Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования

**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)**

Кафедра автоматизации обработки информации (АОИ)

Ю. В. Морозова

ИНФОРМАТИКА И ПРОГРАММИРОВАНИЕ

**Методические указания
к выполнению курсовой работы
для студентов заочной формы обучения
с применением ДОТ
(уровень бакалавриата)**

Томск 2018

Корректор: А. Н. Миронова

Морозова Ю. В.

Информатика и программирование : методические указания к выполнению курсовой работы для студентов заочной формы обучения с применением ДОТ (уровень бакалавриата) / Ю. В. Морозова. – Томск : ФДО, ТУСУР, 2018. – 30 с.

В пособие внесены изменения в 2024 г.

ОГЛАВЛЕНИЕ

Введение 4
1 Методические указания к выполнению курсовой работы 5
1.1 Общие требования 5
1.2 Выбор варианта контрольной работы 5
2 Тематика курсовой работы 6
3 Порядок выполнения работы 7
4 Требования к содержанию и оформлению отчета по курсовой работе 14
5 Темы курсовой работы 16
Рекомендуемые источники 27
Приложение А Пример оформления титульного листа 29
Приложение Б Бланк задания на курсовую работу 30

ВВЕДЕНИЕ

Важным фактором подготовки специалистов в области информационных технологий является умение программировать, используя современные языки, включающие объектные возможности, знакомство с основными методами и современными технологиями программирования, в том числе с использованием объектных библиотек конкретных языков.

Курсовая работа по дисциплине «Информатика и программирование» имеет целью получение навыков самостоятельной разработки программного продукта в соответствии с принципами объектно-ориентированного программирования.

В результате изучения дисциплины ***студент должен***:

- ***знать***: методы обработки и способы реализации основных структур данных в объектно-ориентированных программных средах;
- ***уметь***: разрабатывать объектно-ориентированные программы в современных инструментальных средах;
- ***владеть***: практическими приемами объектно-ориентрованного программирования.

Курсовая работа позволяет сформировать способности будущего специалиста к самостоятельному решению практических задач и инженерных проблем с использованием теоретических положений, а также знаний и умений, полученных в ходе обучения объектно-ориентированному программированию.

1 МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ

1.1 Общие требования

Задачей курсовой работы является разработка программного продукта для заданной предметной области, включающая в себя:

- построение объектной модели предметной области;
- разработку диаграммы классов и интерфейсов;
- программную реализацию;
- тестирование и отладку программы;
- подготовку и написание отчета.

Выполнение курсовой работы предполагает самостоятельное изучение дополнительных вопросов по объектно-ориентированным языкам программирования (в частности, C++ и C#, Java), средам проектирования Windows-приложений, а также получение практического опыта объектно-ориентированного анализа и программирования, оформления соответствующей документации на программную разработку.

1.2 Выбор варианта контрольной работы

Выбор варианта контрольной работы осуществляется по общим правилам с использованием следующей формулы:

$$V = (N \times K) \text{ div } 100,$$

где V – искомый номер варианта,

N – общее количество вариантов,

div – целочисленное деление,

при $V = 0$ выбирается максимальный вариант,

K – код варианта.

2 ТЕМАТИКА КУРСОВОЙ РАБОТЫ

В ходе выполнения курсовой работы студенты должны на практике освоить общий методологический подход, используемый при проектировании и программной реализации системы классов, соответствующей объектно-ориентированной парадигме программирования.

Создаваемая система классов описывает (моделирует) определённую предметную область и может служить основой для полноценной информационной системы, решающей задачи данной области.

Спроектированная система классов должна быть не только реализована в виде программы на одном из объектно-ориентированных языков с графическим интерфейсом, но и протестирована.

Примерный перечень тем для курсовой работы

1. Разговорник.
2. Игра «Сапер».
3. Игра «Змейка».
4. Игра «Бродилка».
5. Редактор.
6. Фракталы.
7. Визуализация генетического алгоритма.
8. Мультфильм.
9. Игра «Города».
10. Графики.
11. Анализатор текста.
12. Микробы.
13. Лабиринты.
14. Магический шар.
15. Игра «Придумай слова».

3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Курсовая работа представляет собой результат выполнения следующих взаимосвязанных этапов:

1. Постановка и формулирование задачи.

2. Проектирование.

3. Кодирование.

4. Отладка и тестирование.

5. Документирование созданного программного продукта и оформление отчета по курсовой работе.

6. Рецензирование.

В рамках *первого этапа* следует:

1) изучить предметную область и осуществить сбор материала в проблемно-ориентированном контексте;

2) определить назначение программы, выработать требования к ней и представить их, если это возможно, в формализованном виде;

3) сформулировать требования к представлению исходных данных и выходных результатов;

4) определить структуры входных и выходных данных;

5) сформулировать ограничения и допущения на исходные и выходные данные.

В рамках *второго этапа* следует:

1) сформировать объектно-ориентированную модель задачи;

2) осуществить выбор метода реализации решения задачи;

3) разработать алгоритм реализации задачи.

При создании любого программного продукта всегда присутствует этап проектирования задачи. При написании реальных программ одна из первых возникающих при этом проблем – определить, должна ли быть программа объектно-ориентированной или нет. Использование объектно-

ориентированного программирования без необходимости только увеличит объем программы и сложность ее написания.

Смешивать два подхода в одном проекте не рекомендуется, поскольку может оказаться, что готовый программный продукт обладает недостатками и структурного, и объектно-ориентированного принципов построения. Но следует также иметь в виду, что даже небольшие программы могут с успехом пользоваться уже существующими библиотеками классов.

Для проектирования объектной модели удобно использовать язык UML. ***UML*** – унифицированный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем. Словарь UML включает три вида строительных блоков:

- диаграммы;
- сущности;
- связи.

Сущности – это абстракции, которые являются основными элементами модели, ***связи*** соединяют их между собой, а ***диаграммы*** группируют представляющие интерес наборы сущностей.

Диаграмма – это графическое представление набора элементов, чаще всего изображенного в виде связного графа вершин (сущностей) и путей (связей). Язык UML включает ***13*** видов диаграмм, среди которых на первом месте в списке – диаграмма классов.

Классы – это базовые элементы любой объектно-ориентированной системы. Классы представляют собой описание совокупностей однородных объектов с присущими им свойствами – атрибутами, операциями, отношениями и семантикой.

Классы в UML изображаются на диаграммах классов, которые позволяют описать систему в статическом состоянии – определить типы объектов системы и различного рода статические связи между ними. Классы отображают типы объектов системы.

Графически класс изображается в виде прямоугольника, разделенного на 3 блока горизонтальными линиями:

- имя класса;
- атрибуты (свойства) класса;
- операции (методы) класса.

Атрибут (свойство) – это именованное свойство класса, описывающее диапазон значений, которые может принимать экземпляр атрибута.

Операция (метод) – это реализация метода класса. Для атрибутов и операций может быть указан один из трех типов видимости:

- \- – private (частный);
- # – protected (защищенный);
- \+ – public (общий).

Видимость для полей и методов указывается в виде левого символа в строке с именем соответствующего элемента.

Абстрактные методы класса обозначаются курсивным шрифтом. Статические методы класса обозначаются подчеркиванием.

Существует четыре типа связей в UML:

- зависимость;
- ассоциация;
- обобщение;
- реализация.

Эти связи представляют собой базовые строительные блоки для описания отношений в UML, используемые для разработки хорошо согласованных моделей.

При создании класса, то есть нового типа данных, следует хорошо продумать его интерфейс – средства работы с классом, доступные использующим его программистам. Интерфейс хорошо спроектированного класса должен быть интуитивно ясен, непротиворечив и обозрим. Как правило, он должен включать в себя только методы и свойства. Поля данных должны быть скрытыми. Это даст возможность впоследствии изменить реализацию класса без изменения в его интерфейсе, а также регулировать доступ к полям класса с помощью предоставляемых пользователю методов. Не нужно расширять интерфейс класса без необходимости, «на всякий случай», поскольку увеличение количества методов будет вести к трудности понимания класса другим программистом, который, возможно, будет использовать его в своей работе. В идеале интерфейс должен быть полным, т. е. предоставлять возможность выполнить любые разумные действия с классом, и минимальным – без дублирования и пересечения возможностей методов.

В рамках *третьего этапа* следует:

1) уточнить структуру входных и выходных данных и определить формат их представления в объектно-ориентированной программе;

2) произвести программирование решения задачи;

3) закомментировать текст программы и составить предварительное описание программы.

После завершения проектирования библиотеки классов можно приступать к её программной реализации. Однако этот процесс невозможен без предварительного решения ряда технических вопросов: на каком языке следует писать программу, какая среда разработки должна использоваться, каких правил именования идентификаторов следует придерживаться, как организовать проектную работу, как документировать процесс программирования, как отслеживать изменения в коде программы.

Рекомендованными в рамках дисциплины языками являются C++ и C#, Java.

Рекомендуемой средой разработки является система для языка Java – Eclipse. Тем не менее, студент имеет право остановиться на каком-либо другом объектно-ориентированном языке высокого уровня, позволяющем разрабатывать независимые оконные приложения. Выбор языка требует обязательного обоснования.

Обоснование строится на основе выполненного анализа предметной области, исходя из следующих определяющих факторов:

- функциональные требования к системе;
- наличие в языке возможностей для реализации функциональных требований;
- трудоёмкость разработки.

Обоснование должно быть оформлено в виде связного текста, содержащего сравнительную оценку альтернативных вариантов выбора по указанным критериям.

Далее необходимо выполнить генерацию иерархии классов на выбранном языке программирования с получением основных классов и структур данных, сформировать архитектуру программного модуля или модулей, определить алгоритмы методов, разработать при необходимости интерфейс программного продукта.

Грамотно спроектированная диаграмма классов позволяет очень легко написать программный код, содержащий общее описание классов (иерархия классов, свойства, прототипы методов). Однако полноценно использовать классы и работать с объектами этих классов можно только в том случае, когда полностью даны определения всем методам. Поэтому в ходе программной реализации системы классов основной задачей является алгоритмизация и программирование методов классов.

В рамках *четвертого этапа* следует:

1) составить тесты для проверки правильности работы программы;

2) произвести обнаружение, локализацию и устранение ошибок в программе, выявленных с помощью тестов;

3) скорректировать код программы и ее описание.

Тестирование программного обеспечения должно показать работоспособность программы, а также выявить ошибки и дефекты. Следует различать процессы тестирования и отладки программного кода.

Отладка выполняется программистом с помощью встроенных средств среды разработки и исходя из опыта написания программного кода. В основном она сводится к выявлению синтаксических и семантических ошибок в тексте программы.

Тестирование – это процесс, требующий планирования и выполнения ряда предварительных процедур, основной из которых является составление набора тестовых сценариев, образующих тест-сьют. Тестовые сценарии в большинстве случаев основаны на функциональных требованиях к системе и могут затрагивать различные уровни разработки (модульное тестирование, интеграционное тестирование, системное тестирование).

В ходе курсовой работы необходимо выполнить упрощённый вариант модульного тестирования, сводящийся к тестированию всех методов разработанной библиотеки классов.

Тестирование библиотеки классов выполняется в соответствии с разработанной методикой. Если все предыдущие этапы работы завершены успешно, то данная процедура в основном сводится к выполнению тестов и фиксации результатов их выполнения. Документирование может быть выполнено в упрощённом режиме. Для каждого выполненного теста фиксируются:

1) номер теста (по нумерации тестов в методике тестирования);

2) входные данные;

3) ожидаемый результат выполнения теста;

4) результат выполнения теста;

5) вывод (результат выполнения теста соответствует /не соответствует ожидаемому).

Если в ходе тестирования выявляются несоответствия, следует выполнить анализ его причин и привести результаты анализа в тексте отчета (например, указать, чем вызвано несоответствие – семантическими или алгоритмическими ошибками, сформулировать рекомендации по исправлению несоответствия и т. п.).

В рамках *пятого этапа* следует:

- оформить курсовую работу в виде отчета согласно требованиям ОС ТУСУР 01-2021.

4 ТРЕБОВАНИЯ К СОДЕРЖАНИЮ И ОФОРМЛЕНИЮ ОТЧЕТА ПО КУРСОВОЙ РАБОТЕ

По результатам курсовой работы оформляется отчет. Оформление должно соответствовать требованиям стандарта ОС ТУСУР 01-2021. Примеры оформления титульного листа и задания на курсовую работу приведены в приложении А.

Для данной курсовой работы рекомендовано следующее содержание отчета:

1. Титульный лист
2. Задание на курсовую работу
3. Оглавление
4. Введение
5. Разработка библиотеки классов
   5.1. Диаграмма классов
   5.2. Выбор языка программирования
   5.3. Реализация классов
6. Тестирование
7. Заключение
8. Список использованных источников
9. Приложение А Диаграмма классов
10. Приложение Б Листинг программы

Объем работы должен составлять не менее 25 страниц основной части. Изложение должно быть последовательным, логичным, конкретным.

Первая страница – титульный лист, вторая – задание, далее – аннотация, оглавление и текст (номера на первых двух страницах не указываются). Оглавление создается автоматически средствами текстового редактора.

Документ может содержать таблицы, рисунки и формулы. На каждую таблицу, рисунок, формулу должна быть ссылка в тексте.

В текст отчета могут быть включены небольшие фрагменты программного кода, обязательно с комментариями. Рекомендуемый шрифт для набора фрагмента кода – Courier New, размер 12 пт.

На материалы, взятые из источников, должны быть даны ссылки с указанием номера источника по списку использованной литературы.

Список составляется в порядке появления ссылок.

В приложениях размещаются листинг основных классов, схемы программы, скриншоты интерфейса. Приложения нумеруются русскими буквами в порядке появления ссылок на них в основном тесте документа.

5 ТЕМЫ КУРСОВОЙ РАБОТЫ

1 Разговорник

Разрабатываемый разговорник должен представлять собой оконное приложение, в котором реализованы функции поиска и сопоставления (русский – английский, английский – русский) слов и словосочетаний при их вводе с клавиатуры или выборе из алфавитного каталога. Разговорник должен содержать не менее 100 слов и не менее 20 словосочетаний.

В рамках курсового проекта должны быть реализованы:

1. Дружественный графический интерфейс программы. Интуитивно понятное управление.

2. Возможность выбора слов и словосочетаний из алфавитного каталога.

3. Возможность ввода слов и словосочетаний с клавиатуры.

4. Отображение текущего количества слов и словосочетаний в разговорнике.

5. Возможность дополнять разговорник новыми словами.

6. Меню «О программе», содержащее вкладки «Справка», «О разработчике».

2 Игра «Сапёр»

Игра-приложение «Сапёр» представляет собой плоское или объёмное игровое поле, которое разделено на смежные ячейки (квадраты, шестиугольники, кубы и т. п.), некоторые из которых «заминированы»; количество «заминированных» ячеек известно. Целью игры является открытие всех ячеек, не содержащих мины.

Игрок открывает ячейки, стараясь не открыть ячейку с миной. Открыв ячейку с миной, он проигрывает. Мины расставляются после первого хода, поэтому проиграть на первом же ходу невозможно. Если под открытой ячей-

кой мины нет, то в ней появляется число, показывающее, сколько ячеек, соседствующих с только что открытой, «заминировано»; используя эти числа, игрок пытается рассчитать расположение мин, однако иногда даже в середине и в конце игры некоторые ячейки всё же приходится открывать наугад. Если под соседними ячейками тоже нет мин, то открывается некоторая «незаминированная» область до ячеек, в которых есть цифры. «Заминированные» ячейки игрок может пометить, чтобы случайно не открыть их. Открыв все «не заминированные» ячейки, игрок выигрывает.

В рамках курсового проекта должны быть реализованы:

1. Дружественный графический интерфейс программы. Интуитивно понятное управление.

2. Алгоритм расстановки «мин» на игровом поле.

3. Отображение времени игры.

4. Отображение текущего количества открытых и скрытых «мин».

5. Возможность помечать «заминированные» ячейки.

6. Меню «О программе», содержащее вкладки «Справка», «О разработчике».

3 Игра «Змейка»

Программа должна представлять собой оконное приложение, которое имитирует движение змеи, позволяет наращивать длину «змейки» путем «поедания мышек». В процессе игры должна быть реализована возможность перехода на следующий уровень при достижении определенной длины «змейки» (увеличение скорости движения «змейки», появление препятствий). В случае, если «змейка» ударяется об стену, препятствие или свой хвост, игра останавливается и выдается сообщение «Вы проиграли!».

В рамках курсового проекта должны быть реализованы:

1. Дружественный графический интерфейс программы. Интуитивно понятное управление.

2. Возможность задавать начальный уровень.

3. Возможность изменять начальную длину змейки.

4. Возможность менять цвета фона и тела «змейки».

5. Отображение текущего рекорда, счета, количества жизней.

6. Меню «О программе», содержащее вкладки «Справка», «О разработчике».

4 Игра «Бродилка»

Игра должна представлять собой платформер. Главный герой прыгает по платформам, собирает предметы, которые отображаются в виде бонусных очков в строке меню. На пути главного героя должны встречаться препятствия, при касании которых игра заканчивается. Для завершения уровня необходимо пройти все платформы и собрать необходимое количество предметов.

В рамках курсового проекта должны быть реализованы:

1. Дружественный графический интерфейс программы. Интуитивно понятное управление. Сохранение результирующего счета.

2. Возможность задавать начальный уровень.

3. Возможность менять цвет главного героя.

4. Отображение текущего счета, количества жизней.

5. Меню «О программе», содержащее вкладки «Справка», «О разработчике».

5 Редактор

Программа должна представлять собой оконное приложение, которое позволяет загружать и просматривать текстовые файлы с возможностью поиска и замены слов и выражений.

В рамках курсовой работы должны быть реализованы:

1. Дружественный графический интерфейс программы. Интуитивно понятное управление.

2. Возможность загрузки текстовых файлов (*.txt).

3. Разработка общего алгоритма программы (создание формы приложения с полем для отображения содержимого загружаемого файла, кнопки поиска/замены).

4. Разработка алгоритма поиска и замены слов и выражений.

5. В случае обнаружения совпадения программа должна выдавать окно с информацией о количестве совпадений.

6. Меню «О программе», содержащее вкладки «Справка», «О разработчике».

6 Фракталы

Разработать генератор геометрических фракталов (снежинка Коха, ковёр Серпинского, кривая дракона, кривая Леви и т. д.) на основе стандартных средств графических библиотек.

В рамках курсового проекта должны быть реализованы:

1. Дружественный графический интерфейс программы. Интуитивно понятное управление. Визуальное представление.

2. Возможность задавать количество итераций.

3. Возможность менять цвет.

4. Меню «О программе», содержащее вкладки «Справка», «О разработчике».

7 Визуализация генетического алгоритма

Генетический алгоритм – алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе. Генетические алгоритмы не

являются гарантированно точными или оптимальными, но достаточны для решения поставленной задачи (Генетические алгоритмы : учеб. пособие / Л. А. Гладков, В. В. Курейчик, В. М. Курейчик. – 2-е изд. – М. : Физматлит, 2006. – 320 с.).

Задача генетического алгоритма формализуется таким образом, чтобы её решение могло быть закодировано в виде вектора («генотипа») генов, где каждый ген может быть неким объектом. В классических реализациях генетического алгоритма предполагается, что генотип имеет фиксированную длину.

Некоторым, обычно случайным, образом создаётся множество генотипов начальной популяции. Они оцениваются с использованием «функции приспособленности», в результате чего с каждым генотипом ассоциируется определённое значение («приспособленность»), которое определяет, насколько хорошо фенотип, им описываемый, решает поставленную задачу. Из полученного множества решений («поколения») с учётом значения «приспособленности» выбираются решения (обычно лучшие особи имеют большую вероятность быть выбранными), к которым применяются «генетические операторы» (в большинстве случаев «скрещивание» и «мутация»), результатом чего является получение новых решений. Для них также вычисляется значение приспособленности и затем производится отбор («селекция») лучших решений в следующее поколение. Этот набор действий повторяется итеративно, так моделируется «эволюционный процесс», продолжающийся несколько жизненных циклов (поколений), пока не будет выполнен критерий остановки алгоритма.

Таким критерием может быть:

- нахождение глобального решения,
- исчерпание числа поколений, отпущенных на эволюцию,
- исчерпание времени, отпущенного на эволюцию,
- исчерпание времени на улучшение предыдущего результата.

Генетические алгоритмы служат главным образом для поиска решений в многомерных пространствах поиска.

В рамках курсового проекта должны быть реализованы:

1. Дружественный графический интерфейс программы. Интуитивно понятное управление. При запуске программы пользователю предлагается разместить на прямоугольной области препятствия и установить точку цели, прежде чем запустить основной цикл программы.

2. С момента запуска основного цикла начинается демонстрация движения особей. Когда текущее поколение прекращает существование, стартует следующая «итерация» основного цикла, особи нового поколения начинают движение из начальной точки. И так далее для последующих поколений. Другими словами, пользователь сможет наблюдать движение особей каждого поколения.

3. В случае если сходимость алгоритма происходит слишком медленно, пользователь имеет возможность пропустить показ некоторого количество поколений.

4. Меню «О программе», содержащее вкладки «Справка», «О разработчике».

8 Мультфильм

В рисованных мультфильмах иллюзия движения создается последовательной сменой кадров, каждый из которых фиксирует очередное положение движущегося объекта. Используя этот принцип, получить мультфильм, показывающий: а) идущего человечка; б) бегущего человечка; в) человечка, выполняющего приседания; г) человечка, выполняющего сигнализацию флажком. Написать программу для движения объекта, отрисовки и смены кадров.

В рамках курсового проекта должны быть реализованы:

1. Дружественный графический интерфейс программы. Интуитивно понятное управление.

2. Алгоритм отрисовки.

3. Возможность устанавливать паузу.

4. Меню «О программе», содержащее вкладки «Справка», «О разработчике».

9 Игра «Города»

Смысл игры «Города» заключается в том, что пользователь поочередно с компьютером называет слова. Название каждого следующего слова должно начинаться с той буквы, которой заканчивается название предыдущего. Если название слова оканчивается на «ь» или «ы», можно использовать предпоследнюю букву. Уже использованные названия не могут повторяться.

Составить программу, позволяющую компьютеру и человеку играть в слова. Предварительно программа объясняет правила игры и позволяет уточнить их в любой момент. Тематикой игры могут быть по выбору города, животные, растения и т. д. Тема выбирается из предложенных компьютером (не менее трех).

В рамках курсового проекта должны быть реализованы:

1. Дружественный графический интерфейс программы. Интуитивно понятное управление.

2. При нажатии кнопки «Начать» («Start») открывается игровое поле, на нем находятся кнопки «Ввод» («Enter») и «Сдаться» («Give up»).

3. Возможность «Сдаться» («Give up») означает, что пользователь хочет покинуть игру или не знает слов, чтобы ее продолжить.

4. Меню «О программе», содержащее вкладки «Справка», «О разработчике».

10 Графики

Составить программу, которая предлагает пользователю некоторый список функций для построения графиков (например, $y = ax2 + bx + c$, $y = ax + b$ и т. д. – до 10 наименований). После выбора соответствующей функции, задания коэффициентов и отрезка, на котором выполняется построение, программа строит указанный график. Затем значение коэффициентов и положение графика можно менять (например, с помощью клавиш управления курсором), после чего график перестраивается и записывается обновленное уравнение соответствующей кривой.

В рамках курсового проекта должны быть реализованы:

1. Дружественный графический интерфейс программы. Интуитивно понятное управление.

2. Выбор функций.

3. Возможность установить коэффициенты и отрисовку нескольких графиков в одних осях.

4. Меню «О программе», содержащее вкладки «Справка», «О разработчике».

11 Анализатор текста

Анализатор текста. Парсинг страниц. Квантирование текста.

Исходные данные представлены в двух файлах. Первый файл содержит словарь ключевых слов, разделенных запятыми. После последнего слова должна стоять точка. Длина текста – не более NUMW слов, длина строки – не более NS символов, длина слова – не более NW символов. Второй файл содержит приставки (прописными буквами), допустимые правилами словообразования. Составить программу, которая выводит исходный текст, список существующих приставок и преобразованный текст, в котором найденные приставки отделены от остальной части слова символом «-».

В рамках курсового проекта должны быть реализованы:

1. Дружественный графический интерфейс программы. Интуитивно понятное управление.

2. Алгоритм сравнения.

3. Возможность загрузки файлов.

4. Меню «О программе», содержащее вкладки «Справка», «О разработчике».

12 Микробы

Составить программу с заданными условиями.

В пробирку посадили микроб ровно в текущее время. Каждую минуту микроб делится на два таких же микроба, те, в свою очередь, через минуту тоже делятся и т. д.

В рамках курсового проекта должны быть реализованы:

1. Дружественный графический интерфейс программы. Интуитивно понятное управление.

2. Сделать так, чтобы пользователь мог отслеживать рост микробов и указывать время, когда количество микробов должно перестать увеличиваться.

3. Меню «О программе», содержащее вкладки «Справка», «О разработчике».

13 Лабиринты

Разработать программу, в которой генерируются лабиринты (простой вариант: вручную; сложный вариант: случайным образом). Ваша задача – найти правильный путь и угадать, куда он приведет.

В рамках курсового проекта должны быть реализованы:

1. Дружественный графический интерфейс программы. Интуитивно понятное управление.

2. Пользователь должен видеть изначальную точку и варианты, куда может привести лабиринт.

3. Дать возможность пользователю производить выбор и выдавать результат (ошибся ли пользователь, или показал правильный путь).

4. Меню «О программе», содержащее вкладки «Справка», «О разработчике».

14 Магический шар

Разработать программу, которая будет как магический шар выдавать случайный результат. К примеру, вы задаете вопрос: «...?», а программа выдает вам результат из предложенного:

- Да
- Нет
- Скорее всего да
- Скорее всего нет
- Возможно
- Имеются перспективы
- Вопрос задан неверно

В рамках курсового проекта должны быть реализованы:

1. Дружественный графический интерфейс программы. Интуитивно понятное управление.

2. Возможность пользователю дополнять ответы.

3. Меню «О программе», содержащее вкладки «Справка», «О разработчике».

15 Игра «Придумай слова»

Известна игра с придумыванием слов, состоящих из тех же букв, что и некоторое фиксированное слово. Например, из слова ПАСКАЛЬ можно

получить слова ЛАК, ЛАСКА, СКАЛА и т. д. Кратные вхождения некоторой буквы в получаемое слово допускаются, если эта буква с наименьшей кратностью входит в слово-образец. Пусть дана последовательность слов, разделенных пробелами. Приняв, что первое слово в последовательности есть образец, выбрать те из остальных членов последовательности, которые могут быть получены из образца по указанному выше правилу. Максимально возможную длину слова считать равной 18.

В рамках курсового проекта должны быть реализованы:

1. Дружественный графический интерфейс программы. Интуитивно понятное управление.

2. Меню «О программе», содержащее вкладки «Справка», «О разработчике».

РЕКОМЕНДУЕМЫЕ ИСТОЧНИКИ

1. Павловская Т. А. C/C++. Программирование на языке высокого уровня : учебник для вузов / Т. А. Павловская. – СПб. : Питер, 2013. – 461 с.

2. Лаптев В. В. C++. Объектно-ориентированное программирование : учеб. пособие / В. В. Лаптев. – СПб. : Питер, 2008. – 464 с.

3. Катаев М. Ю. Объектно-ориентированное программирование : учеб. пособие / М. Ю. Катаев, А. Я. Суханов. – Томск : ТМЦДО, 2007. – 160 с.

4. Романенко В. В. Объектно-ориентированное программирование : учеб. пособие / В. В. Романенко. – Томск : ФДО, ТУСУР, 2014. – 475 с.

5. Панов С. А. Объектно-ориентированное программирование : курс лекций / С. А. Панов, Т. В. Ганджа. – Томск : ТУСУР, 2015. – 110 с.

6. Лаптев В. В. C++. Объектно-ориентированное программирование. Задачи и упражнения : учеб. пособие для вузов / В. В. Лаптев, А. В. Морозов, А. В. Бокова. – СПб. : Питер, 2007. – 288 с.

7. Ашарина И. В. Язык C++ и объектно-ориентированное программирование в C++ : лабораторный практикум / И. В. Ашарина. – М. : Горячая Линия – Телеком, 2015. – 232 с.

8. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений / Г. Буч, Р. А. Максимчук, М. У. Энгл и др. – М. : Вильямс, 2010. – 720 с.

9. Васильев А. Н. C#. Объектно-ориентированное программирование / А. Н. Васильев. – СПб. : Питер, 2012. – 320 с.

10. Лафоре Р. Объектно-ориентированное программирование в C++ / Р. Лафоре. – СПб. : Питер, 2015. – 928 с.

11. Лесневский А. С. Объектно-ориентированное программирование для начинающих / А. С. Лесневский. – М. : Бином. Лаборатория знаний, 2010. – 232 с.

12. Пол А. Объектно-ориентированное программирование в C++ / А. Пол. – М. : Бином, 2001. – 464 с.

13. Рассел Дж. Объектно-ориентированное программирование / Дж. Рассел. – М. : Книга по требованию, 2012. – 74 с.

14. Хорев П. Б. Объектно-ориентированное программирование / П. Б. Хорев. – М. : Академия, 2012. – 448 с.

15. Шлеер С. Объектно-ориентированный анализ: моделирование мира в состояниях / С. Шлеер, С. Меллор. – М. : Диалектика, 1993. – 240 с.

16. Образовательный стандарт ТУСУР 01-2021. Работы студенческие по направлениям подготовки и специальностям технического профиля. Общие требования и правила оформления (утвержден приказом ректора ТУСУРа от 25.11.2021 № 1100). – URL: https://regulations.tusur.ru/documents/70 (дата обращения: 11.01.2024).

ПРИЛОЖЕНИЕ А

Пример оформления титульного листа

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

ТЕМА РАБОТЫ

Курсовая работа по дисциплине

«Информатика и программирование»

Студент гр. ___

_____ И. О. Фамилия

«___»_____20_ г.

Руководитель

доцент каф. АОИ,

канд. техн. наук

_____ Ю. В. Морозова

«___»_____20_ г.

Томск 20__

ПРИЛОЖЕНИЕ Б

Бланк задания на курсовую работу

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

УТВЕРЖДАЮ
Зав. каф. АОИ
_____ А. А. Сидоров
«___» _____ 20__ г.

ЗАДАНИЕ
на курсовую работу по дисциплине
«Информатика и программирование»

1. Тема работы:
«_____».
2. Срок сдачи работы: «____» _____ 20__ г.
3. Цель работы:
_____.
4. Формулировка задания:
_____.
5. Содержание отчета:
_____.

Задание выдал:
_____
«__» _____ 20__ г.
Принял к выполнению:
_____
«__» _____20__ г.