МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(МОСКОВСКИЙ ПОЛИТЕХ)

ПРОГРАММИРОВАНИЕ И ОСНОВЫ АЛГОРИТМИЗАЦИИ

Методические указания к выполнению курсовых работ для студентов, обучающихся по направлениям подготовки 27.03.04 «Управление в технических системах» и 15.03.04 «Автоматизация технологических процессов и производств»

Авторы: Е.В. Пикалов, Н.В. Груненков

Текстовое электронное издание

Москва
2024

Дисциплина «Программирование и основы алгоритмизации» преподаётся бакалаврам направлений 27.03.04 «Управление в технических системах» и 15.03.04 «Автоматизация технологических процессов и производств» на втором курсе обучения. Изучение дисциплины завершается выполнением обучающимися курсовой работы, чему и посвящены данные методические указания.

В состав издания входят краткие теоретические сведения, справочные материалы и рекомендации к выполнению, а также сами темы курсовых работ с подробным пояснением необходимого функционала разрабатываемых студентами программ и ожидаемого их ввода и вывода.

Рецензент:
А.В. Кузнецов, к.т.н., доцент, заведующий кафедрой «Автоматика и управление» Московского Политеха

Рекомендовано к изданию на заседании кафедры «Автоматика и управление»

»

Системные требования: PC-совместимый процессор 1,3 ГГц и выше. Оперативная память (RAM): 256 Мб. Необходимо на винчестере: 350 Мб. Операционные системы: Windows, Mac OS. Видеосистема: разрешение экрана 1024x768. Дополнительные программные средства: Adobe Acrobat Reader 9 и выше.

Разработано с помощью программного обеспечения Microsoft Office Word, Adobe Acrobat Pro

Издается в авторской редакции

Компьютерная верстка: ###
Подписано к использованию ###
Объем издания ###. Тираж ###. Заказ № ###

Издательство Московского Политеха
115280, Москва, Автозаводская, 16
www.mospolytech.ru; e-mail: izdat.mospolytech@yandex.ru;
тел. (495) 276-33-67

СОДЕРЖАНИЕ

Введение

При подготовке будущего специалиста по направлениям, связанным с автоматизацией и управлением в технических системах, нельзя обойти стороной такую дисциплину, как программирование и алгоритмизация. Студент должен изучить как принципы построения блок-схем алгоритмов, логику построения программ и взаимодействия программных блоков, так и познакомиться с конкретным языком программирования. Этот язык должен иметь широкое применение в мире, и давать достаточный базис, необходимый для понимая основных аспектов работы компьютера и компьютерных программ. В будущем это бы способствовало более простому освоению любого другого языка при необходимости, так как основа – логическое структурированное мышление, умение произвести декомпозицию задачи и навык разработки алгоритмов – есть основа для профессиональных навыков любого успешного программиста.

Именно эти навыки призвана развить данная дисциплина, чтобы у будущих инженеров-автоматизаторов и робототехников было сформировано понимание как должна строиться разработка и отладка управляющей программы для любого устройства в общем случае.

В то же время курсовая работа по дисциплине «Программирование и основы алгоритмизации» является важнейшей составляющей самостоятельной работы студентов. В связи с этим, данная курсовая работа является также и творческой задачей, и эта составляющая преднамеренно заложена в любую тематику.

Само выполнение курсовых работ по данным методическим указаниям **несёт в себе цели**:

- развить у студента навыки по самостоятельному поиску информации в предметной области;
- развить у студента навыки самостоятельной работы над задачей;
- изучить язык программирования;
- понять основные принципы и логику построения алгоритмов и программ на их основе;

- научиться самостоятельно проводить тестирование программных продуктов, производить их доработку и исправлять ошибки;
- получить навыки по грамотному составлению описательной документации, в виде отчёта по курсовой работе, отражающего все аспекты разработанной программы.

Для достижения этих целей обучающийся решает множество задач, касающихся полного цикла разработки программного обеспечения, но не обязательно все. Вот некоторые из них:

- определение целей и задач в рамках конкретной темы;
- организация собственной самостоятельной работы, грамотное распределение времени и задач на весь период реализации курсовой работы;
- формирование требований к разрабатываемой программе;
- составление блок-схем алгоритмов, структурных и функциональных схем взаимодействия различных блоков разрабатываемой программы;
- написание программного кода;
- отладка, поиск и исправление ошибок в разрабатываемой программе;
- тестирование разрабатываемой программы, исправление некорректной её работы;
- составление отчёта по курсовой работе и т.д.

Слова благодарности

Данные методические указания являются первой серьёзной авторской работой старших преподавателей кафедры «Автоматика и управление» Пикалова Е.В. и Груненкова Н.В., которые по такому случаю не могут обойти вниманием людей, всячески помогавших им.

Авторы выражают благодарность лично доценту, кандидату технических наук, доценту кафедры «Автоматика и управление» Кузнецову Александру Валерьевичу за предоставленный простор

для самореализации, полезные советы и возможность заниматься любимым делом.

Преподавателю программирования МГИУ/МАМИ Курасову Юрию Викторовичу за ценное наставничество и помощь в формировании профессиональных компетенций.

Инженеру автоматизации тестирования и контроля качества («a1qa», «Альфа-банк») Коноплёву Андрею Сергеевичу за полезную критику и предложения, активный интерес, проявленный к данным методическим указаниям.

А также активным студентам групп 201-251, 211-251, 211-291 Московского Политехнического Университета за высказанные идеи, пожелания и отзывы.

Общие положения и теоретические сведения

Формат тем

Как было сказано во введении, данная курсовая работа является самостоятельной работой студента, включающей творческую составляющую. Подобное выражается в отсутствии чётких требований, например, к количеству и качеству реализуемых функций, набору команд, оформлению пользовательского интерфейса и некоторым другим аспектам выполняемой работы. Таким образом, студент получает достаточный простор и свободу для мысли и творческого решения задачи. Это выливается в уникальную программу, отражающую его индивидуальное видение, а также полученные им за изученный курс умения и навыки, в рамках выбранной темы.

В соответствии с этим, темы курсовых работ оформлены в виде задач на разработку программы, включающих в первую очередь требования к их функционалу. Так же, дополнительно, могут быть прописаны требования к интерфейсу пользователя, некоторым аспектам реализации и оформления.

Вид разрабатываемой программы

Разрабатываемые студентами программы представляют из себя **исключительно консольные приложения**. Программы могут быть реализованы как с предоставлением пользовательского меню, так и с обработкой пользовательских команд. Первый вариант является базовым и подразумевает следующую структуру вывода программы:

« --- Меню ---
1. Действие 1
2. Действие 2
3. Помощь
4. Выход »,

где пользователю необходимо ввести лишь число, соответствующее пункту меню.

Вариант с обработкой пользовательских команд, например, «help», «quit» и т.д., оценивается выше при защите. Подробнее в разделе 1.6 «Модификаторы сложности».

Обязательно следует предусмотреть в программе наличие пункта меню или команды для завершения программы. Непредвиденное завершение программы является ошибкой.

Если программа ожидает пользовательский ввод, то следует обозначить это соответствующим приглашением к вводу. Например, знаками «больше» «>>», знаком двоеточия «:» или любым другим понятным и удобным знаком.

Любая программа должна содержать справочную информацию по её работе, которую можно наблюдать или вызвать в процессе работы с этой программой. Это может быть реализовано при помощи отдельного пункта меню «Помощь», команды «help», справочной информации-легенды в доступной области окна консоли. Справочная информация должна содержать краткое описание назначения программы, её возможностей, список всех функций с описанием их назначения, дополнительных ключей и примерами использования.

При запуске программа обязательно должна выводить приветственное сообщение любого формата на усмотрение разработчика. Его содержание должно как минимум сообщать о штатном начале работы и предлагать пользователю ввести команду или вызвать меню помощи.

Так же, программа должна отзываться на все вводимые команды, либо совершая какое-либо действие и оповещая пользователя об его успешности, либо сообщая о неудаче и ошибке. Качественной программой считается та, которая содержит достаточное количество обработок исключений – особых ситуаций её использования, не являющихся штатными, но предусмотренных заранее при разработке. Целью обработки исключений является предупреждение неожиданных завершений программы и появления каких-либо ошибок, делающих невозможным штатное продолжение работы программы.

Нет определённых требований к использованию при организации исходного кода какого-либо конкретного стиля отступов. Однако, настоятельно рекомендуется в самом начале работы выбрать один конкретный стиль отступов, и придерживаться его на

протяжении всей работы в целях повышения читаемости исходного кода программы.

Порядок выполнения курсовой работы

Существуют различные модели организации процесса разработки программных продуктов, однако, во многих этапах они сходятся. Для реализации программы в рамках данных методических указаний предлагается придерживаться этих этапов. Все из них на данный момент практикуются и в промышленной разработке управляющих программ, и в крупных IT-гигантах, и даже в качественно организованных IT-стартапах.

Таким образом, выделим всего 4 основных этапа, важных для грамотной организации процесса разработки программы курсовой работы:

- анализ задачи, составление требований к будущей программе;
- проектирование алгоритма работы программы;
- написание программного кода;
- тестирование программы.

Придерживаясь этих этапов, можно гарантировать, что процесс разработки программы будет удобнее и проще, а итоговый результат качественнее, чем, например, если сразу браться за написание кода.

Рассмотрим каждый этап немного подробнее:

1. **Анализ задачи, составление требований к будущей программе**
   На первом этапе необходимо дать ответ на все вопросы, возникающие к формулировке задачи, осветить все неоднозначные моменты. Этап можно считать пройденным, когда студент чётко понимает и может подробно описать, как должна работать его программа, какой набор команд исполнять, как должен выглядеть интерфейс пользователя на каждом этапе работы программы.
2. **Проектирование алгоритма работы программы**
   На данном этапе необходимо ответить на вопрос «как должна работать программа?». Это означает связать пред-

ставление о наборе команд с представлением о пользовательском интерфейсе, ввод с выводом. В свою очередь, это выливается в составление всех необходимых блок-схем алгоритмов, структурных и функциональных схем, в наиболее полной мере описывающих принцип действия будущей программы. Важно помнить, что алгоритм должен быть построен так, чтобы его мог прочитать человек, знающий любой язык программирования, или же не знающий никакого из них. Алгоритм должен быть исчерпывающим, но не избыточным. Остальные требования к оформлению алгоритмов будут описаны в разделе 4.3 «Требования к оформлению алгоритмов».

3. **Написание программного кода**
   Этот этап, включённый в приведённую систему, являет собой реализацию разработанных алгоритмов действия программы на определённом языке программирования. Не рекомендуется начинать курсовую работу сразу с этого этапа, так как это вызовет лишь неоправданную потерю времени, а избежать исполнения предыдущих этапов не представляется возможным. Окончанием этого этапа можно считать готовую в первом приближении программу студента, реализующую основной необходимый функционал.
4. **Тестирование программы**
   Под последним этапом подразумевается проверка работоспособности разработанной программы, соответствования всем поставленным требованиям и отказоустойчивости. Рекомендуется применить последовательно два основных вида тестирования по сценарию, а именно **позитивное** и **негативное**:
   В первую очередь, в рамках **позитивного** тестирования, проверяются сценарии, соответствующие штатному поведению разработанной программы. Вводятся ожидаемые команды, и программа возвращает ожидаемый вывод. Далее проводится **негативное** тестирование, то есть проверяется отказоустойчивость программы. Вводятся не предопределённые команды, или же ожидаемые команды

видоизменяются. Могут изменятся типы вводимых данных, размеры окна консоли, вызываться команды в непредсказуемом порядке и так далее.

По итогам проведения каждого тестирования все обнаруженные ошибки должны быть исправлены.

В результате завершения этого этапа повышается отказоустойчивость и качество работы программы, что особенно внимательно оценивается при защите курсовой работы.

Помимо этого, стоит подходить осмысленно к решению любой задачи на всех уровнях. Прежде всего, для упрощения самостоятельной разработки программы рекомендуется прибегать к такому приёму, как *декомпозиция задачи*. Этот приём заключается в разбиении крупной задачи на более мелкие подзадачи, связанные между собой, и решение их отдельно друг от друга в определённом или свободном порядке.

Самым простым примером декомпозиции в случае данных курсовых работ может быть решение задачи написания программы, которая ищет в целочисленном массиве наибольшее значение. Такая задача может быть разбита на подзадачи: чтение (вывод) элемента массива; чтение (вывод) всех элементов массива в цикле; сравнение всех элементов массива с одним значением; перезапись сравниваемого значения, если значение массива больше него.

Поступательно решая такие равноценные по сложности подзадачи, разработчик двигается по нарастающей к усложнению структуры своей программы и решению итоговой, общей задачи, более простым и удобным путём, чем попытка браться за всё разом.

Среды разработки и компилятор

Для удобства выполнения, а также последующей проверки студенческой курсовой работы предлагается использовать свободно распространяемое программное обеспечение в виде среды разработки (IDE) Code::Blocks последней версии.

Стоит отметить, что данное ПО на текущий момент поставляется в комплекте с набором MinGW (компилятор GCC) НЕ последней версии. Получить последнюю версию компилятора можно по ссылке из раздела 5.2 «Полезные ссылки». После его установки в самом Code::Blocks необходимо будет указать путь к нему на вкладке Settings -> Compiler -> … «Путь к папке mingw».

Настоятельно **НЕ рекомендуется** пользоваться избыточными для данных задач средами разработки, такими как Visual Studio, VS Code, XCode и прочими. Также не рекомендуется использовать онлайн-компиляторы. В случае, если обучающий не располагает техническими возможностями для самостоятельной работы, он может обратиться к преподавателю или на кафедру и ему будет выделено достаточно отдельного от занятий времени в компьютерной аудитории.

Разрабатываемые программы должны свободно компилироваться MinGW последней версии на **ОС Windows**. На защиту курсовой работы предоставляется бумажный отчёт по курсовой работе, исходный код программы в электронном виде или на переносном носителе и итоговый .exe-файл. При сборке итогового .exe-файла, предоставляемого на защиту, следует **обязательно** установить в IDE флаг статической компиляции (gcc ключ «-static»). Если файл студента не запускается по ошибке студента – такая работа не может быть оценена более чем на 0 баллов.

Все ссылки на необходимое программное обеспечение можно найти в соответствующем разделе.

Разбиение тем по сложности

Разбиение задач по сложности преследует цель обеспечить удобный выбор студентом темы по собственным амбициям и возможностям, областям интересов, и выливается в следующую классификацию:

Блок тем класса С «начальная сложность»

Данные темы предлагается брать студентам, которые не изучали программирование в школе, либо плохо знакомы с программированием как таковым, и для которых знакомство с ним началось лишь в рамках дисциплины «Программирование и основы ал-

горитмизации», к которой принадлежат эти методические указания. Темы в этом блоке достаточно просты для освоения такими студентами, и в то же время имеют достаточный потенциал для развития необходимой базы понимания основ алгоритмизации и получения основных навыков программирования.

Блок тем класса B «нормальная сложность»

Темы этого блока рассчитаны на студентов, уже знакомых с программированием со школы, изучавших такие языки программирования, как: Pascal, Basic, C и др. Выполнение заданий этого блока более тесно познакомит обучающегося с синтаксисом и особенностями языка C++, продемонстрирует особенности применения языка программирования для решений различных задач, расширит навыки алгоритмизации и даст достаточный базовый прикладной опыт программирования.

Блок тем класса A «повышенная сложность»

Данный блок тем рассчитан на студентов, интересующихся программированием, имеющих желание развивать профессиональные навыки, связанные с написанием программ и чувствующих себя уверенно в таких языках как: Pascal, Basic, C, C++, C#, Python, Java и др. Реализация программ из этого блока тем потребует гибкости в применении различных функций, методов и конструкций языка, вызовет необходимость самостоятельно изучить предметную область и особые возможности языка, не рассмотренные в рамках лабораторных работ по дисциплине. Задания в этом блоке требуют внимательного алгоритмического планирования и структурированного, вдумчивого подхода к их решению.

Блок тем класса S «специальная сложность»

Блок тем «высокая сложность» подготовлен специально для студентов, обладающих высокими навыками и большим опытом в области программирования, способным решать сложные алгоритмические задачи, грамотно писать, оформлять и структурировать свой код, или для групп студентов от 2-ух человек. Задания этого блока могут потребовать дополнительных консультаций с преподавателем или автором настоящих методических указаний, а получение этой темы, также, как и любой другой, не гарантирует получение какой-либо конкретной оценки. Особенностью тем блока S является более широкая направленность тематик, предусмотренных, прежде всего, для энтузиастов.

Таким образом, в данных методических указаниях предлагается к выполнению:

13 тем класса C «начальная сложность»

10 тем класса B «нормальная сложность»

5 тем класса A «повышенная сложность»

3 темы класса S «специальная сложность»,

что позволит каждому обучающемуся подобрать себе тему согласно собственным интересам и возможностям, а при желании получить повышенную оценку, добавив в программу специальные модификаторы сложности.

Модификаторы сложности

Одним из основных критериев оценки курсовой является объём и качество проделанной студентом работы. Реализованная в полном соответствии с заданием любой сложности и подкреплённая правильно оформленным отчётом работа оценивается на *«хорошо»* при соответствующей защите. Для увеличения шанса получения более высокой оценки обучающемуся предлагается расширить функционал своей программы за счёт реализации специальных *модификаторов сложности*.

Модификаторы сложности представляют из себя различные методы, подходы и отдельные подзадачи, которые можно интегрировать в готовую или разрабатываемую программу по любой из тем. В некоторых темах такие модификаторы включены по умолчанию, и обойти их не представляется возможным. Например, тема №26 «Мессенджер через веб-сокеты» требует обязательной реализации сетевого взаимодействия, в отличии от многих других тем.

В то же время, для каждой темы есть уникальные модификаторы сложности, немного повышающие шанс получить хорошую оценку. Они описаны в специальном разделе каждой темы.

Интеграция в программу модификаторов сложности укрепляет позиции студента на защите, а также влечёт за собой дополнительные вопросы, верно ответив на которые студент может претендовать на оценку *«отлично»*.

Список доступных модификаторов сложности:

МС-1. Интеграция набора команд. Реализация обработчика пользовательского ввода и набора команд, вместо пользовательского меню. Таким образом, ввод пользователя становится более мнемонически ясным, например, ввод команды «quit» будет вызывать завершение программы с соответствующим сообщением. Вместе с этим, может понадобиться обрабатывать дополнительные ключи для команд. Например, «quit -s» также завершит программу, но, на этот раз в «тихом режиме», без вывода какого-либо сообщения, в то время как обычный «quit» выведет предупреждение о завершении работы программы на несколько секунд.

МС-2. Взаимодействие с файлом. Чтение исходных данных, необходимых для работы программы (заполнения массивов, списков и иных структур данных) из текстового файла на выбор пользователя. Запись результатов работы программы, сортированных массивов, собранных списков, графов, логов работы и т.д. на усмотрение разработчика и преподавателя в файл с определённым именем на выбор пользователя. Пользователь вводит относительный или абсолютный путь к файлу с расширением «.txt» или «.json» и программа обрабатывает его содержимое.

МС-3. Динамический вывод, консольная графика. В общем случае, применение данного модификатора означает повышение удобства пользовательского интерфейса с применением различных доступных средств разработки консольных приложений. Одним из таких инструментов является заголовочный файл «wingdi.h», содержащий функции, необходимые для рисования различных геометрических фигур в консоли. Другим вариантом является перемещение курсора в консоли на определённую позицию при помощи структуры COORD и соответствующих функций из заголовочного файла *windows.h*. Базовый пример реализации этого модификатора приводится в Приложении 4.

МС-4. Многопоточное выполнение программы. В простейшем случае главный поток отвечает за обработку ввода пользователя, а другие за выполнение соответствующих действий. Пример реализации этого модификатора приводится в Приложении 5.

МС-5. Сетевое взаимодействие посредством веб-сокетов. В первом приближении разнести по сети можно сервер, как исполнитель пользовательских команд и главный обработчик, и клиент как простейший интерфейс ввода-вывода, передающий данные от

пользователя-клиента на сервер. Базовый пример реализации этого модификатора приводится в конце данных методических указаний: Приложение 2 – серверная сторона, Приложение 3 – клиентская сторона.

Данные модификаторы действуют для каждой темы. По вопросам интеграции определённого модификатора в программу студент может обратиться к преподавателю или автору данных методических указаний.

Перечень тем

Далее приводится список тем для выполнения курсовых работ в порядке повышения их сложности.

Важно понимать, что сложность выбранной темы не является ни единственным, ни даже основным критерием получения той или иной оценки. Оценка за курсовую работу складывается из многих параметров, включающих как сложность исходного задания, так и использование дополнительных модификаторов сложности, приёмы программирования, использованные разработчиком, его теоретическая подготовка и практические навыки на момент защиты, качество отчёта и т.д. Подробнее о критериях оценивания в разделе 3 «Критерии оценивания».

Блок тем класса C «начальная сложность»

Тема №1. Алгоритмы поиска в тексте

Необходимо разработать программу, реализующую возможность поиска в тексте количества определённых:

- символов;
- слов;
- подстрок.

Исходный текст, в котором будет производиться поиск может быть введён пользователем прямо в консоль, или же прочитан из файла.

Вводом является выбор соответствующего пункта пользовательского меню, например, «1», для пункта «1. Поиск символов». Далее ввод символа, который необходимо найти, например, «а».

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «find sym a»

Выводом программы в обоих случаях будет являться количество найденных в предложенном тексте символов «a», а также расстояние от начала текста до первого вхождения символа «a». Например, «Символ «a» встречается 42 раз(а), первое вхождение на позиции 14.»

Особые возможности для повышения оценки:

- Вывод содержимого файла с выделением найденных символов, слов, подстрок.

Тема №2. Очередь и стек

Необходимо разработать программу, позволяющую создавать очередь и стек и взаимодействовать с ними:

- добавлять элементы;
- удалять элементы;
- читать элементы;
- изменять значения элементов.

В программе должно быть реализовано меню выбора, отвечающее за то, куда (стек или очередь), и какой элемент (например, символ) будет записываться, изменяться или считываться.

Вводом может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Запись в стек». Далее ввод символа, который будет записан в стек, например, «B».

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «stack push B».

Выводом программы же является демонстрация содержимого текущей активной структуры данных, или обеих структур, после выполнения каждой команды. Например,

« Stack: 5 – 3 – 6 <
Queue: 8 – 3 < »

Особые возможности для повышения оценки:

- Возможность работы с несколькими стеками и очередями по названию.

Тема №3. Сортировка одномерных массивов

Разработать программу, принимающую на вход последовательность вещественных чисел, разделённых пробелом, и сортирующая их 5-ю различными методами на выбор пользователя.

В программе должно быть реализовано меню для ввода последовательности чисел и опция выбора сортировки различными методами. Например, вводится «1» для пункта «1. Сортировка методом пузырька».

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «sort bubble».

Выводом программы является демонстрация введенной последовательности чисел, отсортированной последовательности по выбранному методу, затраченное на сортировку время, объём памяти и количество итераций.

Также предусмотреть возможность автоматического заполнения массива исходных данных случайными числами в количестве и диапазоне на выбор пользователя.

Особые возможности для повышения оценки:

- Реализовать 7 и более методов сортировки.
- Предусмотреть возможность чтения исходного набора значений из файла, а также возможность добавления новых значений.
- Добавить опцию, осуществляющую сортировку всеми методами, и вывод их результатов в виде сравнительной таблицы.
- Добавить дополнительные фильтры для сортировки: по модулю, по четности, по делению на заданное число.

Тема №4. Алгоритмы поиска в линейных структурах

Реализовать программу, осуществляющую поиск в линейной структуре данных медианы, моды и среднего арифметического. А также поиск первого вхождения элемента по значению в сортированном различными методами.

В программе должно быть реализовано меню, позволяющее пользователю либо ввести содержимое структуры самостоятельно

вручную, либо заполнить её в автоматическом режиме случайными числами из определённого диапазона на выбор.

Вводом может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Поиск медианы».

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «find median».

Выводом программы может являться результат поиска, демонстрация содержимого линейной структуры данных, затраченное на поиск время, объем занимаемой памяти и количество итераций.

Особые возможности для повышения оценки:

- Добавить опцию, осуществляющую поиск значения всеми методами, и вывод их результатов в виде сравнительной таблицы.

Тема №5. Моделирование работы биржи

Реализовать программу, моделирующую жизненный цикл валют и других ценных бумаг (минимум 4 позиции) в банке с течением времени. Курс валют и ценных бумаг периодически случайно изменяется. Есть возможность обменять валюты и ценные бумаги, внести вклад (банк выплачивает процент с течением времени), взять кредит (банк получает процент с течением времени).

В программе должно быть реализовано меню, позволяющее пользователю брать кредиты, покупать или продавать валюту, вносить средства на вклад и пропускать определённое количество дней для изменения курса валют. Иными словами, реализовать модель биржевого брокера.

Вводом может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Отобразить текущий курс валют».

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «currency rate».

Выводом программы является демонстрация ресурсов пользователя и текущего курса валют банка, например,

« У вас на счету: 0 рублей 0 @ 0 & 0 Э

Валюты:	Продать	Купить
@ (собаки)	156.01	179.00
& (ишки)	10.54	13.37
Э (э-коины)	42.42	50.50
Ценные бумаги:	Продать	Купить
Серббак	1991.06	2023.06
Тунцофф	1516.17	1819.20 »

Для организации более удобного постоянного вывода текущих курсов валют настоятельно рекомендуется применить модификатор MC-3 или MC-4.

Особые возможности для повышения оценки:

- Организовать ежедневные и ежемесячные вычеты средств с основного счета в качестве расходов.
- Увеличить количество позиций до 8 и больше, добавив ценные металлы.

Тема №6. Моделирование учёта товара в аптеке

Реализовать программу, ведущую учёт товарно-материальных ценностей розничного аптекарского магазина (несколько наименований товара, их количество) изменение формата поставок, поставка при достижении определённого уровня, покупку и бронирование товара.

В программе должна быть реализована возможность взаимодействовать с объектами магазина, узнать количество для различных товаров, заказать недостающие или отсутствующие позиции.

Вводом может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Отобразить количество товаров N».

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «amount N».

Выводом программы является демонстрация наименований и их количество, например,

« N есть в наличии, количество: 10
Критическое значение: 3
Объём в следующей поставке: 20 »

Особые возможности для повышения оценки:

- Добавить в систему возможность пропуска определённого количества дней, за которые случайным образом «покупаются» – вычитаются из наличия различные наименования.
- Реализовать истечение срока годности товаров (отсчёт идёт от момента поставки). Имеет смысл добавлять только если реализован предыдущий пункт.

Тема №7. Уравновешивание химический уравнений

Реализовать программу, уравнивающую химические формулы **по количеству элементов** в левой и в правой частях.

После начала работы программа просит пользователя ввести химическую реакцию.

Вводом может служить сама химическая реакция, например, «H2 + O2 = H2O».

Выводом программы является уравненная формула химической реакции с выставленными коэффициентами, например,

«Результат уравновешивания:

2H2 + O2 = 2H2O »

Особые возможности для повышения оценки:

- Добавить вывод валентности химических элементов, определения типа реакции, окислителя и восстановителя.

Тема №8. Сжатие, хэширование и сверка ключей

Разработать программу, которая способна сжимать текстовую строку. Сжатие осуществляется за счёт повторяющихся символов, например, ааа = а3. В исходном сообщении предполагается наличие только букв. Также программа должна уметь высчитывать как минимум 2 различных типа хэш-ключа для любой строки и совершать "разархивирование" для сжатой строки.

Вводом может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Сжать строку». Далее вводятся сами строковые данные.

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «zip …».

Выводом программы является отображение сжатой и «разархивированной» строки, или хэш-ключей на выбор пользователя.

Особые возможности для повышения оценки:

- Добавить возможность сжатия методами:
  - кодированием переменной длины;
  - Хаффмана;

  на выбор разработчика и преподавателя.

Тема №9. Булева алгебра

Разработать программу, реализующую двоичные вычисления операциями AND, OR, NOR, XOR, XNOR, NAND, NOT и операции побитового сдвига.

В консоли должно быть реализовано меню, где дается краткая сводка, в каком формате и какие операции могут быть введены, например, «Введите X OPERATION Y для двух операндов». Пример операции «1010 XOR 1101». В случае ввода незначащих нулей программа должна отрабатывать операцию штатно.

Выводом в консоль служат операнды, операция и результат выполнения операции, например,

« 0011
OR 0101
0111 »

Особые возможности для повышения оценки:

- Предусмотреть выбор системы счисления для вводимых операндов и для отображения результата.
- Использовать возможность использования флага переноса, в случае добавления операций побитового сдвига или операций сложения и вычитания.

Тема №10. Полный калькулятор

Реализовать программу, моделирующую полный набор функций стандартного, инженерного и программистского калькулятора.

В программе должно быть реализовано меню, позволяющее пользователю производить операции над числами и выбирать тип калькулятора, например, инженерный.

Выбором функционала калькулятора может служить один из пунктов пользовательского меню, например, «1» для пункта «1. Стандартный калькулятор».

Вводом должна служить строка, например, «10+28». Выводом программы является результат операции.

Особые возможности для повышения оценки:

- Реализация обработки целого выражения, содержащего различные операции и скобки, например, «1+(sin(90)*8^2-1)».

Тема №11. Статистическая обработка данных

Реализовать программу, считывающую вещественные числа, разделённые пробелами, сохраняющая их как набор данных, и по команде рассчитывающую и выводящую для них:

- среднее;
- медиану;
- моду;
- дисперсию;
- среднеквадратическое отклонение;
- принадлежность к нормальному распределению по критерию хи-квадрат.

Вводом для такой программы может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Ввести исходные данные». Далее вводится имя набора данных, например, «data1» и сами вещественные числа через пробел.

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «new data1» и ввод чисел следом.

Выводом программы является результат расчёта того или иного параметра.

Также предусмотреть возможность автоматического заполнения массива исходных данных случайными числами в количестве и диапазоне на выбор пользователя.

Особые возможности для повышения оценки:

- Добавить возможность провести регрессионный и корреляционный анализ для двух наборов данных.

Тема №12. Аппроксимация, интерполяция, экстраполяция

Реализовать программу, позволяющую произвести аппроксимацию, интерполяцию и экстраполяцию для функции по введённым отсчётам. Данные могут быть предоставлены пользователем как через известные интервалы, то есть с постоянным шагом, так и в формате "X X;Y Y".

Вводом для такой программы может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Произвести линейную аппроксимацию набора данных». Далее вводится имя набора данных, например, «data1» (формат ввода данных схож с темой №11 «Статистическая обработка данных»), с которым планируется работать.

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «approx data1».

Выводом программы является результат расчёта того или иного параметра.

Особые возможности для повышения оценки:

- Реализовать графическое представление исходного набора данных и рассчитываемых значений.

Тема №13. Строительный калькулятор

Реализовать программу для расчета количества рулонов обоев, упаковок напольного покрытия, количества мешков сухой смеси, емкостей краски, клея или грунтовки и расходных материалов (шпателей, валиков и т.д.) в зависимости от объема помещения.

В программе должно быть реализовано меню, позволяющее пользователю ввести начальные данные (ширину, длину и высоту помещения) и данные о товаре (количество шт. в упаковке или объем в емкости, а также площадь покрываемой товаром поверхности).

Вводом может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Введите данные о помещении», после чего вводятся параметры помещения, например, «10 5 3», если речь идёт про формат «%длина% %ширина% %высота%».

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «newroom 10 5 3».

Выводом программы является рассчитанное минимальное количество требуемого товара и цена.

Особые возможности для повышения оценки:

- Реализовать возможность рассчитывать товар для нескольких помещений.
- Учитывать допуски, например, в случае с напольными покрытиями.
- Добавить в оценку расчет цены и количества товара в зависимости от количества окон, дверей и их площади.

Блок тем класса B «нормальная сложность»

Тема №14. Работа с файловой системой

Реализовать программу, осуществляющую полный цикл взаимодействия с файловой системой:

- вывод содержимого директории;
- переход в поддиректории;
- перемещение директорий и/или файлов;
- удаление директорий и/или файлов;
- добавление директорий и/или файлов.

В программе должно быть реализовано меню для работы с файлами и директориями, позволяя пользователю перемещаться по файловой системе, создавать, удалять и перемещать файлы и директории, а также просматривать содержимое текущей директории.

Вводом может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Просмотр текущей директории».

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «ls» - сокр. от англ. «list». Для перемещения будет использоваться команда «cd testdir», где «testdir» - имя директории, в которую осуществляется переход. Для перехода обратно (выше) по дереву каталогов используется «cd ..».

Выводом программы является демонстрация содержимого текущей директории: файлов, папок, текущего активного пути, выполненного действия и т.д., например,

« Активный путь: C://Program Files/Tesing/
Список содержимого:
directory1
directory2
file1.txt
file2.jpg »

« Активный путь: C://Program Files/Tesing/
Директория testdir успешно создана»

Особые возможности для повышения оценки:

- Вывод содержимого директории на выбор пользователя:
  - только имена;
  - с размерами директорий и файлов;
  - с флагами доступа,

  в виде таблицы.
- Добавить возможность чтения текстовых файлов, вывода их содержимого на экран и их редактирования с перезаписью.

Тема №15. Обработка строк

Разработать программу, позволяющую обрабатывать введённую пользователем строку и подсчитать в ней количество слов, символов, символов с пробелами.

В программе должно быть реализовано меню для работы с содержимым исходной строки, позволяющее: найти количество

определённых слов и символов, заменить определённые символы или слова.

Вводом может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Подсчет количества символов». Далее ввод символа, по которому будет проведен поиск, например, «A».

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «find char A».

Выводом программы является демонстрация содержимого исходного файла, подпункты меню и, если необходимо, измененное содержимое файла.

Особые возможности для повышения оценки:

- Предусмотреть возможность поиска, с выделением в исходном тексте и замены определённых символов и подстрок.
- Реализовать возможность вставки после определённого символа подстроки, вводимой отдельно в консоль.

Тема №16. Тамагочи

Реализовать программу, моделирующую жизненный цикл воображаемого существа. Программа должна предусматривать следующий функционал:

- создание существа с именем;
- рост существа, как функцию времени, что выражается в изменении его параметров;
- возможность изменить параметры существа различными взаимодействиями, например, «накормить», «поиграть», «помыть» и т.д.

В зависимости от разных факторов существо будет продолжать жить, или может погибнуть. Предусмотреть возможность пропуска определённого времени с учётом изменения параметров.

Вводом может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Накормить питомца», которое понизит уровень параметра голода.

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «feed».

Выводом программы будет являться информация о всех параметрах существа – его статус.

Особые возможности для повышения оценки:

- Реализовать вывод изображения существа, изменяющегося во времени и зависящего как минимум от 2-ух параметров.
- Реализовать возможность «кормить» существо различными видами «еды» и «играть» с существом в различные «игры», что будет влиять на качество производимых операций.

Тема №17. Односвязные и двусвязные списки

Реализовать программу, осуществляющую полный цикл взаимодействия с односвязными и двусвязными списками:

- создание списка;
- вывод содержимого списков;
- добавление элемента:
  - в начало списка;
  - в конец списка;
  - по индексу;
  - после определённого значения;
- удаление или изменение элемента:
  - по индексу;
  - по содержимому;
  - в конце списка;
  - в начале списка;

Вводом может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Создать односвязный список». Следом, программа потребует ввести значение для заполнения элемента, например, целое число: «15».

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «create ol 15» (сокр. от англ. «one linked» – односвязный).

Выводом такой программы обычно является содержимое списков на текущий момент, например,

«ОС: 15 > 23 > 2

ДС: 13 <> 37 <> 42 <> 86 ».

Особые возможности для повышения оценки:

- Реализовать возможность произвести сортировку списков по содержимому в порядке:
  - возрастания;
  - убывания;
  - по модулю.

Тема №18. Графы

Реализовать программу, позволяющую в полной мере взаимодействовать с неориентированными графами:

- создание;
- добавление вершины;
- вывести в виде списка;
- вывести в виде таблицы;
- нахождение кратчайшего пути;
- поиск в ширину;
- поиск в глубину.

Вводом может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Создать граф». Следом, программа потребует ввести количество вершин, например, «4», и рёбер графа, например, «4», после чего необходимо будет указать все пары вершин, связанные рёбрами. Например, через пробел по одной:

«1 2

1 3

1 4

2 4 »

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «create 4 4», а далее перечисление пар в том же формате, приведённом выше.

Выводом такой программы обычно является результат проделанного действия. Выводятся графы либо в формате списка связей, либо в формате таблицы связей.

Особые возможности для повышения оценки:

- Реализовать возможность создания ориентированного графа с сохранением остального функционала программы.
- Реализовать возможность задавать веса рёбрам графа с сохранением остального функционала программы.

Тема №19. Криптография

Разработать программу, способную расшифровать строковое сообщение, введённое в консоль одним из методов шифрования на выбор пользователя (Хилла, Плейфера, Виженера, Цезаря, A1Z26, Атбаш).

В программе должно быть реализовано начальное меню, позволяющее пользователю ввести текст для дальнейшего шифрования/дешифрования, а также выбрать метод шифрования/дешифрования.

Вводом может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Расшифровать строку методом Хилла».

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «decode Hill».

Выводом программы является отображение начальной и расшифрованной строк, согласно выбранному пользователем методу.

Особые возможности для повышения оценки:

- Реализовать дешифровку как минимум с трёх различных языков при помощи частотного анализа.

Тема №20. Игра Калах

Реализовать программу, позволяющую сыграть в Калах на одном компьютере для двух игроков. Вся игра должна проводиться в рамках одного консольного приложения. Для ознакомления с правилами игры обучающемуся предлагается обратиться к

преподавателю, составителю данных методических указаний или к любым литературным и Интернет-источникам.

Стандартным вводом такой программы-игры будет выбор ячейки в соответствии с правилами игры.

Выводом программы должна являться демонстрация содержимого каждой ячейки и калахов обоих игроков, информации о том, какой игрок делает ход и какие ходы были сделаны.

Особых возможностей для повышения оценки не предусмотрено. Если студент высказывает желание получить оценку «отлично» ему рекомендуется организовать работу по сети (МС-4) или обратиться к преподавателю.

Тема №21. Игра «Twenty one»

Реализовать программу, позволяющую сыграть с компьютером в Twenty-one. Для ознакомления с правилами игры обучающемуся предлагается обратиться к преподавателю, составителю данных методических указаний или к любым литературным и Интернет-источникам.

В программе должно быть реализовано меню, позволяющее пользователю ввести количество карт в колоде (36 или 52) и возможность брать по одной карте или сбрасывать карты, а также добавить перемешивание.

Вводом может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Взять карту».

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «take».

Выводом программы является количество карт пользователя с указанием их номинала, например, «6D» – 6 of diamonds, и количество карт компьютера.

Особые возможности для повышения оценки:

- Ввести рейтинговую таблицу.
- Учитывать конец колоды при подряд идущих играх. Замешивать колоду заново при желании одного из игроков.

Тема №22. IP-калькулятор

Реализовать программу, позволяющую производить базовые расчёты компьютерных сетей согласно стеку TCP/IP.

Программа должна решать следующие задачи:

- расчёт IPv4 адреса сети на основании IPv4 и маски сети;
- расчёт количества доступных для подключения устройств и диапазон адресов для них, а также адреса широковещательного канала, на основе IPv4 адреса сети и маски сети;
- разбиение сети на несколько подсетей (целое число в допустимом диапазоне) на основе IPv4 адреса сети и маски сети.

Вводом может служить выбор одного из пунктов пользовательского меню, например «1» для пункта «1. Получить адрес сети». Далее вводится адрес машины в сети и маска сети, например, «172.21.55.3 255.255.255.0».

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «get subnet 172.21.55.3 255.255.255.0».

Выводом программы является отображение результата расчётов, для данного случая вывод будет содержать «172.21.55.0».

Особые возможности для повышения оценки:

- Реализовать обработку ввода IPv4 адреса и маски в формате «172.21.55.3/24».

Тема №23. Судоку

Реализовать программу, составляющую и выводящую на экран решаемое поле для игры в судоку. Для ознакомления с правилами игры обучающемуся предлагается обратиться к преподавателю, составителю данных методических указаний или к любым литературным и Интернет-источникам.

Вводом может служить выбор одного из пунктов пользовательского меню, например «1» для пункта «1. Создать поле».

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «create».

Выводом программы является отображение поля для игры в судоку, которое имеет решение.

Особые возможности для повышения оценки:

- Реализовать возможность заполнять полученное поле – решать судоку.

Блок тем класса А «повышенная сложность»

Тема №24. Бинарные деревья

Разработать программу, которая позволяет:

- создавать;
- редактировать – добавлять и удалять вершины;
- выводить на экран,

бинарные деревья любой высоты (до 10).

Вводом может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Создать бинарное дерево». Далее вводится целочисленное значения для заполнения начальной вершины (узла), например, «9».

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «create 9».

Выводом программы является вывод бинарного дерева в виде списка или таблицы связей.

Особые возможности для повышения оценки:

- Реализовать методы добавления, удаления, поиска и обхода вершин на основании рекурсии.
- Реализовать вывод на экран в виде бинарного дерева:
  « 9
  / \
  5 3 »
- Реализовать возможность проводить сортировку, переориентацию дерева в бинарное дерево поиска и сбалансированное бинарное дерево поиска.

Тема №25. Двумерная графика

Разработать программу, которая выводит на экран двумерный массив (поле) заданных элементов (цифра/символ), и позволяет отображать на нём геометрические формы, состоящие из иных символов (квадрат, круг, прямоугольник, ромб) различных размеров, а также перемещать их. Таким образом формируется двумерная графика.

В программе должно быть реализовано начальное меню, позволяющее пользователю выбрать отображаемую фигуру и задать ее размеры.

Вводом может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Нарисовать круг». Далее вводится целочисленное значения для указания габаритов фигуры, для круга, например, диаметр и символ отображения, «5 #».

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «circle 5 #».

Выводом программы является добавление отображения соответствующих фигур на поле.

Для организации более удобного постоянного вывода графики параллельно с обработкой пользовательского ввода настоятельно рекомендуется применить модификатор МС-4.

Особые возможности для повышения оценки:

- Реализовать "бегущую строку", состоящую из определённых символов.
- Реализовать динамический вывод диагональных, горизонтальных или вертикальных "бегущих волн".

Тема №26. Мессенджер через веб-сокеты

Реализовать программу, осуществляющую передачу между двумя ПК в сети строкового сообщения при помощи механизма веб-сокетов. Передача может осуществляться по IP или по имени ПК в сети. В процессе написания программы, её отладки и тестирования можно запускать клиентскую и серверную части на одной машине, обращаясь к «localhost» или по адресу «127.0.0.1».

В программе, со стороны клиента, должно быть реализовано начальное меню, позволяющее пользователю выбрать собеседника для обмена текстовыми сообщениями.

Вводом может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Написать пользователю». Далее вводится целочисленное значение из приведенной таблице активных пользователей, например, «2».

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод команды «connect», после чего, просмотрев список возможных подключений, пользователь может сделать соответствующий выбор, например, «2».

Выводом программы, помимо статусной информации, является отображение отсылаемых и принимаемых сообщений, адреса отправителя и времени получения/отправки.

Особые возможности для повышения оценки:

- Подключение к одному «чату» более чем двух клиентов.

Тема №27. Передача файлов через веб-сокеты

Реализовать программу, осуществляющую передачу между двумя ПК файла случайного размера посредством веб-сокетов. Принимающая сторона должна сохранить получаемый файл, получив его без искажений.

В программе должно быть реализовано меню, позволяющее пользователю выбрать файл для отправки.

Вводом может служить выбор одного из пунктов пользовательского меню, например «1» для пункта «1. Отправить файл». Далее вводится путь к файлу, например, абсолютный путь «C://Program Files/Tesing/file.txt» или же его относительная версия.

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «send file %путь к файлу%».

Выводом программы является отображение информации об активных подключениях и статус отправляемых файлов (доставлен/не доставлен).

Особые возможности для повышения оценки:

- Организация «серверного хранилища»: вывод со стороны клиента информации о доступных файлах для скачивания с сервера, иерархии каталогов с возможностью навигации в ней; просмотр размеров и прав доступа к файлам; загрузка и скачивание с сервера файлов с различными расширениями.

Тема №28. Моделирование работы лифтов

Реализовать программу, моделирующую работу N < 10 лифтов одновременно, рассчитываемых для M < 20 этажей на выбор пользователя. В этой теме обязательно необходимо реализовать графическое отображение столбцов, один из блоков которых является лифтом.

Вводом может служить выбор одного из пунктов пользовательского меню, например, «1» для пункта «1. Переместить лифт».

Стандартным вводом в случае использования пользовательских команд является команда формата "N M", после получения которой соответствующий лифт N вызывается на этаж M, передвигаясь по этажу в секунду, либо, если он занят, игнорирует задачу.

Выводом программы является отображение положения лифтов и их перемещения.

Настоятельно рекомендуется реализовать модификатор МС-4 в данной задаче, чтобы реализовать асинхронную обработку положения лифтов и ввода команд.

Особые возможности для повышения оценки:

- Реализовать для каждого лифта очередь задач, чтобы вводимая во время его перемещения команда помещалась в неё и выполнялась далее по порядку.
- Реализовать приоритет перемещений, чтобы лифт мог остановиться на каком-либо этаже, что называется «по пути», если такая задача есть в очереди.

Блок тем класса S «специальная сложность»

Тема №29. Минимизация булевых функций

Реализовать программу, принимающую на вход булеву функцию нескольких переменных и минимизирующая (упрощающая) её любым способом на выбор студента:

- законы булевых операций;
- метод Квайна;
- карты Карно и т.д.

Вводом в такой программе служит исходная функция, содержащая логические операции конъюнкции, дизъюнкции, логического следования, эквивалентности и отрицания, скобки и до трёх переменных (A, B, C). Например, «A+(!B>C)*A*!(A*C)».

Выводом является минимизированный вид исходной функции, полностью эквивалентный ей по таблице истинности, например, «A».

Особые возможности для повышения оценки:

- Расширить возможное количество переменных как минимум до 6 штук. Дать возможность пользователю использовать любые символы, не только A, B и C.
- Реализовать больше одного метода в одной программе.

Тема №30. Игра «Tap-In-Time»

Реализовать программу, предоставляющую пользователю возможность сыграть в «Tap-In-Time». Суть игры заключается в том, чтобы вовремя нажимать на соответствующие кнопки, постоянно приближающиеся к контрольной линии. Для ознакомления с правилами игры обучающемуся предлагается обратиться к преподавателю, составителю данных методических указаний или к любым литературным и Интернет-источникам.

Вводом может служить выбор одного из пунктов пользовательского меню, например «1» для пункта «1. Начать игру», после чего пользователь должен вовремя нажимать последовательность кнопок, например, «5 4 4 5 3 2 3 1 3 2 3» (с определённым интервалом).

Если же программа обрабатывает пользовательские команды, для выполнения такого же действия предполагается ввод следующего формата «start».

Выводом программы является отображение игрового поля – линии, по которой перемещаются символы. Как только символы оказываются на контрольной линии, пользователь должен нажать соответствующий символ на клавиатуре, и тогда ему засчитывается успешное нажатие и начисляются баллы.

Особые возможности для повышения оценки:

- Реализовать игру на нескольких линиях одновременно.

Тема №31. Свободная тема

Реализовать программу, которая удовлетворяет основным условиям, изложенным в данных методических указаниях, но не относится ни к одной из перечисленных тем, представляя из себя оригинальную задумку студента. Основным условием является то, что предлагаемая студентом тема должна быть предварительно обговорена и утверждена на встрече с преподавателем. Только после выполнения этого условия имеет смысл приступать к выполнению задания по этой теме. Стоит учесть, что к заданию, относящемуся к свободной теме, преподаватель может относится с повышенным вниманием.

Возможности для повышения оценки общие, рекомендуется применять модификаторы сложности, или обсудить особые возможности с преподавателем индивидуально.

Критерии оценивания

Классические критерии

Работа оценивается *отлично*, если студентом были выполнены все поставленные перед ним задачи, реализована программа по одной из предложенных тем, верно сформулирована цель и задачи курсовой работы.

Студент оперирует полученными знаниями при ответе на поставленные вопросы, уверенно ориентируется в предметной области. При этом могут быть допущены незначительные неточности.

В реализованной программе применён как минимум один модификатор сложности или особая возможность для повышения оценки.

Работа оценивается *хорошо*, если студентом были выполнены все поставленные перед ним задачи, реализована программа по одной из предложенных тем, верно сформулирована цель и поставлены задачи курсовой работы.

При этом были допущены некритичные ошибки:

- некорректно реализован функционал разрабатываемой программы;
- разработанная блок-схема алгоритма содержит незначительные ошибки;
- код программы простроен на готовых функциях и классах, относящихся к теме курсовой работы, или были использованы простые алгоритмические конструкции.

Студент оперирует полученными знаниями при ответе на поставленные вопросы, ориентируется в предметной области. При этом могут быть допущены некоторые неточности или незначительная ошибка.

Работа оценивается *удовлетворительно*, если студентом были выполнены все поставленные перед ним задачи.

При этом были допущены некоторые ошибки:

- не проведен анализ выбранной темы или некорректно подобран функционал разрабатываемой программы;
- разработанная блок-схема алгоритма содержит явные ошибки или алгоритм не соответствует предоставленной на защиту программе;
- код программы простроен на готовых функциях и классах, относящихся к теме курсовой работы, или были использованы простые алгоритмические конструкции;
- не проведено тестирование программы, на этапе защиты программа завершается непредвиденно, или её поведение не соответствует поставленным требованиям.

Студент испытывает затруднения при ответе на поставленные вопросы, плохо ориентируется в предоставленной на защиту программе, периодически допускает ошибки в своих ответах.

Работа оценивается *неудовлетворительно*, если студентом не были выполнены одна или более поставленных перед ним задач.

Были допущены значительные ошибки:

- не проведен анализ поставленной проблемы;
- не подобран, или неверно подобран функционал разрабатываемой программы;
- разработанная блок-схема алгоритма содержит критичные ошибки или алгоритм не соответствует предоставленной на защиту программе;
- код программы простроен на готовых функциях и классах, относящихся к теме курсовой работы, или были использованы простые алгоритмические конструкции;
- отчёт по курсовой работе не предоставлен, или оформлен не в соответствии с поставленными требованиями, не содержит основной информации о работе.
- не проведено тестирование программы, на этапе защиты программа завершается непредвиденно, или её поведение не соответствует поставленным требованиям.

Студент систематически допускает значительные ошибки при ответе на поставленные вопросы, не ориентируется в предоставленной на защиту программе.

Авторские критерии

На усмотрение преподавателя, принимающего выполненную студентом курсовую работу, для оценки может быть выбрана 100-балльная авторская система оценивания. Далее приводятся критерии, максимальное количество баллов по этому критерию в круглых скобках и его краткое описание. За общие критерии можно получить в сумме 10 баллов, за программу – 50, на защите можно добрать остальные 40. Такая значимая составляющая – более трети баллов – отводимая на устное собеседование на защите здесь заложена сознательно. Таким образом появляется возможность не при-

нимать работу студента в случае, если предоставляемая им программа соответствует всем требованиям, но сам студент в ней и в программировании не ориентируется, и не может подтвердить свои компетенции.

Сроки (5) – насколько студент уложился в объявленные сроки по сдаче курсовой работы.

Оформление (5) – насколько оформление самой курсовой работы соответствует выдвинутым требованиям (приводятся в разделе 4 «Требования к оформлению»).

Блок-схемы, логичность (10) – проверяется качество структурной организации программы, логичность связей подфункций, методов, вызовов циклов и т.д. Критерием выступает минимизация избыточности.

Соответствие условиям (10) – насколько предоставленная на защиту программа соответствуем выданному студенту заданию.

Удобство и интерфейс (10) – насколько программа эргономична, меню или команды удобны в обращении, интуитивно понятны для использования, восприятия и взаимодействия.

Отказоустойчивость (10) – проверяется обработка исключений: стабильность работы программы в штатном режиме, реакция программы на нестандартный ввод, непредвиденное завершение и т.д.

Читаемость и организация кода (10) – насколько понятны имена переменных и функций, насколько логично выделение функций и размещение их в различных файлах программы, обращение к ним, удобно ли будет масштабировать или редактировать предлагаемый код.

Устный ответ (40) – в первую очередь проверяется насколько студент разбирается в предоставляемой им программе. В остальном, оценивание по этому критерию отводится на усмотрение проверяющего. Могут быть заданы любые вопросы, касающиеся предмета и пройденного материала, или области, не затронутой на занятиях, но имеющей первостепенное отношение к конкретной студенческой работе.

В результате оценки согласно этим критериям, студенту могут быть предложены следующие оценки:

До 60 баллов – «не удовлетворительно»;

От 60 включительно до 75 баллов – «удовлетворительно»;

От 75 включительно до 90 баллов – «хорошо»;
От 90 баллов включительно – «отлично».

Требования к оформлению

Структура работы

Структура курсовой работы определяется следующими компонентами:

- титульный лист (Приложение 1);
- содержание – включает перечень всех разделов и подразделов с указанием номеров страниц, с которых они начинаются;
- введение – содержит описание тематики курсовой работы, дается анализ предметной области и задачи, ставятся цель курсовой работы и формулируются задачи для успешного ее достижения;
- теоретическая часть – описывается предметная область, существующие решения и методы в рамках выбранной тематики;
- практическая часть – описывается структура разработанной программы, набор команд и функций, приводятся блок-схемы алгоритмов реализованных функций, структурные и функциональные схемы;
- заключение – обобщается результат проделанной работы, формулируется вывод, насколько реализованная программа соответствует тематике и заданию, была ли достигнута цель курсовой работы;
- список литературы – содержит перечисление всех источников, к которым прибегал студент во время выполнения курсовой работы, прежде всего, в теоретической части (рекомендуемый объём – 5-7 источников);
- приложение – особый раздел, в который помещается полный листинг всей программы. Если программа разбита на несколько файлов, то прикладывается полный листинг **всех** файлов.

Общий объем работы должен составлять от 15 до 30 страниц. Рекомендуется отвести под теоретическую часть не более 5-7 страниц, основной частью работы должна служить практическая составляющая.

Требования к оформлению текста

Текст отчета к курсовой работе должен быть оформлен на компьютере в электронном виде и распечатан на стандартных листах белой бумаги формата А4 (297х210 мм) на одной стороне, с полями слева 3 см, а с других сторон – по 2 см. Формат основного текста настраивается на вкладке «Главная»:

- **Шрифт Times New Roman;**
- **Размер шрифта (кегль) 14.**

В пункте меню «Абзац» выставляются следующие значения:

- **Выравнивание: по ширине;**
- **Отступ: слева 0, справа 0, первая строка – отступ 1,25 см;**
- **Интервал: перед 0, после 0, междустрочный – 1,5 строки.**

Для удобства обозначения содержания и чтения отчета, его текст следует разбить на главы, выделяемые нумерованными заголовками. Введение, заключение и список литературы ***не нумеруются***.

Каждая ***глава*** (не путать с подглавами) начинается с новой страницы.

Ссылки на литературу указываются в тексте в квадратных скобках номерами в порядке появления, которые расшифровываются в списке использованной литературы.

Нумерация рисунков и таблиц должна быть порядковой, вестись арабскими цифрами. Пример: «Рис. Х. Название рисунка», где Х – порядковый номер рисунка в главе. Точка в конце подписи к рисунку не ставится.

Требования к оформлению алгоритмов

Блок схема алгоритма составляется из блоков определенной формы. На изображение схем алгоритмов существует ГОСТ

19.701-90, согласно которому каждой группе действий соответствует блок особой формы.

При разработке алгоритма каждое действие должно обозначаться соответствующим блоком, показывая их последовательность линиями со стрелками на конце. Важно соблюдать основные правила составления блок-схем, например, чтобы линия входила сверху, а выходила снизу, что актуально для большинства видов блоков (кроме циклов и ветвлений). Если линии идут не слева направо и не сверху вниз, то стрелка в конце линии обязательна. В случае, когда схема алгоритма не умещается на листе, необходимо использовать специальные блоки-соединители. В блок схеме приветствуются и положительно оцениваются комментарии.

Рекомендуемая литература и ссылки

Рекомендуемая к изучению литература

1. Керниган Б., Ритчи Д., Язык программирования Си.\ Пер. с англ., 3-е изд., испр. – СПб.: «Невский диалект», 2001. – 352 с: ил.
2. Страуструп Б., Язык программирования C++ = The C++ Programming Language / Пер. с англ. — 3-е изд. — СПб.; М.: Невский диалект — Бином, 1999. — 991 с. — 3000 экз. — ISBN 5-7940-0031-7 (Невский диалект), ISBN 5-7989-0127-0 (Бином), ISBN 0-201-88954-4 (англ.).
3. Страуструп Б., Программирование: принципы и практика использования C++, испр. изд.: Пер. с англ. — М.: ООО “И.Д. Вильямс”, 2011. — 1248 с.: ил. — Парал. тит. англ. ISBN 978-5-8459-1705-8 (рус.)
4. Andrew Koenig and Barbara E. Moo. 2000. Accelerated C++: practical programming by example. Addison-Wesley Longman Publishing Co., Inc., USA.

Полезные ссылки

1. Интернет-справочник по языкам C и C++, URL: https://en.cppreference.com/w/

2. C++ глоссарий Бьярна Страуструпа, URL: https://www.stroustrup.com/glossary.html
3. Официальный справочник по языку C++ от Microsoft, URL: https://learn.microsoft.com/ru-ru/cpp/cpp/?view=msvc-170
4. Онлайн-туториал по языкам C и C++, ресурс cprogrm-ming.com, URL: https://www.cprogramming.com/tutorial/c++-tutorial.html
5. Онлайн-туториал по языку C++, ресурс geeksforgeeks.org, URL: https://www.geeksforgeeks.org/c-plus-plus/
6. Онлайн-туториал по языку C++ с высоким уровнем наглядности и графическими материалами, ресурс www3.ntu.edu.sg, URL: https://www3.ntu.edu.sg/HOME/EHCHUA/PROGRAMMING/cpp/cp0_Introduction.html
7. Онлайн-туториал по языкам C и C++ для продвинутых, ресурс tenouk.com, URL: https://www.tenouk.com/Sitemap.html
8. Скачать Code::Blocks с официального сайта, URL: https://www.codeblocks.org/downloads/binaries/
9. Скачать компилятор GCC и набор MinGW для Windows, URL: https://winlibs.com/

Список литературы

1. Керниган Б., Ритчи Д., Язык программирования Си.\ Пер. с англ., 3-е изд., испр. – СПб.: «Невский диалект», 2001. – 352 с: ил.
2. Страуструп Б., Язык программирования C++ = The C++ Programming Language / Пер. с англ. — 3-е изд. — СПб.; М.: Невский диалект — Бином, 1999. — 991 с. — 3000 экз. — ISBN 5-7940-0031-7 (Невский диалект), ISBN 5-7989-0127-0 (Бином), ISBN 0-201-88954-4 (англ.).
3. Страуструп Б., Программирование: принципы и практика использования C++, испр. изд.: Пер. с англ. — М.: ООО “И.Д. Вильямс”, 2011. — 1248 с.: ил. — Парал. тит. англ. ISBN 978-5-8459-1705-8 (рус.)

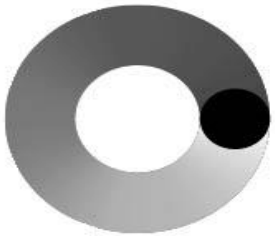
4. Running the Winsock Client and Server Code Sample, GitHub – MicrosoftDocs/win32 (Online), URL: https://github.com/MicrosoftDocs/win32/blob/docs/desktop-src/WinSock/finished-server-and-client-code.md

Приложение 1

МИНИСТРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное автономное
образовательное учреждение
высшего образования
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГАОУ ВО «Московский Политех»)



МОСКОВСКИЙ ПОЛИТЕХ

Факультет: «Машиностроение»

Кафедра: «Автоматика и управление»

Дисциплина: «Программирование и основы алгоритмизации»

Курсовая работа

Тема: «…»

Группа …
Выполнил:
…
Проверил:
…

Москва 2023

Приложение 2

```
#include <iostream>
#include <ws2tcpip.h>

// размера буфера принимаемого сообщения
#define DEFAULT_BUFLEN 512
// порт, используемый для ожидания подключения
#define DEFAULT_PORT "27015"

using namespace std;

int main() {
// структура с информацией о реализации сокетов
    WSADATA wsaData;
// число принятых от клиента байт
    int iResult;
// результирующая структура данных о подключении
    struct addrinfo *result = NULL;
// исходная структура данных о подключении
    struct addrinfo hints;
// буфер для записи получаемого сообщения
    char recvbuf[DEFAULT_BUFLEN];
// размера буфера принимаемого сообщения
    int recvbuflen = DEFAULT_BUFLEN;
// объявление сокета, инициализация ошибкой
    SOCKET ClientSocket = INVALID_SOCKET;

// инициализация сокета Winsock
    WSAStartup(MAKEWORD(2,2), &wsaData);
// наполнение структуры hints
    ZeroMemory(&hints, sizeof(hints));
    hints.ai_family = AF_UNSPEC;
    hints.ai_socktype = SOCK_STREAM;
    hints.ai_protocol = IPPROTO_TCP;
    hints.ai_flags = AI_PASSIVE;
// создание структуры result
    getaddrinfo(NULL, DEFAULT_PORT, &hints, &result);
// инициализация сокета на стороне сервера
    SOCKET ServerSocket = socket(result->ai_family, result->ai_socktype,
                                                        result->ai_protocol);
// закрепление сокета на адресе
    bind(ServerSocket, result->ai_addr, (int)result->ai_addrlen);
// освобождение структуры result
```

```
    freeaddrinfo(result);
// начало прослушивания сокета
    listen(ServerSocket, SOMAXCONN);
// инициализация сокета клиента, если есть подключение
//(блокирует выполнение программы, ожидание подключения)
    ClientSocket = accept(ServerSocket, NULL, NULL);
// закрытие сокета сервера
    closesocket(ServerSocket);

// цикл обработки получаемых сообщений
    do {
// запись в буфер, количество байт в iResult
        iResult = recv(ClientSocket, recvbuf, recvbuflen, 0);
// если получено сообщение
        if (iResult > 0) {
// вывод количества полученных байт
            cout << "Bytes received: " << iResult << endl;
        }
// если получено пустое сообщение
        else if (iResult == 0)
// завершение соединения
            cout << "Connection closing...\n";
// пока есть сообщение
    } while (iResult > 0);

// завершение передачи данных, остановка и очистка сокета
    shutdown(ClientSocket, SD_SEND);
    closesocket(ClientSocket);
    WSACleanup();
// ожидание ввода для завершения программы
    getchar();
// завершение работы программы с кодом 0
    return 0;
}
```

Приложение 3

```
#include <iostream>
#include <ws2tcpip.h>

// IP-адрес сервера
#define IP_ADDRESS "localhost"
// порт, используемый для подключения к серверу
#define DEFAULT_PORT "27015"

using namespace std;

int main()
{
// структура с информацией о реализации сокетов
    WSADATA wsaData;
// результирующая структура данных о подключении
    struct addrinfo *result = NULL;
// структура-счётчик, используемая в цикле
    struct addrinfo *ptr = NULL;
// исходная структура данных о подключении
    struct addrinfo hints;
// передаваемое сообщение
    const char *sendbuf = "123456";
// объявление сокета, инициализация ошибкой
    SOCKET ConnectSocket = INVALID_SOCKET;

// инициализация сокета Winsock
    WSAStartup(MAKEWORD(2,2), &wsaData);
// наполнение структуры hints
    ZeroMemory(&hints, sizeof(hints));
    hints.ai_family = AF_UNSPEC;
    hints.ai_socktype = SOCK_STREAM;
    hints.ai_protocol = IPPROTO_TCP;
    hints.ai_flags = AI_PASSIVE;
// создание структуры result
    getaddrinfo(IP_ADDRESS, DEFAULT_PORT, &hints, &result);

// попытка подключиться к одному из адресов в списке
    for(ptr=result; ptr != NULL ;ptr=ptr->ai_next) {
// создание сокета для подключения к серверу
        ConnectSocket = socket(ptr->ai_family, ptr->ai_socktype, ptr->ai_protocol);
// подключение к серверу
        connect( ConnectSocket, ptr->ai_addr, (int)ptr->ai_addrlen);
```

```
// окончание цикла после первой итерации
        break;
    }
// освобождение структуры result
    freeaddrinfo(result);
// если подключение не удалось
    if (ConnectSocket == INVALID_SOCKET) {
// вывод сообщения об ошибке
        cout << "Unable to connect!" << endl;
// очистка сокета Winsock
        WSACleanup();
// завершение программы с кодом 4
        return 4;
    }
// отправка сообщения из буфера
    send(ConnectSocket, sendbuf, (int)strlen(sendbuf), 0);
// вывод длины отправленного сообщения
    cout << "Bytes Sent: " << (int)strlen(sendbuf) << endl;

// завершение передачи данных, остановка и очистка сокета
    shutdown(ConnectSocket, SD_SEND);
    closesocket(ConnectSocket);
    WSACleanup();
// ожидание ввода для завершения программы
    getchar();
// завершение работы программы с кодом 0
    return 0;
}
```

Приложение 4

```
#include <iostream>
#include <windows.h>

using namespace std;

// функция перемещения каретки
void gotoxy(int x, int y) {
// объявление структуры, содержащей координаты каретки
    COORD coord;
// заполнение координат каретки
    coord.X = x;
    coord.Y = y;
// перенос каретки
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}

int main() {
// вывод «Hello world!»
    cout << "Hello world!" << endl;
// вызов функции gotoxy (выше)
    gotoxy(1, 1);
// ожидание 2 секунды
    Sleep(2000);
// вывод «Ok by now. »
    cout << "Ok by now." << endl;
// завершение работы программы с кодом 0
    return 0;
}
```

Приложение 5

```
#include <iostream>
#include <thread>
#include <windows.h>
// максимальное число итераций цикла for в функции act
#define MAX_COUNTER 10

using namespace std;
// описание класса
class PrinterClass {
public:
// метод act класса PrinterClass, которая будет циклично выводить номер итерации
      void act(const int &counter = MAX_COUNTER, int *adder = nullptr,
const bool &must_close = false) {
// вывод сообщения, когда начинается дочерний поток
         cout << " --- Child thread started ---" << endl;
// начало цикла с выводом сообщения в каждой итерации
         for(int x; x < counter; x++) {
// вывод сообщения с номером текущей итерации
         cout << "\rIteration number #" << x << endl;
// блокирующее дочерний поток ожидание 500 мс
         Sleep(500);
// если adder содержит положительное число, добавляем его к номеру текущей итерации и
// обнуляем adder
         if (*adder > 0) {
            x+=*adder;
            *adder = 0;
         }
// если флаг must_close = true, означает сигнал от родительского потока, о закрытии дочернего
         if (must_close) {
// завершение цикла for
            break;
         }
       }
// вывод сообщения о завершении дочернего потока
         cout << " --- Child thread closed ---" << endl;
// завершение функции в дочернем потоке
         return;
```

```
// также может быть вызвана функция exit(), завершающая выполнение всей программы
// из дочернего потока
        }
};

int main() {
// объявление строковой переменной для обработки пользовательского ввода в родительском
// потоке
    string command = "";
// целочисленные переменные максимального количества итераций и пропуска итераций
    int max_count = 0, adder = 0;
// флаг для завершения дочернего потока
    bool close_child = false;
// вывод приветственного сообщения
    cout << "Enter ONLY numbers:\n\
        first, enter how many iterations of threaded cycle must be done;\n\
        then you can enter number to skip some iterations;\n\
        to quit the program enter skip number > first number and press ENTER.\n";
// чтение пользовательского ввода
    cin >> max_count;
// исключение из потока ввода символа переноса строки чтобы избежать реакции на getchar()
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
// объявление объекта класса PrinterClass с именем printer1
        PrinterClass printer1;
// объявление объекта класса thread с именем cycle (дочерний поток – метод act)
        thread cycle(&PrinterClass::act, ref(printer1), max_count, &adder, close_child);
        //cycle.detach();
// бесконечный цикл в родительском потоке, обрабатывающий ввод пользователя
        while(true) {
// ввод команды, в данном случае ожидаются только числа для пропуска количества итераций
        cin >> command;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
// эхо-вывод
        cout << "User input:\n\t" << command << endl;
// преобразование пользовательского ввода в число
        adder = stoi(command);
// если число итерация для пропуска меньше, чем максимальное количество итераций
        if(adder < max_count) {
```

```
// продолжение
            continue;
        }
//в ином случае
        else {
// передача дочернему потоку сообщение о необходимости завершения
            close_child = true;
// подключение родительского потока к дочернему, родительский блокируется
до окончания
// выполнения дочернего
            cycle.join();
// завершение цикла while в родительском процессе
            break;
        }
    }
// ожидание нажатия ENTER до завершения программы
        getchar();
// завершение работы программы с кодом 0
        return 0;
}
```