Практична робота №12

Тема: Робота з винятками для обробки помилок під час виконання програми. Логування подій засобами фреймворку Qt.
Мета: Навчитись обробляти виняткові ситуації та обробляти помилки періоду виконання програми, а також навчитись зберігати (логувати) повідомлення про такі помилки.

Хід роботи

1. У Git-репозиторію із попередніх практичних робіт створюю нову гілку «PR12» і переходжу в неї для виконання даної практичної роботи.

2. Запускаю Qt Creator і відкриваю проєкт «MDI» з попередньої практичної роботи.

3. Виявляю у проєкті місця можливого виникнення виняткових ситуацій. Забезпечую обробку можливих помилок.

4. Підключаю логування помилок у файл «logfile.txt».

5. В усіх блоках обробки помилок додаю виклик необхідних функції (qDebug(), qWarning(), qCritical(), qFatal()).

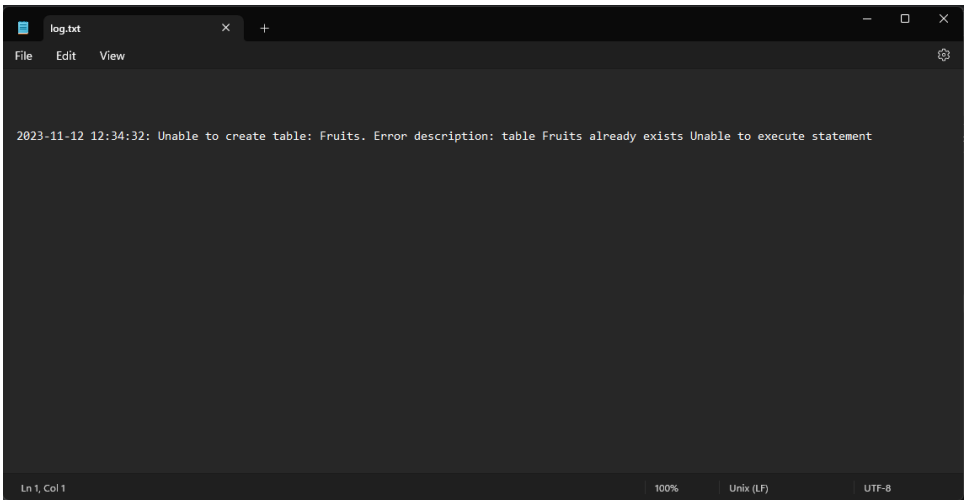
6. Перевіряю роботу програми.

7. Роблю коміт проєкту із повідомленням «done practical work №12».

8. Об’єдную гілку PR12 у гілку main і надсилаю зміни у гілці main у віддалений репозиторій.

9. Оформляю звіт.

Скріншот тексту файлу logfile.txt:



Вихідний код:

customessagehendler.h:

```
#ifndef CUSTOMMESSAGEHANDLER_H
#define CUSTOMMESSAGEHANDLER_H

#include <QCoreApplication>
#include <QFile>
#include <QTextStream>
#include <QtDebug>

class CustomMessageHandler
{
public:
    CustomMessageHandler();

    static void handleError(QtMsgType errorType, const QMessageLogContext& context, const
QString& errorMessage);
};

#endif // CUSTOMMESSAGEHANDLER_H
```

customessagehendler.cpp:

```
#include "custommessagehandler.h"
#include <QDateTime>
#include <QDebug>

CustomMessageHandler::CustomMessageHandler()
{

}

void CustomMessageHandler::handleError(QtMsgType errorType, const QMessageLogContext& context,
const QString& errorMessage)
{
    QFile logFile("log.txt");
    if (logFile.open(QIODevice::Append)) {
        QTextStream stream(&logFile);

        switch (errorType) {
        case QtInfoMsg:
            stream << "Info: ";
            break;
        case QtDebugMsg:
            stream << "Debug: ";
            break;
        case QtWarningMsg:
            stream << "Warning: ";
            break;
        case QtCriticalMsg:
            stream << "Critical: ";
            break;
        case QtFatalMsg:
            stream << "Fatal: ";
            break;
        }
```

```
        stream << "\n\n\n" + QDateTime::currentDateTime().toString("yyyy-MM-dd hh:mm:ss") + ":
" +
                  errorMessage << " (" << context.file << ":" << context.line << ", " <<
context.function << ")\n";
        logFile.close();
    }
}
```

Висновок: під час виконання даної практичної роботи, здобуто навички обробки виняткових ситуацій та обробки помилок періоду виконання програми, а також зберіганяя (логуваняя) повідомлення про такі помилки.