Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Санкт-Петербургский государственный университет промышленных технологий и дизайна»

|  |
| --- |
| **Институт информационных технологий и автоматизации** |

|  |  |
| --- | --- |
| Кафедра: | Цифровых и аддитивных технологий |
| Направление подготовки: | 09.03.03 Прикладная информатика |
| Профиль подготовки: | Цифровые технологии в финансах |

|  |
| --- |
| **КУРСОВАЯ РАБОТА**  **(Курсовой проект)** |

|  |  |  |
| --- | --- | --- |
| на тему |  | Автомобильные перевозки. Отдел продаж |
|  | | |
| по дисциплине |  | Алгоритмизация и программирование |
|  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Исполнитель - студент учебной группы: | |  | 2-МД-21 |
|  | |  | *(группа)* | |
|  | |  | Коровашев Дмитрий Константинович | |
|  | |  | | *(Ф.И.О./ подпись)* | |

|  |  |  |
| --- | --- | --- |
| Руководитель курсовой работы: |  | Стяжкин Андрей Анатольевич |
|  | |  | *(должность, ученая степень / звание, Ф.И.О../ подпись)* |

Санкт-Петербург

2024

**Министерство науки и высшего образования Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**высшего образования**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ПРОМЫШЛЕННЫХ ТЕХНОЛОГИЙ И ДИЗАЙНА»**

|  |  |
| --- | --- |
| **Утверждаю** | **Институт информационных**  **технологий и автоматизации** |
| Заведующий кафедрой |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. В. Сошников | Кафедра Цифровых и аддитивных технологий |
| «\_\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |  |

**ЗАДАНИЕ**

**на курсовую работу (проект)**

Студенту Коровашеву Дмитрию Константиновичу

1. Тема курсовой работы (проекта) Автомобильные перевозки. Отдел продаж. Разработка автомобильного конфигуратора для автосалона Lada

2. Срок сдачи студентом законченной курсовой работы (проекта) 28 мая 2024 года

3. Исходные данные по курсовой работе (проекту). Изучить типы контейнеров и разработать программу на языке C++

4. Перечень подлежащих разработке в курсовой работе (проекте) вопросов или краткое содержание работы:

Введение. Цель курсовой работы: формирование навыков комплексного решения конкретных вычислительных задач, реализованных в одной из систем программирования.

Постановка задачи. В данной курсовой работе необходимо разработать и реализовать программу для отдела продаж автосалона Lada, предназначенную для автоматизации процесса продажи автомобилей. Целевая аудитория программы включает покупателей автомобилей, менеджеров по продажам и ИТ-специалистов, которые взаимодействуют с системой в соответствии со своими задачами и потребностями.

Изучение контейнеров. List - двусвязный список, каждый элемент которого содержит данные и два указателя: на предыдущий и следующий элементы. В программе используется для хранения конфигураций автомобилей и выбранных аксессуаров, что обеспечивает удобное добавление и удаление элементов. Map - ассоциативный контейнер, хранящий элементы в виде пар ключ-значение, где ключи уникальны и используются для быстрого доступа к значениям. Этот контейнер применяется для хранения данных о моделях автомобилей, типах кузова, комплектациях и цветах, что позволяет эффективно управлять и быстро находить необходимую информацию.

Математическая модель. Для описания предметной области задачи была использована математическая модель.

Блок-схема программы. Создание блок-схемы является ключевым шагом в проектировании структуры программы перед началом написания кода. Этот этап важен для планирования и организации будущего кода, чтобы он был логичным и структурированным.

Тестирование программы. Цель тестирования – убедиться, что программа выполняет свои функции согласно требованиям и обеспечивает необходимый уровень надёжности и производительности. Пользовательская история фокусируется на конечных пользователях, избегая технического языка, и предоставляет команде контекст для разработки.

Руководство пользователя. Основная цель руководства – предоставить пользователю необходимую информацию для успешного выполнения задач с программой.

Разработка структуры данных. Хорошо спроектированная структура данных оптимизирует производительность программы, обеспечивая быстрый доступ к данным, их эффективное изменение и удобное управление.

Разработка основного меню программы. В разработанной программе основное меню представляет собой интерактивный пользовательский интерфейс, который позволяет пользователям выбирать различные операции для управления заказами услуг.

Заключение. В ходе выполнения данной курсовой работы была разработана программа на языке программирования C++ для автосалона Lada, предназначенная для автоматизации процесса продажи автомобилей.

Дата выдачи задания 01.04.2024 Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Стяжкин А.А.\_ (дата) (подпись) (ФИО)

Задание принял к исполнению 01.04.2024 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Коровашев Д.К.

(дата) (подпись) (ФИО)

## СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc168070117)

[1 ПОСТАНОВКА ЗАДАЧИ 7](#_Toc168070118)

[2 ИЗУЧЕНИЕ КОНТЕЙНЕРОВ LIST И MAP 9](#_Toc168070119)

[3 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ 13](#_Toc168070120)

[4 БЛОК-СХЕМА ПРОГРАММЫ 16](#_Toc168070121)

[5 ТЕСТИРОВАНИЕ ПРОГРАММЫ 19](#_Toc168070122)

[5.1 User story 20](#_Toc168070123)

[6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 22](#_Toc168070124)

[7 РАЗРАБОТКА СТРУКТУРЫ ДАННЫХ 33](#_Toc168070125)

[8 РАЗРАБОТКА ОСНОВНОГО МЕНЮ ПРОГРАММЫ 41](#_Toc168070126)

[ЗАКЛЮЧЕНИЕ 43](#_Toc168070127)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 45](#_Toc168070128)

ВВЕДЕНИЕ

Сегодня информационные технологии играют важнейшую роль в жизни общества, так как они способствуют автоматизации и оптимизации процессов в различных сферах. Одними из ключевых элементов в области ИТ являются алгоритмизация и программирование, которые позволяют создавать эффективные алгоритмы для решения разнообразных задач. Алгоритмизация подразумевает разработку пошаговых инструкций для выполнения задач, тогда как программирование представляет собой процесс реализации этих инструкций на конкретном языке программирования.

Язык программирования C++ является объектно-ориентированным расширением языка Си, разработанным в начале 1980-х годов Бьерном Страуструпом. Первоначально были добавлены возможности работы с классами и объектами, а в 1983 году, после внесения множества синтаксических изменений, язык получил название «C++». Первый коммерческий выпуск языка состоялся в октябре 1985 года [1].

Целью данной курсовой работы является приобретение навыков комплексного решения вычислительных задач, реализованных в одной из систем программирования.

Задачи курсовой работы:

- закрепление, углубление и систематизация знаний, полученных при изучении дисциплины «Алгоритмизация и программирование», а также приобретение практических навыков решения комплексных задач;

- накопление опыта аналитической и расчетной работы, развитие навыков работы со специализированной литературой и другими источниками информации;

- формирование умений делать логически обоснованные выводы, предложения и рекомендации на основе результатов выполненной работы;

- развитие навыков грамотной подготовки презентаций и защищаемых работ;

- формирование умений выступать перед аудиторией, защищая курсовую работу, компетентно отвечать на вопросы, вести профессиональные дискуссии и убеждать оппонентов в правильности своих решений;

- развитие системного мышления и совершенствование профессиональной устной и письменной речи.

В рамках данной курсовой работы была разработана программа на языке программирования C++ для автосалона Lada, предназначенная для автоматизации процесса продажи автомобилей. Данная система, являясь конфигуратором, позволяет управлять процессом выбора и продажи автомобилей, обеспечивая удобный интерфейс для настройки автомобилей по желанию клиента. Основная цель программы – улучшить эффективность и качество обслуживания клиентов, автоматизировав процессы, связанные с продажей автомобилей.

1. ПОСТАНОВКА ЗАДАЧИ

Формулировка задачи представляет собой один из ключевых этапов при разработке программы. Основная цель этого этапа состоит в создании четкого и ясного описания того, что именно требуется разработать и какие функции должна выполнять программа.

В рамках данной курсовой работы предстоит разработать и реализовать программу для автосалона Lada, которая будет предназначена для автоматизации процесса продажи автомобилей. Разработанная программа должна включать в себя следующие функции и возможности:

- просмотр всех доступных моделей автомобилей, предлагаемых автосалоном;

- настройка выбранной модели, включая выбор типа кузова, комплектации, цвета и дополнительных аксессуаров;

- расчёт итоговой стоимости автомобиля с учётом выбранных аксессуаров, скидок и кредитных условий;

- создание заявки на покупку автомобиля с вводом контактных данных и информации о заказе;

- просмотр, обработка и отслеживание полученных заявок на покупку.

Цель данной программы – автоматизировать процесс продажи автомобилей в автосалоне Lada и повысить эффективность обработки заявок. Внедрение данной системы в работу автосалона позволит сократить время на обработку заказов, а также предоставить удобный интерфейс для пользователей и менеджеров.

Целевая аудитория программы включает несколько основных сегментов пользователей:

- основными пользователями программы являются клиенты автосалона, которым необходимо выбрать и персонализировать автомобиль, а также оставить заявку на его покупку;

- менеджеры автосалона, которые будут использовать программу для управления процессом продажи автомобилей, включая просмотр текущего модельного ряда, добавление и удаление моделей, отслеживание статуса заявок и управление наличием модельного ряда в базе данных;

- руководство автосалона, которое сможет использовать данные, предоставляемые программой, для анализа предпочтений клиентов, планирования закупок и маркетинговых стратегий.

Правильная постановка задачи является важнейшим этапом, который определяет успех всего проекта. Тщательно проработанные требования и функции программы для автосалона Lada обеспечат автоматизацию процесса продажи автомобилей, повысят эффективность работы автосалона и улучшат взаимодействие с клиентами. Реализованная программа будет удобной и функциональной как для клиентов, так и для менеджеров автосалона, что в конечном итоге приведёт к увеличению продаж и удовлетворённости клиентов.

1. ИЗУЧЕНИЕ КОНТЕЙНЕРОВ LIST И MAP

Контейнеры являются фундаментальными элементами стандартной библиотеки C++, предоставляя структуры данных для хранения и управления коллекциями объектов. Они обеспечивают удобный и эффективный способ работы с данными, что особенно важно при разработке сложных программных систем. Контейнеры List и Map предлагают различные методы доступа, вставки и удаления элементов, что делает их чрезвычайно гибкими и мощными инструментами для разработчиков.

Контейнер List представляет собой двусвязный список, который позволяет эффективно вставлять и удалять элементы в любом месте списка. В отличие от массивов и векторов, где элементы хранятся последовательно, List хранит каждый элемент отдельно, используя узлы, которые содержат указатели на предыдущий и следующий элемент. Это делает операции вставки и удаления быстрыми и постоянными по времени, независимо от размера списка. Синтаксис данного контейнера представлен на рисунке 1.



Рисунок 1 – Синтаксис контейнера List

**Основные операции:**

1. push\_back(const T& val) - добавляет элемент в конец списка;
2. push\_front(const T& val) - добавляет элемент в начало списка;
3. pop\_back() - удаляет последний элемент;
4. pop\_front() - удаляет первый элемент;
5. insert(iterator position, const T& val) - вставляет элемент перед заданной позицией;
6. erase(iterator position) - удаляет элемент в заданной позиции;
7. size() - возвращает количество элементов в списке;
8. clear() - удаляет все элементы из списка;
9. begin() - возвращает итератор на первый элемент;
10. end() - возвращает итератор на позицию после последнего элемента.

**Преимущества данного контейнера:**

1. эффективные операции вставки и удаления, независимо от размера списка;
2. возможность вставки и удаления в любом месте списка без необходимости перемещения других элементов.

**Недостатки данного контейнера:**

1. нет прямого доступа к элементам по индексу, что делает операции случайного доступа медленными;
2. более высокий расход памяти по сравнению с массивами или векторами из-за дополнительных указателей.

Внутренняя структура контейнера List представлена на рисунке 2.

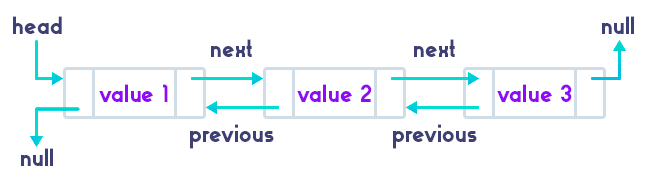
****

Рисунок 2 – Внутренняя структура контейнера List

Контейнер Map является ассоциативным контейнером, который хранит элементы в виде пар ключ-значение. Он обеспечивает быстрый доступ к значениям по их ключам, используя внутреннюю структуру данных, обычно реализованную как красно-черное дерево. Это позволяет поддерживать элементы отсортированными и обеспечивает логарифмическую сложность для операций вставки, удаления и поиска. Синтаксис данного контейнера представлен на рисунке 3.



Рисунок 3 – Синтаксис контейнера List

**Основные операции:**

1. operator[](const key\_type& k) - доступ к элементу по ключу. Если элемент с таким ключом не существует, он будет создан;
2. insert(const value\_type& val) - вставляет элемент. Возвращает пару, состоящую из итератора и булева значения, указывающего на успешность вставки;
3. erase(const key\_type& k) - удаляет элемент по ключу. Возвращает количество удаленных элементов (0 или 1);
4. find(const key\_type& k) - возвращает итератор на элемент с заданным ключом или end(), если элемент не найден;
5. clear() - удаляет все элементы из контейнера;
6. size() - возвращает количество элементов в контейнере;
7. begin() - возвращает итератор на первый элемент.;
8. end() - возвращает итератор на позицию после последнего элемента.

**Преимущества данного контейнера:**

1. быстрый доступ к элементам по ключу;
2. автоматическая сортировка элементов по ключу;
3. обеспечивает гарантированную логарифмическую сложность для основных операций.

**Недостатки данного контейнера:**

1. более высокий расход памяти из-за хранения ключей и значений, а также внутренней структуры данных;
2. операции вставки и удаления могут быть медленнее по сравнению с последовательными контейнерами, такими как вектор или массив.

Внутренняя структура контейнера Map представлена на рисунке 4.

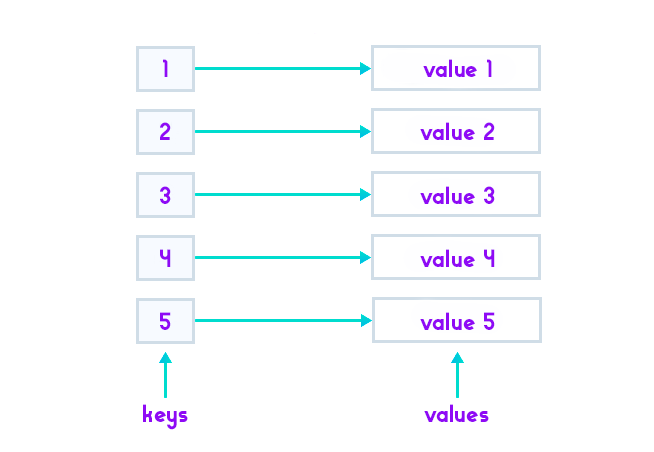
****

Рисунок 4 – Внутренняя структура контейнера Map

Таким образом, контейнеры List и Map обеспечивают гибкость и эффективность при работе с различными наборами данных. List идеально подходит для сценариев, требующих частых вставок и удалений, в то время как Map обеспечивает быстрый доступ и манипуляции с элементами на основе ключей. Понимание их синтаксиса, основных операций, преимуществ и недостатков помогает выбирать правильные инструменты для решения различных задач в программировании.

1. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

Для описания предметной области задачи была использована следующая математическая модель:

#### Дано:

1. **C = {M, B, T, K, P}** - конфигурация автомобиля (Car configuration):

* **M** - модель автомобиля (Model);
* **B** - тип кузова автомобиля (Body type);
* **T** - комплектация автомобиля (Trim level);
* **K** - цвет автомобиля (Color);
* **P** - цена автомобиля (Price);

1. **A = {a1, a2, ..., an}** - аксессуары (Accessories):
   * + **ai = {name, price}** - каждый аксессуар характеризуется именем и ценой;
2. **D = {C, A}** - сделка (Deal):

* **C** - выбранная конфигурация автомобиля;
* **A** - выбранные аксессуары;
* **finalPrice** - итоговая цена с учётом скидок и кредитных условий;
* **discountAmount** - сумма скидок;
* **loanAmount** - сумма кредита;
* **monthlyPayment** - ежемесячный платеж по кредиту;
* **totalPayment** - общая сумма выплат по кредиту;
* **overpayment** - сумма переплаты по кредиту;

1. **L = {D1, D2, ..., Dm}** - история сделок (Deal history):
   * **m** - количество сделок;
   * **L** - список завершённых сделок.

#### Найти:

1. **GetAvailableModels()** - получить список доступных моделей:
   * **GetAvailableModels() = {M1, M2, ..., Mk}** - возвращает список всех доступных моделей автомобилей;
2. **SelectConfiguration(C, M, B, T, K)** - выбрать конфигурацию автомобиля:
   * **SelectConfiguration(C, M, B, T, K) = {M, B, T, K, P}** - устанавливает выбранную конфигурацию и цену автомобиля;
3. **GetAvailableAccessories()** - получить список доступных аксессуаров:
   * **GetAvailableAccessories() = {a1, a2, ..., an}** - возвращает список всех доступных аксессуаров;
4. **AddAccessory(D, ai)** - добавить аксессуар в сделку:
   * **AddAccessory(D, ai) = D'** - добавляет выбранный аксессуар к текущей сделке;
5. **RemoveAccessory(D, ai)** - удалить аксессуар из сделки:
   * **RemoveAccessory(D, ai) = D'** - удаляет выбранный аксессуар из текущей сделки;
6. **ApplyDiscounts(D)** - применить скидки:
   * **ApplyDiscounts(D) = D'** - рассчитывает и применяет скидки к текущей сделке;
7. **CalculateLoan(D, loanTerm)** - рассчитать кредитные условия:
   * **CalculateLoan(D, loanTerm) = D'** - рассчитывает сумму кредита, ежемесячные платежи и переплату;
8. **FinalizeDeal(D)** - финализировать сделку:
   * **FinalizeDeal(D) = D'** - завершает сделку, учитывая все выбранные опции и условия;
9. **SaveDealToFile(D)** - сохранить сделку в файл:
   * **SaveDealToFile(D)** - сохраняет детали сделки в файл.

Математическая модель программы включает описание всех ключевых компонентов и операций, связанных с конфигурацией автомобилей, выбором аксессуаров, расчётом скидок и кредитных условий, а также сохранением историй сделок. Эта модель обеспечивает чёткую структуру для реализации программы, автоматизирующей процесс продажи автомобилей в автосалоне Lada.

1. БЛОК-СХЕМА ПРОГРАММЫ

Блок-схема представляет собой графическое изображение алгоритма или процесса, демонстрирующее последовательность шагов и операций. Она используется для упрощения понимания, анализа и документирования сложных процессов и систем. Визуальное представление блок-схемы позволяет разработчикам и другим специалистам легко визуализировать и анализировать последовательность действий, что помогает выявлять ошибки, улучшать процессы и эффективно передавать информацию [2].

Основное назначение блок-схем состоит в следующем:

* + 1. упрощение понимания процесса. Благодаря визуальному представлению, блок-схемы делают сложные процессы и алгоритмы более понятными. Каждый шаг процесса отображается в виде символов и соединительных линий, что позволяет увидеть общую картину и последовательность действий;
    2. анализ и улучшение процессов. Блок-схемы помогают выявлять узкие места, избыточные шаги или потенциальные ошибки в алгоритме или процессе. Это позволяет проводить анализ и вносить улучшения, оптимизируя работу системы и повышая её эффективность;
    3. документация и обучение. Блок-схемы служат важным инструментом для документирования процессов и алгоритмов. Они обеспечивают наглядное представление, которое легко использовать для обучения новых сотрудников или передачи знаний между специалистами. Это особенно важно в условиях, когда необходимо быстро передать информацию или объяснить сложные процессы;
    4. коммуникация между командами. Блок-схемы способствуют эффективному общению между различными командами и специалистами. Визуальное представление процесса позволяет легко обсуждать и объяснять детали алгоритма, что способствует более слаженной работе и лучшему пониманию задач и целей.

Блок-схемы широко применяются в различных областях, включая программирование, инженерное дело и бизнес-процессы. Они помогают систематизировать информацию, упростить сложные задачи и улучшить качество принимаемых решений [2].

Блок-схема, представленная на рисунке 5, отражает основные этапы работы программы, предназначенной для автоматизации процесса продажи автомобилей.

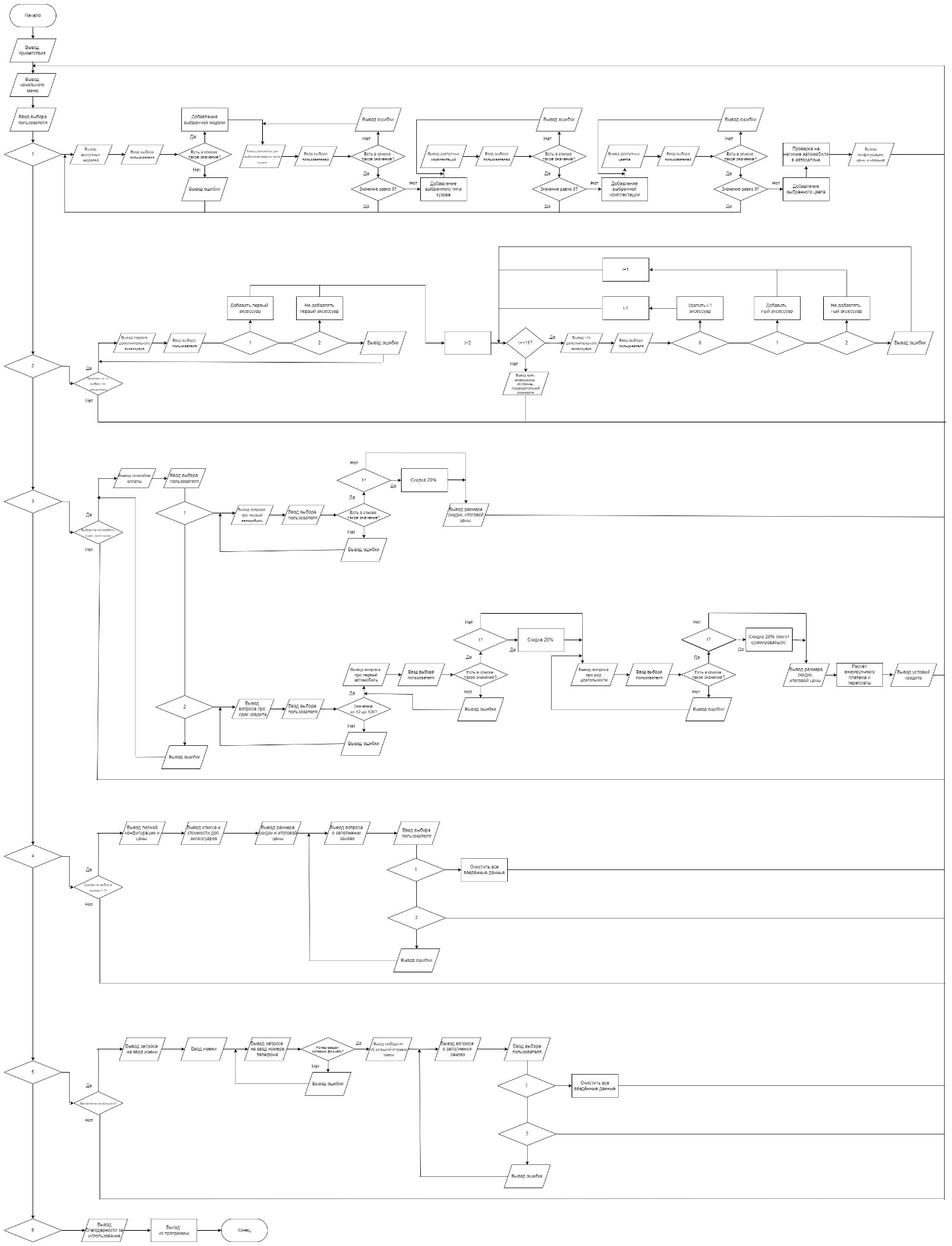


Рисунок 5 - Блок-схема программы

1. ТЕСТИРОВАНИЕ ПРОГРАММЫ

Тестирование программы является важным этапом при разработке программных кодов. Оно направлено на выявление ошибок, проверку корректности выполнения функций, оценку производительности и обеспечение надежности работы программы. Основная цель тестирования заключается в том, чтобы гарантировать соответствие программного кода функциональным требованиям и стандартам качества, а также обеспечить его стабильность и безопасность [3].

В процессе тестирования необходимо учитывать следующие основные аспекты:

* + 1. проверка функциональности. Программа должна быть полностью рабочей, все функции и возможности должны выполняться корректно и соответствовать описанию, представленному на этапе постановки задачи и составления блок-схемы;
    2. выявление ошибок. В случае нарушения стабильности работы программы или возникновения проблем при выполнении определённой функции, необходимо провести анализ возможных ошибок. При серьёзных проблемах разработчики должны вернуться к предыдущим этапам разработки для устранения выявленных недостатков;
    3. разработка тестовых сценариев. Тестировщики должны определить конкретные тестовые случаи и сценарии, которые будут использованы для проверки функциональности программы. Это необходимо для того, чтобы удостовериться в том, что программа удовлетворяет всем потребностям основных сегментов целевой аудитории.

План тестирования (Test Plan) — это документ, который описывает цели, подходы, ресурсы и график запланированных тестовых активностей. Он определяет объекты тестирования, характеристики, подлежащие тестированию, задачи, ответственных за выполнение задач, степень независимости каждого тестировщика, тестовое окружение, метод проектирования тестов. Также он устанавливает критерии входа и выхода, причины их выбора и любые риски, требующие планирования на случай чрезвычайных обстоятельств. План тестирования помогает организовать процесс тестирования, обеспечивая систематический подход к проверке программы и достижению высокого уровня качества.

Эффективное тестирование программного кода позволяет обеспечить высокое качество и надежность программного обеспечения, минимизировать риски возникновения ошибок в эксплуатации и повысить удовлетворенность пользователей [3].

* 1. User story

User Story (пользовательская история) — это метод описания функциональности программного обеспечения с точки зрения конечного пользователя. Пользовательские истории помогают понять, какие функции и возможности программы важны для пользователей, и позволяют разработчикам лучше сфокусироваться на их потребностях и ожиданиях. Пользовательские истории обычно включают в себя описание пользователя, его цели и причины, по которым он стремится достичь этой цели [4].

Примеры User story для данной программы:

* + 1. Артемий Сонькин планирует приобрести новый автомобиль в автосалоне Lada. Он использует конфигуратор для выбора подходящей модели автомобиля. После выбора модели, Артемий добавляет необходимые дополнительные аксессуары, чтобы адаптировать автомобиль под свои потребности. Затем он использует функцию расчёта стоимости, чтобы узнать общую цену с учётом всех выбранных опций. Поскольку Артемий планирует взять автомобиль в кредит, он использует функцию расчёта кредита, чтобы узнать ежемесячный платёж и общую переплату. Артемию также важно узнать о возможных скидках, так как это его первый автомобиль. Он вводит данные в конфигуратор и видит, что ему доступна скидка 20 процентов. Убедившись, что все условия устраивают его, Артемий отправляет заявку на покупку автомобиля через конфигуратор;
    2. Ольга Смирнова, менеджер автосалона Lada, ежедневно использует конфигуратор для помощи клиентам. Когда к ней обращается клиент, Ольга открывает конфигуратор и вместе с клиентом выбирает подходящую модель автомобиля. Она добавляет в конфигуратор все необходимые аксессуары, чтобы клиент мог видеть полную комплектацию и стоимость автомобиля. После этого Ольга проводит расчёт итоговой стоимости с учётом возможных скидок и кредитных предложений. Она отслеживает все поступившие заявки через конфигуратор, чтобы обеспечить своевременную обработку и ответ клиентам. Ольга также следит за наличием моделей в базе данных и обновляет информацию о доступных автомобилях, чтобы клиенты всегда имели актуальные данные при выборе автомобиля.

1. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Руководство пользователя представляет собой справочник, к которому пользователь может обратиться в процессе работы, что обеспечивает удобство при решении задач с помощью данного программного обеспечения. Основная цель руководства – предоставить пользователю необходимую информацию для успешного выполнения всех нужных операций. Ниже приведено подробное руководство пользователя по использованию данной программы.

При запуске программы пользователь видит приветственное сообщение, в котором представлена информация о том, что это конфигуратор автосалона Lada, кто его создал, а также список доступных действий:

* выбрать модель автомобиля;
* выбрать дополнительные аксессуары;
* рассчитать скидки и акции;
* рассчитать итоговую стоимость;
* оставить заявку;
* закрыть конфигуратор.

Окно при запуске программы представлено на рисунке 4.

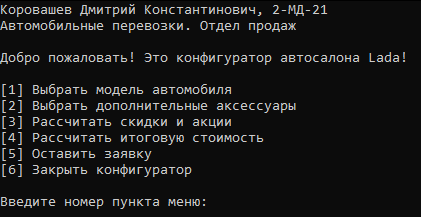


Рисунок 6 – Окно при запуске программы

В течение всей работы с программой в случае ввода некорректного значения она выведет сообщение об ошибке и предложит ввести ответ заново, чтобы пользователь мог сделать это корректно. Пример вывода ошибки из-за некорректно введённого значения представлен на рисунке 7. Помимо этого, при ожидании ответа от пользователя выводится сообщение, связанное с этим.

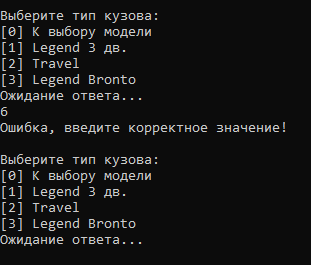


Рисунок 7 – Пример вывода ошибки

При выборе пункта 1 – «Выбрать модель автомобиля» пользователю предоставляется список всех доступных моделей автомобилей. Пользователь должен ввести номер пункта, соответствующий выбранной модели. После этого программа переходит к следующему шагу, предлагая выбрать тип кузова для выбранной модели. Если пользователь вводит цифру «0» на этом этапе, программа возвращается к выбору модели, стирая предыдущий выбор.

После выбора типа кузова программа переходит к выбору комплектации автомобиля. Пользователь вводит номер пункта, соответствующий выбранной комплектации. Как и в предыдущем шаге, при вводе цифры «0» программа возвращается к выбору модели, аннулируя предыдущие шаги.

После выбора комплектации программа предложит выбрать цвет автомобиля. Пользователь вводит номер пункта, соответствующий желаемому цвету. Как и ранее, цифра «0» позволяет вернуться к выбору модели.

После успешного выбора цвета программа выводит полное наименование автомобиля, подробности его конфигурации и итоговую цену. Программа также проверяет наличие выбранного автомобиля в базе данных автосалона и выводит сообщение о наличии автомобиля или необходимости его заказа. Фрагмент базы данных с автомобилями и их наличием в автосалоне представлен на рисунке 8.

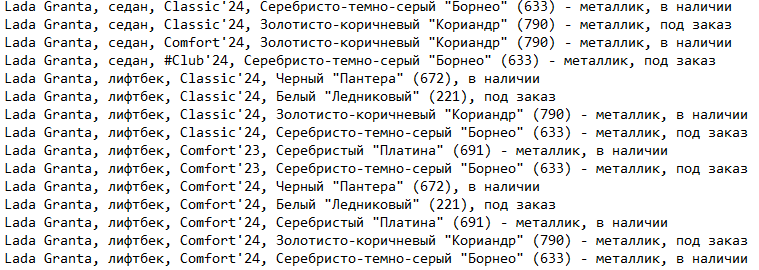


Рисунок 8 - Фрагмент базы данных с автомобилями и их наличием

Завершающим этапом программа сообщает, что модель выбрана, и возвращается в главное меню, предоставляя пользователю возможность продолжить работу с программой.

Пример выбора модели автомобиля представлен на рисунке 9.

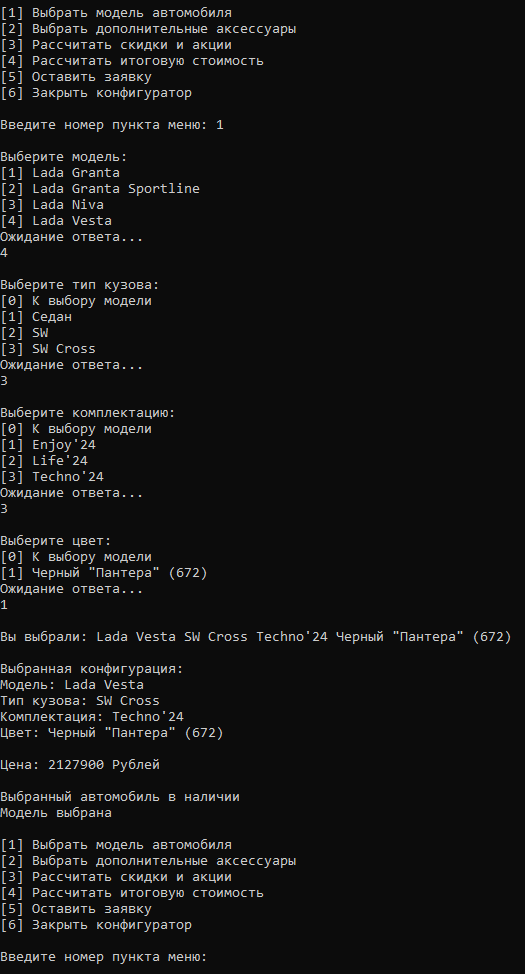


Рисунок 9 – Пример выбора модели автомобиля

При выборе пункта 2 – «Выбрать дополнительные аксессуары» пользователю последовательно предоставляются доступные аксессуары, каждый с указанием названия и стоимости. Пользователю предлагается ввести одно из нескольких значений: «1» - добавить в заказ, «2» - не добавлять в заказ, «0» - вернуться к предыдущему аксессуару. Если пользователь вводит цифру «0», он возвращается к предыдущему аксессуару, что позволяет пересмотреть выбор. Возвращение к предыдущему аксессуару возможно только со второго аксессуара - на первом аксессуаре вернуться назад невозможно.

Программа начинает с первого аксессуара и предлагает пользователю сделать выбор. Если пользователь решает добавить аксессуар, программа отмечает его в заказе. Если пользователь отказывается от добавления, программа переходит к следующему аксессуару. При каждом шаге ввода некорректного значения программа выведет сообщение об ошибке и предложит ввести ответ заново.

Процесс продолжается до тех пор, пока не будет рассмотрен последний, одиннадцатый аксессуар. После завершения цикла предложений программа выводит итоговый список всех выбранных аксессуаров, их суммарную стоимость, а также промежуточную стоимость автомобиля с учётом дополнительных аксессуаров. Это предоставляет пользователю полное представление о конфигурации автомобиля и его текущей стоимости с учётом всех выбранных опций. После вывода информации программа возвращается в главное меню.

Пример выбора дополнительных аксессуаров представлен на рисунке 10.

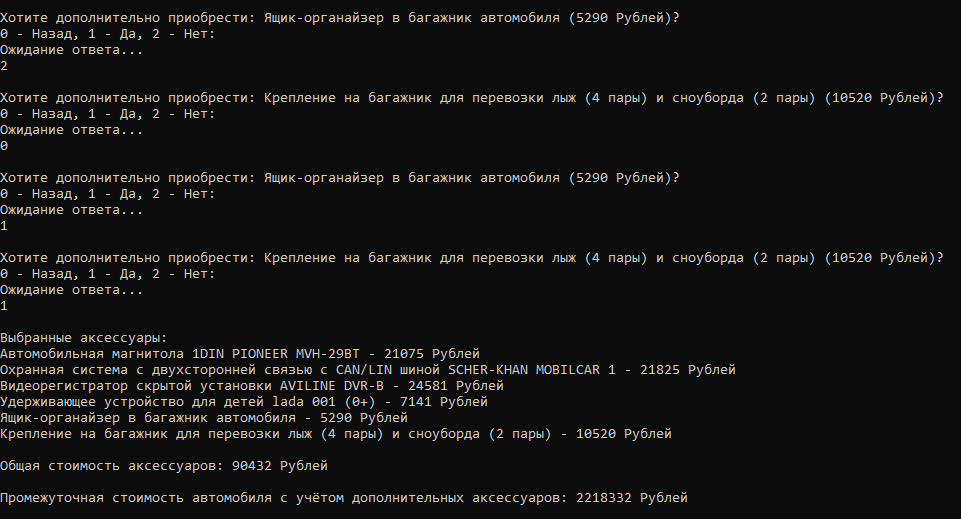


Рисунок 10 - Пример выбора дополнительных аксессуаров

Результат вывода программы, если не выбран ни один дополнительный аксессуар, представлен на рисунке 11.

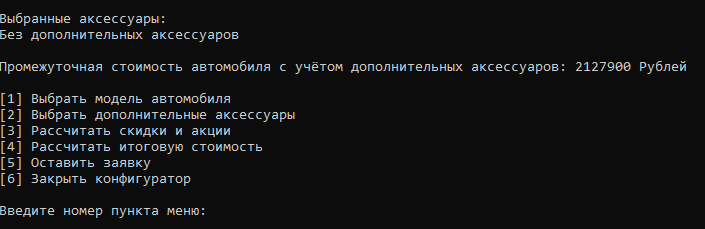


Рисунок 11 - Вывод после пункта 2, если не выбран ни один аксессуар

При выборе пункта 3 – «Рассчитать скидки и акции» пользователю предлагается выбрать способ оплаты: наличные или кредит. На экране появляется соответствующий вопрос, и пользователь вводит свой выбор.

Если пользователь выбирает оплату наличными, программа задаёт вопрос о том, является ли данный автомобиль первым автомобилем для покупателя. В случае утвердительного ответа, на автомобиль предоставляется скидка в размере 20 процентов. Программа выводит размер скидки и итоговую цену автомобиля с учетом скидки. Пример выбора оплаты наличными и получения скидки представлен на рисунке 12.

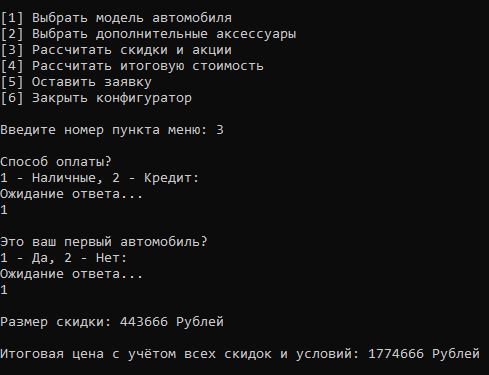


Рисунок 12 - Пример выбора оплаты наличными

Если пользователь выбирает оплату в кредит, программа снова задаёт вопрос о первом автомобиле. При утвердительном ответе также предоставляется скидка в размере 20 процентов. Далее программа задаёт вопрос о том, является ли покупатель учителем, медработником или военнослужащим. При утвердительном ответе на этот вопрос предоставляется дополнительная скидка в размере 20 процентов. Эти скидки могут суммироваться, что позволяет существенно снизить цену автомобиля. После получения ответов на все вопросы программа выводит размер общей скидки и итоговую цену автомобиля с учетом всех возможных скидок. Кроме того, для оплаты в кредит программа выводит условия кредита, включая ежемесячный платеж и переплату. Пример выбора оплаты в кредит и получения скидки представлен на рисунке 13.

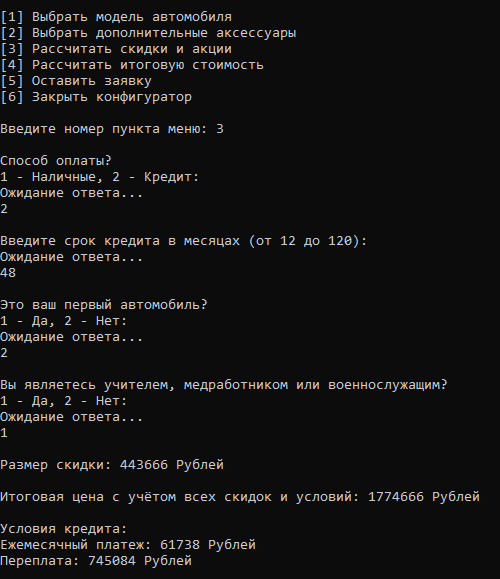


Рисунок 13 - Пример выбора оплаты в кредит

В конце всегда происходит возвращение в главное меню, что позволяет пользователю продолжить работу с программой или завершить настройку автомобиля.

При выборе пункта 4 – «Рассчитать итоговую стоимость» пользователю предоставляется вся собранная ранее информация из пунктов 1, 2 и 3. Программа выводит полное наименование выбранного автомобиля, его тип кузова, комплектацию, цвет, перечень всех добавленных аксессуаров, суммарную стоимость выбранных аксессуаров, размер скидки (если есть) и итоговую цену автомобиля с учётом всех дополнительных услуг и скидок.

После вывода всей информации пользователю предлагается выбор: заполнить анкету заново или продолжить и оставить заявку на покупку. Программа выводит соответствующий вопрос и ожидает ввода пользователя.

Если пользователь выбирает заполнить анкету заново, то все введённые ранее данные очищаются, и открывается главное меню, предоставляя возможность начать процесс конфигурации автомобиля с самого начала.

Если пользователя устраивает итоговая цена и он не хочет заново заполнять анкету, то программа возвращает его в главное меню, при этом все введённые данные сохраняются. Пользователь может продолжить работу с программой, сохранив все введённые данные для дальнейшего оформления заявки на покупку.

Пример вывода итоговой стоимости представлен на рисунке 14.

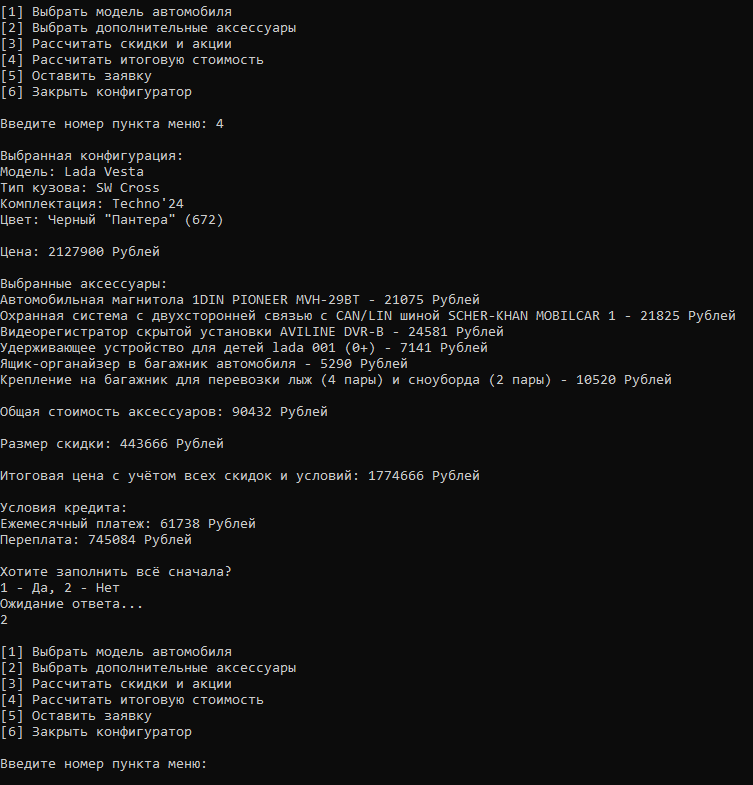


Рисунок 14 - Пример вывода итоговой стоимости

При выборе пункта 5 – «Оставить заявку» пользователю предоставляется возможность оформить заявку на покупку автомобиля. Сначала программа предлагает пользователю ввести своё имя. После ввода имени программа запрашивает номер телефона в необходимом формате, начиная с +7 и следуя шаблону +7 (xxx) xxx-xx-xx. Если формат введённого номера телефона не соответствует требуемому, программа выводит сообщение об ошибке и просит ввести номер телефона заново.

После того как номер телефона введён корректно, программа выводит сообщение о том, что заявка успешно отправлена. Затем пользователю предлагается выбор: заполнить все данные заново или вернуться в главное меню.

Если пользователь соглашается заполнить всё заново, все ранее введённые данные очищаются, и программа возвращается в главное меню, предоставляя возможность начать процесс конфигурации автомобиля с самого начала.

Если пользователь не хочет заполнять данные заново, программа возвращает его в главное меню, сохранив все введённые данные. Из главного меню пользователь может ещё раз просмотреть данные своей заявки, либо выйти из программы и ожидать звонка от представителя автосалона.

Пример подачи заявки на покупку выбранного автомобиля представлен на рисунке 15.

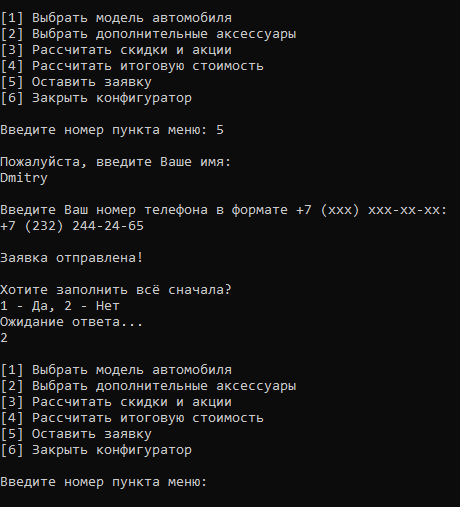


Рисунок 15 - Пример подачи заявки на покупку

Полученная информация добавляется в базу данных заявок на покупку автомобиля. Её фрагмент представлен на рисунке 16.

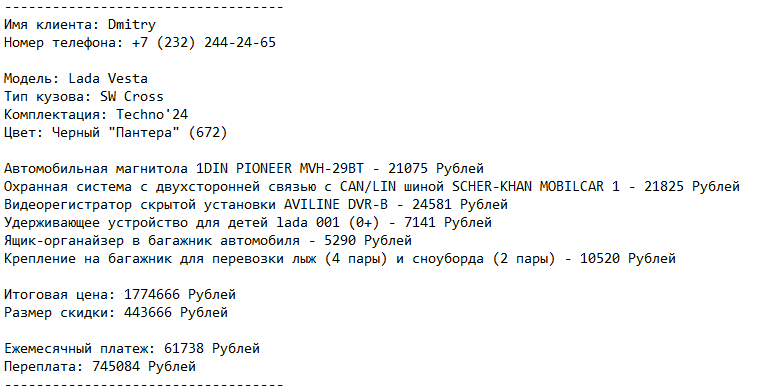


Рисунок 16 - Фрагмент базы данных заявок

При выборе пункта 6 – «Закрыть конфигуратор» программа выводит сообщение с благодарностью за использование конфигуратора. Пользователь увидит текст: «Спасибо за использование конфигуратора. До свидания!». После этого программа завершает свою работу и закрывается. Этот пункт позволяет пользователю удобно завершить процесс работы с конфигуратором без необходимости принудительного закрытия программы. Демонстрация работы пункта 6 представлена на рисунке 17.

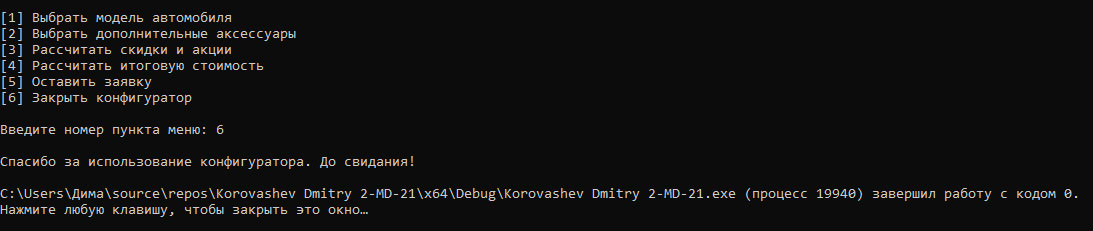


Рисунок 17 - Пример закрытия конфигуратора

Это руководство предназначено для того, чтобы помочь пользователям в полной мере использовать функционал программы и успешно решать возникающие в процессе работы вопросы.

1. РАЗРАБОТКА СТРУКТУРЫ ДАННЫХ

Структура данных представляет собой метод организации информации, который способствует её более эффективному использованию. В программировании под структурой данных подразумевается совокупность данных, объединённых определённым образом. Она определяет формат данных, способы их доступа и методы взаимодействия с ними. Грамотно разработанная структура данных позволяет существенно повысить производительность программы, обеспечивая быстрый доступ к информации, её эффективное изменение и удобное управление [5]. Структура данных для данной программы представлена на рисунке 24.

Для программы конфигуратора автомобилей автосалона Lada была разработана структура данных, включающая несколько ключевых классов: «Car», «Configuration», «Accessories» и «Deal». Эти классы обеспечивают возможность выбора автомобиля, добавления дополнительных аксессуаров, расчёта итоговой стоимости с учётом скидок и условий кредита, а также ведения информации о сделке и клиенте. В данной главе подробно описаны структуры данных, используемые в программе.

Класс «Car» является центральным элементом системы и включает в себя информацию о выбранной конфигурации автомобиля, выбранных дополнительных аксессуарах и сделке. Он предоставляет методы для выбора модели автомобиля, добавления аксессуаров, расчёта скидок и условий кредита, а также оформления и сохранения заявки. Класс представлен на рисунке 18.

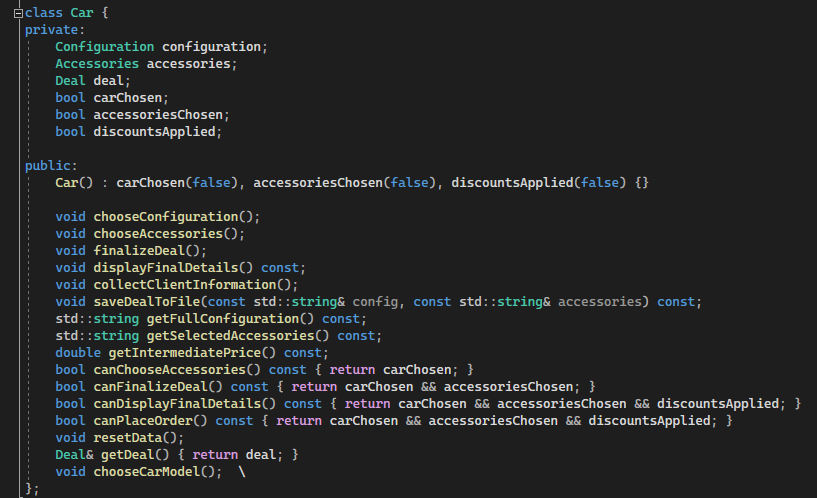


Рисунок 18 - Класс «Car»

Основные методы:

* + 1. конструктор Car(): инициализирует состояние автомобиля (выбрана ли модель, аксессуары, применены ли скидки);
    2. метод chooseCarModel(): позволяет пользователю выбрать модель автомобиля;
    3. метод chooseAccessories(): позволяет пользователю выбрать дополнительные аксессуары;
    4. метод finalizeDeal(): применяет скидки и рассчитывает условия кредита;
    5. метод displayFinalDetails() const: отображает итоговую информацию о заказе;
    6. метод collectClientInformation(): собирает информацию о клиенте;
    7. метод saveDealToFile(const std::string& config, const std::string& accessories) const: сохраняет информацию о заказе в файл;
    8. метод canChooseAccessories() const: проверяет, можно ли выбрать аксессуары;
    9. метод canFinalizeDeal() const: проверяет, можно ли применить скидки и акции;
    10. метод canDisplayFinalDetails() const: проверяет, можно ли отображать итоговую информацию;
    11. метод canPlaceOrder() const: проверяет, можно ли оформить заказ;
    12. метод resetData(): сбрасывает все данные о заказе.

Класс «Configurator» представляет собой основной класс для конфигурации автомобиля в автосалоне. Основные функции класса включают выбор модели автомобиля, выбор дополнительных аксессуаров, применение скидок и акций, расчёт кредита и отображение итоговой информации о заказе. Класс представлен на рисунке 19.

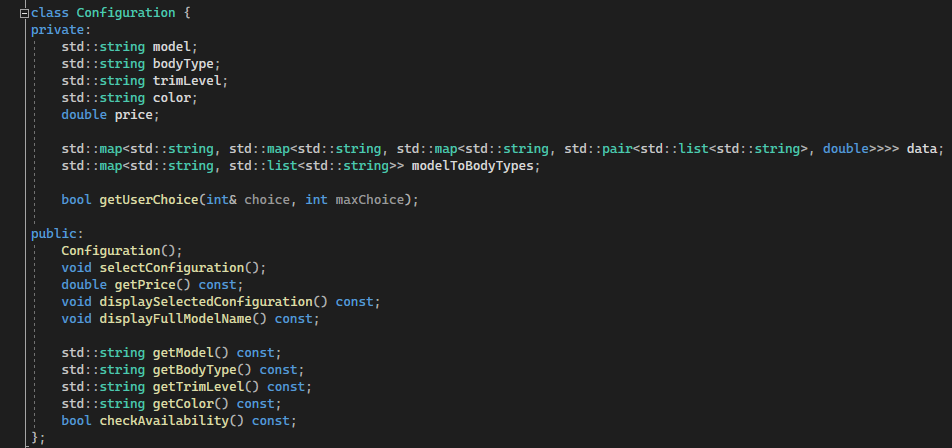


Рисунок 19 - Класс «Configurator»

Основные методы:

* + 1. конструктор Configurator(): инициализирует состояние конфигуратора;
    2. метод showMainMenu(): отображает главное меню и обрабатывает выбор пользователя;
    3. метод chooseCarModel(): позволяет пользователю выбрать модель автомобиля;
    4. метод chooseAccessories(): позволяет пользователю выбрать дополнительные аксессуары;
    5. метод applyDiscountsAndLoans(): применяет скидки и рассчитывает условия кредита;
    6. метод displayTotalCost(): отображает итоговую информацию о заказе;
    7. метод placeOrder(): собирает информацию о клиенте и сохраняет заказ;
    8. метод resetData(): сбрасывает все данные о заказе.

Класс «Accessories» представляет собой набор дополнительных аксессуаров, которые могут быть выбраны пользователем для автомобиля. Основные функции класса включают отображение доступных аксессуаров, выбор аксессуаров и расчёт их общей стоимости. Класс представлен на рисунке 20.

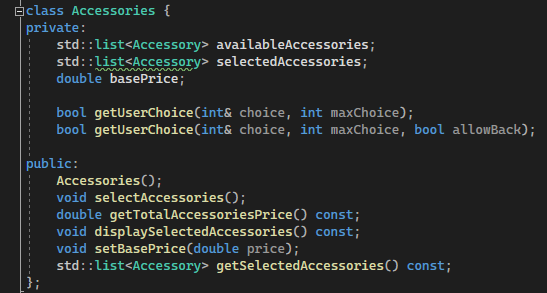


Рисунок 20 - Класс «Accessories»

Основные методы:

* + 1. конструктор Accessories(): инициализирует список доступных аксессуаров и базовую цену;
    2. метод selectAccessories(): позволяет пользователю выбрать дополнительные аксессуары;
    3. метод getTotalAccessoriesPrice() const: возвращает общую стоимость выбранных аксессуаров;
    4. метод displaySelectedAccessories() const: отображает выбранные аксессуары и их стоимость;
    5. метод setBasePrice(double price): устанавливает базовую цену автомобиля;
    6. метод getSelectedAccessories() const: возвращает список выбранных аксессуаров.

Класс «Deal» представляет собой сделку, включающую информацию о цене автомобиля, применённых скидках и условиях кредита. Основные функции класса включают установку базовой цены, получение пользовательских вводов, применение скидок, расчёт условий кредита и отображение итоговой информации о сделке. Класс представлен на рисунке 21.



Рисунок 21 - Класс «Deal»

Основные методы:

* 1. конструктор Deal(): инициализирует состояния сделки (цены, скидки, кредиты);
  2. метод setBasePrice(double price): устанавливает базовую цену автомобиля;
  3. метод getUserChoice(int& choice, int maxChoice): получает выбор пользователя в виде целого числа;
  4. метод getDoubleInput(double& input): получает ввод пользователя в виде числа с плавающей точкой;
  5. метод getIntInput(int& input): получает ввод пользователя в виде целого числа;
  6. метод getPhoneNumberInput(std::string& input): получает ввод пользователя в виде строки, проверяет формат номера телефона;
  7. метод applyDiscounts(): применяет доступные скидки к базовой цене автомобиля;
  8. метод calculateLoanDetails(): рассчитывает условия кредита, включая ежемесячный платёж и переплату;
  9. метод finalizeDeal(): завершает сделку, применяя скидки и условия кредита;
  10. метод displayDealDetails() const: отображает информацию о скидках и условиях кредита;
  11. метод collectClientInformation(): собирает информацию о клиенте (имя, номер телефона);
  12. метод saveDealToFile(const std::string& config, const std::string& accessories) const: сохраняет информацию о сделке в файл;
  13. метод getFinalPrice() const: возвращает итоговую цену автомобиля с учётом всех скидок и условий кредита;
  14. метод reset(): сбрасывает все данные о сделке.

Основной файл программы: «main.cpp», он содержит основной цикл и обработку пользовательского ввода. Обеспечивает взаимодействие пользователя с программой через консольное меню. Код файла представлен на рисунках 22-23.

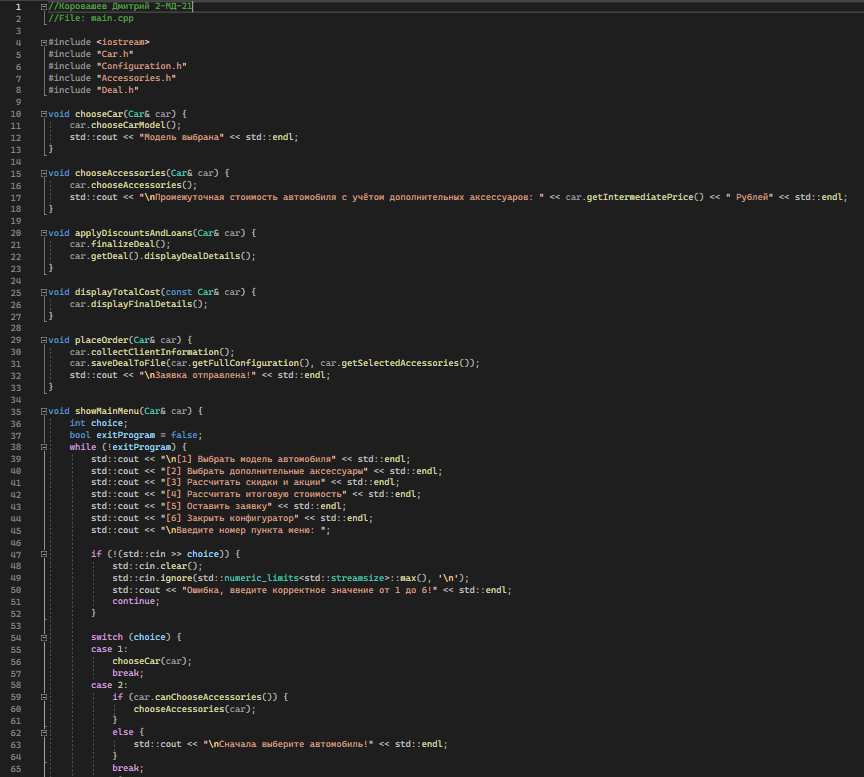


Рисунок 22 - Код файла «main.cpp»

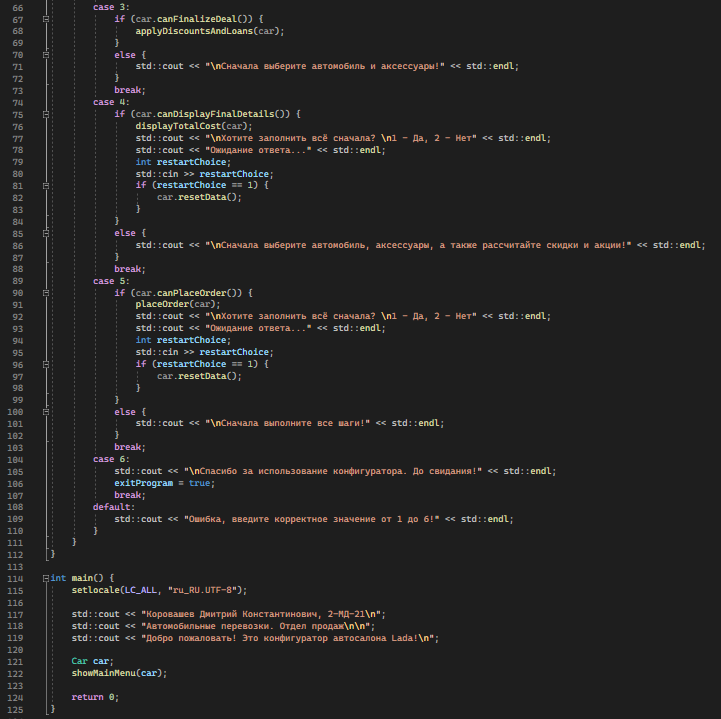


Рисунок 23 - Код файла «main.cpp»

Структура данных представлена в виде схемы на рисунке 24.

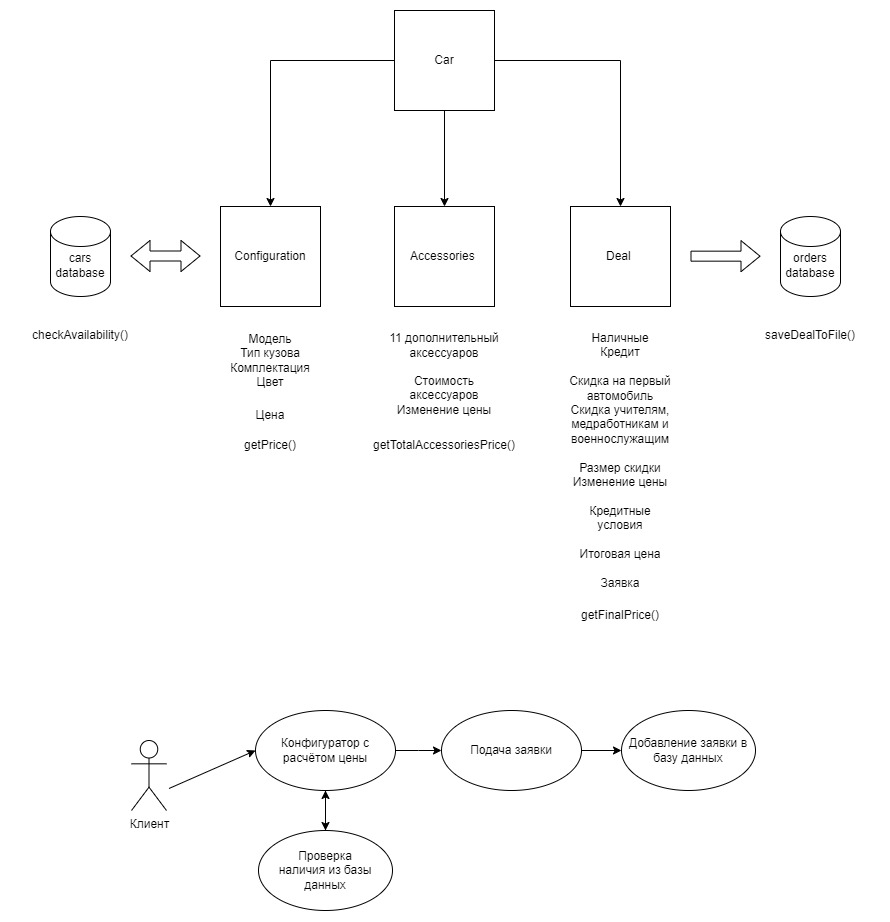


Рисунок 24 – Структура данных проекта

1. РАЗРАБОТКА ОСНОВНОГО МЕНЮ ПРОГРАММЫ

В разработанной программе основное меню представляет собой интерактивный пользовательский интерфейс, позволяющий пользователям выполнять различные операции для конфигурации и заказа автомобилей. Меню предоставляет следующие опции:

* выбрать модель автомобиля;
* выбрать дополнительные аксессуары;
* рассчитать скидки и акции;
* рассчитать итоговую стоимость;
* оставить заявку на покупку;
* закрыть конфигуратор.

Реализация основного меню представлена на рисунке 25.

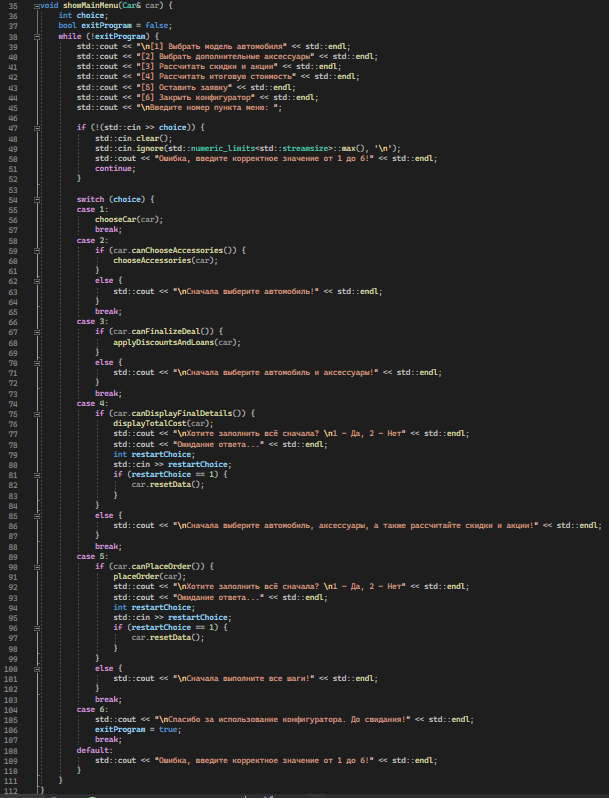


Рисунок 25 – Реализация вывода основного меню с управляющим циклом программы

Управление вводом в программе осуществляется с использованием цикла while, что позволяет повторно отображать меню до тех пор, пока пользователь не решит завершить работу с программой (выбор пункта «6»). Перед началом цикла программа объявляет переменную int choice, которая используется для считывания ввода пользователя через клавиатуру.

В начале каждой итерации цикла программа выводит список доступных действий, а затем считывает число, введенное пользователем. Используя оператор switch, программа определяет, какое действие следует выполнить в зависимости от выбора пользователя:

* + 1. Case 1 – выбор модели автомобиля;
    2. Case 2 – выбор дополнительных аксессуаров;
    3. Case 3 – расчёт скидок и акций;
    4. Case 4 – расчёт итоговой стоимости;
    5. Case 5 – оставление заявки на покупку;
    6. Case 6 – закрытие конфигуратора.

Если введённое число не соответствует ни одному из предложенных вариантов, программа выводит сообщение о некорректном вводе и предлагает повторить выбор.

Каждая из операций меню взаимодействует с соответствующими классами и методами для выполнения задач, таких как выбор модели автомобиля, добавление аксессуаров, расчёт скидок, или оформление заказа.

Основное меню и его функции обеспечивают пользователю удобство в конфигурировании автомобиля и оставлении заявки, а также гарантируют корректное выполнение всех этапов процесса.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была разработана программа для автосалона Lada, предназначенная для автоматизации процесса продажи автомобилей. Основной целью данного проекта было создание удобного и функционального инструмента, который позволит менеджерам автосалона и клиентам эффективно и быстро конфигурировать автомобили, проверять их наличие, рассчитывать итоговую стоимость, а также оформлять заявки на покупку.

Программа реализована на языке программирования C++ и включает в себя несколько классов, каждый из которых отвечает за определённый аспект работы системы. Были разработаны классы для управления конфигурацией автомобиля, выбора аксессуаров, расчёта скидок и условий кредита, а также для обработки и сохранения данных о заявках клиентов.

В процессе разработки программы для автосалона Lada были изучены и использованы контейнеры List и Map из стандартной библиотеки шаблонов C++. Контейнер List был выбран за его эффективность в операциях вставки и удаления элементов, что делает его идеальным для хранения и управления списками аксессуаров и конфигураций автомобилей. Контейнер Map, в свою очередь, предоставляет возможность быстрого доступа к элементам по ключу, что позволяет эффективно управлять данными о различных моделях автомобилей и их характеристиках.

Для программы, предназначенной для автосалона, была разработана математическая модель, включающая описание автомобилей и аксессуаров, а также основные операции, такие как выбор модели автомобиля, добавление и удаление аксессуаров, расчёт скидок и условий кредита, и оформление заявки на покупку. Эта модель послужила основой для реализации всей функциональности программы, обеспечивая удобство и гибкость в работе с данными.

Тестирование программы является важнейшим этапом разработки, так как оно позволяет убедиться в правильности работы всех компонентов системы и выявить возможные ошибки. Проведенное тестирование показало, что разработанная программа соответствует поставленным требованиям и выполняет все необходимые функции. Программа обеспечивает удобный интерфейс для пользователя, позволяя ему легко и быстро выполнять необходимые задачи.

Таким образом, разработанная программа для автосалона Lada, предназначенная для автоматизации процесса продажи автомобилей, успешно прошла тестирование и готова к использованию.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Кому и для чего нужен C++ // SKILLFACTORY MEDIA URL: https://blog.skillfactory.ru/cplus-komu-i-dlya-chego-nuzhen/ (дата обращения: 20.05.2024)
2. Что такое блок-схемы и как их использовать? // Dropbox URL: https://experience.dropbox.com/ru-ru/resources/flowcharts (дата обращения: 23.05.2024)
3. Что такое тестирование программ и зачем оно нужно // Skillbox Media URL: https://skillbox.ru/media/code/chto\_takoe\_testirovanie\_programm/ (дата обращения: 23.05.2024)
4. Что такое User Story и как её создать // Блог Яндекс Практикума URL: https://practicum.yandex.ru/blog/chto-takoe-user-story-i-kak-napisat/ (дата обращения: 23.05.2024)
5. Структура данных // SKILLFACTORY MEDIA URL: https://blog.skillfactory.ru/glossary/struktura-dannyh/ (дата обращения: 24.05.2024)