

Темы индивидуальных заданий (курсовых проектов) по дисциплине «Объектно-ориентированное программирование»

1.	Статистические расчеты	2
2.	Вектор.....	2
3.	Динамический массив.....	2
4.	Разреженный массив	2
5.	Ассоциативный контейнер	3
6.	Матрица.....	3
7.	Разреженная матрица	3
8.	Полином	4
9.	Комплексное число	4
10.	Неограниченные числа	4
11.	Точные вычисления	5
12.	Рациональная дробь	5
13.	Граф	5
14.	Дерево.....	5
15.	Множество	6
16.	Системы счисления	6
17.	Генератор случайных чисел	6
18.	Файловый драйвер	7
19.	Алгоритмы сортировки.....	7
20.	Алгоритмы шифрования данных.....	7
21.	Графические примитивы	8
22.	Поле игры «Морской бой»	8
23.	Поле игры «Шашки».....	8
24.	Поле игры «Реверси»	8
25.	Поле игры «2048».....	9
26.	Кроссворд.....	9
27.	Поле игры «Сапер»	9
28.	Игра «Змейка»	10
29.	Поле игры «Охота на лис»	10
30.	Поле игры «Лабиринт»	11

При выполнении курсового проекта необходимо создать класс (систему классов), который реализует некоторую математическую абстракцию или определенные проблемно-ориентированные функции в соответствии с заданием. Для проверки адекватности реализованных функций необходимо также создать приложение, демонстрирующее возможности созданного класса (системы классов).

Приведенные темы курсовых проектов являются ориентировочными, задания на разработку должны быть уточнены и окончательно сформированы в процессе анализа задачи. Также возможно формирование заданий в соответствии с предложениями студентов по другим тематикам, имеющим практическую значимость или при наличии обоснованной заинтересованности студента в разработке. В этом случае целью проекта может являться разработка класса, системы классов или приложения пользователя с использованием объектно-ориентированного подхода.

Выполнение курсового проекта и представление результатов разработки регламентируется методическим пособием «Курсовое проектирование. Разработка программного обеспечения».

1. Статистические расчеты

Класс статистических расчетов должен позволять выполнять над выборкой данных ряд операций:

- добавление, в том числе путем объединения двух выборок, и удаление элементов выборки, доступ к ним как к элементам массива, а также очистку выборки;
- вычисление суммы и суммы квадратов значений элементов выборки;
- отыскание значений математического ожидания, дисперсии, среднеквадратического отклонения выборки и т.п.;
- отыскание значений гистограммы, представленных в виде объекта этого же класса (выборки).

2. Вектор

Класс вектора должен реализовывать различные операции векторной арифметики в n -мерном пространстве:

- присваивание и сравнение векторов, доступ к отдельным элементам вектора как к элементам массива и к количеству измерений вектора;
- арифметические операции между двумя векторами, вектором и скаляром;
- вычисление модуля вектора, длин и углов проекций вектора на плоскость, заданную номерами двух измерений;
- определение ортогональности, коллинеарности и принадлежности между двумя векторами, определение ортогональности и принадлежности вектора плоскости;
- преобразование вектора в строку и обратно, используя форму записи $\{ a_1, \dots a_n \}$, где a_i – числовое значение i -го элемента вектора.

3. Динамический массив

Класс динамического массива вещественных чисел должен реализовывать необходимые операции управления памятью и данными:

- задание размера массива и его изменение (в сторону увеличения и уменьшения, с сохранением значений «старых» элементов);
- заполнение всех элементов массива заданным значением, значениями из обычного массива и обратно;
- доступ к отдельным элементам массива по индексу (начиная с 0);
- присваивание, конкатенация и поэлементное сравнение массивов;
- поэлементные арифметические операции между двумя динамическими массивами;
- вставка и удаление элементов со сдвигом последующих элементов;
- выделение части элементов, заданных начальным и конечным номерами, в виде подмассива.

4. Разреженный массив

Класс разреженного массива вещественных чисел ориентирован на эффективное использование памяти для представления массивов, в которых значительное количество элементов имеют нулевые значения, и должен обеспечивать выполнение основных операций над массивами и их элементами:

- присваивание и поэлементное сравнение разреженных массивов, заполнение разреженного массива значениями из обычного массива и наоборот;
- доступ к отдельным элементам разреженного массива, получение номеров следующего и предыдущего ненулевых элементов для заданного номера;
- поэлементные арифметические операции между двумя разреженными массивами;
- выделение части элементов, заданных начальным и конечным номерами, в виде подмассива, очистка (заполнение нулями) части элементов массива;
- вставка и удаление элементов со сдвигом последующих элементов;
- отыскание минимального и максимального номеров элементов, содержащих ненулевые значения, подсчет количества и суммы ненулевых значений в массиве.

5. Ассоциативный контейнер

Класс ассоциативного контейнера предназначен для хранения данных некоторого типа совместно с текстовыми ключами для их поиска и должен обеспечивать выполнение следующих операций:

- присваивание содержимого контейнеров вместе с ключами;
- добавление в контейнер элемента и его ключа, удаление элементов по ключу;
- доступ к отдельным элементам по порядковому номеру и по ключу;
- определение количества элементов, содержащихся в контейнере;
- сортировка элементов в контейнере по возрастанию и убыванию значений ключа;
- очистка контейнера.

6. Матрица

Класс матрицы вещественных чисел должен реализовывать различные операции матричной арифметики:

- присваивание значений матриц, склеивание матриц, удаление, добавление и перестановки строк и столбцов матрицы, формирование единичной и нулевой матрицы, изменение размера матрицы;
- доступ к отдельным элементам матрицы как к элементам двумерного массива;
- арифметические операции между двумя матрицами, между матрицей и скаляром, поэлементные арифметические операции для двух матриц одного размера;
- отыскание минорной, транспонированной, треугольной и обратной матриц, вычисление определителя матрицы.

7. Разреженная матрица

Класс разреженной матрицы вещественных чисел ориентирован на эффективное использование памяти для представления матриц, в которых лишь относительно небольшое количество элементов имеют ненулевые значения, и должен реализовывать основные операции над матрицами:

- присваивание значений матриц, изменение размера матрицы и доступ к отдельным элементам матрицы как к элементам двумерного массива;
- арифметические операции между двумя матрицами, между матрицей и скаляром;
- поэлементные арифметические операции для двух матриц;
- формирование единичной и нулевой матрицы;
- удаление, добавление и перестановки строк и столбцов матрицы;

- отыскание минимальных и максимальных номеров строк и столбцов, содержащих ненулевые элементы.

8. Полином

Класс полинома должен реализовывать различные операции над полиномами:

- присваивание значений, доступ к значениям коэффициентов полинома и его порядка;
- арифметические операции между двумя полиномами, полиномом и скаляром, вычисление композиции полиномов;
- вычисление значения полинома при заданном значении аргумента, отыскание действительных и комплексных корней полинома;
- отыскание производной заданного порядка и неопределенного интеграла как объект этого же класса, отыскание значения определенного интеграла полинома;
- преобразование полинома в строку и обратно, используя при этом общепринятые формы записи полиномов и векторов.

9. Комплексное число

Разрабатываемый класс должен реализовывать вычисления с комплексными числами с наибольшей возможной точностью:

- присваивание значений комплексных чисел и стандартных типов данных;
- арифметические операции с комплексными числами и со стандартными типами данных, вычисление трансцендентных функций комплексного аргумента, включая гиперболические и обратные функции;
- определение действительной и мнимой составляющей комплексного числа, его модуля и угла;
- преобразование комплексного числа в строку и обратно, используя при этом как алгебраическую, так и показательную формы записи.

10. Неограниченные числа

Класс должен реализовывать вычисления с неограниченными по длине целочисленными значениями, количество разрядов которых превышает установленное для стандартных типов данных. Класс должен обеспечивать:

- присваивание значений, преобразование значения стандартного типа данных в неограниченное целое и обратно, если это возможно;
- арифметические операции между неограниченными числами;
- преобразование неограниченного целого в строку, содержащую его представление в десятичной, в двоичной системах счисления (со знаком) и обратное преобразование с проверкой правильности записи строки;
- определение количества байт, требуемых для хранения двоичного представления числа в памяти.

11. Точные вычисления

Класс точного числа должен реализовывать вычисления с заданной точностью, превышающей точность стандартных типов данных, в том числе с неограниченным количеством значащих разрядов:

- присваивание значений, преобразование точного числа в значения стандартных типов данных и обратно;
- арифметические операции между двумя точными числами, точным числом и скаляром;
- доступ к значениям разрядов числа, к знаку числа и позиции десятичной запятой;
- преобразование точного числа в строку и обратно, используя при этом общепринятые формы записи чисел.

12. Рациональная дробь

Разрабатываемый класс должен обеспечивать вычисления с числами, представленными в виде рациональной дроби:

- присваивание и сравнение дробей, приведение рациональной дроби к стандартным типам данных и обратно;
- арифметические операции между двумя рациональными дробями, рациональной дробью и скаляром, отыскание общего знаменателя двух дробей, сокращение дроби;
- отдельный доступ к числителю и знаменателю дроби;
- преобразование рациональной дроби в строку и обратно, используя при этом общепринятые формы записи чисел и запись в виде дроби с использованием символов «/» или «:».

13. Граф

Класс многодольного ориентированного мультиграфа должен обеспечивать:

- задание вида графа (однодольный или многодольный, ориентированный или неориентированный, с кратными дугами или без) и преобразование вида графа;
- присваивание значений и объединение графов;
- поиск изолированных подграфов, циклов и маршрутов, определение отношений инцидентности элементов графа;
- запись графа в файл и восстановление графа из файла.

14. Дерево

Класс должен реализовывать произвольное дерево, каждый узел которого содержит целочисленный ключ, и обеспечивать выполнение следующих операций:

- присваивание деревьев;
- вставка и удаление узла с заданным ключом;
- обход всех узлов дерева, поиск узла по ключу, доступ к родительскому и дочерним узлам для заданного узла;
- сравнение двух деревьев без учета ключей (только структура) и с учетом значений ключей;
- выделение ветви как отдельного дерева и включение дерева как ветви в другое дерево;

- определение ширины и высоты дерева, количества узлов в дереве, максимального количество потомков одного узла;
- одинарные и двойные повороты влево и вправо, произвольная перестановка узлов и ветвей;
- сохранение дерева в файл и загрузка из файла.

15. Множество

Класс множества (элементами являются целочисленные значения) должен обеспечивать операции:

- присваивание и сравнение множеств;
- задание вида множества (множество или мультимножество) и преобразование множества в мультимножество и обратно;
- добавление и удаление элементов множества, определение принадлежности значения множеству, перебор всех значений множества, отыскание мощности множества и количества уникальных значений;
- отыскание результата объединения, пересечения, разности, симметрической разности двух множеств с учетом их вида.

16. Системы счисления

Класс должен выполнять преобразование целочисленного значения в произвольную систему счисления с основанием 2 или более и должен обеспечивать:

- задание основания системы счисления;
- задание набора символов, используемых для записи числа (цифры, буквы большие и/или маленькие, с учетом регистра или без, другие знаки и т.п.);
- представление хранимого значения в виде строки символов заданного набора в заданной системе счисления;
- преобразование строки символов в число с проверкой правильности записи;
- присваивание, хранение и считывание целочисленного значения;
- присваивание и сравнение значений объектов.

17. Генератор случайных чисел

Класс генератора случайных чисел должен обеспечивать выполнение следующих функций:

- независимая генерация псевдослучайных последовательностей несколькими одновременно существующими экземплярами класса;
- реализация различных законов распределения случайной величины (равномерный, нормальный и др.);
- считывание очередного значения последовательности как целочисленного или вещественного;
- генерацию значений только в определенном интервале;
- заполнение массива заданного типа и размера значениями последовательности;
- сброс генератора в произвольный момент времени.

18. Файловый драйвер

Класс предназначен для организации стека потоков ввода и должен обеспечивать:

- перемещение текущего используемого потока в стек при подключении нового потока и восстановление предыдущего потока при завершении или отключении текущего;
- использование в качестве входных потоков буферизированного и небуферизированного ввода стандартной библиотеки языка C, потоков ввода языка C++;
- подключение ранее открытого потока в качестве входного, открытие указанного файла в заданном режиме (текстовый или двоичный);
- отключение и/или закрытие текущего потока при его завершении и принудительное (в произвольный момент времени);
- принудительное перемещение к началу текущего потока (при наличии возможности);
- посимвольное чтение из потока, чтение данных стандартных типов, чтение блока (аналогично read и fread);
- формирование признака конца файла при окончании текущего потока и опустошении стека.

19. Алгоритмы сортировки

Требуется разработать класс, позволяющий исследовать различные алгоритмы сортировки. Класс должен обеспечивать:

- выбор алгоритма сортировки;
- задание массива исходных данных для сортировки;
- генерацию массива исходных данных с заданными характеристиками (заполненный значениями в соответствии с заданным законом распределения случайной величины, упорядоченный и обратно упорядоченный, частично упорядоченный и др.);
- определение количества сравнений и перестановок, выполненных в процессе сортировки массива;
- сравнение показателей различных алгоритмов, выполненных над идентичными массивами.

Рекомендация: отдельные алгоритмы сортировки и алгоритмы заполнения массивов рекомендуется выполнить в виде классов, образующих соответствующие семейства.

20. Алгоритмы шифрования данных

Требуется разработать класс, осуществляющий кодирование и декодирование информации с целью устранения избыточности (сжатия), шифрования и/или повышения надежности передачи и хранения (введение избыточности, использование кодов, устойчивых к ошибкам). Каждый алгоритм должен быть представлен в виде отдельно класса, который обеспечивает:

- преобразование (кодирование/декодирование) данных исходного массива и формирование результирующего массива;
- считывание исходного массива из потока и запись результирующего массива в поток;
- задание параметров кодирования/декодирования (при необходимости).

21. Графические примитивы

Класс графического объекта позволяет описывать такие графические примитивы, как точка, отрезок, эллипс, прямоугольник и др. в двумерном декартовом пространстве и должен обеспечивать выполнение различных операций над ними:

- присваивание значений объектов;
- определение взаимного расположения двух объектов (идентичность, полное перекрытие, пересечение, вписывание, касание границ) и взаимного расположения их прямоугольных областей;
- изменение положения, размеров объекта и других характеристик;
- определение координат прямоугольной области, охватывающей множество объектов;
- перегрузку операций ввода и вывода описаний объектов в текстовый поток в формате SVG;
- генерацию пролога и эпилога XML-документа в формате SVG посредством соответствующих статических методов класса.

Рекомендация: отдельные графические примитивы рекомендуется выполнить в виде классов, образующих одно семейство.

22. Поле игры «Морской бой»

Класс реализует хранение и обработку информации об игровом поле одного игрока и должен обеспечивать:

- расстановку кораблей и проверку правильности расстановки;
- проверку правильности ходов противника, определение их результата и соответствующую обработку ходов;
- хранение последовательности событий и отмену любого количества ходов;
- сохранение в поток и загрузку из потока игрового поля и последовательности ходов.

23. Поле игры «Шашки»

Разрабатываемый класс предназначен для хранения и обработки информации об игровом поле и должен обеспечивать:

- задание размера поля, автоматическую начальную расстановку в соответствии с правилами игры, произвольную расстановку и контроль правильности расстановки;
- проверку правильности ходов, определение их результата, определение наличия возможных и обязательных ходов, обработку ходов;
- хранение последовательности событий в игре, отмену любого количества ходов;
- сохранение в поток и загрузку из потока игрового поля, последовательности ходов.

24. Поле игры «Реверси»

Разрабатываемый класс предназначен для хранения и обработки информации об игровом поле и должен обеспечивать:

- задание размера поля, автоматическую расстановку начальной позиции в соответствии с правилами, произвольную расстановку и контроль правильности расстановки;
- проверку правильности ходов, определение их результата, определение наличия возможных ходов, обработку ходов;

- хранение последовательности событий в игре, отмену любого количества ходов;
- сохранение в поток и загрузку из потока игрового поля, последовательности ходов.

25. Поле игры «2048»

Разрабатываемый класс предназначен для хранения и обработки информации об игровом поле и должен обеспечивать:

- задание размера поля, автоматическую расстановку начальных значений, произвольную расстановку и контроль правильности расстановки;
- задание параметров игры (ее сложности: необходимое для выигрыша значение, набор значений, появляющихся на поле и др.);
- проверку правильности ходов, определение их результата и обработку ходов, определение наличия возможных ходов;
- определение условий окончания игры и выигрыша, подсчет очков;
- хранение последовательности событий в игре, отмену любого количества ходов;
- сохранение в поток и загрузку из потока игрового поля, последовательности ходов.

26. Кроссворд

Класс предназначен для реализации компьютерных программ для составления и решения классических (обычных) кроссвордов.

Режим составления (создания) кроссворда должен обеспечивать:

- задание (изменение) общего размера поля;
- разметку поля (размещение слов по горизонтали и вертикали, проверку правильности пересечения слов, удаление слов), нумерацию слов в соответствии с правилами составления кроссвордов;
- поиск слов (совпадающих полностью, содержащих заданную букву, содержащих заданную букву в заданной позиции), определение состояния любой клетки в пределах кроссворда (содержит или не содержит букву);
- сохранение кроссворда в файл и загрузку его из файла.

Режим решения (разгадывания) кроссворда должен обеспечивать (для уже созданного кроссворда):

- размещение слова в соответствии с его номером и направлением (по горизонтали или вертикали), проверку возможности его размещения с учетом длины и пересечений, определение конфликтов пересечений (номер несовпадающей буквы и номер пересекаемого слова);
- удаление слова с учетом пересечений;
- проверку заполненности кроссворда, проверку правильности заполнения каждого слова (соответствие изначально заданным словам);
- сохранение текущего состояния кроссворда в файл и чтение его из файла.

27. Поле игры «Сапер»

Разрабатываемый класс предназначен для хранения и обработки информации об игровом поле известной компьютерной игры и должен обеспечивать:

- задание параметров игры, определяющих ее сложность: размер поля, количество «мин»;
- создание поля и автоматическую начальную расстановку «мин» в случайном порядке (новая игра в соответствии с заданными параметрами);

- определение состояния каждой клетки поля по ее координатам (закрыта, отмечена, открыта с указанием количества «мин» в соседних закрытых клетках);
- проверку правильности ходов («открыть», «отметить»), их обработку и определение результата;
- определение условий окончания игры и выигрыша, подсчет очков;
- сохранение в поток и загрузку из потока игрового поля (продолжение отложенной игры).

28. Игра «Змейка»

Разрабатываемое приложение представляет собой реализацию известной компьютерной игры. Приложение должно обеспечивать:

- реализацию игры в соответствии с ее правилами (правила необходимо определить);
- приостановку и возобновление игры, сброс игры (начать заново);
- определение условий завершения игры;
- подсчет количества очков;
- ведение таблицы рекордов.

Рекомендуется реализовать сохранение и восстановление отложенной игры, задание уровней сложности игры.

Обязательным является применение объектно-ориентированного подхода при проектировании и объектно-ориентированного языка программирования при разработке приложения.

29. Поле игры «Охота на лис»

Разрабатываемый класс предназначен для хранения и обработки информации об игровом поле компьютерной реализации технического вида спорта «Охота на лис». Во многом она похожа на игру «Сапер», но в открывающихся клетках игрового поля отражается количество «лис» (передатчиков), пеленгуемых из данной клетки – находящихся на одной вертикали, горизонтали или диагонали. Задача игрока – вычислить и пометить все передатчики, «наступить» на них нельзя (открытие клетки с передатчиком означает проигрыш).

Разрабатываемый класс должен обеспечивать:

- задание параметров игры, определяющих ее сложность: размер поля, количество «лис»;
- создание поля и автоматическую начальную расстановку «лис» в случайном порядке (новая игра в соответствии с заданными параметрами);
- определение состояния каждой клетки поля по ее координатам (закрыта, отмечена, открыта с указанием количества пеленгуемых «лис»);
- проверку правильности ходов («открыть», «отметить»), их обработку и определение результата;
- определение условий окончания игры и выигрыша, подсчет очков;
- сохранение в поток и загрузку из потока игрового поля (продолжение отложенной игры);
- возможность использования одинаковых параметров и одного игрового поля несколькими объектами класса (организация соревнований).

30. Поле игры «Лабиринт»

Требуется разработать систему классов для построения с ее использованием игр, заключающихся в прохождении лабиринтов.

Класс лабиринта должен описывать лабиринт как прямоугольный двумерный массив байт заданного размера, каждый элемент которого обозначает наличие в соответствующей клетке свободного пространства или стены. Этот класс должен учитывать наличие и расположение в лабиринте (в свободных ячейках) объектов лабиринта – неподвижных (артефактов) и подвижных (участники), а также контролировать правильность их перемещения и пересечение в одной ячейке лабиринта.

Класс артефакта является базовым для размещения в лабиринте различных объектов, обладающих некоторыми свойствами, обусловленными правилами создаваемой игры.

Класс участника является базовым для создания классов игрока и других активных объектов, как управляемых пользователем, так и имеющих собственный алгоритм поведения (в зависимости от логики создаваемого приложения). Объект класса участника взаимодействует с объектом класса лабиринта для определения состояния 4-х соседних с ним ячеек по вертикали и горизонтали, перемещения на 1 шаг в этих направлениях, обнаружения других объектов в зоне «прямой видимости».

Должны быть предусмотрены средства сохранения в файл (поток) всей информации о поле лабиринта, включая состояние всех объектов в нем, и загрузку этой информации из файла (потока).