ВВЕДЕНИЕ

Магазин - юридическое лицо, которое выполняет функции розничной торговли. Торговля - обширнейшая область предпринимательской деятельности и сфера приложения труда - получила в последние годы новые импульсы своего развития. В нее влилось много новых предприимчивых людей, в ряде случаев коренным образом поменявших профессию и жизненные ориентиры.

Торговля является одной из крупнейших отраслей экономики любой страны как по объему деятельности, так и по численности занятого в ней персонала, а предприятия этой отрасли являются наиболее массовыми. Деятельность торговых предприятий связана с удовлетворением потребностей каждого человека, подвержена влиянию множества факторов и охватывает широкий спектр вопросов организационно-технологического, экономического и финансового характера, требующих повседневного решения.

Программирование можно рассматривать как искусство, науку, ремесло. Программирование — это искусство получения ответов от машины. Для этого в узком смысле нужно составить специальный код для технического устройства, а в широком — разработать программы на языках программирования, т. е. не просто составить код, а выполнить интеллектуальную работу по составлению высокоразумных программ для решения различных задач во всех сферах человеческой деятельности.

Целью курсовой работы является закрепление знаний и навыков, полученных в процессе изучения дисциплины. Получение практических навыков по реализации программных средств на языке программирования Си.

При подготовке и написании курсовой работы перед студентами ставятся следующие задачи:

* рассмотрения история возникновения и этапы развития языков;
* проведение классификация языков программирования и средств разработки;
* анализ предметной области «магазин»;
* разработка консольного приложения.

Объектом курсовой работы является процесс создание консольного приложения для предметной области «магазин»

Предмет исследования различные точки зрения разработки консольного приложения на языке СИ.

Разработка консольного приложения посредством использования языка СИ. Именно из-за этого проведения классификации языков программирования, изучение теоретических этапов их развития и изучение языка СИ.

1. История и этапы развития программирования

С глубокой древности известны попытки создать устройства, ус­коряющие и облегчающие процесс вычислений. Еще древние греки и римляне применяли приспособление, подобное счетам, — абак. Такие устройства были известны и в странах Древнего Востока. В XVM в. немецкие ученые В. Шиккард (1623), Г.Лейбниц (1673) и французский ученый Б. Паскаль (1642) создали механические вы­числительные устройства — предшественники всем известного арифмометра. Вычислительные машины совершенствовались в те­чение нескольких веков. Но при этом не применялось понятие «программа и программирование» [1]

В 1854 г. английский математик Джордж Буль опубликовал кни­гу «Законы мышления», в которой развил алгебру высказываний —Булеву алгебру. На ее основе в начале 80-х гг. XIX в. построена тео­рия релейно-контактных схем и конструирования сложных дискрет­ных автоматов. Алгебра логики оказала многогранное влияние на развитие вычислительной техники, являясь инструментом разработ­ки и анализа сложных схем, инструментом оптимизации большого числа логических элементов, из многих тысяч которых состоит со­временная ЭВМ.

Идеи Ч. Бэббиджа реализовал американский ученый Г. Холлерит, который с помощью построенной счетно-аналитической маши­ны и перфокарт за три года обработал результаты переписи населе­ния в США по состоянию на 1890 г. В машине впервые было ис­пользовано электричество. В 1896 г. Холлеритом была основана фирма по выпуску вычислительных перфорационных машин и пер­фокарт. [1]

С появлением цифровых программно-управляемых машин родилась новая область прикладной математики — программирование. Как область науки и профессия она возникла в 1950-х гг. Первона­чально программы составлялись вручную на машинных языках (в машинных кодах). Программы были громоздки, их отладка — очень трудоемка. Для упрощения приемов и методов составления и отладки программ были созданы мнемокоды, по структуре близкие к машинному языку и использующие символьную адресацию. Ассемблеры переводили программу, записанную в мнемокоде, на машинный язык и, расширенные макрокомандами, используются и в настоящее время. Далее были созданы автокоды, которые можно применять на различных машинах, и позволившие обмениваться программами. Автокод — набор псевдокоманд для решения специализиро­ванных задач, например, научных или инженерных. Для таких задач имеется развитая библиотека стандартных программ. [2]

В основе вторых обычно лежит методология, определяющая совокупность методов, используемых на разных этапах разработки.

1 этап – «стихийное» программирование. Первые программы имели простейшую структуру, состояли из программы на машинном языке и обрабатываемых ею данных. [3]

2 этап – структурный подход к программированию - представляет собой совокупность рекомендуемых технологических приемов, охватывающих выполнение всех этапов разработки ПО. В основе – декомпозиция сложных систем с целью последующей реализации в виде отдельных небольших подпрограмм.

3 этап – объектный подход к программированию Объектно-ориентированное программирование - технология создания сложного ПО, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного типа (класса), а классы образуют иерархию с наследованием свойств.

4 этап – компонентный подход и CASE-технологии. Компонентный подход - построение ПО из отдельных компонентов – физически отдельно существующих частей программного обеспечения, которые взаимодействуют между собой через стандартизованные двоичные интерфейсы.

1. Классификация языков программирования и сред разработки

Все языки программирования можно условно классифицировать по некоторым основным признакам. Ниже приведена краткая классификация языков программирования:

По назначению:

* процедурные;
* непроцедурные.

Процедурная (алгоритмическая) программа — это система формальных предписаний, направленных на решение конкретных задач, которые выполняет ЭВМ.

Непроцедурное программирование представляет собой прямо противоположную методологию (парадигму) разработки, когда компьютеру ставится определённая задача в более или менее общем виде, без написания формализованного алгоритма, который отдаётся на усмотрение машины.

Процедурные языки отличаются тем, на кого в первую очередь направлены: на машину или человека. Они подразделяются на две категории:

* низкого уровня (или машинно-ориентированные);
* высокого уровня.

Низкоуровневые языки ориентированы на конкретные компьютерные архитектуры и учитывают их особенности. Они являются следующим уровнем после машинного кода и следуют конкретным указаниям, исходящим от процессора.

К языкам низкого уровня относится программирование в машинных кодах, ассемблер, макроассемблер.

Высокоуровневые языки более понятны для человека. Соответственно, программировать на них гораздо проще и удобнее. В них не учитываются особенности конкретных типов процессоров, поэтому такие программы легко переносятся с одной платформы на другую.

Языки высокого уровня делятся на универсальные и проблемно-ориентированные.

Наиболее распространенные универсальные языки C#, C++, Basic, Pascal (Delphi) используются для разработки Windows-приложений. Большой вклад в программирование на начальных этапах внесли языки Fortran, Cobol, Algol, C и др.

Непроцедурные языки включают две основные языковые группы:

* объектно-ориентированные;
* декларативные.

Объектно-ориентированные состоят из ряда независимых объектов, которые функционируют как отдельные компьютеры. С помощью этих блоков можно решать задачи, не вникая во «внутреннюю кухню» их работы.  
 Работа с декларативным языком подразумевает установление взаимосвязей между исходными информационными структурами и свойствами конечного результата. При этом в нём не существует понятия «команда», а программист не создаёт алгоритмы. [4]

Декларативные языки подразделяются на два семейства:

* логические;
* функциональные.

Логическое программирование описывает проблемы в виде фактов и формул, а система решает их посредством механизмов логического вывода.

Функциональное, в свою очередь, формулирует задачу как совокупность определённых функций.

При разработке программных средств используется в той или иной мере компьютерная поддержка процессов разработки и сопровождения ПС [16.1]. Это достигается путем представления хотя бы некоторых программных документов ПС (прежде всего, программ) на компьютерных носителях данных (например, на дискетах) и предоставлению в распоряжение разработчика ПС специальных ПС или включенных в состав компьютера специальных устройств, созданных для какой-либо обработки таких документов. В качестве такого специального ПС можно указать компилятор с какого-либо языка программирования. Компилятор избавляет разработчика ПС от необходимости писать программы на языке компьютера, который для разработчика ПС был бы крайне неудобен, - вместо этого он составляет программы на удобном ему языке программирования, которые соответствующий компилятор автоматически переводит на язык компьютера. В качестве специального устройства, поддерживающего процесс разработки ПС, можно указать, например, эмулятор какого-либо языка. Эмулятор позволяет выполнять (интерпретировать) программы на языке, отличном от языка компьютера, поддерживающего разработку ПС, например, на языке компьютера, для которого эта программа предназначена.

ПС, предназначенное для поддержки разработки других ПС, будем называть программным инструментом разработки ПС, а устройство компьютера, специально предназначенное для поддержки разработки ПС, будем называть аппаратным инструментом разработки ПС.

Инструменты разработки ПС могут использоваться в течение всего жизненного цикла ПС для работы с разными программными документами. Так текстовый редактор может использоваться для разработки практически любого программного документа. С точки зрения функций, которые инструменты выполняют при разработке ПС, их можно разбить на следующие четыре группы:

* редакторы,
* анализаторы,
* преобразователи,
* инструменты, поддерживающие процесс выполнения программ.

Редакторы поддерживают конструирование (формирование) тех или иных программных документов на различных этапах жизненного цикла. Как уже упоминалось, для этого можно использовать один какой-нибудь универсальный текстовый редактор. Однако, более сильную поддержку могут обеспечить специализированные редакторы: для каждого вида документов - свой редактор. В частности, на ранних этапах разработки в документах могут широко использоваться графические средства описания (диаграммы, схемы и т.п.). В таких случаях весьма полезными могут быть графические редакторы. На этапе программирования (кодирования) вместо текстового редактора может оказаться более удобным синтаксически управляемый редактор, ориентированный на используемый язык программирования. [4]

Анализаторы производят либо статическую обработку документов, осуществляя различные виды их контроля, выявление определенных их свойств и накопление статистических данных (например, проверку соответствия документов указанным стандартам), либо динамический анализ программ (например, с целью выявление распределения времени работы программы по программным модулям).

Преобразователи позволяют автоматически приводить документы к другой форме представления (например, форматеры) или переводить документ одного вида к документу другого вида (например, конверторы или компиляторы), синтезировать какой-либо документ из отдельных частей и т.п.

Инструменты, поддерживающие процесс выполнения программ, позволяют выполнять на компьютере описания процессов или отдельных их частей, представленных в виде, отличном от машинного кода, или машинный код с дополнительными возможностями его интерпретации. Примером такого инструмента является эмулятор кода другого компьютера. К этой группе инструментов следует отнести и различные отладчики. По существу, каждая система программирования содержит программную подсистему периода выполнения, которая выполняет программные фрагменты, наиболее типичные для языка программирования, и обеспечивает стандартную реакцию на возникающие при выполнении программ исключительные ситуации (такую подсистему мы будем называть исполнительной поддержкой). Такую подсистему также можно рассматривать как инструмент данной группы.

Совокупность инструментальных сред можно разбивать на разные классы, которые различаются значением следующих признаков:

* ориентированность на конкретный язык программирования,
* специализированность,
* комплексность,
* ориентированность на конкретную технологию программирования,
* ориентированность на коллективную разработку,

Ориентированность на конкретный язык программирования (языковая ориентированность) показывает: ориентирована ли среда на какой-либо конкретный язык программирования (и на какой именно) или может поддерживать программирование на разных языках программирования. В первом случае информационная среда и инструменты существенно используют знание о фиксированном языке (глобальная ориентированность), в силу чего они оказываются более удобным для использования или предоставляют дополнительные возможности при разработке ПС. Но в этом случае такая среда оказывается не пригодной для разработки программ на другом языке. Во втором случае инструментальная среда поддерживает лишь самые общие операции и, тем самым, обеспечивает не очень сильную поддержку разработки программ, но обладает свойством расширения (открытости). Последнее означает, что в эту среду могут быть добавлены отдельные инструменты, ориентированные на тот или иной конкретный язык программирования, но эта ориентированность будет лишь локальной (в рамках лишь отдельного инструмента).

Специализированность инструментальной среды показывает: ориентирована ли среда на какую-либо предметную область или нет. В первом случае информационная среда и инструменты существенно используют знание о фиксированной предметной области, в силу чего они оказываются более удобными для использования или предоставляют дополнительные возможности при разработке ПС для этой предметной области. Но в этом случае такая инструментальная среда оказывается не пригодной или мало пригодной для разработки ПС для других предметных областей. Во втором случае среда поддерживает лишь самые общие операции для разных предметных областей. Но в этом случае такая среда будет менее удобной для конкретной предметной области, чем специализированная на эту предметную область.

Комплексность инструментальной среды показывает: поддерживает ли она все процессы разработки и сопровождения ПС или нет. В первом случае продукция этих процессов должна быть согласована. Поддержка инструментальной средой фазы сопровождения ПС, означает, что она должна поддерживать работу сразу с несколькими вариантами ПС, ориентированными на разные условия применения ПС и на разную связанную с ним аппаратуру, т.е. должна обеспечивать управление конфигурацией ПС [16.1, 16.3].

Ориентированность на конкретную технологию программирования показывает: ориентирована ли инструментальная среда на фиксированную технологию программирования [16.2] либо нет. В первом случае структура и содержание информационной среды, а также набор инструментов существенно зависит от выбранной технологии (технологическая определенность). Во втором случае инструментальная среда поддерживает самые общие операции разработки ПС, не зависящие от выбранной технологии программирования.

Ориентированность на коллективную разработку показывает: поддерживает ли среда управление (management) работой коллектива или нет. В первом случае она обеспечивает для разных членов этого коллектива разные права доступа к различным фрагментам продукции технологических процессов и поддерживает работу менеджеров [16.1] по управлению коллективом разработчиков. Во втором случае она ориентирована на поддержку работы лишь отдельных пользователей.

1. Анализ предметной области «Магазин»

Предметная область информационной системы - это материальная система или система, характеризующая элементы материального мира, информация о которой хранится и обрабатывается.

Анализ должен заканчиваться подробным описанием информации об объектах предметной области, которая требуется для решения конкретных задач с кратким описанием алгоритмов их решения, описанием выходных документов, которые должны генерироваться в системе.

Магазин, устар. лавка — предприятие розничной торговли, размещённое в стационарном здании, оборудованном для продаж товаров и оказания услуг. Здание или часть здания, где располагается современный магазин, обычно включает как собственно торговый зал, так и подсобные, административно-бытовые помещения и склады.

Виды деятельности магазина:

* розничная куплю продажи товаров;
* осуществлению всех видов внешнеэкономической деятельности;
* оказание помощи покупателю в совершении покупки и при её использовании;

Так как куплю-продажи — это взаимодействия клиентов с продавцами и предоставление какого-либо товара, то нужно иметь средство оптимизации хранение информации о клиентах и имеющимся товаре для более удобной работы.

Для этого будет разработана программа консольного приложения с следующим функционалом:

* возможность добавлять клиентов;
* возможность удалять клиентов;
* возможность чтение клиентов;
* возможность поиска клиента по фамилии;
* возможность добавлять товар;
* возможность удалять товар;
* возможность чтения товара;
* возможность поиска товара по названию.

На основе задач определяются сущности данных которые необходимо обрабатывать в программе:

Сущности товар состоит из:

* наименования;
* стоимости.

Сущности клиент состоит из:

* фамилия;
* возраст;
* номер телефона.

Данная программа будет предоставлять интерфейс через консольную команду и разработана на языке программирование С++.

1. Разработка программного обеспечения

Для работы программы запускается батник после его открытия видно главное меню программы.

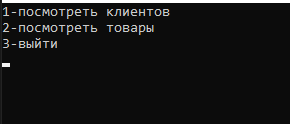


Рисунок 1 - главное меню программы

Код работы главного меню представлен ниже, реализации следующих имеющихся меню имеет подобную кодовую базу.

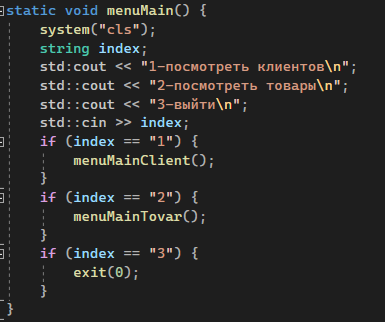


Рисунок 2 - код главного меню программы

С главного меню можно перейти в меню 1-посмотреть клиентов, которое предоставлено ниже

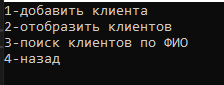


Рисунок 3 - меню клиента

С главного меню можно перейти в меню 2-посмотреть товары, которое предоставлено ниже

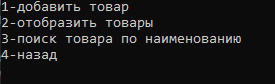


Рисунок 3 - меню товаров

Далее рассмотрим функционал добавить клиента для этого в меню клиента нажмем на 1 и нас перенес в меню где нужно ввести данные о клиенте, после чего они будут сохранны в файл, результат работы представлен ниже.

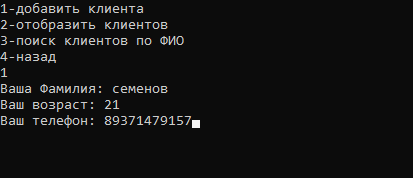


Рисунок 4 – сохранение клиента

Код работы сохранение клиента в файл предоставлен ниже, сохранения товара имеет подобную кодовую базу.

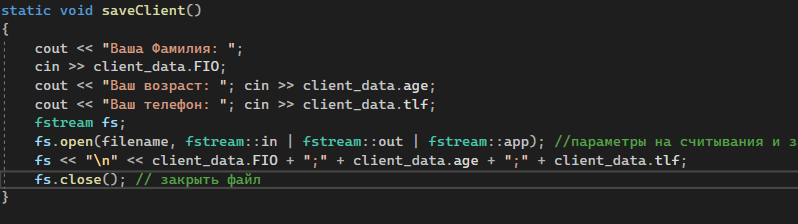


Рисунок 5 – код сохранение клиента

Далее, чтобы посмотреть список клиентов в меню клиентов нажимается кнопка 2, результат работы представлен ниже.

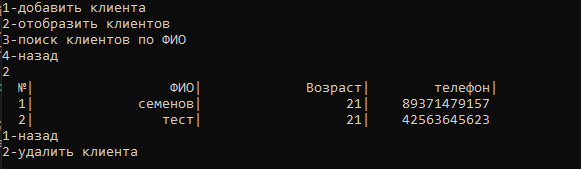


Рисунок 6 – отображение клиентов

Для удаления клиента во время просмотра клиентов нажимаем 2 и требуется ввести № строки которую нужно удалить.

Удаления элемента в файле представлен отдельной функцией, которая принимает название файла, код функции представлен ниже.

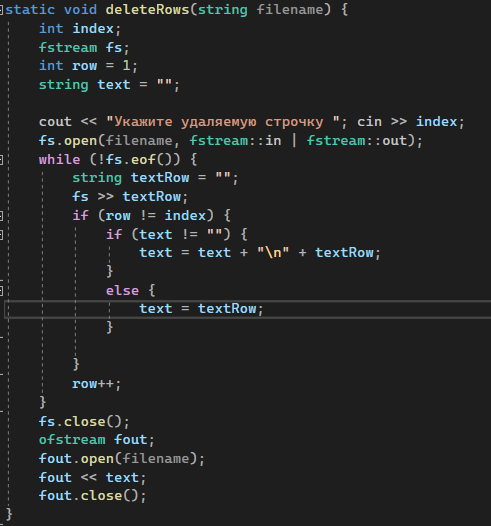


Рисунок 7 – удаление элементов в файле

Далее, чтобы найти клиента по фамилии в меню клиента нужно нажать на 3 и ввести фамилию результат работы представлен ниже.

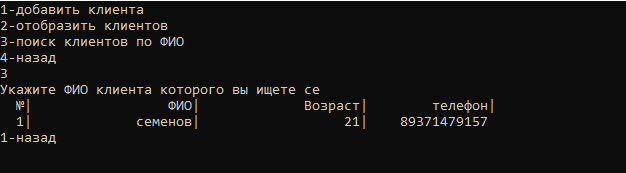


Рисунок 8 – поиск клиента по фамилии

Данный поиск ищет не только по полному совпадению фамилии, но и может найти фамилию по подстроке код реализации представлен ниже.

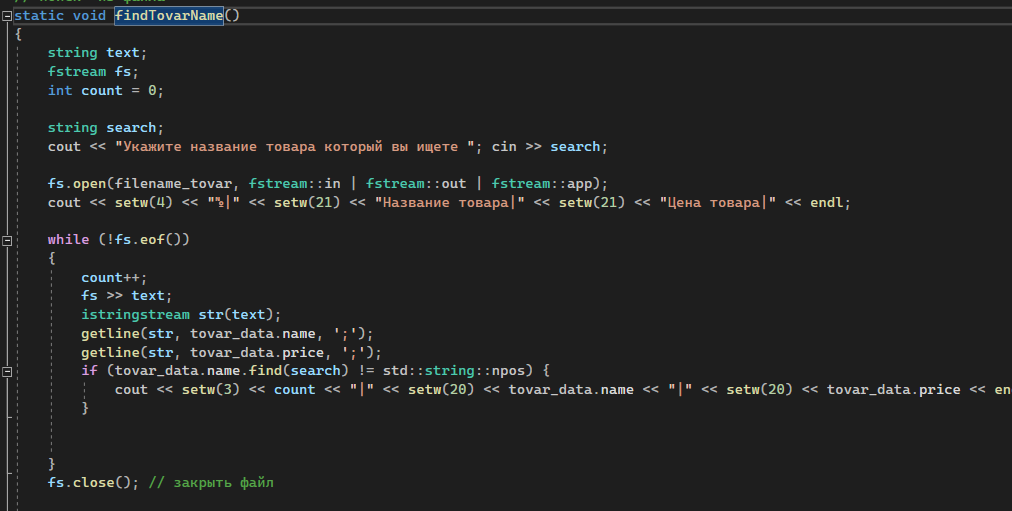


Рисунок 9 – код поиск клиента по фамилии

Далее рассмотрим функционал добавить товара для этого в меню товара нажмем на 1 и нас перенес в меню где нужно ввести данные о товаре, после чего они будут сохранны в файл, результат работы представлен ниже.

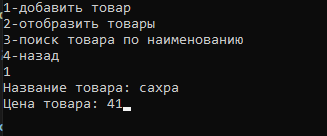


Рисунок 10 – сохранение товара

Далее, чтобы посмотреть список товаров в меню товаров нажимается кнопка 2, результат работы представлен ниже.

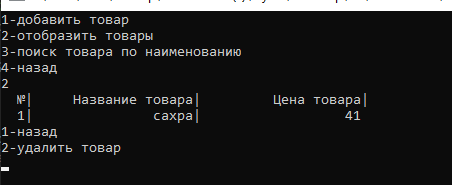


Рисунок 11 – отображение товара

Для удаления товара во время просмотра товара нажимаем 2 и требуется ввести № строки которую нужно удалить.

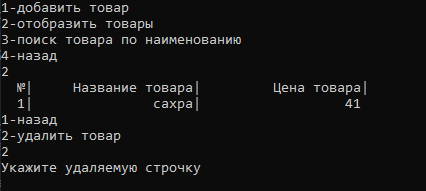


Рисунок 12 – удаление товара

Далее, чтобы найти товар по названию в меню товар нужно нажать на 3 и ввести наименование товара результат работы представлен ниже.

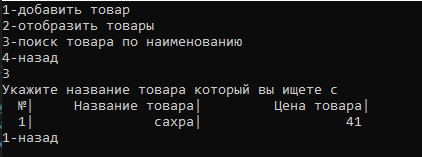


Рисунок 13 – поиск товара по названию

Заключение

Результатами данной курсовой работы является разработанное программное обеспечение в предметной области «Магазин». В процессе выполнения курсовой работы была достигнута цель и решены поставленные задачи.

В ходе выполнения курсовой работы были выполнены следующие задачи:

* изучена история развития языков программирование и определены основные этапы развития языков программирования;
* определена классификация языков программирования и изучены средства разработки программного обеспечения;
* проведен анализ предметной области «Магазин», поставлены требуемые задачи от ожидаемого программного обеспечение в предметной области «Магазин», определены сущности и их атрибуты программном обеспечении в предметной области «Магазин», определено что программное обеспечение в предметной области «Магазин» будет разрабатываться на языке с++;
* разработано программное обеспечение предметной области «Магазин» с помощью языка с++, рассмотрен весь функционал написанного программное обеспечение предметной области «Магазин», также изучены особенности кодовой базы в программном обеспечении предметной области «Магазин».

Была выполнена цель курсовой работы закреплены знания и навыки, в изучении дисциплины. Получены практические навыки по реализации программных средств на языке программирования Си.

Список использованных источников

1. Хвостов А. А. Разработка интерфейса оператора технологического процесса на языке С++ с использованием его математической модели / А.А. Хвостов, В.К. Битюков, С.Г. Тихомиров. - Воронеж : Воронежский университет инженерных технологий, 2014. - 116 с. - ISBN 978-5-00032-048-8. - URL: https://ibooks.ru/bookshelf/344487/reading (дата обращения: 14.01.2022). - Текст: электронный.
2. Немцова Т.И. Программирование на языке высокого уровня. Программирование на языке С++ / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев. - Москва : Форум, 2021. - 512 с. - ISBN 978-5-8199-0699-6. - URL: https://ibooks.ru/bookshelf/361544/reading (дата обращения: 14.01.2022). - Текст: электронный.
3. Русакова З. Н. Структуры данных в С++ : учебное пособие / З.Н. Русакова, И.В. Рудаков. - Москва : МГТУ им. Н.Э. Баумана, 2020. - 157 с. - ISBN 978-5-7038-5256-9. - URL: https://ibooks.ru/bookshelf/374738/reading (дата обращения: 14.01.2022). - Текст: электронный.
4. Тюгашев А. Языки программирования: Учебное пособие. Стандарт третьего поколения. — (Серия «Учебник для вузов»). / А. Тюгашев. - Санкт-Петербург : Питер, 2021. - 336 с. - ISBN 978-5-4461-9407-0. - URL: https://ibooks.ru/bookshelf/378758/reading (дата обращения: 14.01.2022). - Текст: электронный.
5. Нужина, О. А. История : учебное пособие / О. А. Нужина. — Сочи : СГУ, 2019. — 54 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/147800 (дата обращения: 14.01.2022). — Режим доступа: для авториз. пользователей.