Министерство науки и высшего образования Российской Федерации

Калужский филиал федерального государственного бюджетного образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **Информатика и управление**

КАФЕДРА **Информационные системы и сети**

ЛАБОРАТОРНАЯ РАБОТА №5
«Создание SDI приложений»

ДИСЦИПЛИНА: "Программирование в среде Windows"

Выполнил: студент гр. ИУК2-51.Б: _____ (Мелкумян Д.Т.)
(Подпись) (Ф.И.О.)

Проверил: _____ (Крысин И. А.)
(Подпись) (Ф.И.О.)

Дата сдачи (защиты): _____

Результаты сдачи (защиты):

-Бальная оценка: ____

-Оценка: _____

Калуга, 2021 г.

Цель: изучить способы создания SDI приложений. Освоить приёмы работы с SDI приложениями.

Теоретическая часть:

DI-приложения имеют одно главное окно и могут при необходимости отображать диалоговые окна, но не имеют подчиненных окон.

Для ***SDI-приложений*** он создается на основе CFrameWnd, а для MDI-приложений - на базе CMDIChildWnd.

Подобно ***SDI-приложению***, класс документа MDI-приложения хранит данные документа и выполняет ввод / вывод файлов. Однако программа с многодокументным интерфейсом создает отдельные экземпляры этого класса для каждого открытого документа вместо повторного использования одного и того же экземпляра.

В ***SDI-приложениях*** главное окно является и окном документа. В MDI-приложениях окна, содержащие открытые документы, располагаются внутри главного окна приложения, при этом активизация того или иного документа может приводить к изменению элементов приложения, например, меню и панелей инструментов.

За основу взято SDI-приложение, построенное с использованием мастера AppWizard, поэтому мы не будем приводить весь код, а ограничимся фрагментами, которые иллюстрируют соответствующие аспекты работы с меню в динамическом режиме. Для того чтобы открыть доступ к нему, необходимо в системном меню выбрать элемент Разблокировать меню Цвет (рис. 61), который при этом будет отмечен галочкой.

Как и в SDI-приложении класс главного окна MDI-приложения управляет главным окном программы. Однако в MDI-программе главное окно не содержит единственное окно представления, служащее для просмотра документа. Вместо этого оно содержит рабочую область приложения. Таким образом, класс главного окна не связан с определенным типом документа и не включен в шаблон документа.

В дополнение к четырем классам, используемым в SDI-приложениях, MDI-приложения используют класс дочернего масштабируемого окна. Этот класс управляет дочерними окнами, создаваемыми для каждого открытого документа. Каждое дочернее окно отображается в рабочей области приложения и содержит окно представления для отображения документа.

Практическая часть:

Задание:

Разработать программу работы с графами через матрицу смежности. Программа должна обеспечивать создание новой матрицы, удаление и добавление вершин графа. Обеспечивать свойство симметричности. Чтение и сохранение информации о графе. Заполнение элементов матрицы должно происходить по двойному щелчку левой кнопки мыши или нажатия клавиши пробел. Изучить основные функции работы с элементом ActiveX (FlexGrid). Каждому пункту должна соответствовать ускоряющая клавиша и кнопка на панели инструментов.

Код программы:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QMessageBox>
#include "QStandardItemModel"
#include "QStandardItem"
#include <QApplication>
#include <QFile>
#include <QMessageBox>
#include <QtGui>
#include <QFileDialog>
#include <QString>

QTableWidgetItem *item;
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

}

MainWindow::~MainWindow()
{
    delete ui;
}
```

```
void MainWindow::on_action_triggered()
{
    QString fileName = QFileDialog::getOpenFileName(this,
                                QString::fromUtf8("Открыть файл"),
                                QDir::currentPath(),
                                "Файл csv (*.csv);;All files (*.*)");
    QTextStream out(stdout);
    ui->tableWidget->setRowCount(0);
    ui->tableWidget->setColumnCount(0);
      // Создаем объект
      QFile file(fileName);

      // С помощью метода open() открываем файл в режиме чтения
      if (!file.open(QIODevice::ReadOnly)) {
        qWarning("Cannot open file for reading"); // если файл не найден, то
выводим предупреждение и завершаем выполнение программы
      }

      // Создаем входящий поток, из которого будут считываться данные, и
связываем его с нашим файлом
      QTextStream in(&file);

      // Считываем файл строка за строкой
      while (!in.atEnd()) { // метод atEnd() возвращает true, если в потоке
больше нет данных для чтения
        QString line = in.readLine(); // метод readLine() считывает одну
строку из потока
        out << line;
        QStringList splited = line.split(";");
        ui->tableWidget->setRowCount(ui->tableWidget->rowCount()+1);
        ui->tableWidget->setColumnCount(splited.length()-1);
        for (int i=0;i<splited.length()-1;i++)
        {
            item = new QTableWidgetItem(splited[i]);
            ui->tableWidget->setItem(ui->tableWidget->rowCount()-1,i,item);


        }

      }

      // Закрываем файл
      file.close();
}


void MainWindow::on_action_2_triggered()
{
    QFile fileOut("test.csv"); // Связываем объект с файлом fileout.txt
    if(fileOut.open(QIODevice::WriteOnly | QIODevice::Text))
    { // Если файл успешно открыт для записи в текстовом режиме
         QTextStream writeStream(&fileOut); // Создаем объект класса
QTextStream
             for (int i=0;i<ui->tableWidget->rowCount();i++){
                for (int j=0;j<ui->tableWidget->columnCount();j++)
                {
                    item =  ui->tableWidget->item(i,j);
                    if (NULL != item) writeStream << item->text()+";"; //
Посылаем строку в поток для записи
                        else writeStream << ";";

                }
                writeStream << "\n";
```

```
            }
        fileOut.close(); // Закрываем файл
    }


}


void MainWindow::on_action_3_triggered()
{
    qApp->quit();
}


void MainWindow::on_pushButton_clicked()
{
    //int count = ui->tableView->model()->rowCount();
    ui->tableWidget->clear();
    ui->tableWidget->setRowCount(ui->lineEdit_2->text().toInt());
    ui->tableWidget->setColumnCount(ui->lineEdit->text().toInt());

}
```

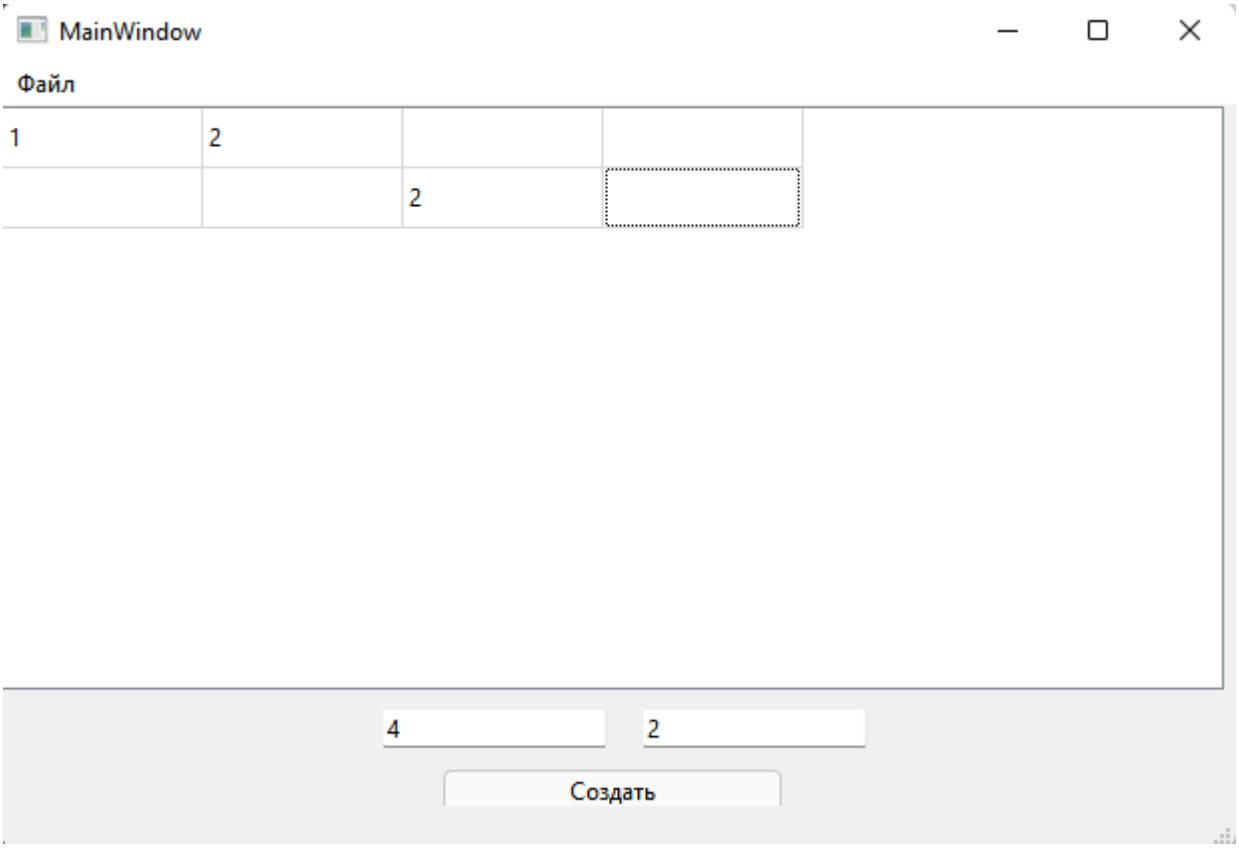

Рис 1. Результат выполнения программы

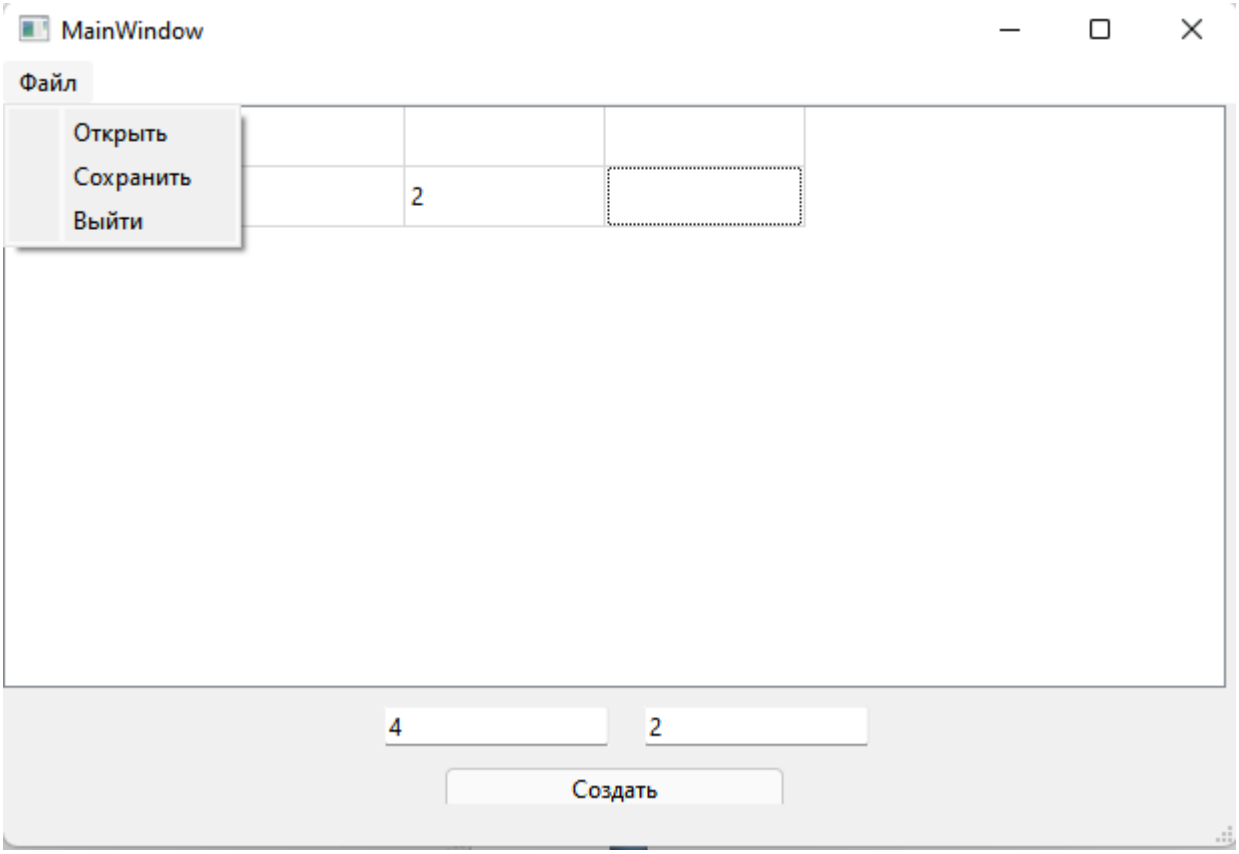

Рис 2. Результат выполнения программы 2

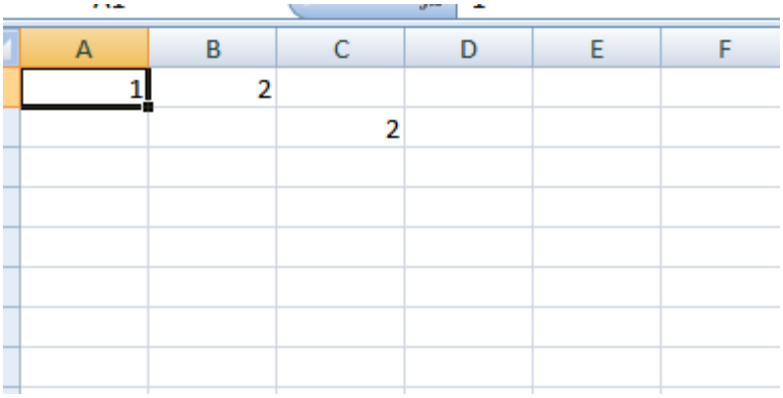

Рис 1. Результат записи в файл

Вывод: В данной лабораторной работе изучены способы создания SDI приложений. Освоены приёмы работы с SDI приложениями.

СПИСОК ЛИТЕРАТУРЫ

1. ***Подбельский В.В***. Язык СИ++: учебное пособие. М.: Финансы и статистика, 2008.

2. ***Павловская Т.А.*** С/С++ Программирование на языке высокого уровня: учебник. СПб.:Питер, 2010.

3. ***Перова В. И.*** Программирование на С++ в среде VISUAL STUDIO.NET. Нижний Новгород: Издательство Нижегородского госуниверситета, 2010., http://elibrary.ru/item.asp?id=19597408

4. ***Крылов Е.В.*** Техника разработки программ: учебник. М.: Высшая школа, 2008.

5. Научная электронная библиотека: http://eLIBRARY.RU.

6. Издательство «Лань». Электронно-библиотечная система. http://e.lanbook.com.