Федеральное государственное автономное образовательное учреждение высшего образования
«Пермский национальный исследовательский политехнический университет»

Творчески работа
Автоматизированное рабочее место «Контролёр технологических процессов»

Выполнили студенты группы ИВТ-23-2Б
Муравьев Дмитрий
Злыгостев Денис
Ломоев Давид
Глебко Георгий
Проверила: доцент кафедры ИТАС
Ольга Андреева Полякова

2023

Описание программы
Программа "Контроллер технологических процессов": Усовершенствование производственных операций

Программа "Контроллер технологических процессов" представляет собой мощное программное решение, разработанное для оптимизации производственных процессов и повышения их эффективности.

Основные функции:

• Отслеживание этапов производства: Полный контроль над производственными этапами, позволяющий визуализировать процесс и выявлять области для улучшения.

• Учет входных ресурсов: Автоматизированный учет материалов, сырья и других ресурсов, включая отслеживание их количества на складе.

• Автоматизированная генерация заказов на ресурсы: Система автоматически определяет недостающие ресурсы и создает заказы на их пополнение, устраняя задержки в производственном процессе.

• Учет выходных ресурсов: Отслеживание готовой продукции и других выходных ресурсов, что обеспечивает своевременное реагирование на изменения спроса и предложения.

• Интеграция с оборудованием: Совместимость с различными типами производственного оборудования, позволяющая получать данные в режиме реального времени и автоматизировать управление процессами.

• Передача указаний от руководства: Получение указаний от руководства и их автоматическое распространение на производственный уровень, устраняя путаницу и повышая точность выполнения.

• Упрощение управления цехом: интуитивно понятный интерфейс, упрощающий работу мастеров цеха, предоставляя им всю необходимую информацию под рукой.

Преимущества:

• Увеличение производительности за счет сокращения простоев и оптимизации процессов.

• Повышение точности и своевременности выполнения заказов.

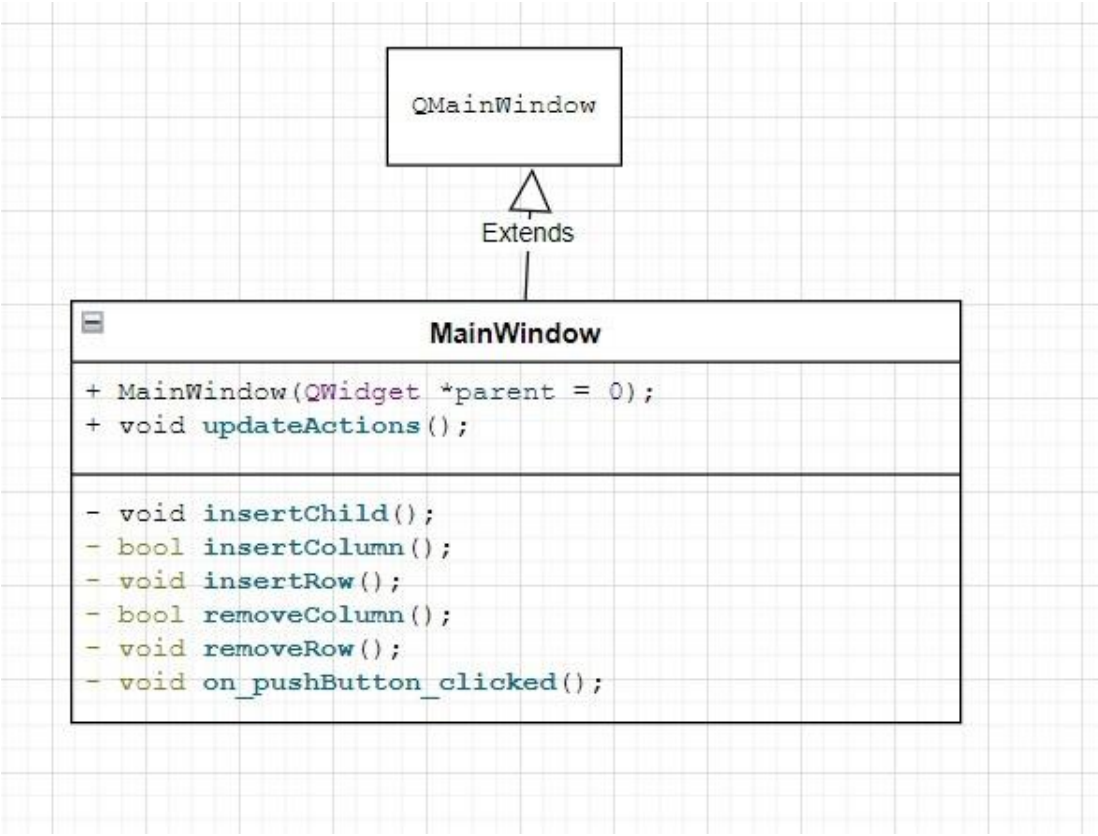
• Улучшение контроля запасов и управления цепочкой поставок.

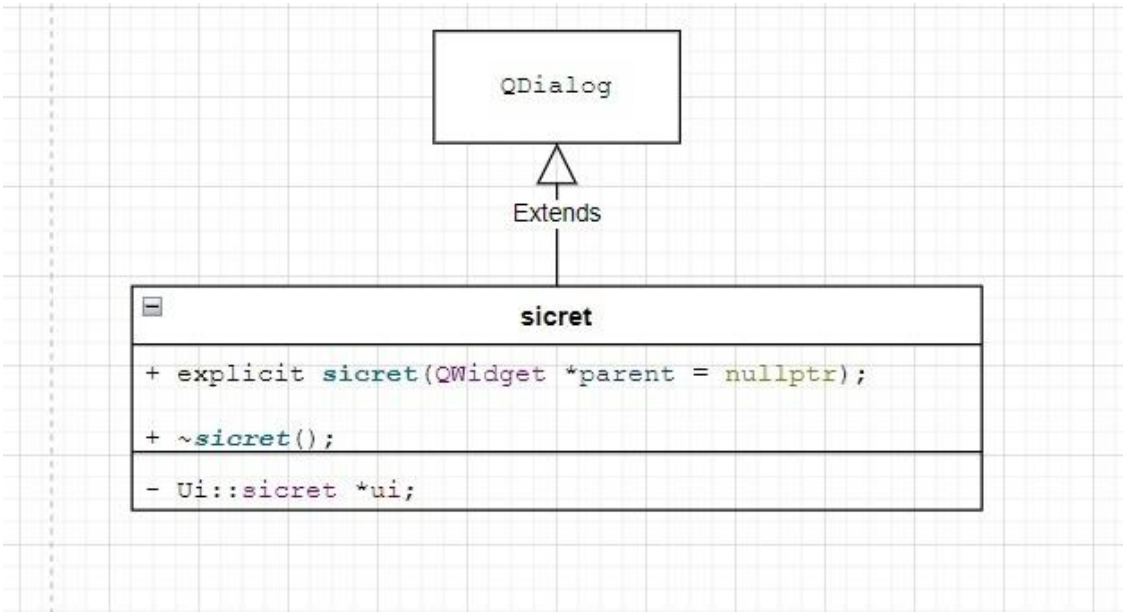
• Сокращение потерь и брака за счет автоматизации и повышения видимости.

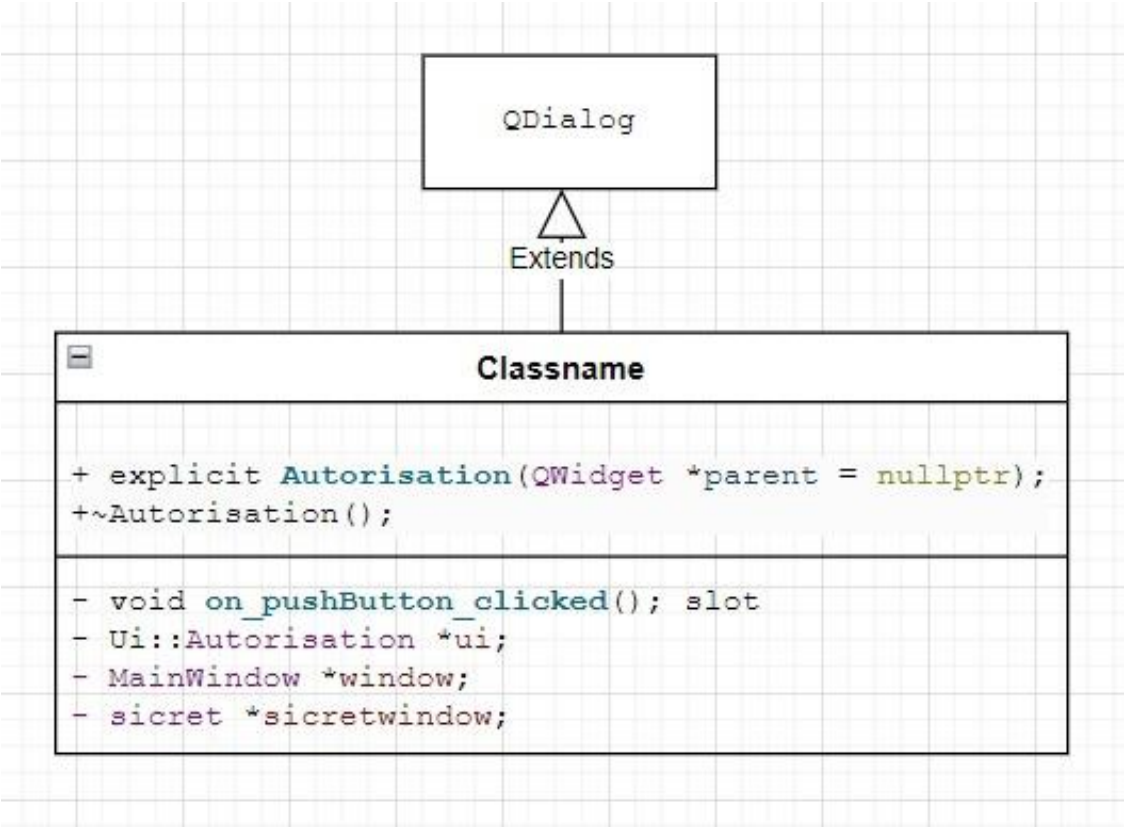
• Упрощение работы мастеров цеха и повышение их эффективности.

Программа "Контроллер технологических процессов" является незаменимым инструментом для предприятий, стремящихся улучшить свои производственные операции, повысить качество продукции и снизить общие затраты.

UML – Диаграмма




QDialog
Extends
sicret
+ explicit sicret(QWidget *parent = nullptr);
+ ~sicret();
- Ui::sicret *ui;


QDialog
Extends
Classname
+ explicit Autorisation(QWidget *parent = nullptr);
+~Autorisation();
- void on_pushButton_clicked(); slot
- Ui::Autorisation *ui;
- MainWindow *window;
- sicret *sicretwindow;

Описание всех классов

Класс Autorisation

Назначение:

Класс Autorisation представляет собой диалоговое окно авторизации, которое запрашивает у пользователя имя пользователя и пароль для доступа к основному окну приложения (MainWindow).

Устройство:

• Приватные переменные-члены:

  • ui: Указатель на объект пользовательского интерфейса, созданный Qt Designer.

  • window: Указатель на объект главного окна (MainWindow).

  • sicretwindow: Указатель на объект окна с секретной информацией (sicret).

• Конструктор:

  • Создает диалоговое окно авторизации и настраивает его пользовательский интерфейс.

• Метод on_pushButton_clicked():

  • Обрабатывает нажатие кнопки "Войти".

  • Проверяет правильность введенного имени пользователя и пароля.

  • Если данные верны, создает окно с секретной информацией и отображает его.

• Деструктор:

  • Удаляет объект диалогового окна авторизации.

Взаимодействие с другими классами:

• Класс Autorisation взаимодействует с классом MainWindow через указатель window. Когда пользователь успешно авторизуется, создается новый объект класса sicret и отображается в окне MainWindow.

Класс MainWindow

Назначение класса:

Класс MainWindow представляет главное окно приложения Qt, предназначенного для работы с табличными данными. Он предоставляет графический пользовательский интерфейс (GUI) и функциональность для управления и редактирования таблицы.

Устройство класса:

Наследование:

• Класс MainWindow наследует от QMainWindow. Это базовый класс для всех главных окон в Qt и предоставляет функции для управления строкой меню, панелью инструментов, строкой состояния и центральным виджетом.

• Класс MainWindow также наследует от Ui::MainWindow, который является автоматически сгенерированным классом Qt, который предоставляет доступ к элементам пользовательского интерфейса, определенным в файле .ui.

Свойства:

• Ui::MainWindow предоставляет доступ к различным элементам пользовательского интерфейса, таким как кнопки, меню и виджеты.

Методы:

• Конструктор: Конструктор создает экземпляр объекта MainWindow и инициализирует его графический пользовательский интерфейс.

• Public Slots:

* updateActions():

Обновляет состояние действий в главном окне в зависимости от состояния таблицы.

• Private Slots:

• insertChild(): Вставляет дочерний элемент в родительский элемент в таблице.

• insertColumn():

Вставляет столбец в таблицу. Возвращает true, если вставка прошла успешно, в противном случае - false.

• insertRow(): Вставляет строку в таблицу.

- removeColumn():

Удаляет столбец из таблицы. Возвращает true, если удаление прошло успешно, в противном случае - false.

- removeRow():

Удаляет строку из таблицы.

Класс TreeItem

Назначение класса:

Класс TreeItem представляет элемент в древовидной структуре данных. Он используется для создания и управления иерархическими данными, такими как древовидные структуры или таблицы.

Устройство класса:

Конструктор:

• TreeItem(const QVector<QVariant> &data, TreeItem *parent = 0):

Конструктор создает экземпляр объекта TreeItem с заданными данными и родительским элементом.

Методы:

• child(int number): Возвращает дочерний элемент с указанным номером.

• childCount() const:

Возвращает количество дочерних элементов.

• columnCount() const:

Возвращает количество столбцов в элементе.

• data(int column) const:

Возвращает данные в указанном столбце.

• insertChildren(int position, int count, int columns): Вставляет указанное количество дочерних элементов в указанную позицию с указанным количеством столбцов. Возвращает true, если вставка прошла успешно, в противном случае - false.

• insertColumns(int position, int columns): Вставляет указанное количество столбцов в указанную позицию. Возвращает true, если вставка прошла успешно, в противном случае - false.

• parent(): Возвращает родительский элемент.

• removeChildren(int position, int count): Удаляет указанное количество дочерних элементов из указанной позиции. Возвращает true, если удаление прошло успешно, в противном случае - false.

• removeColumns(int position, int columns): Удаляет указанное количество столбцов из указанной позиции. Возвращает true, если удаление прошло успешно, в противном случае - false.

• childNumber() const: Возвращает номер текущего элемента среди дочерних элементов его родителя.

• setData(int column, const QVariant &value): Устанавливает данные в указанном столбце. Возвращает true, если установка данных прошла успешно, в противном случае - false.

Класс TreeModel

Назначение класса:

Класс TreeModel представляет древовидную модель данных, основанную на классе TreeItem. Он предоставляет универсальный интерфейс для управления иерархически организованными данными, которые отображаются с помощью QTreeView. TreeModel поддерживает различные операции редактирования и манипуляции данными, такие как вставка, удаление и изменение строк и столбцов.

Устройство класса:

Наследование:

• Класс TreeModel наследует от QAbstractItemModel. Это базовый класс для всех моделей, предоставляющих данные для представлений Qt, таких как QTreeView и QTableView.

Конструктор:

• TreeModel(const QStringList &headers, const QString &data, QObject *parent = 0): Конструктор создает экземпляр объекта TreeModel с заданными заголовками столбцов, данными и родительским объектом.

Методы:

• data(const QModelIndex &index, int role) const:

Возвращает данные, связанные с указанным индексом модели, для указанной роли (например, отображаемый текст или данные модели).

• headerData(int section, Qt::Orientation orientation,int role = Qt::DisplayRole) const:

Возвращает данные заголовка для указанного раздела и ориентации.

• index(int row, int column, const QModelIndex &parent = QModelIndex()) const:

Возвращает индекс модели для указанной позиции в указанном родительском индексе.

• parent(const QModelIndex &index) const:

Возвращает родительский индекс для указанного индекса модели.

• rowCount(const QModelIndex &parent = QModelIndex()) const:

Возвращает количество строк в указанном родительском индексе.

• columnCount(const QModelIndex &parent = QModelIndex()) const:

Возвращает количество столбцов в указанном родительском индексе.

• flags(const QModelIndex &index) const:

Возвращает флаги для указанного индекса модели, которые определяют поведение редактирования и выбора.

• setData(const QModelIndex &index, const QVariant &value,int role = Qt::EditRole):

Устанавливает данные для указанного индекса модели и роли.

• setHeaderData(int section, Qt::Orientation orientation, const QVariant &value, int role = Qt::EditRole):

Устанавливает данные заголовка для указанного раздела, ориентации и роли.

• insertColumns(int position, int columns,const QModelIndex &parent = QModelIndex()):

Вставляет указанное количество столбцов в указанную позицию в указанном родительском индексе.

• removeColumns(int position, int columns,const QModelIndex &parent = QModelIndex()):

Удаляет указанное количество столбцов из указанной позиции в указанном родительском индексе.

• insertRows(int position, int rows,const QModelIndex &parent = QModelIndex()):

Вставляет указанное количество строк в указанную позицию в указанном родительском индексе.

• removeRows(int position, int rows, const QModelIndex &parent = QModelIndex()):

Удаляет указанное количество строк из указанной позиции в указанном родительском индексе.

Класс sicret

Назначение класса:

Класс sicret представляет секретное окно в приложении. Оно предназначено для отображения списка всех возможных логинов и паролей, если пользователь вводит правильные учетные данные при авторизации в основном приложении.

Устройство класса:

Наследование:

Класс sicret наследует от QDialog. Это базовый класс для всех диалоговых окон в Qt и предоставляет функции для управления окном, его размером, положением и взаимодействием с пользователем.

Члены данных:

• Ui::sicret ui: Указатель на автоматически сгенерированный класс пользовательского интерфейса Qt, который предоставляет доступ к элементам пользовательского интерфейса, определенным в файле .ui.

• QPoint p: Переменная, используемая для хранения положения окна.

Методы:

Конструктор:

Конструктор создает экземпляр объекта sicret и инициализирует его графический пользовательский интерфейс.

Деструктор:

Деструктор освобождает память, выделенную для объекта sicret.

Защищенные методы:

Переопределенные защищенные методы предоставляют дополнительную функциональность для управления окном и обработкой событий.

Задача коммивояжера

1. Постановка задачи:

Задача коммивояжера заключается в поиске оптимального маршрута, проходящего через указанные города по одному разу с последующим возвратом в исходный город.

2. Описание решения:

Программу можно разбить на несколько условных зон:

- Полотно

- Панель настроек

- Кнопка запуска алгоритма

Полотно:

Данная область обеспечивает возможность создания графа. Для того, чтобы добавить вершину графа, необходимо нажать на

случайном пустом месте на полотне, после чего появится

вершина графа (и если раньше уже были созданы вершины, то от

текущей вершины и к существующим будут созданы ребра из

случайно заданными величинами от 1 до 25).

Также введено ограничение на размещение вершин, в местах

непосредственной близости к другим вершинам, на других вершинах и на цифрах.

Панель настроек:

Программа содержит различные инструменты на панели

настроек, обеспечивающих работу с графом.

Выпадающий список «Номер города:»:

Позволяет выбрать вершину из списка, для дальнейших операций над ней. При добавлены вершин на холст, список автоматически обновляется.

Поле «Количество дорог ведущих из города:»:

Данное поле показывает количество ребер, которые прилегают к выбранной вершине.

Редактирование ребер:

Выпадающие списки «Дорога между:» и «и» позволяют выбрать ребро для редактирования, между выбранными из списка, вершинами.

Поле ввода «Значение ребра:»:

Предоставляет возможность изменить значение ребра. Значение перезаписывается автоматически, сразу после ввода\стирания цифры. Диапазон: 1 - 100.

Кнопка «Удалить ребро»:

При нажатии удаляет ребро, однако при условии, что в случае удаления данного ребра, граф останется гамильтоновым (останется возможность обрезать все вершины графа один раз и вернуться к начальной точке).

Добавление ребер:

Выпадающие списки «Дорога между:» и «и» позволяют выбрать вершины, между которыми можно добавить новое ребро.

Поле ввода «Значение ребра:»:

Предоставляет возможность задать значение для нового ребра.

Диапазон: 1 - 100.

Кнопка «Добавить ребро»:

При нажатии добавляет ребро между указанными вершинами.

Чекбокс «Показать ограничения»:

При постановке галочки, отобразятся зоны-ограничения, которые указывают область, в которой невозможно создать новую вершину.

Кнопка запуска алгоритма:

Наконец, после создания графа, и настройке его вершин, можно запустить алгоритм поиска оптимального пути.

Данное действие осуществляется нажатием кнопки «Метод ветвей и границ»

После нажатия кнопки, программа найдет и отобразит найденный путь, перекрасив ребра, входящие в данный путь, в красный цвет.

Код программ можно посмотреть в github репозитории

https://github.com/Dmitriy-Mur/Creative-work