
    Необходимые методы:
    reserve(n) – зарезервировать ещё n элементов
    getCapacity() – получить текущую ёмкость
    getLength() – получить текущую длину
    insert(elem,n) – добавить элемент на позицию n
    remove(elem,n) – удалить элемент на позиции n

    Перегрузить следующие операции:
    + – добавить элемент в конец массива
    - – удалить первое вхождение элемента в массиве (для типа char* должна быть своя специализация шаблона)
    [] – доступ к элементу по индексу
    == и != – проверка на идентичность и не идентичность массивов (для типа char* должна быть своя специализация шаблона)

20. **Динамически-инициализированный массив,** количество элементов задаётся через non-type параметр, выделение памяти происходит в конструкторе при создании объекта.
    Необходимые методы:
    sort(bAscending) – сортировать элементы по возрастанию (**bAscending=true**), либо по убыванию. Для типа char* должна быть своя специализация шаблона, сортирующая по длине строк.
    getMaxSize() – получить максимальный размер массива
    getLength() – получить текущее кол-во элементов
    insert(elem,n) – добавить элемент на позицию n
    remove(elem,n) – удалить элемент на позиции n

    Перегрузить следующие операции:
    + – добавить элемент в конец массива
    - – удалить первое вхождение элемента в массиве (для типа char* должна быть своя специализация шаблона)
    [] – доступ к элементу по индексу
    < и > – проверка, что в двух одинакового размера массивов все элементы больше или меньше элементов другого массива (для типа char* и Vec2 должна быть своя специализация шаблона: char* исходя из длины строки, для Vec2 – исходя из длины вектора).

21. **Множество**, динамическая структура данных, где через non-type параметр типа enum задаётся будет ли использоваться

мультимножество (множество с повторениями элементов), либо простое множество (все элементы уникальны).
Необходимые методы:
getLength() – получить текущее кол-во элементов
getSizeInBytes() – получить текущий размер всей структуры в памяти в байтах (для типа char* должна быть своя специализация шаблона исходя из длины строки)
getRepetitions(elem) - получить текущее кол-во повторений элемента для мультимножества (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)

Перегрузить следующие операции:
+ – добавить элемент в множество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
- – удалить элемент из множества (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
[] – проверить вхождение элемента в множество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
> и < – проверить на подмножество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).

22. **Множество**, динамическая структура данных, где через non-type параметр задаётся максимально допустимый размер множества (больше этого кол-ва нельзя увеличить множество).
Необходимые методы:
getLength() – получить текущее кол-во элементов
isExist(elem) – получить вхождение элемента в множество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
unite(set) – объединить с другим множеством.

Перегрузить следующие операции:
+ – добавить элемент в множество
- – удалить элемент из множества (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
***** – получить пересечение двух множеств (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).
== и != – проверка множеств на тождественность \ не тождественность (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).

23. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимая характеристика добавляемого элемента (для типов int и float это модуль значения, для char* - длина строки, для Vec2 – длина вектора). Т.е. элементы с характеристикой, превышающей заданное значение нельзя добавить в список.
    Необходимые методы:
    getLength() – получить текущее кол-во элементов
    isExist(elem) – получить вхождение элемента в список (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
    insert(elem,n) – добавить элемент на позицию n
    remove(elem,n) – удалить элемент на позиции n

    Перегрузить следующие операции:
    + – добавить элемент в начало списка.
    - – удалить все вхождения элемента из списка (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
    -- – удалить первый элемент из начала списка.
    [] – доступ к элементу на заданной позиции.

24. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимое кол-во элементов в списке.
    Необходимые методы:
    getLength() – получить текущее кол-во элементов
    insert(elem,n) – добавить элемент на позицию n
    remove(elem,n) – удалить элемент на позиции n

    Перегрузить следующие операции:
    + – добавить элемент в конец списка.
    + – добавить все элементы из другого списка.
    - – удалить первое и последнее вхождение элемента из списка (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
    [] – доступ к элементу на заданной позиции.
    == и != – проверка списков на идентичность и не идентичность (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).

25. **Динамический массив**, использующий избыточное резервирование памяти под элементы. Увеличивает ёмкость на количество элементов,

заданных в non-type параметре.
Необходимые методы:
reserve(n) – зарезервировать ещё n элементов
getCapacity() – получить текущую ёмкость
getLength() – получить текущую длину
insert(elem,n) – добавить элемент на позицию n
remove(elem,n) – удалить элемент на позиции n

Перегрузить следующие операции:
+ – добавить элемент в конец массива
- – удалить первое вхождение элемента в массиве (для типа char* должна быть своя специализация шаблона)
[] – доступ к элементу по индексу
== и != – проверка на идентичность и не идентичность массивов (для типа char* должна быть своя специализация шаблона)

26. **Динамически-инициализированный массив,** количество элементов задаётся через non-type параметр, выделение памяти происходит в конструкторе при создании объекта.
Необходимые методы:
sort(bAscending) – сортировать элементы по возрастанию (**bAscending=true**), либо по убыванию. Для типа char* должна быть своя специализация шаблона, сортирующая по длине строк.
getMaxSize() – получить максимальный размер массива
getLength() – получить текущее кол-во элементов
insert(elem,n) – добавить элемент на позицию n
remove(elem,n) – удалить элемент на позиции n

Перегрузить следующие операции:
+ – добавить элемент в конец массива
- – удалить первое вхождение элемента в массиве (для типа char* должна быть своя специализация шаблона)
[] – доступ к элементу по индексу
< и > – проверка, что в двух одинакового размера массивов все элементы больше или меньше элементов другого массива (для типа char* и Vec2 должна быть своя специализация шаблона: char* исходя из длины строки, для Vec2 – исходя из длины вектора).

27. **Множество**, динамическая структура данных, где через non-type параметр типа enum задаётся будет ли использоваться мультимножество (множество с повторениями элементов), либо простое множество (все элементы уникальны).

Необходимые методы:
getLength() – получить текущее кол-во элементов
getSizeInBytes() – получить текущий размер всей структуры в памяти в байтах (для типа char* должна быть своя специализация шаблона исходя из длины строки)
getRepetitions(elem) - получить текущее кол-во повторений элемента для мультимножества (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)

Перегрузить следующие операции:
+ – добавить элемент в множество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
- – удалить элемент из множества (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
[] – проверить вхождение элемента в множество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
> и < – проверить на подмножество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).

28. **Множество**, динамическая структура данных, где через non-type параметр задаётся максимально допустимый размер множества (больше этого кол-ва нельзя увеличить множество).
Необходимые методы:
getLength() – получить текущее кол-во элементов
isExist(elem) – получить вхождение элемента в множество (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
unite(set) – объединить с другим множеством.

Перегрузить следующие операции:
+ – добавить элемент в множество
- – удалить элемент из множества (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
***** – получить пересечение двух множеств (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).
== и != – проверка множеств на тождественность \ не тождественность (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).

29. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимая характеристика добавляемого элемента (для типов int и float это модуль значения, для

char* - длина строки, для Vec2 – длина вектора). Т.е. элементы с характеристикой, превышающей заданное значение нельзя добавить в список.
Необходимые методы:
getLength() – получить текущее кол-во элементов
isExist(elem) – получить вхождение элемента в список (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
insert(elem,n) – добавить элемент на позицию n
remove(elem,n) – удалить элемент на позиции n

Перегрузить следующие операции:
+ – добавить элемент в начало списка.
- – удалить все вхождения элемента из списка (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
-- – удалить первый элемент из начала списка.
[] – доступ к элементу на заданной позиции.

30. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимое кол-во элементов в списке.
Необходимые методы:
getLength() – получить текущее кол-во элементов
insert(elem,n) – добавить элемент на позицию n
remove(elem,n) – удалить элемент на позиции n

Перегрузить следующие операции:
+ – добавить элемент в конец списка.
+ – добавить все элементы из другого списка.
- – удалить первое и последнее вхождение элемента из списка (для типа char* должна быть своя специализация шаблона исходя из содержимого строки)
[] – доступ к элементу на заданной позиции.
== и != – проверка списков на идентичность и не идентичность (для типа char* должна быть своя специализация шаблона исходя из содержимого строки).

Библиографический список

1. Каширин И.Ю., Новичков В.С. От С к С++ Горячая линия , Телеком 2005 г. 324с.
2. Лаврентьев С.И., Наумов Д.А. : Объектно-ориентированное программирование на С++, Методические указания к курсовому проектированию, РГРТА, Рязань, 2003г. 48 с.
3. Программирование на языке СИ в системе Турбо-Си: Методические указания к лабораторным работам / Рязан. гос. радиотехн. акад.; Сост. В.С. Богданов, Р.М. Ганеев, М.А. Зубов, С.И. Лаврентьев, Б.В. Никичкин; Рязань, 1999. 72 с. № 2806 .
4. Программирование на языке СИ в системе Турбо-Си: Методические указания к лабораторным работам / Рязан. гос. радиотехн. акад.; Сост. В.С. Богданов, Р.М. Ганеев, М.А. Зубов, Л.В. Маликова, С.И. Лаврентьев, Б.В. Никичкин, В.И. Никитин; Рязань, 2001. 56 с. № 3192.
5. Турбо-Си.: Методические указания к лабораторным работам № 1 – 4/ Рязан. радиотехн. ин-т; Под ред. В.С. Новичкова. Рязань, 1992. –64 с. № 2003.
6. Турбо-Си. Производные типы данных: Методические указания к лабораторным работам № 5 – 8/ Рязан. радиотехн. ин-т; Под ред. И.Ю. Каширина. Рязань,1993. –60 с. № 2152 .
7. Программирование в среде VISUAL C++: Методические указания к лабораторным работам / Рязан. гос. радиотехн. унив.; С.И. Лаврентьев, В.К. Столчнев. Рязань, 2008, 40 с. //FS/Work/Docs/ МО дисциплин кафедры/ информатика.
8. Основы программирования в С++ Builder. С.И. Лаврентьев, Б.В. Никичкин; РГРТУ Рязань, 2007. 48 с.
9. Справочник по языку С++, msdn.microsoft.com, дата просмотра 01.08.2018.

Составил: к.т.н., доцент кафедры «Вычислительная и прикладная математика» Антипов О.В.

Заведующий кафедрой вычислительной и прикладной математики, д-р техн. наук, профессор Овечкин Г.В.