**4 Создание (конструирование) программного средства**

**4.1 Подробное описание классов, атрибутов и методов**

**4.1.1** Класс TfrMainForm модуля MainUnit

* + 1. createNewObj принимает ссылку на создаваемый объект и флаг isShop. Выделяет память под запись, заполняет поля (имя, улица, дом, корпус, вместимость), задаёт координаты, создаёт графический круг (TShape) с уникальным ключом и цветом, настраивает события мыши и отображает объект на карте.

1. imgMapMouseUp (OnMouseUp) — обработчик отпускания кнопки мыши по компоненту imgMap. Скрывает все панели, сбрасывает формы создания и редактирования, запоминает координаты щелчка, отображает маркер spMapPoint и показывает панель pnCreateSelect для выбора типа создаваемого объекта.
2. btnCreateSelectCancelClick скрывает все всплывающие панели и убирает маркер spMapPoint с карты.
3. btnCreateObjCancelClick очищает все поля формы создания (имя, улица, дом, корпус, вместимость), скрывает панели и убирает маркер spMapPoint.
4. btnCreateSelectClick скрывает все панели, подготавливает панель pnCreateObj (в зависимости от тега кнопки: 1 – магазин, 2 – склад), позиционирует её в точке щелчка и делает видимой.
5. btnCreateObjConfirmClick проверяет корректность заполнения полей и уникальность имени (через дерево имён), при успешной валидации и подтверждении пользователем вызывает createShop или createWarehouse, сбрасывает форму и скрывает панели.
6. OnClickValidateLength проверяет длину текста в переданном поле TEdit и окрашивает фон в красный, если поле пустое, или возвращает стандартный цвет при ненулевой длине.
7. OnClickValidateLetters проверяет, что в поле TEdit содержатся только буквы (с помощью validateLetters), и окрашивает фон поля в красный при наличии недопустимых символов.
8. OnClickvalidateAll последовательно выполняет проверки validateLength и validateLetters для поля TEdit и окрашивает фон в зависимости от результата обеих проверок.
9. createShop создаёт новый объект‑магазин: вызывает createNewObj с isShop = true, вставляет узел в декартово дерево shops и добавляет имя в дерево имён shopsNames.
10. createWarehouse создаёт новый объект‑склад: вызывает createNewObj с isShop = false, вставляет узел в дерево warehouses и добавляет имя в дерево имён warehousesNames.
11. validateCreateObj выполняет набор валидационных функций (validateLength, validateLetters и др.) для полей формы создания и возвращает true при отсутствии ошибок.
12. validateEditObj аналогично проверяет поля формы редактирования объекта и возвращает true, если все проверки прошли успешно.
13. validateNumberFromText конвертирует непустую строку в целое число, возвращая –1, если строка пуста, или соответствующее значение при корректном вводе.
14. FormCreate (OnCreate) инициализирует генератор случайных чисел, счётчики ключей магазинов и складов, очищает фильтр и инициализирует деревья (shops, warehouses, shopsNames, warehousesNames) и хеш‑таблицу.
15. hideAllPanels скрывает все всплывающие панели интерфейса: pnCreateObj, pnCreateSelect, pnSelectObject, pnEditObj и pnFilterParams.
16. showPanel позиционирует переданную панель рядом с указанными координатами (с учётом границ imgMap) и делает её видимой.
17. resetPnCreateObj очищает текст всех полей формы создания объекта.
18. resetPnEditObj очищает текст всех полей формы редактирования объекта.
19. pnSelectObjectShow (OnMouseUp для TShape) скрывает панели, вычисляет центр графического элемента, сохраняет его тег в pnSelectObject.Tag и отображает панель действий над выбранным объектом.
20. pnObjectInfoShow (OnMouseEnter для TShape) определяет тип (магазин/склад) по тегу, извлекает данные из соответствующего дерева, заполняет метки панели информации (lbObjInfo\*) и показывает её рядом с объектом.
21. pnObjectInfoHide (OnMouseLeave) скрывает панель информации об объекте.
22. setFilterPanel сбрасывает все поля и флажки панели фильтрации к значениям по умолчанию.
23. cntFilteredItems подсчитывает активные условия фильтра (типы объектов, заполненные поля) и возвращает их количество.
24. btnSelectObjDeleteClick после подтверждения пользователя удаляет графический элемент TShape, удаляет узел из декартового дерева по ключу и удаляет соответствующее имя из дерева имён по хешу.
25. btnSelectObjCancelClick скрывает панель действий над выбранным объектом.
26. btnSelectObjEditClick подготавливает и отображает панель pnEditObj для редактирования выбранного объекта, заполняя поля текущими значениями из дерева.
27. btnEditObjConfirmClick при успешной валидации и подтверждении обновляет данные выбранного объекта (имя, адрес, вместимость), очищает форму и скрывает панели.
28. btnFilterClick скрывает все панели, заполняет панель pnFilterParams текущими значениями из структуры filter (тип объекта, адрес, диапазоны) и показывает её.
29. btnFilterCancelClick скрывает панель фильтрации.
30. btnFilterDefaultClick сбрасывает панель фильтрации к значениям по умолчанию через setFilterPanel.
31. btnFilterConfirmClick проверяет поля фильтрации, создаёт структуру filter через createFilter, обновляет надпись кнопки btnFilter с количеством активных условий и скрывает панели.
32. edFilterCapacityFromValChange при изменении значения проверяет диапазон «от–до» (capacity) и подсвечивает второе поле при нарушении.
33. edFilterCapacityToValChange при изменении значения проверяет диапазон «от–до» (capacity) и подсвечивает второе поле при нарушении.
34. edFilterUsedCapacityFromValChange при изменении значения проверяет диапазон «от–до» (usedCapacity) и подсвечивает второе поле при нарушении.
35. edFilterUsedCapacityToValChange при изменении значения проверяет диапазон «от–до» (usedCapacity) и подсвечивает второе поле при нарушении.
36. edFilterCapacityFromValExit при потере фокуса проверяет диапазон «от–до» (capacity) и окрашивает поле при ошибке.
37. edFilterCapacityToValExit при потере фокуса проверяет диапазон «от–до» (capacity) и окрашивает поле при ошибке.
38. edFilterUsedCapacityFromValExit при потере фокуса проверяет диапазон «от–до» (usedCapacity) и окрашивает поле при ошибке.
39. edFilterUsedCapacityToValExit при потере фокуса проверяет диапазон «от–до» (usedCapacity) и окрашивает поле при ошибке.
40. N11Click (пункт меню) скрывает все панели и центрирует панель pnCreateShipment внутри контейнера pnMapWrap.

**4.2 Описание процедур и функций модулей**

**4.2.1** Модуль CartesianTree

1. InitTree инициализирует декартово дерево, устанавливая корень в nil.
2. FindTreap выполняет рекурсивный поиск узла по заданному ключу в дереве; возвращает указатель на узел или nil, если ключ не найден.
3. ExistsPriority рекурсивно проверяет, присутствует ли в дереве узел с указанным приоритетом; возвращает True, если приоритет найден, иначе – False.
4. GenerateUniquePriority генерирует случайное целое число с помощью Random(MaxInt), пока оно не станет уникальным в дереве (проверяется через ExistsPriority), и возвращает его.
5. SplitTreap разбивает дерево t на два поддерева L и R по значению ключа: в L попадают узлы с ключом меньше заданного, в R – с ключом не меньше.
6. MergeTreap объединяет два декартовых дерева L и R в одно, сохраняя свойства кучи по полю Priority; возвращает новый корень.
7. InsertTreap создаёт новый узел с данными Data, генерирует для него уникальный приоритет, выполняет SplitTreap исходного дерева по ключу Data^.Key, присоединяет полученные поддеревья к новому узлу и назначает его корнем.
8. EraseTreap удаляет из дерева узел с заданным ключом: рекурсивно находит его, затем заменяет корень поддерева на результат MergeTreap его левого и правого поддеревьев и освобождает память удалённого узла.
9. ClearTreap рекурсивно освобождает память всех узлов дерева (обход «лево‑право‑корень»), после чего устанавливает корень в nil.

**4.2.2** Модуль CartesianTreeByName

1. InitTreeName процедура, инициализирующая корень дерева, устанавливая его в значение nil. Используется для создания пустого дерева перед началом работы с ним.
2. FindTreapName функция, выполняющая рекурсивный поиск узла по заданному ключу. Если узел с таким ключом найден, возвращает указатель на него. В противном случае — nil.
3. ExistsPriorityName функция, проверяющая наличие узла с указанным приоритетом в дереве. Используется для генерации уникальных приоритетов, чтобы избежать дубликатов.
4. GenerateUniquePriorityName функция, которая генерирует уникальный приоритет (целое число), не встречающееся в текущем дереве. Повторяет генерацию, пока не найдёт уникальный вариант.
5. SplitTreapName процедура, разделяющая дерево на два поддерева в зависимости от ключа. Все узлы с ключами меньше заданного попадают в левое поддерево, остальные — в правое.
6. MergeTreapName функция, объединяющая два декартовых дерева в одно. Выбор корня дерева при объединении осуществляется на основе приоритетов узлов.
7. InsertTreapName процедура вставки нового узла по имени. Генерирует ключ на основе хеш-функции от строки, создаёт новый узел, разбивает существующее дерево и объединяет его с новым узлом.
8. EraseTreapName процедура удаления узла по заданному ключу. Если узел найден, он удаляется, а его левое и правое поддеревья объединяются в одно дерево.
9. ClearTreapName процедура, полностью очищающая дерево. Рекурсивно освобождает память всех узлов, после чего дерево становится пустым.

**4.2.3** Модуль Filter

1. InitFilter процедура, инициализирующая объект фильтра с начальными значениями. Все поля объекта устанавливаются в значения по умолчанию (например, для типа здания — 3, для улицы — пустая строка, для емкостей — -1). Используется для создания пустого фильтра перед началом работы с ним.
2. CreateFilter процедура, создающая фильтр с заданными параметрами. Заполняет поля объекта типа TFilter значениями, переданными в качестве аргументов, такими как тип объекта (магазин, склад и т. д.), улица, номера дома и здания, а также диапазоны емкостей.

**4.2.4** Модуль GetKeys

1. getShopKey функция, возвращающая уникальный ключ для магазина. Каждый вызов функции возвращает текущий ключ shopKey, а затем увеличивает его на 1, обеспечивая уникальность ключей для каждого следующего вызова.
2. getWarehouseKey функция, возвращающая уникальный ключ для склада. Каждый вызов функции возвращает текущий ключ warehouseKey, а затем увеличивает его на 1, обеспечивая уникальность ключей для каждого следующего вызова.

**4.2.4** Модуль Hash

1. initHash процедура, инициализирующая параметры хеш-функции. Устанавливает значения для переменных p и m, а также вычисляет степени числа p по модулю m и сохраняет их в массив pows. Этот массив используется для вычисления хеша строк.
2. getHash функция, вычисляющая хеш строки на основе предварительно инициализированных параметров. Для каждого символа строки она вычисляет значение, умножая его ASCII-код на соответствующую степень p (с учетом индекса символа) и суммируя результаты. Итоговое значение возвращается по модулю m.

**4.2.5** Модуль Messages

1. getConfirmation функция, отображающая диалоговое окно с запросом на подтверждение действия. В окне отображаются кнопки "Да" и "Нет". Если пользователь нажимает "Да", функция возвращает True, если "Нет" — возвращает False. Диалоговое окно также настраивает заголовок и кнопки в зависимости от результата.
2. showError процедура, отображающая диалоговое окно с сообщением об ошибке. Окно содержит кнопку "Ок", которая закрывает диалог после нажатия. Сообщение для отображения передается как строка аргумента в процедуру.

**4.2.6** Модуль Validation

1. validateLength функция, проверяющая длину текста в поле ввода. Если текст пустой (после удаления начальных пробелов), цвет поля ввода меняется на красный, а функция возвращает false. Если текст не пустой, возвращает true.
2. validateLetters функция, проверяющая, состоит ли текст в поле ввода только из букв, пробелов и цифр. Если встречаются символы, не являющиеся пробелом, буквой или цифрой, цвет поля ввода меняется на красный, а функция возвращает false. Если все символы в допустимом диапазоне, функция возвращает true.
3. validateAll функция, которая комбинирует валидацию длины текста и проверки на буквы. Если обе проверки успешны, функция возвращает true, иначе — false.
4. validateFromTo функция, проверяющая, что значение в первом поле ввода не больше значения во втором поле ввода. Если первое значение больше второго, функция возвращает false. В противном случае — true.

**5 Тестирование, проверка работоспособности и анализ полученных результатов**

**5.1 Описание тестов, результаты тестирования**

Таблица 5.1 – Тестирование функционала программного средства

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Специфика тестирования** | **Номер теста** | **Вводимые данные** | **Ожидаемый результат** | **Полученный результат** |
| Запуск программы | 1 | Двойной щелчок левой кнопкой мыши по программе | Появление окна программы с картой | Тест пройден |
| Фильтрация | 2 | Нажатие на кнопку “фильтр” на главном экране | Появление окна со значениями для фильтра по центру экрана | Тест пройден |
| Создание магазина | 3 | Нажатие на кнопку "Создать магазин" и ввод имени, улицы, дома, емкости | Появление магазина на карте с соответствующими данными | Тест пройден |
| Создание склада | 4 | Нажатие на кнопку "Создать склад" и ввод имени, улицы, дома, емкости | Появление склада на карте с соответствующими данными | Тест пройден |
| Редактирование магазина | 5 | |  | | --- | |  |   Выбор магазина для редактирования, изменение данных (имя, улица, дом) | Обновление данных магазина на карте | Тест пройден |
| Редактирование склада | 6 | Выбор склада для редактирования, изменение данных (имя, улица, дом) | Обновление данных склада на карте | Тест пройден |
| Валидация данных при создании объекта | 7 | Ввод неправильных данных (недопустимые символы в имени или улице) | Появление ошибки и запрет на создание объекта | Тест пройден |
| Валидация данных при фильтрации | 8 | Ввод некорректных значений в фильтре (нечисловые значения в поле "Вместимость") | Появление сообщения об ошибке или запрет на фильтрацию с ошибочными данными | Тест пройден |
| Выбор объекта на карте | 9 | Клик по объекту (магазин/склад) на карте | Появление панели с подробной информацией о выбранном объекте | Тест пройден |
| Фильтрация объектов по типу | 10 | Установка фильтра "Только магазины" и нажатие кнопки "Применить" | Отображение только объектов типа "Магазин" на карте | Тест пройден |
| Фильтрация объектов по адресу | 11 | Ввод улицы и дома в фильтре, нажатие кнопки "Применить" | Отображение объектов, которые соответствуют фильтру по адресу | Тест пройден |
| Сброс фильтра | 12 | Нажатие на кнопку "Сброс" | Очистка всех фильтров, возврат к состоянию с отображением всех объектов | Тест пройден |
| Работа с полями фильтрации (интервал емкости) | 13 | Ввод минимального и максимального значения емкости в фильтре и нажатие кнопки "Применить" | Отображение объектов, которые соответствуют диапазону емкости | Тест пройден |
| Успешное создание объекта с уникальным именем | 14 | Ввод уникального имени для магазина/склада и нажатие кнопки "Подтвердить" | Создание нового объекта на карте с уникальным именем и соответствующими данными | Тест пройден |
| Проверка существующего имени объекта | 15 | Ввод существующего имени магазина/склада и попытка создать новый объект | Появление сообщения об ошибке и отказ в создании нового объекта | Тест пройден |
| Проверка обновления объектов после редактирования | 16 | Редактирование данных объекта (например, изменения вместимости склада) | Обновление отображаемых данных на карте и на панели информации | Тест пройден |
| Тестирование кнопки "Отмена" в процессе создания объекта | 17 | Нажатие кнопки "Отмена" в процессе создания объекта | Очистка введенных данных и возврат в начальное состояние формы | Тест пройден |
| Проверка работы кнопки "Удалить" | 18 | Нажатие на кнопку "Удалить" на объекте | Удаление объекта с карты и его исчезновение из базы данных | Тест пройден |

Все тесты прошли успешно.