Київський національний університет

імені Тараса Шевченка

Факультет РЕКС

**Звіт з лабораторної роботи №6**

на тему: « ООП + Проект + Структури даних»

Виконав:

Біленко А.Р. , 1 курс, група ПФНКТ

Прийняла:

Єфіменко Світлана Володимирівна

Київ – 2023

## **Постановка завдання**

## Мета роботи:

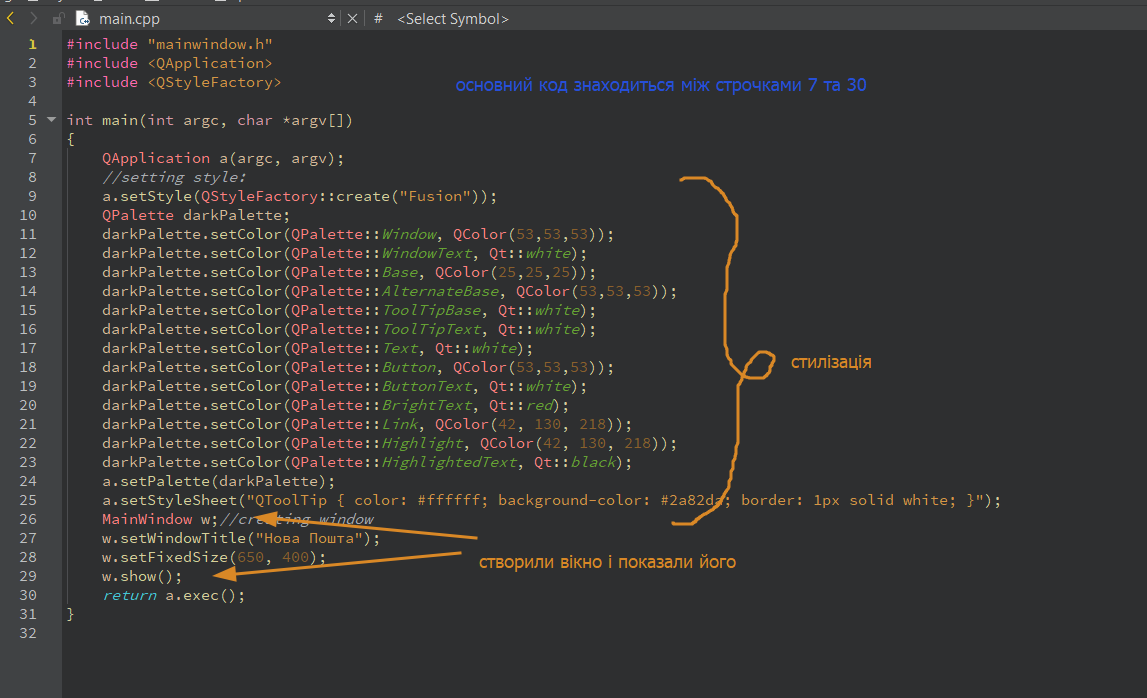
Реалізувати чергу, зробити проект.

## Умова:

1. Реалізувати чергу (Queque) інформації про відправлення Нової пошти (для елементів черги виконується принцип FIFO – first input, first output, тобто відправлення, що прибуло на склад першим, першим і відправляється, отже, видаляється із голови черги. А нове відправлення може додаватись лише в кінець черги). У кожній структурі – елементі черги, передбачити збереження інформації про номер накладної, вагу відправлення, місце призначення, категорію відправлення, наприклад, одяг, їжа, книга, тощо). Реалізувати методи побудови такої черги, визначення кількості відправлень у конкретне задане місце призначення, обчислення сумарної ваги відправлень певної категорії, пошук в черзі відправлення з заданим номером товарної накладної.

## Хід роботи:

main.cpp:



mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "queue.hpp"

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public *QMainWindow*

{

    Q\_OBJECT

public:

    MainWindow(*QWidget* \**parent* = nullptr);//constructor

    ~MainWindow();//destructor

private slots:

    ///

    /// \brief on\_deliver\_button\_clicked

    /// dequeues and displays a message (what was delivered)

    void on\_deliver\_button\_clicked();

    ///

    /// \brief on\_deliver\_all\_button\_clicked

    /// delivers untill queue is empty

    void on\_deliver\_all\_button\_clicked();

    ///

    /// \brief on\_browse\_button\_clicked

    /// that one "..." button, triggers file dialog and sets file\_path\_lineEdit to a chosen by user file path

    void on\_browse\_button\_clicked();

    ///

    /// \brief on\_add\_from\_file\_button\_clicked

    /// reads all packages from selected file info in format:

    /// {invoice} {weight} {delivery type (one word)} {destination place (one word)}

    /// and adds those packages to a queue

    void on\_add\_from\_file\_button\_clicked();

    ///

    /// \brief on\_add\_delivery\_manually\_button\_clicked

    /// reads user's input (comboboxes and weight\_lineEdit) and enqueues the created package

    /// shows error message if lineEdit input cannot be converted to a number or not all comboboxes are selected

    void on\_add\_delivery\_manually\_button\_clicked();

    ///

    /// \brief on\_find\_button1\_clicked

    /// finding number of deliveries to selected destination, shows output as a messageBox

    void on\_find\_button1\_clicked();

    ///

    /// \brief on\_find\_button2\_clicked

    /// finding summary weight of deliveries in selected category, shows output as a messageBox

    void on\_find\_button2\_clicked();

    ///

    /// \brief on\_find\_button3\_clicked

    /// lookup by invoice, invoice example: "INV210432". Shows output in a messageBox

    void on\_find\_button3\_clicked();

    ///

    /// \brief on\_exit\_button\_clicked

    /// asks to resumbit and quits the whole program.

    void on\_exit\_button\_clicked();

    ///

    /// \brief on\_show\_all\_button\_clicked

    /// traverses the whole queue and shows all the packages info in a messageBox.

    void on\_show\_all\_button\_clicked();

private:

    Ui::MainWindow \*ui;

    Queue\* queue{nullptr};//will be created in constructor and deleted in destructor

};

#endif // MAINWINDOW\_H

MainWindow наслідує від передвизначеного в фреймворку класу QMainWindow.

Неймспейс та кістяк класу був попередньо згенерований. Неймпспейс тут для того, щоб компілятор розрізняв два класи MainWindow та той що згенерується з ui файлу.

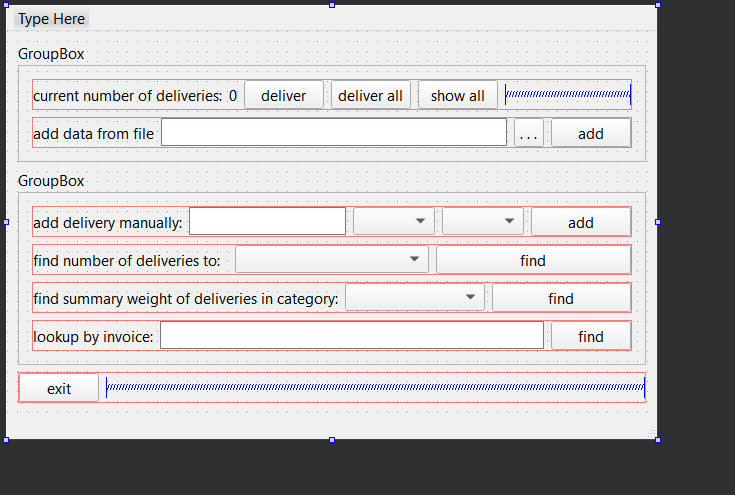
Q\_OBJECT означає що даний клас працюватиме за законами об’єктів цього фреймворку, тобто: не можна їх вручну видаляти(використовувати delete до них) (це зробить система),

працюють відносини батьківський\_клас-дочірній\_клас (щоб влаштувати ці відносини потрібно в коструктор дочірного класу при створенні об’єкту додати покажчик на батьківський. При видаленні батьківського видаляються всі дочірні об’єкти),

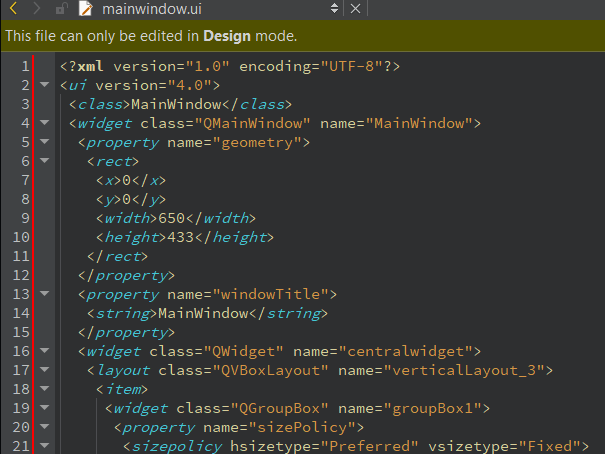
працює також концепція сигналів-слотів: наприклад, в даній роботі сигналами є наскання на кнопки, а слотами - фукції які як наслідок виконуються.(це лише один простий приклад, направді можливостей безліч)

Можна було (і було б безумовно краще) з’єднувати сигнали зі слотами вручну використовуючи метод connect(), але з огляду на обмеження в часі було використано функції по типу on\_button\_clicked(), з’єднання яких з кнопками влаштовується за нас. Проблема таких функцій в тому, що вони легко перестають працювати належним чином при зміні назви кнопки (з чим я і стикнувся в ході переплановки дизайну програми), тим не менш, вони просто є трішки застарілими, а в іншому працюють як заплановано.

mainwindow.ui

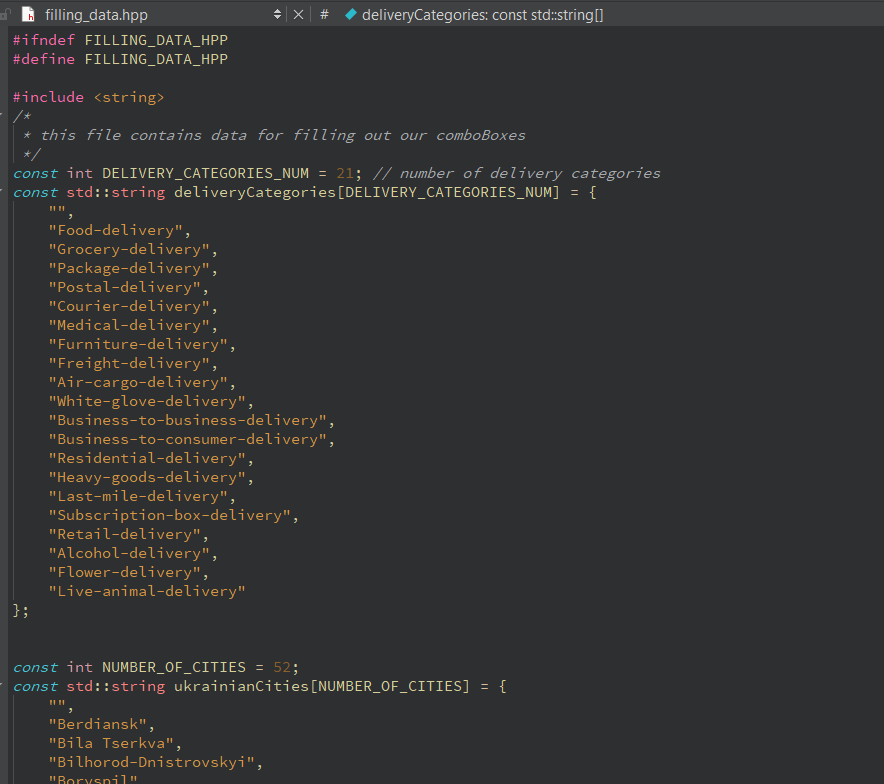


Це звичайний xml файл:



Використовувати ui файли дуже просто (як віндовс форми). Можна обійтися і без них, але можу сказати, що програмувати елементи вручну (а особливо layout-и) може стати дуже марудною справою оскільки важче бачити кінцеві зміни вікна.

filling\_data.hpp:



Файл, що містить елементи для заповнення комбо-боксів.

queue.hpp:

#ifndef QUEUE\_HPP

#define QUEUE\_HPP

#include <string>

///

/// \brief Package \struct is a node, and our queue will contain list of these nodes

///

struct *Package*{

    double weight{};

    std::*string* invoice\_number{""};

    std::*string* destination{""};

    std::*string* kind\_of\_delivery{""};

*Package*\* next{nullptr};

};

///

/// \brief The Queue \class had to be manually constructed (that was the main task)

///

class *Queue*{

public:

    /// \brief destrcutor

    /// empties the queue

    ~Queue();

    ///

    /// \brief Enqueue \fn inserts a package to the rear of the queue

    /// \param *package* will be inserted to a queue

    ///

    void Enqueue(const *Package* *package*);

    ///

    /// \brief Dequeue \fn pops the package at the front if available

    ///

    void Dequeue();

    ///

    /// \brief search\_deliveries\_to \fn traverses the queue in order to find number of occurencies of

    /// packages with destination = place \param.

    /// \param *place* is an input parameter

    /// \param *num\_of\_deliveries* is an output parameter

    ///

    void search\_deliveries\_to(const std::*string* *place*, int& *num\_of\_deliveries*);

    ///

    /// \brief sum\_weight\_in\_category \fn traverses the queue in order to find sum of weights of

    /// packages with kind\_of\_delivery = category \param.

    /// \param *category* is an input parameter

    /// \param *weight* is an output parameter

    ///

    void sum\_weight\_in\_category(const std::*string* *category*, double& *weight*);

    ///

    /// \brief find\_by\_invoice \fn finds the first from head package with invoice\_number = invoice \param.

    /// \param *invoice* is an input parameter

    /// \param *package* is an output parameter

    ///

    void find\_by\_invoice(const std::*string* *invoice*, *Package*& *package*);

    ///

    ///getters:

*Package*\* get\_head();

*Package*\* get\_rear();

private:

    //member variables

*Package*\* m\_head{nullptr};

*Package*\* m\_rear{nullptr};

};

#endif // QUEUE\_HPP

Чергу реалізовано як однозв’язний сиписок. Складається зі структур Package. Містить всі необхідні функції (і три додаткові для завдання) та специфікації в коді.

queue.cpp:

(реалізації)

#include "queue.hpp"

Queue::~Queue()

{

    while(m\_head != nullptr){

        Dequeue();

    }

}

void Queue::Enqueue(const *Package* *package*){

    Package\* tmp = **new** Package;

    tmp->weight = package.weight;

    tmp->invoice\_number = package.invoice\_number;

    tmp->destination = package.destination;

    tmp->kind\_of\_delivery = package.kind\_of\_delivery;

    if((m\_head == nullptr) && (m\_rear == nullptr)){

        m\_head = m\_rear = tmp;

        return;

    }

    m\_rear->next = tmp;

    m\_rear = tmp;

}

void Queue::Dequeue(){

    Package\* tmp = m\_head;

    if(m\_head == nullptr){

        return;

    }

    if(m\_head == m\_rear){

        m\_head = m\_rear = nullptr;

    }else{

        m\_head = m\_head->next;

    }

    delete tmp;

}

void Queue::search\_deliveries\_to(const std::*string* *place*, int& *num\_of\_deliveries*)

{

    Package\* tmp = m\_head;

    num\_of\_deliveries = 0;

    while(tmp != nullptr){

        if(tmp->destination == place){

            num\_of\_deliveries++;

        }

        tmp = tmp->next;

    }

}

void Queue::sum\_weight\_in\_category(const std::*string* *category*, double& *weight*)

{

    Package\* tmp = m\_head;

    weight = 0;

    while(tmp != nullptr){

        if(tmp->kind\_of\_delivery == category){

            weight += tmp->weight;

        }

        tmp = tmp->next;

    }

}

void Queue::find\_by\_invoice(const std::*string* *invoice*, *Package*& *package*)

{

    Package\* tmp = m\_head;

    while(tmp != nullptr){

        if(tmp->invoice\_number == invoice){

            package = \*tmp;

            return;

        }

        tmp = tmp->next;

    }

}

*Package* \*Queue::get\_head()

{

    return m\_head;

}

*Package* \*Queue::get\_rear()

{

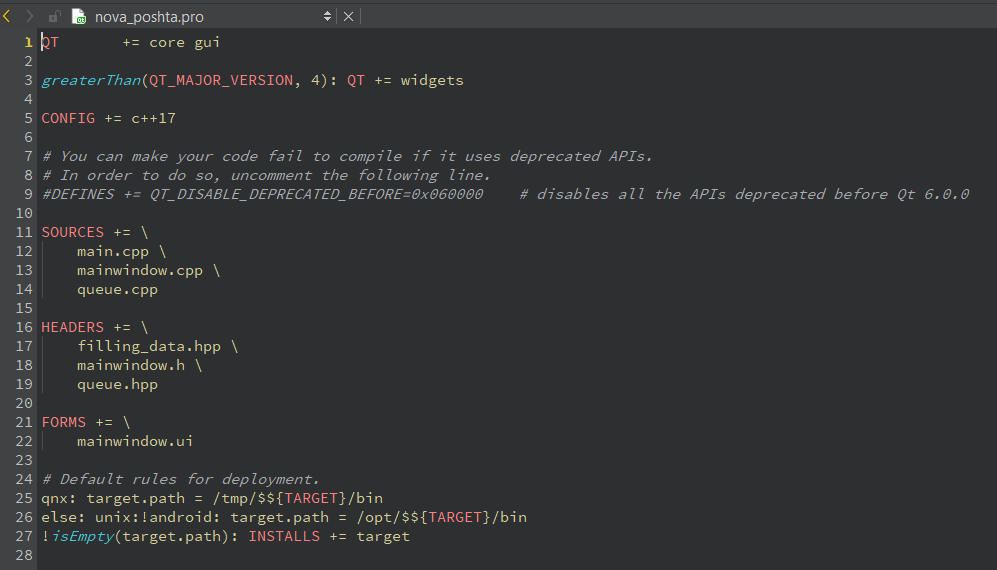
    return m\_rear;

}

**Ще два файли:**

nova\_poshta.pro

Файл проекту (qmake, щось на кшталт CMake, але для даного фреймворку):



mainwindow.cpp:

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "filling\_data.hpp"

#include <QMessageBox>

#include <QFileDialog>

#include <sstream>

#include <fstream>

MainWindow::MainWindow(QWidget \*parent)//MainWindow constructor

    : QMainWindow(parent)

    , ui(**new** Ui::MainWindow)

{

    ui->setupUi(*this*);

    //populating comboboxes

    for(int i = 0; i < NUMBER\_OF\_CITIES; i++){

        ui->comboBox1\_1->addItem(QString::fromStdString(ukrainianCities[i]));

    }

    ui->comboBox1\_1->setStyleSheet("combobox-popup: 0;");

    for(int i = 0; i < NUMBER\_OF\_CITIES; i++){

        ui->comboBox2->addItem(QString::fromStdString(ukrainianCities[i]));

    }

    ui->comboBox2->setStyleSheet("combobox-popup: 0;");

    for(int i = 0; i < DELIVERY\_CATEGORIES\_NUM; i++){

        ui->comboBox1\_2->addItem(QString::fromStdString(deliveryCategories[i]));

    }

    ui->comboBox1\_2->setStyleSheet("combobox-popup: 0;");

    for(int i = 0; i < DELIVERY\_CATEGORIES\_NUM; i++){

        ui->comboBox3->addItem(QString::fromStdString(deliveryCategories[i]));

    }

    ui->comboBox3->setStyleSheet("combobox-popup: 0;");

    //placeholder texts:

    ui->file\_path\_lineEdit->setPlaceholderText("there will be path to file");

    ui->invoice\_lineEdit->setPlaceholderText("invoice");

    ui->weight\_lineEdit->setPlaceholderText("weight (kilograms)");

    ui->file\_path\_lineEdit->setReadOnly(true);

    //exit button color:

    QPalette Pal(palette());

    Pal.setColor(QPalette::Button, QColor(53,0,0));

    ui->exit\_button->setAutoFillBackground(true);

    ui->exit\_button->setPalette(Pal);

    //initialisinng queue:

    queue = **new** Queue;

}

MainWindow::~MainWindow()//MainWindow destructor

{

    delete ui;

    delete queue;

}

void MainWindow::on\_deliver\_button\_clicked()

{

    if(queue->get\_head() != nullptr){

        QMessageBox::information(*this*, "delivered",

                                 (QString::fromStdString(queue->get\_head()->kind\_of\_delivery) + " package is of weight "

                                  + QString::number(queue->get\_head()->weight)+ "kg delivered to " + QString::fromStdString(queue->get\_head()->destination)

                                  + "\ninvoice: " + QString::fromStdString(queue->get\_head()->invoice\_number)));

        queue->Dequeue();

        //subtracting 1 from the dispayed number

        int count{-1};

        bool ok{true};

        count += ui->number\_of\_deliveries\_label->text().toInt(&ok, 10);

        ui->number\_of\_deliveries\_label->setText(QString::number(count));

    }else{

        QMessageBox::information(*this*, "message", "nothing to deliver");

    }

}

void MainWindow::on\_deliver\_all\_button\_clicked()

{

    while(queue->get\_head() != nullptr){

        queue->Dequeue();

        //subtracting 1 from the dispayed number

        int count{-1};

        bool ok{true};

        count += ui->number\_of\_deliveries\_label->text().toInt(&ok, 10);

        ui->number\_of\_deliveries\_label->setText(QString::number(count));

    }

    QMessageBox::information(*this*, "notification", "everything is delivered");

}

void MainWindow::on\_browse\_button\_clicked()

{

    ui->file\_path\_lineEdit->setText(QFileDialog::getOpenFileName());

}

void MainWindow::on\_add\_from\_file\_button\_clicked()

{

    if(ui->file\_path\_lineEdit->text() != ""){

        QString file\_path = ui->file\_path\_lineEdit->text();

        std::ifstream fin;

        fin.open(file\_path.toStdString());

        if(fin){

            std::stringstream ss;

            ss << fin.rdbuf();

            std::string line;

            while (std::getline(ss, line)) {//splitting data line by line

                if (line != "") {

                    std::stringstream lineStream(line);

                    //initialising variables for reading

                    std::string invoice{ "" };

                    double weight{};

                    std::string kind\_of\_delivery{ "" };

                    std::string destination{""};

                    //reading

                    if ((lineStream >> invoice >> weight >> kind\_of\_delivery >> destination)) {

                        Package tmp;

                        tmp.invoice\_number = invoice;

                        tmp.weight = weight;

                        tmp.kind\_of\_delivery = kind\_of\_delivery;

                        tmp.destination = destination;

                        queue->Enqueue(tmp);

                        int count{1};//changing number displayed +1

                        bool ok{true};

                        count += ui->number\_of\_deliveries\_label->text().toInt(&ok, 10);

                        ui->number\_of\_deliveries\_label->setText(QString::number(count));

                    }

                    else {

                        QMessageBox::critical(*this*, "error", "error reading data from file occured.");

                        break;

                    }

                }

            }

        }

        else

        {

            QMessageBox::warning(*this*, "error", "file could not be opened.");

            fin.close();

            return;

        }

        fin.close();

        return;

    }

}

void MainWindow::on\_add\_delivery\_manually\_button\_clicked()

{

    if((ui->comboBox1\_1->currentText() != "") && (ui->comboBox1\_1->currentText() != "") && (ui->weight\_lineEdit->text() != "")){

        std::string weight\_string = ui->weight\_lineEdit->text().toStdString();

        double weight{};

        try{

        weight = std::stod(weight\_string);//can throw std::invalid\_argument on user's wrong input

        }catch(...){

            QMessageBox::warning(*this*, "warning", "error for weight was found.");

            return;

        }

        if(weight <= 0){

            QMessageBox::warning(*this*, "warning", "error for weight was found.");

            return;

        }else{

            Package package;

            //filling package

            package.weight = weight;

            package.destination = ui->comboBox1\_1->currentText().toStdString();

            package.kind\_of\_delivery = ui->comboBox1\_2->currentText().toStdString();

            Package\* tmp = queue->get\_rear();

            int invoice\_num{};

            if(tmp != nullptr){

                std::string last\_invoice\_num = tmp->invoice\_number;

                last\_invoice\_num.erase(last\_invoice\_num.begin(), last\_invoice\_num.begin()+3);

                invoice\_num = std::stoi(last\_invoice\_num) + 1;

            }else{

                invoice\_num = 210401;

            }

            package.invoice\_number = ("INV" + std::to\_string(invoice\_num));

            queue->Enqueue(package);//enqueueing

            QMessageBox::information(*this*, "recorded", ("invoice: " + QString::fromStdString(package.invoice\_number)));

            int count{1};//changing number displayed +1

            bool ok{true};

            count += ui->number\_of\_deliveries\_label->text().toInt(&ok, 10);

            ui->number\_of\_deliveries\_label->setText(QString::number(count));

        }

    }else{

        QMessageBox::warning(*this*, "warning", "Not all fields are specified.");

    }

}

void MainWindow::on\_find\_button1\_clicked()//finding number of deliveries to x

{

    if(ui->comboBox2->currentText() != ""){

        int num\_of\_deliveries{};

        queue->search\_deliveries\_to(ui->comboBox2->currentText().toStdString(), num\_of\_deliveries);

        QMessageBox::information(*this*, QString::number(num\_of\_deliveries), ("number of deliveries to " + ui->comboBox2->currentText()));

    }else{

        QMessageBox::warning(*this*, "warning", "Not all fields are specified.");

    }

}

void MainWindow::on\_find\_button2\_clicked()//find summary weight of deliveries in category x

{

    if(ui->comboBox3->currentText() != ""){

        double weight{};

        queue->sum\_weight\_in\_category(ui->comboBox3->currentText().toStdString(), weight);

        QMessageBox::information(*this*, QString::number(weight)+”kg”, ("sum weight in category " + ui->comboBox3->currentText()));

    }else{

        QMessageBox::warning(*this*, "warning", "Not all fields are specified.");

    }

}

void MainWindow::on\_find\_button3\_clicked()//lookup by invoice

{

    if(ui->invoice\_lineEdit->text() != ""){

        Package package;

        queue->find\_by\_invoice(ui->invoice\_lineEdit->text().toStdString(), package);

        if(package.invoice\_number == ""){

            QMessageBox::information(*this*, "package not found", "please check if the invoice code is correct");

        }else{

            double weight = package.weight;

            QString invoice = QString::fromStdString(package.invoice\_number);

            QString destination = QString::fromStdString(package.destination);

            QString kind\_of\_delivery = QString::fromStdString(package.kind\_of\_delivery);

            QMessageBox::information(*this*, "package found",

                                     ("your " + kind\_of\_delivery + " package is of weight " + QString::number(weight)

                                      + "kg heading to " + destination + "\ninvoice: " + invoice));

        }

    }else{

        QMessageBox::warning(*this*, "warning", "Not all fields are specified.");

}

}

void MainWindow::on\_exit\_button\_clicked()

{

    QMessageBox::StandardButton reply = QMessageBox::question(*this*, "please resubmit", "do you want to quit?", QMessageBox::Yes | QMessageBox::No);

    if(reply == QMessageBox::Yes){

        QCoreApplication::quit();//exiting whole app

    }

}

void MainWindow::on\_show\_all\_button\_clicked()

{

    Package\* tmp = queue->get\_head();

    QString message = {""};

    if(tmp == nullptr){

        QMessageBox::information(*this*, "message", "the queue is empty.");

        return;

    }

    while(tmp != nullptr){

        message += QString::fromStdString(tmp->kind\_of\_delivery) + " package is of weight "

                   + QString::number(tmp->weight)+ "kg going to " + QString::fromStdString(tmp->destination)

                   + "\ninvoice: " + QString::fromStdString(tmp->invoice\_number) + "\n";

        tmp = tmp->next;

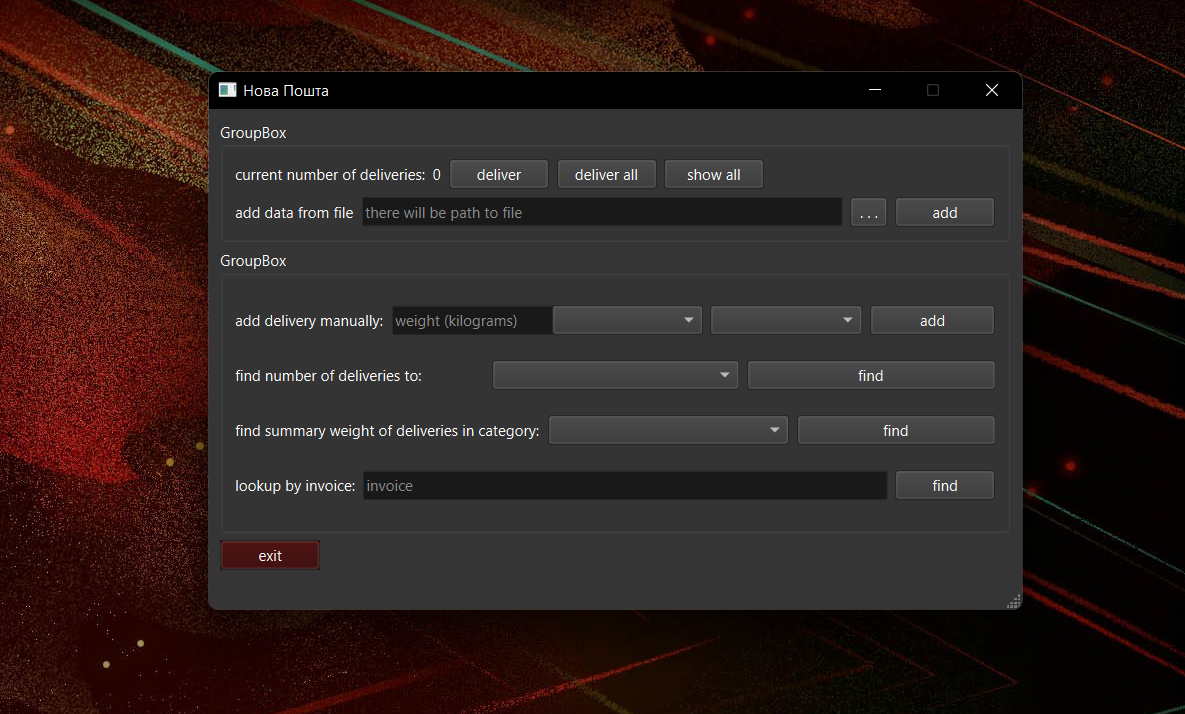
    }

    QMessageBox::information(*this*, "queue:",message);

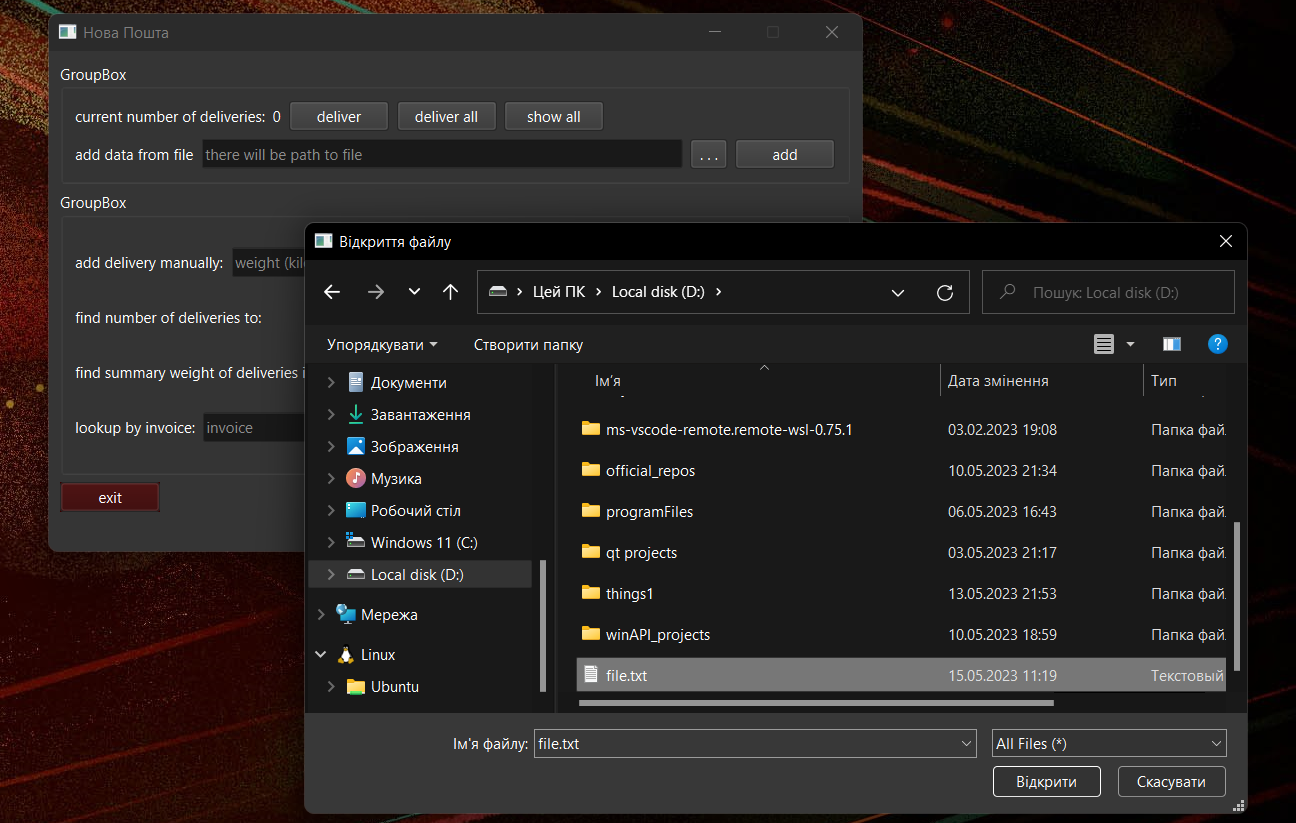
}

## Результат виконання:

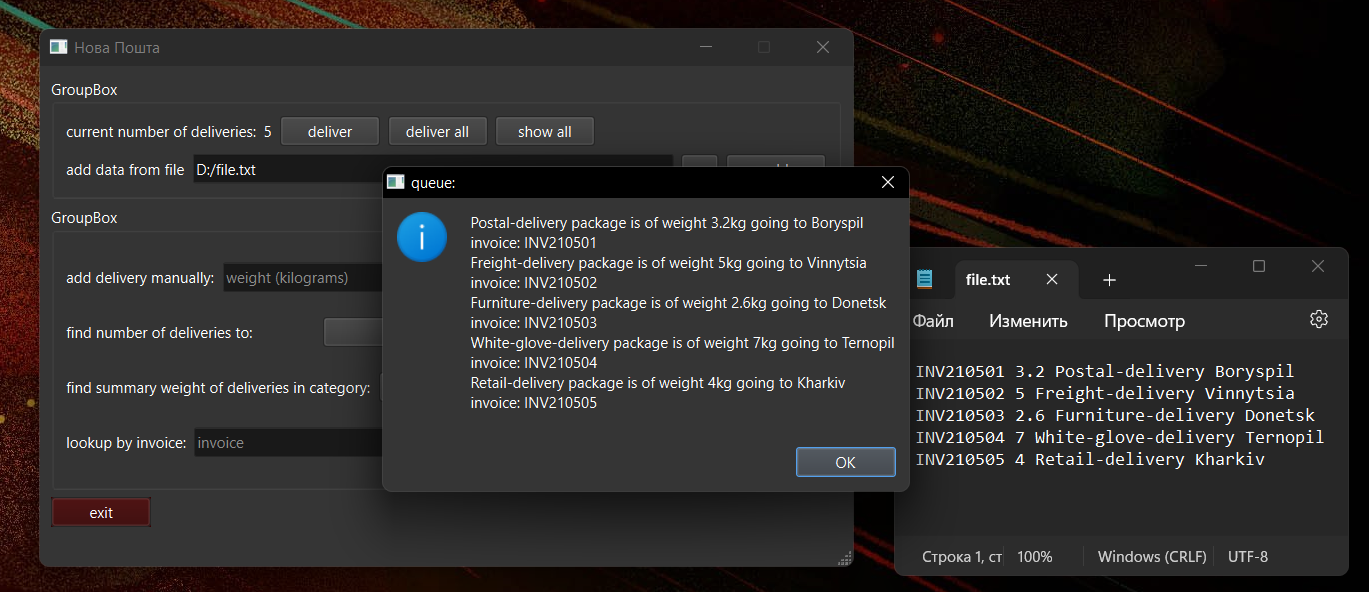
Вигляд:

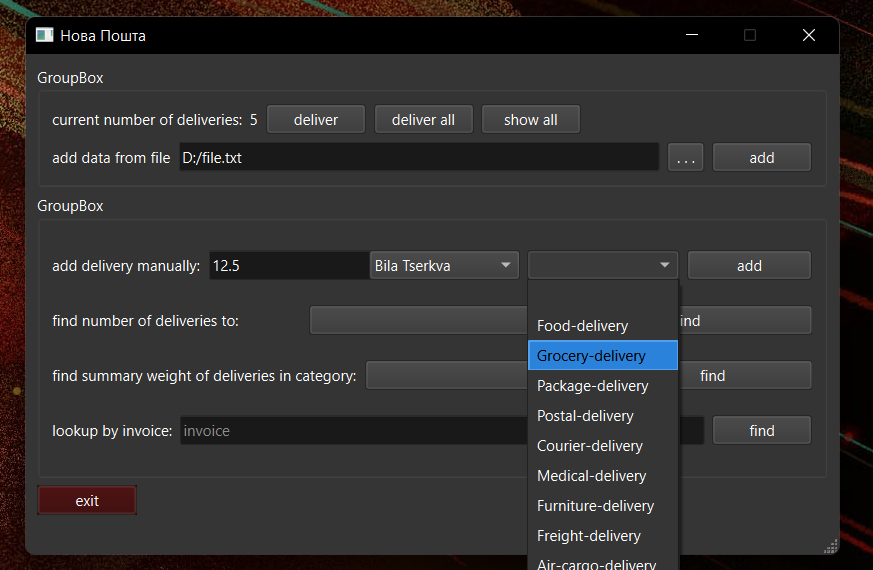


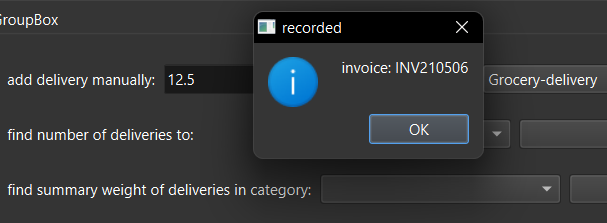
При натисканні “...” кнопки відкривається новий діалог (провідник мого Windows):



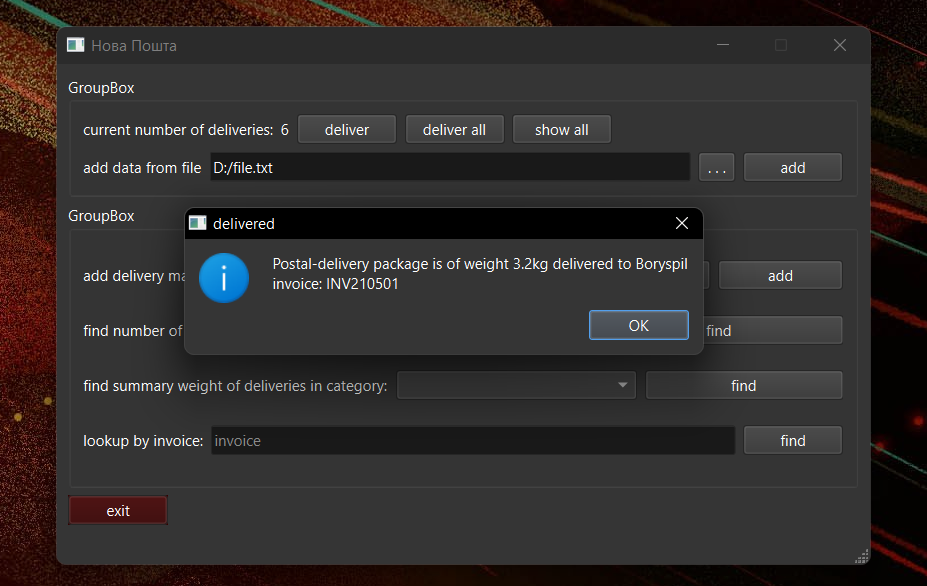
Натиснув add (data from file) та show all :

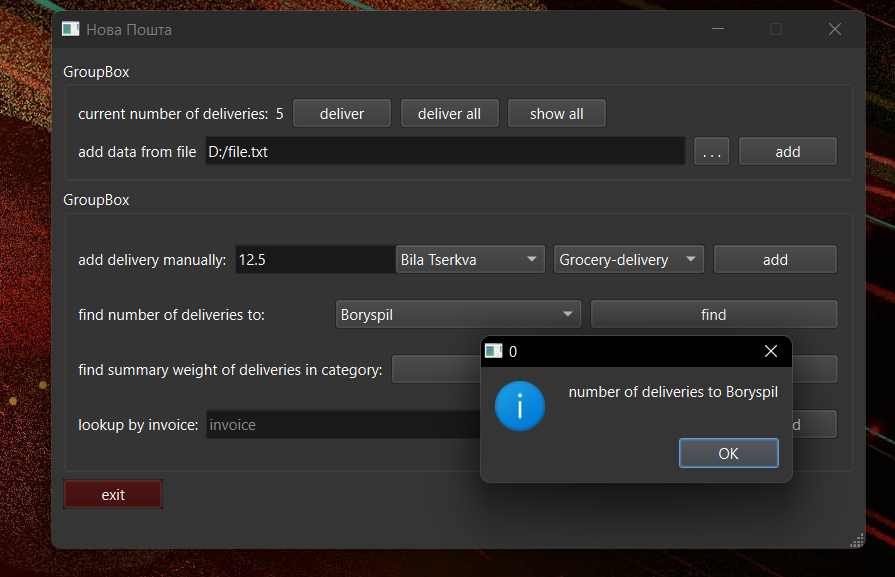


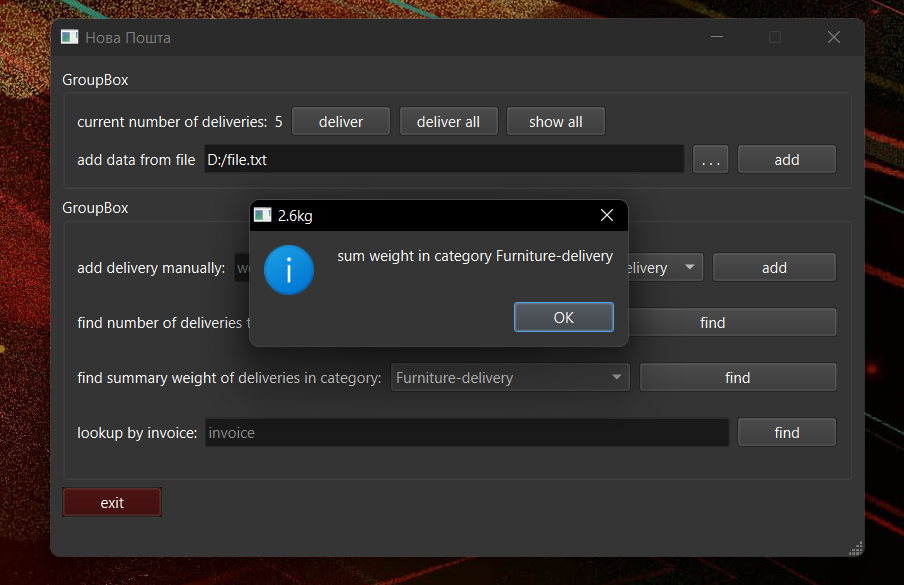


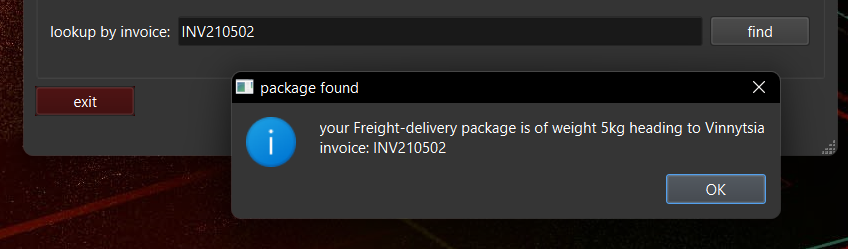


Після натискання deliver:

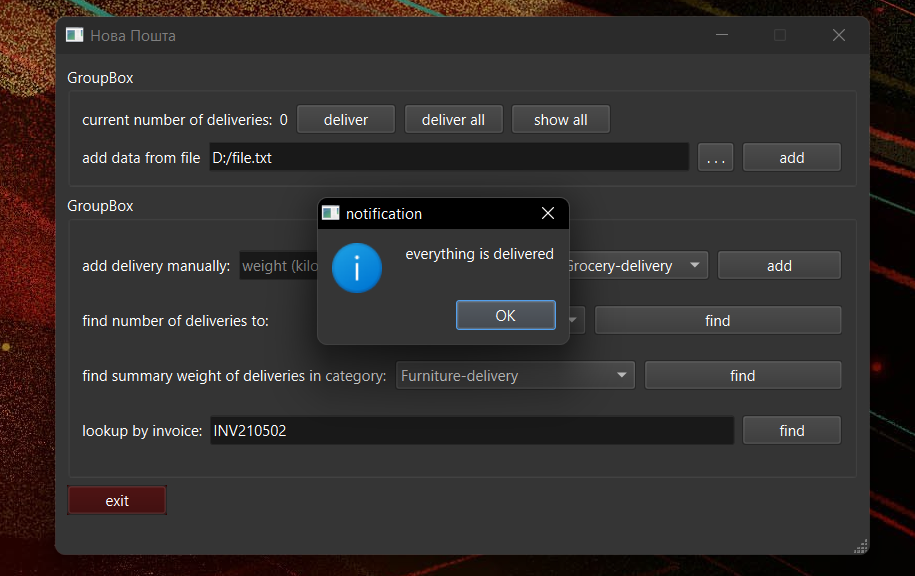


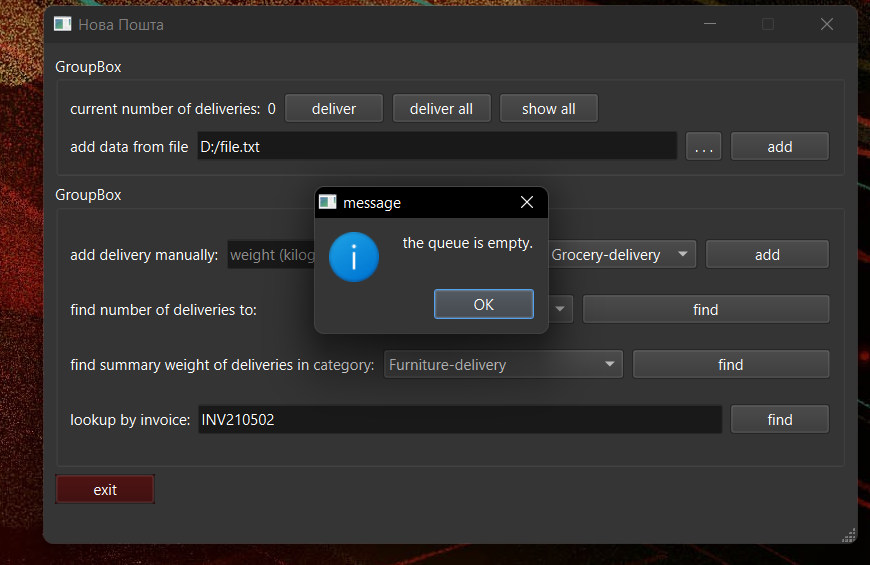


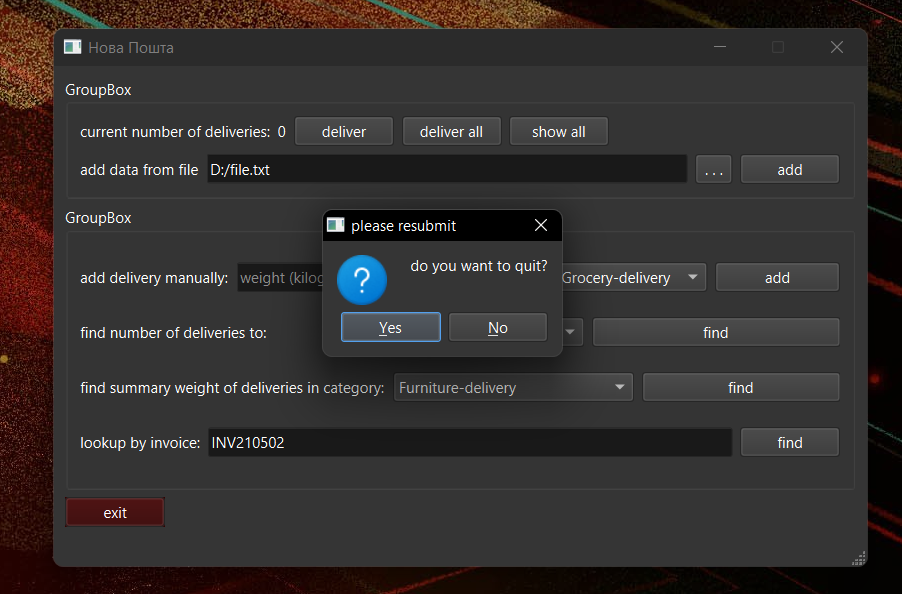




По натисканню deliver all:







**Висновки:**

Лабораторну роботу №6 успішно виконано. Використав Qt framework. Використовував функції on\_button\_clicked() (це як невеличкий мінус роботі). Тут немає перевірок на те, чи немає двох однакових накладних, можна купу разів додавати одні й ті ж дані з файлу та очевидно програма в певній мірі має купу недоробок (навіть попри велику кількість перевірок). Спочатку ідея була під’єднувати програму до текстового файлу як до певної бази даних, але через значну ускладненість ідей було прийнято рішення: файл тут просто для того щоб вручну не наповнювати чергу. Якщо приймати це до уваги, то у всьому іншому програма працює як належить.