-- Создание базы данных

CREATE DATABASE RepairRequestsDB;

GO

USE RepairRequestsDB;

GO

-- Таблица пользователей

CREATE TABLE Users (

UserId INT IDENTITY(1,1) PRIMARY KEY,

UserName NVARCHAR(100) NOT NULL,

Email NVARCHAR(100) NOT NULL UNIQUE,

CreatedAt DATETIME DEFAULT GETDATE()

);

GO

-- Таблица статусов заявок

CREATE TABLE RequestStatuses (

StatusId INT IDENTITY(1,1) PRIMARY KEY,

StatusName NVARCHAR(50) NOT NULL UNIQUE

);

GO

-- Заполнение таблицы статусов

INSERT INTO RequestStatuses (StatusName) VALUES ('В ожидании');

INSERT INTO RequestStatuses (StatusName) VALUES ('В процессе');

INSERT INTO RequestStatuses (StatusName) VALUES ('Завершено');

INSERT INTO RequestStatuses (StatusName) VALUES ('Отменено');

GO

-- Таблица оборудования

CREATE TABLE Equipment (

EquipmentId INT IDENTITY(1,1) PRIMARY KEY,

EquipmentName NVARCHAR(100) NOT NULL,

SerialNumber NVARCHAR(100) NOT NULL UNIQUE,

CreatedAt DATETIME DEFAULT GETDATE()

);

GO

-- Таблица заявок на ремонт

CREATE TABLE RepairRequests (

RequestId INT IDENTITY(1,1) PRIMARY KEY,

UserId INT NOT NULL,

EquipmentId INT NOT NULL,

StatusId INT NOT NULL,

Description NVARCHAR(255) NOT NULL,

CreatedAt DATETIME DEFAULT GETDATE(),

FOREIGN KEY (UserId) REFERENCES Users(UserId),

FOREIGN KEY (EquipmentId) REFERENCES Equipment(EquipmentId),

FOREIGN KEY (StatusId) REFERENCES RequestStatuses(StatusId)

);

GO

БЛОК СХЕМА

Блок-схема функции расчета количества заявок

[Овал] Начало

|

v

[Параллелограмм] Ввод списка заявок

|

v

[Прямоугольник] Инициализация счетчика = 0

|

v

[Ромб] Для каждой заявки в списке?

|

+--- Да ---> [Прямоугольник] Увеличить счетчик на 1

| |

| v

| [Ромб] Есть еще заявки?

| |

| +--- Да ---> [Прямоугольник] Увеличить счетчик на 1

| |

| +--- Нет ---->

| |

v v

[Прямоугольник] Возврат значения счетчика

|

v

[Овал] Конец

▎Обозначения элементов:

1. [Овал]: Используется для обозначения начала и конца процесса.

• Пример: "Начало", "Конец".

2. [Параллелограмм]: Используется для ввода и вывода данных.

• Пример: "Ввод списка заявок".

3. [Прямоугольник]: Используется для обозначения процессов или действий.

• Пример: "Инициализация счетчика = 0", "Увеличить счетчик на 1", "Возврат значения счетчика".

4. [Ромб]: Используется для принятия решений (условий).

• Пример: "Для каждой заявки в списке?", "Есть еще заявки?".

▎Описание блок-схемы

1. [Овал] Начало: Начало процесса расчета.

2. [Параллелограмм] Ввод списка заявок: Пользователь вводит список заявок, который необходимо обработать.

3. [Прямоугольник] Инициализация счетчика = 0: Создается переменная-счетчик, которая инициализируется значением 0.

4. [Ромб] Для каждой заявки в списке?: Начало цикла, который проходит по всем заявкам в списке.

5. [Прямоугольник] Увеличить счетчик на 1: Для каждой заявки в списке счетчик увеличивается на 1.

6. [Ромб] Есть еще заявки?: Проверка, остались ли еще заявки для обработки.

7. [Прямоугольник] Возврат значения счетчика: Функция возвращает значение счетчика, которое представляет общее количество заявок.

8. [Овал] Конец: Завершение процесса.

Создание алгоритма в виде таблицы по правилам ГОСТ 2.105 требует структурированного подхода, где каждый шаг алгоритма представлен в виде отдельной строки таблицы с указанием номера шага, описания действия и других необходимых атрибутов.

Вот пример алгоритма расчета количества заявок, оформленный в соответствии с ГОСТ 2.105:

▎Таблица алгоритма

| Номер шага | Описание действия | Примечание |

|------------|-------------------------------------------|----------------------------------|

| 1 | Начало | |

| 2 | Ввод списка заявок | Пользователь вводит данные |

| 3 | Инициализация счетчика = 0 | Создаем переменную-счетчик |

| 4 | Проверка: есть ли заявки в списке? | Условие: если список не пуст |

| 5 | Да | Переход к шагу 6 |

| 6 | Увеличить счетчик на 1 | Счетчик увеличивается на 1 |

| 7 | Проверка: есть ли еще заявки? | Условие: если список не пуст |

| 8 | Да | Переход к шагу 6 |

| 9 | Нет | Переход к шагу 10 |

| 10 | Возврат значения счетчика | Возвращаем общее количество |

| 11 | Конец | Завершение алгоритма |

▎Объяснение таблицы:

1. Номер шага: Уникальный номер для каждого действия, который помогает отслеживать последовательность выполнения.

2. Описание действия: Четкое и лаконичное описание того, что происходит на каждом шаге.

3. Примечание: Дополнительная информация о действии, например, условия или пояснения.

▎Примечания по оформлению:

• Таблица должна быть оформлена в соответствии с требованиями, включая единый шрифт и размеры ячеек.

• Заголовки таблицы должны быть выделены (например, жирным шрифтом).

• Если алгоритм длинный, можно разбить его на несколько таблиц для удобства восприятия.