МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №33

КУРСОВАЯ РАБОТА (ПРОЕКТ)
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

профессор, канд. техн. наук		С.Г. Фомичева
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ

РАЗРАБОТКА ПРОГРАММЫ ПОСТРОЕНИЯ БЛОК-СХЕМЫ АЛГОРИТМА ПО ИСХОДНОМУ ПРОГРАММНОМУ ФАЙЛУ.

по дисциплине: ТЕХНОЛОГИИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 3133		Е. Н. Юшкова
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2023

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

БСА — блок-схема алгоритма.

Структурный блок — логически выделенный блок в программном коде, в котором последовательно располагаются либо структурные элементы, либо структурные блоки.

Структурный эл-т — самая простая (элементарная) структурная единица кода. Например, объявление переменной или вывод.

ВВЕДЕНИЕ

Программный код бывает достаточно сложен для понимания, поэтому, чтобы максимально доходчиво, по-простому рассказать о сложных моментах, нужна хорошая визуализация. В данном случае отличным решением является БСА.

Блок-схема — это схематичное представление компьютерного алгоритма, системы или процесса. Цель блок-схемы состоит в том, чтобы продемонстрировать процесс в пошаговом режиме, выделив различные возможные результаты, основанные на различных возможных путях. Создание и анализ потоковой диаграммы позволяет визуализировать, с чего начать, конечную цель и каждую задачу, которую необходимо выполнить, чтобы достичь ее. Возможность изложить процесс на бумаге в легко понятном документе позволяет сократить лишние этапы времени и создать самый быстрый и эффективный путь к достижению конечной цели.

БСА часто применяются в различных сферах деятельности, чтобы документировать, изучать, планировать, совершенствовать и объяснять сложные процессы с помощью простых для понимания и логичных диаграмм. Благодаря блок-схемам можно определить цели и масштабы рабочего процесса.

Существует множество онлайн сервисов и приложений для рисования блок-схем, однако создание вручную, особенно тех, которые визуализируют достаточно сложные алгоритмы, требует немало времени, усилий и человеческих ресурсов. При этом при их создании происходит довольно алгоритмизированная работа, а значит, ее можно программно реализовать. Такое приложение значительно уменьшит временные затраты при составлении отчетов.

Таким образом, основной задачей данной курсовой работы является создание приложения для облегчения визуализации программного кода.

1. Назначение и цели создания проекта

Приложение для построения БСА является важным инструментом для программистов, учащихся и профессионалов, которые занимаются разработкой программного обеспечения. Это приложение предназначено для создания графических представлений алгоритмов, которые могут быть использованы для лучшего понимания логики программы и улучшения процесса разработки.

Основная цель приложения для построения блок-схем алгоритмов заключается в том, чтобы помочь пользователям создавать графические представления алгоритмов, которые могут быть использованы для лучшего понимания логики программы и улучшения процесса разработки. Это приложение позволяет пользователям создавать блок-схемы, которые отображают последовательность шагов, условия и циклы в программе.

Создание БСА имеет несколько преимуществ. Во-первых, блок-схемы упрощают понимание сложных алгоритмов, особенно для начинающих программистов. Во-вторых, блок-схемы позволяют быстро и легко отслеживать ошибки в алгоритмах, что помогает сократить время на отладку кода. В-третьих, блок-схемы могут быть использованы для документирования программного кода, что облегчает его понимание другими программистами.

Приложение для построения блок-схем алгоритмов может быть использовано в различных областях, включая программирование, математику, физику, инженерию и т.д. Это приложение может быть полезно для учащихся, которые изучают основы программирования и алгоритмов, а также для профессиональных программистов, которые хотят улучшить свой процесс разработки.

Назначение приложения для построения блок-схем - помочь пользователям создавать графические диаграммы, которые описывают последовательность выполнения процессов или алгоритмов.

Обобщая все выше сказанное, основными целями создания приложения для построения БСА являются:

- упрощение процесса проектирования и разработки программного обеспечения, позволяя разработчикам легко создавать и отлаживать алгоритмы;
- улучшение понимания сложных процессов и алгоритмов, позволяя пользователям визуализировать их шаг за шагом;
- повышение эффективности и точности работы, позволяя пользователям легко оптимизировать процессы и выявлять ошибки;
- улучшение коммуникации и сотрудничества, позволяя пользователям легко обмениваться блок-схемами и работать над ними совместно;
- предоставление пользователю инструментов для создания профессионально выглядящих диаграмм, которые могут быть использованы в презентациях и отчетах.

2. Формализация задачи

2.1. Обзор методов решения

Основной задачей данной курсовой работы является создание приложения, которое воспроизводит БСА по исходному программному файлу. Для осуществления поставленной задачи необходимо определиться с методом ее решения. Существует несколько методов построения блок-схем, которые могут быть использованы программистами и учащимися для создания графических представлений алгоритмов.

Один из методов построения БСА — это метод "потока управления". Этот метод основан на последовательности выполнения операций и отображает порядок выполнения шагов в программе. Блок-схемы, созданные с использованием метода потока управления, состоят из блоков, которые соединены стрелками, показывающими направление выполнения операций.

При построении блок-схемы методом "потока управления" необходимо следить за правильной последовательностью блоков и связей между ними. Каждый блок должен быть связан с предыдущим и следующим блоками, чтобы обеспечить правильный поток управления в программе.

Еще один метод построения блок-схем — это метод "потока данных". Этот метод используется для построения блок-схемы программы, которая представляет собой графическое представление потоков данных между операциями или функциями. Этот метод начинается с определения входных и выходных данных программы.

Процесс построения блок-схемы методом "потока данных" включает следующие шаги:

1). Определение входных и выходных данных — это этап, на котором определяются данные, которые будут вводиться в программу и данные, которые будут выводиться из программы.

2). Определение операций или функций — это этап, на котором определяются операции или функции, которые необходимы для решения задачи программы.

3). Определение потоков данных — это этап, на котором определяются потоки данных между операциями или функциями. Потоки данных могут быть как входными, так и выходными.

4). Построение блок-схемы — это этап, на котором все операции или функции организуются в виде блоков, а потоки данных представляются стрелками, указывающими направление потока данных.

При построении блок-схемы методом "потока данных" необходимо следить за правильной организацией потоков данных и связей между операциями или функциями. Каждый блок должен быть связан с предыдущим и следующим блоками, чтобы обеспечить правильный поток данных в программе.

Третий метод построения блок-схем — это метод "объектно-ориентированного моделирования". Этот метод используется для построения блок-схем программы, которая представляет собой графическое представление объектов и их взаимодействия. Этот метод начинается с определения объектов и их свойств.

Процесс построения блок-схем методом "объектно-ориентированного моделирования" включает следующие шаги:

1). Определение объектов — это этап, на котором определяются объекты, которые будут использоваться в программе. Объекты могут быть как реальными, так и абстрактными.

2). Определение свойств объектов — это этап, на котором определяются свойства каждого объекта. Свойства могут быть как простыми типами данных, так и другими объектами.

3). Определение методов объектов — это этап, на котором определяются методы или функции, которые могут быть вызваны для выполнения операций над объектами.

4). Определение взаимодействия объектов — это этап, на котором определяются взаимодействия между объектами. Взаимодействия могут быть как прямыми, так и косвенными.

5). Построение блок-схемы — это этап, на котором все объекты организуются в виде блоков, а их взаимодействия представляются стрелками, указывающими направление взаимодействия.

При построении блок-схемы методом "объектно-ориентированного моделирования" необходимо следить за правильной организацией объектов и их взаимодействий. Каждый объект должен быть связан с другими объектами, чтобы обеспечить правильное взаимодействие между ними в программе.

2.2.Сравнительный анализ методов решения

Метод "потока управления" является простым и понятным для начинающих программистов, так как он основывается на последовательности выполнения операций.

Метод "потока данных" также основывается на последовательности выполнения операций, но в отличие от метода "потока управления", он учитывает структуру данных и их потоки. Этот метод может быть полезен для построения алгоритмов, связанных с обработкой больших объемов данных.

Метод "объектно-ориентированного моделирования" является более сложным для понимания, но он позволяет создавать более сложные алгоритмы, которые отображают не только логику программы, но и ее структуру. Этот метод основывается на моделировании объектов и их взаимодействия в программе, что делает его полезным для построения объектно-ориентированных программ.

В целом, каждый из этих методов имеет свои преимущества и может быть использован в зависимости от типа программы и задачи, которую нужно решить. Исходя из поставленной задачи, наиболее целесообразно выбрать метод "потока управления". Для данного выбора есть ряд причин.

Во-первых, чаще всего необходима визуализация последовательности выполнения операций в определенном методе. Каждый метод из программного кода будет иметь собственную БСА. Таким образом разделение большой системы на малые подалгоритмы позволит реализовать облегченное понимание программного кода.

Во-вторых, даже несмотря на то, что вводимый код будет написан на объектно-ориентированном языке, реализовать построение БСА с помощью метода "потока управления" не доставит проблем.

В-третьих, метод "потока данных" и метод "объектно-ориентированного моделирования" чаще всего разрабатываются для каждой программы индивидуально, т.е. нет какого-то определенного шаблона алгоритма, с помощью которого можно графически изобразить БСА для любой программы, что в разы увеличивает сложность создания приложения.

Таким образом, метод "потока управления" является наиболее оптимальным решением.

2.3. Подробная реализация выбранного алгоритма

После выбора метода решения поставленной задачи, его необходимо рассмотреть подробнее. Для этого построим контекстный уровень диаграммы IDEF0, который представлен на рис. 1.

Рис. 1. Контекстный уровень диаграммы IDEF0.

На входе программа получает программный код, при этом на выходе должна быть либо полученная блок-схема, либо сообщение об ошибках в введенном коде, не позволяющих построить БСА. Механизмами являются: сам пользователь, который вводит программный код, и программа Flowchart. Управлением являются: документация ЯП C# (для определения корректности

вводимого кода), совокупность структурных эл-тов и правила построения блок-схем.

В общем и целом, для построения БСА методом "потока управления" существует лишь один способ: декомпозировать представленный структурный блок на структурные элементы. Исходя из этого, задачу построения БСА можно разделить на следующие подзадачи:

1) проверка корректности введенного кода;
2) выделение структурных блоков;
3) определение входящих в блок структурных эл-тов;
4) прорисовка эл-тов в необходимом расположении.

Т.к. определились с подзадачами построим уровень декомпозиции диаграммы IDEF0, который представлен на рис. 2.

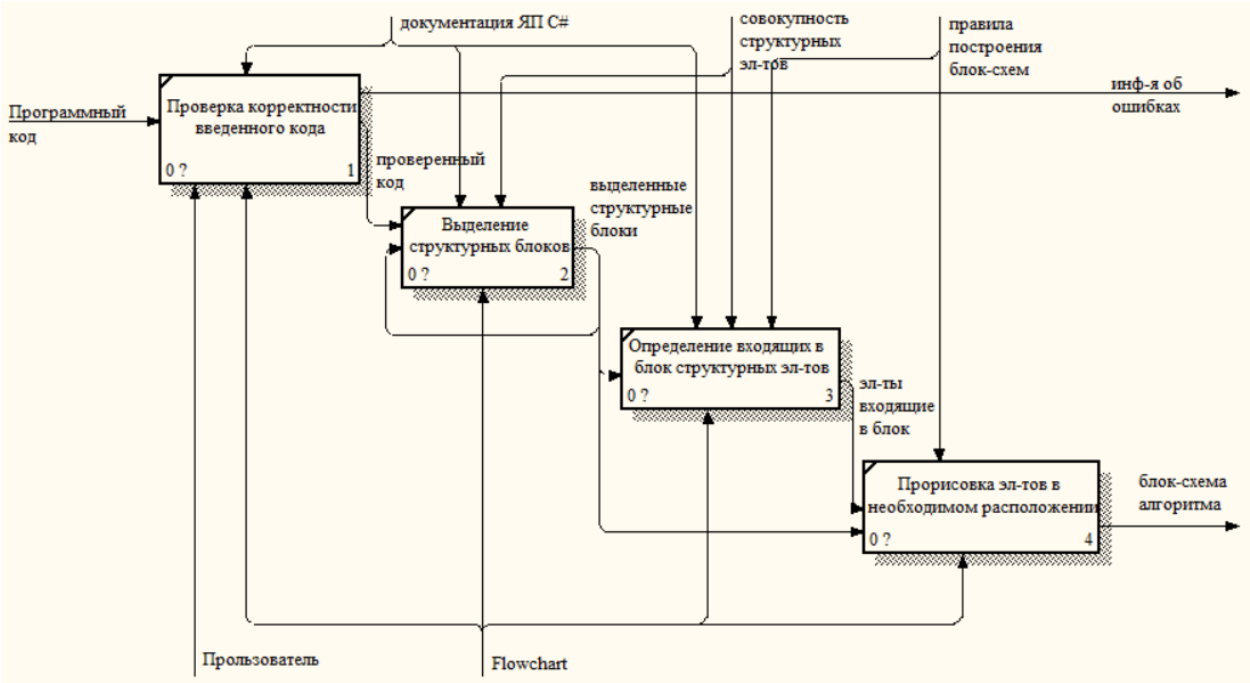


Рис. 2. Уровень декомпозиции диаграммы IDEF0.

После проверки корректности введенного кода есть два варианта событий: либо из данного программного кода можно составить БСА, либо нельзя. При этом, если составить нельзя, программа должна выводить пользователю сообщение об ошибке, если можно, значит приступаем к следующему шагу решения поставленной задачи.

Выделение структурных блоков является одной из главных подзадач в данном алгоритме. Этот этап необходим для последующей корректной работы программы. По сути, этот процесс подразумевает собой декомпозицию программного кода на известные структурные эл-ты и облегчению построения БСА (от простого к сложному).

Выделенные структурные блоки отправляются на вход в следующие процессы: выделение структурных блоков (для дальнейшего разложения на структурные блоки), определение входящих в блок структурных эл-тов (для выделения структурных элементов в данном структурном блоке) и прорисовка эл-тов в необходимом расположении (структурные блоки определяют месторасположение структурных эл-тов в БСА).

Определение входящих в блок структурных эл-тов: на данном этапе определяются самые простые эл-ты, из которых состоит БСА. Обозначение данных элементов, правила их построения и внешний вид программе будут известны.

После определения всех структурных блоков и всех структурных элементов в программном коде, на этапе прорисовки эл-тов в необходимом расположении происходит объединение всех полученных данных в единое целое, т.е. БСА.

2.4.Оценка оптимальности алгоритма

При проведении оценки оптимальности можно проанализировать следующие параметры процесса:

- Простота и понятность.

БСА, построенная методом "потока управления", является достаточно простой и понятной как для начинающих программистов, так и для более опытных. Поэтому на выходе процесса получаем БСА, ориентированную на любых пользователей.

- Функциональность.

Метод "потока данных" и метод "объектно-ориентированного моделирования" чаще всего разрабатываются для каждой программы

индивидуально, т.е. нет какого-то определенного шаблона алгоритма, с помощью которого можно графически изобразить БСА для любой программы, что в разы увеличивает сложность создания приложения. В отличии от них, метод "потока управления" обладает достаточной функциональностью для решения различных задач и создания сложных алгоритмов. Однако стоит отметить, что чем сложнее будет программный код, тем запутаннее будет БСА.

- Эффективность.

Метод "потока управления" обеспечивает эффективность создания блок-схем алгоритмов и занимает меньше времени на создание сложных алгоритмов в сравнении с методами "потока данных" и "объектно-ориентированного моделирования".

- Расширяемость.

Метод "потока управления" расширяем: можно добавлять новые функции и возможности в приложение в будущем. Например, добавлять новые правила в документацию ЯП C#, новые структурные эл-ты в совокупность структурных элементов и новые правила в правила построения БСА.

3. Схема взаимодействия объектов

Всего в процессе построения блок-схемы участвуют две внешние сущности: пользователь и сама программа Flowchart. При этом пользователь предоставляет программный код, по которому надо составить БСА, возвращается же ему либо БСА, либо инф-я об ошибках. Программа Flowchart является управляющим органом. На основе этих данных построим контекстный уровень диаграммы DFD, представленный на рис. 3.

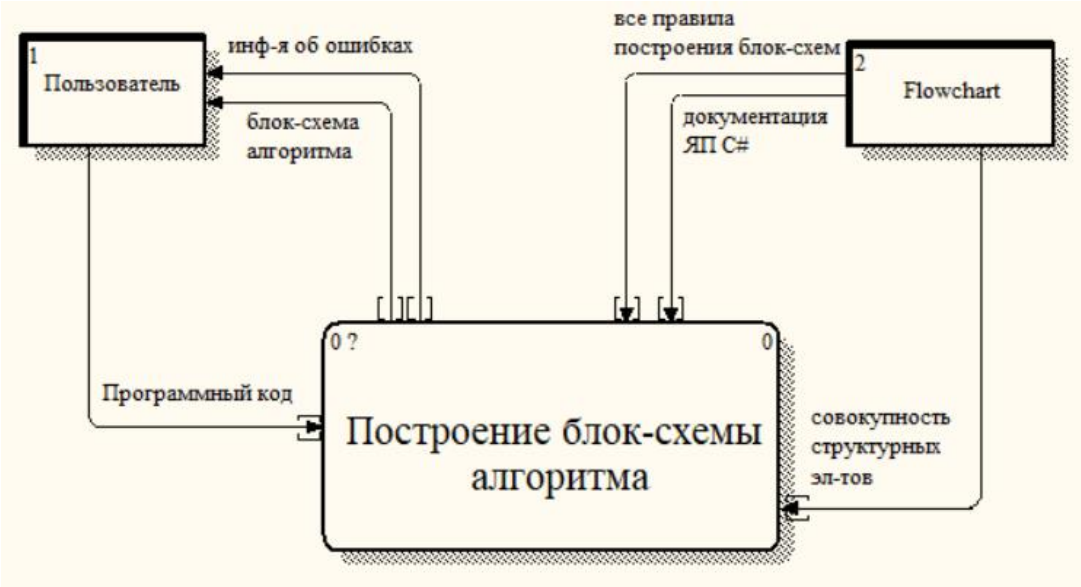


Рис. 3. Контекстный уровень диаграммы DFD.

На основе ранее построенной диаграммы IDEF0 построим контекстный уровень диаграммы DFD, представленный на рис. 4.

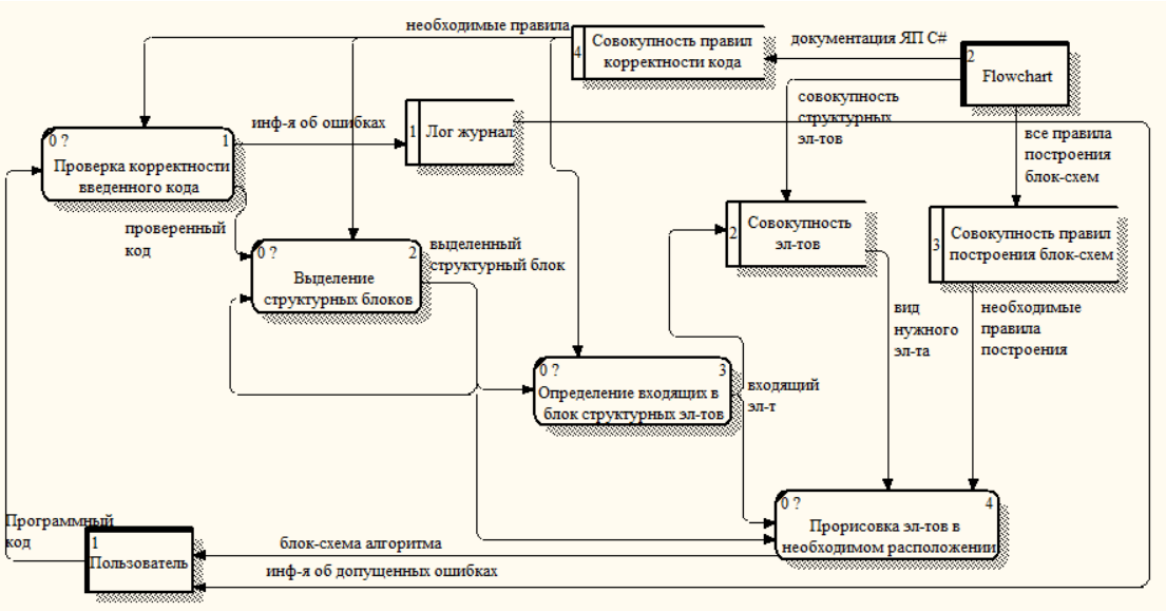


Рис. 4. Уровень декомпозиции диаграммы DFD.

В основном логика взаимодействия процессов, представленная на диаграмме DFD повторяет логику на диаграмме IDEF0, что, собственно, ожидаемо, т.к. мы описываем один и тот же процесс. Однако необходимо описать важные дополнения.

Вся информация об ошибках выводится не только пользователю, но и записывается в накопитель «Лог журнал».

Т.к. программа Flowchart является управляющим органом, то она предоставляет все необходимые управляющие инструменты, а именно: документация ЯП C#, совокупность структурных элементов и все правила построения БСА. Все эти правила хранятся в накопителях «Совокупность правил корректности кода», «Совокупность эл-тов» и «Совокупность правил построения блок-схем» соответственно. По необходимости инф-я из этих накопителей используется для построения БСА.

3.1. Диаграмма классов

При разработке ПО, реализующего функциональность поставленной задачи, получились следующие классы:

- Block – класс, соответствующий структурному блоку алгоритма;
- Flowchart_main_form – форма, на которой отображается основной функционал программы;
- Show_details – форма, на которой отображается информация об структурном элементе блок-схемы
- Form_save_name – форма, на которой пользователь записывает название для сохранения в БД;
- Form_open_flowchart – форма, на которой пользователь выбирает какую блок-схему, записанную в БД открыть;
- ApplicationContext – класс взаимодействия с БД;
- FlowChart – класс записываемых в контекст данных.

Полученная диаграмма классов представлена на рис.5.

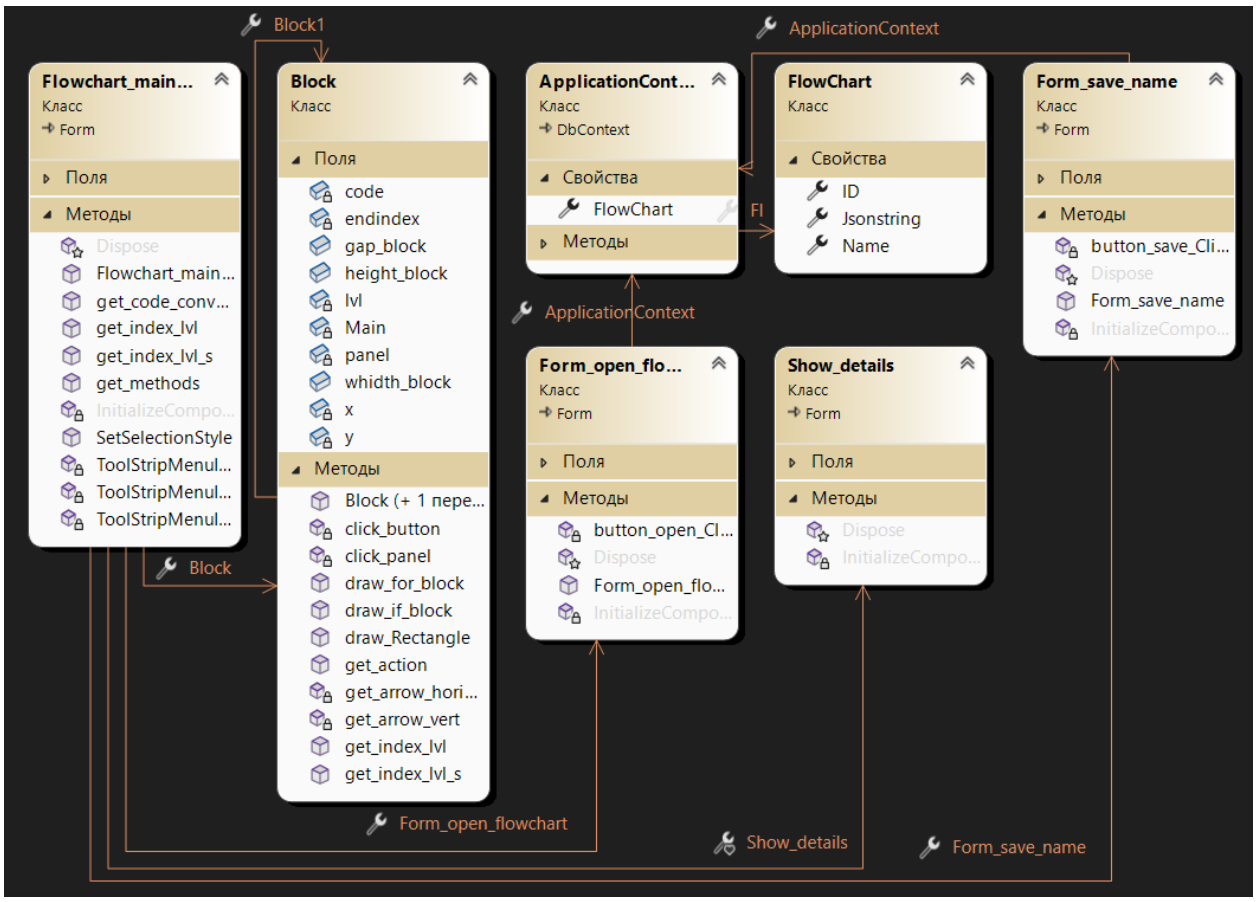


Рис. 5. Диаграмма классов.

Основные методы класса Block (код класса представлен в приложении1):

- Block – метод инициализации элемента (определение основных параметров таких как: расположение и размеры структурного блока, размеры структурного элемента, расстояние между блоками, начальное значение координат x и y – и вызов метода get_action).
- click_button – метод, присваиваемый всем структурным элементам, для открытия формы Show_details для просмотра информации о данном структурном элементе.
- click_panel – метод, присваиваемый структурному блоку, для отображения его в отдельном окне.
- draw_for_block, draw_if_block, draw_Rectangle – методы отрисовки структурных элементов.
- get_action – метод, который парсит структурный блок на структурные элементы, при этом определяя координаты x и y для следующего блока.

- get_arrow_horisont, get_arrow_vert – методы для отрисовки линий связей между структурными элементами.
- get_index_lvl, get_index_lvl_s – методы для нахождения индекса парной фигурной и круглой скобки.

Основные методы класса Flowchart_main_form (код класса представлен в приложении 2):

- get_code_conversion – метод изменяющий входящий код проекта для дальнейшего правильного определения входящих в него элементов (изменяет блок switch на соответствующие блоки else if, в блоки if, после которых нет фигурных скобок, и в блоки else if добавляет фигурные скобки).
- get_methods – метод поиска методов во входном программном коде и создания структурного блока для каждого метода.
- ToolStripMenuItem_create_new_program_fail_Click – метод открытия нового программного кода для создания по нему блок-схемы.
- ToolStripMenuItem_open_Click – метод открытия блок-схем сохраненных в БД.
- ToolStripMenuItem_save_into_BD_Click – метод сохранения открытого программного кода в БД.

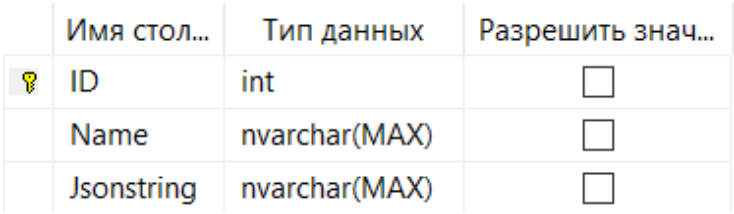
4. Используемые технологии и техники программирования

Для реализации проекта используется среда разработки Visual Studio. Язык программирования – C#.

Построение БСА основывается на рекурсии, реализуемой в классе Block. Это значит, что элемент не будет создан, пока все входящие в него структурные блоки не будут созданы. Если рассматривать БСА как дерево, то реализуемый обход – прямой обход дерева в глубину.

Используемая БД – MS SQL Server.

В БД приложение записывает название программного кода и сам код. Структура таблицы FlowChart представлена на рис. 6.



	Имя стол...	Тип данных	Разрешить знач...
🔑	ID	int	☐
	Name	nvarchar(MAX)	☐
	Jsonstring	nvarchar(MAX)	☐

Рис. 6. Структура таблицы FlowChart.

Взаимодействие с БД происходит с помощью класса ApplicationContext. Программный код представлен в листинге 1

Листинг 1. Программный код класса ApplicationContext.

```
public class ApplicationContext : DbContext
{
    public DbSet<FlowChart> FlowChart { get; set; } = null!;
    public FlowChart FlowChart1
    {
        get => default;
        set{}
    }
    public ApplicationContext()
    {
        Database.EnsureCreated();
    }
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        ConfigurationBuilder builder = new ConfigurationBuilder();
        // установка пути к текущему каталогу
```

```
            //MessageBox.Show(Directory.GetCurrentDirectory());
            builder.SetBasePath(Directory.GetCurrentDirectory());
            // получаем конфигурацию из файла appsettings.json
            builder.AddJsonFile("jsonconfigconnection.json");
            // создаем конфигурацию
            var config = builder.Build();
            // получаем строку подключения
            string connectionString = config.GetConnectionString("DefaultConnection");
            optionsBuilder.UseSqlServer(connectionString);
        }
    }
```

Проектирование моделей данных проводим методом code-first.

Программная реализация использует объектно-ориентированный подход, платформа пользовательского интерфейса – Windows forms.

5. Результаты работы приложения

При открытии приложении открывается стартовое окно программы, представленное на рис.7.

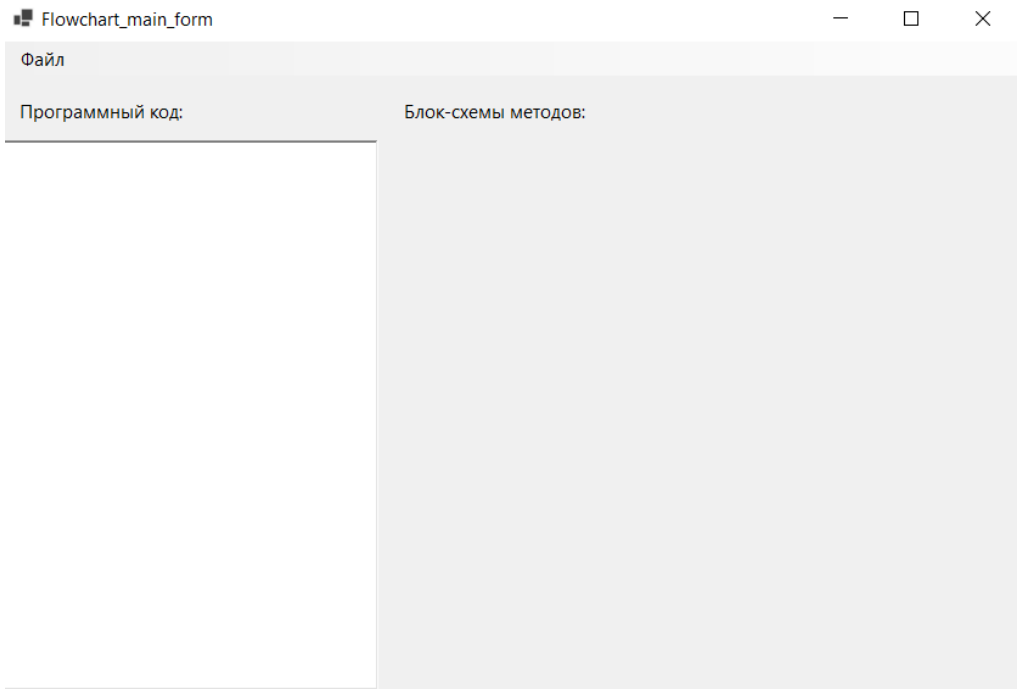


Рис. 7. Стартовое окно программы.

Есть три варианта действий с файлом: создать, сохранить в БД и открыть. Выбрать можно в меню, представленном на рис. 8.

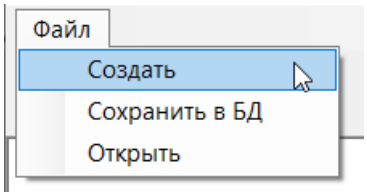


Рис. 8. Меню выбора действий с файлом.

При создании открывается диалоговое окно выбора файла программного кода с расширением *.cs. После выбора файла программа обрисовывает его методы. Реализация отрисовки различных структур таких как if, else if, for, while, switch и простых блоков представлены на рис. 9-12.

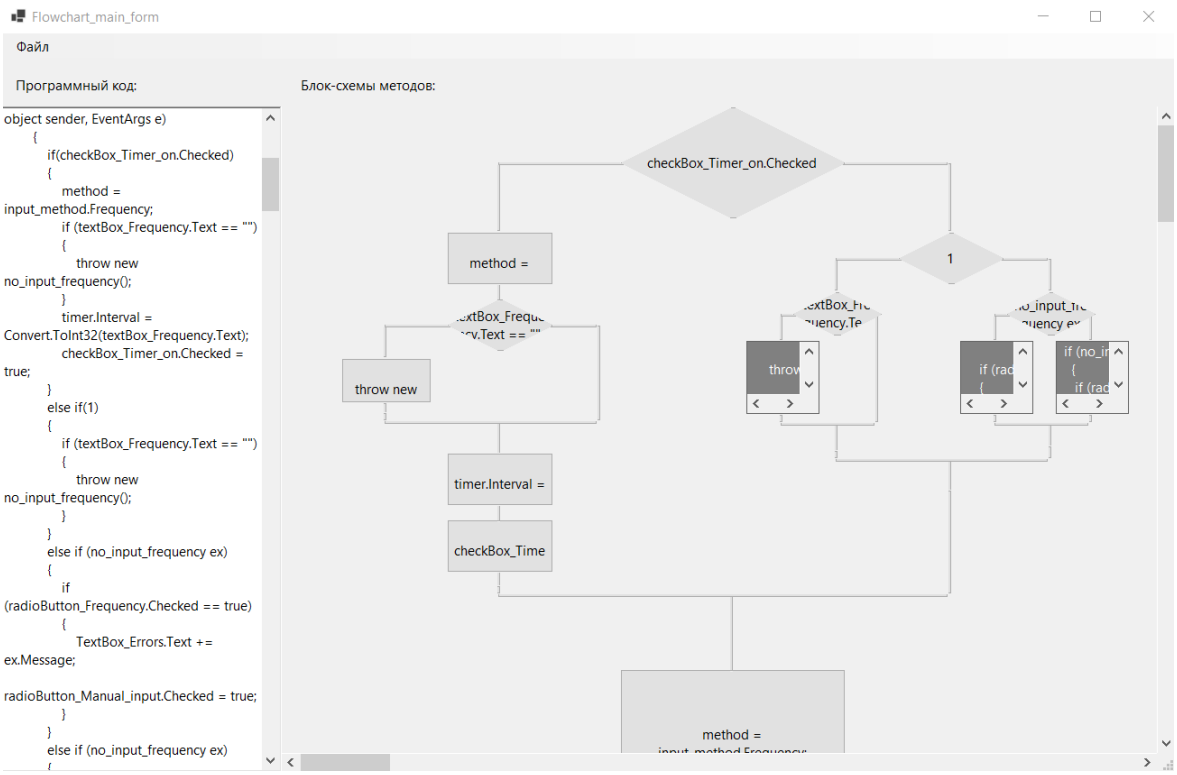


Рис. 9. Отрисовка методов со структурами if и else if.

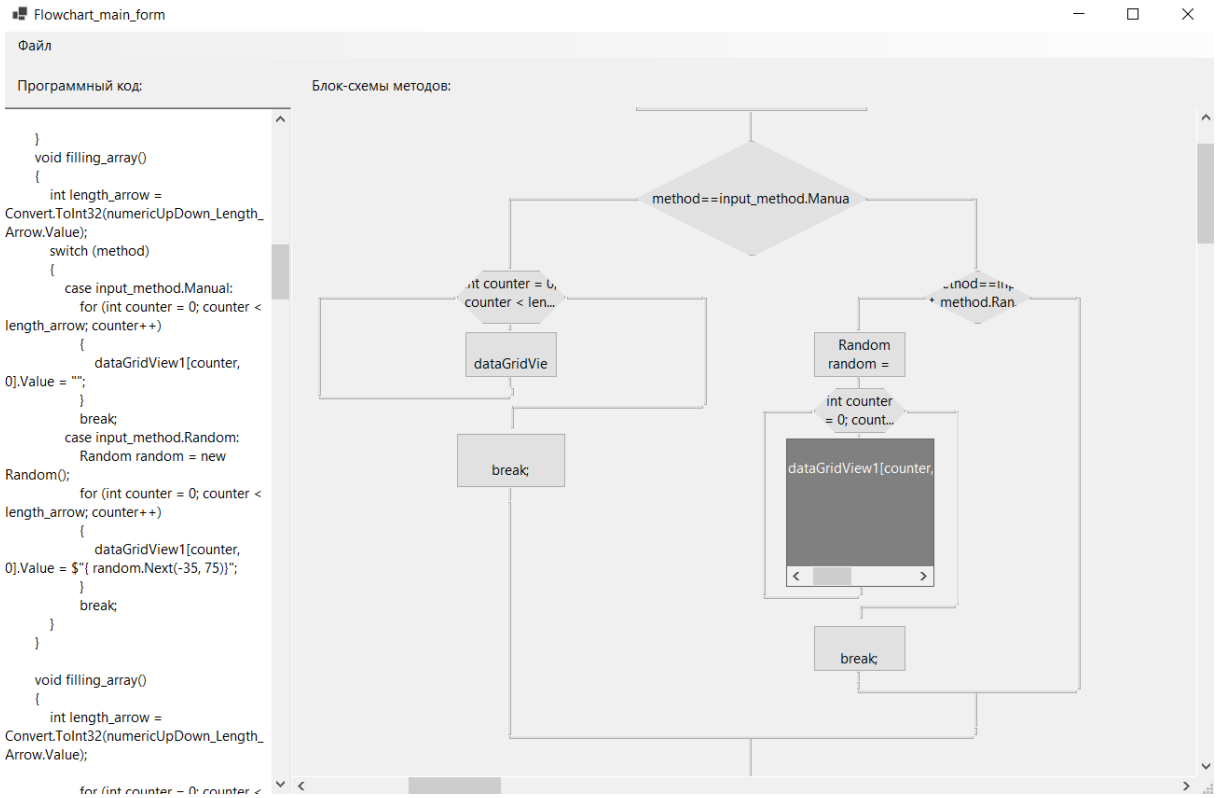


Рис. 10. Отрисовка методов со структурами switch и for

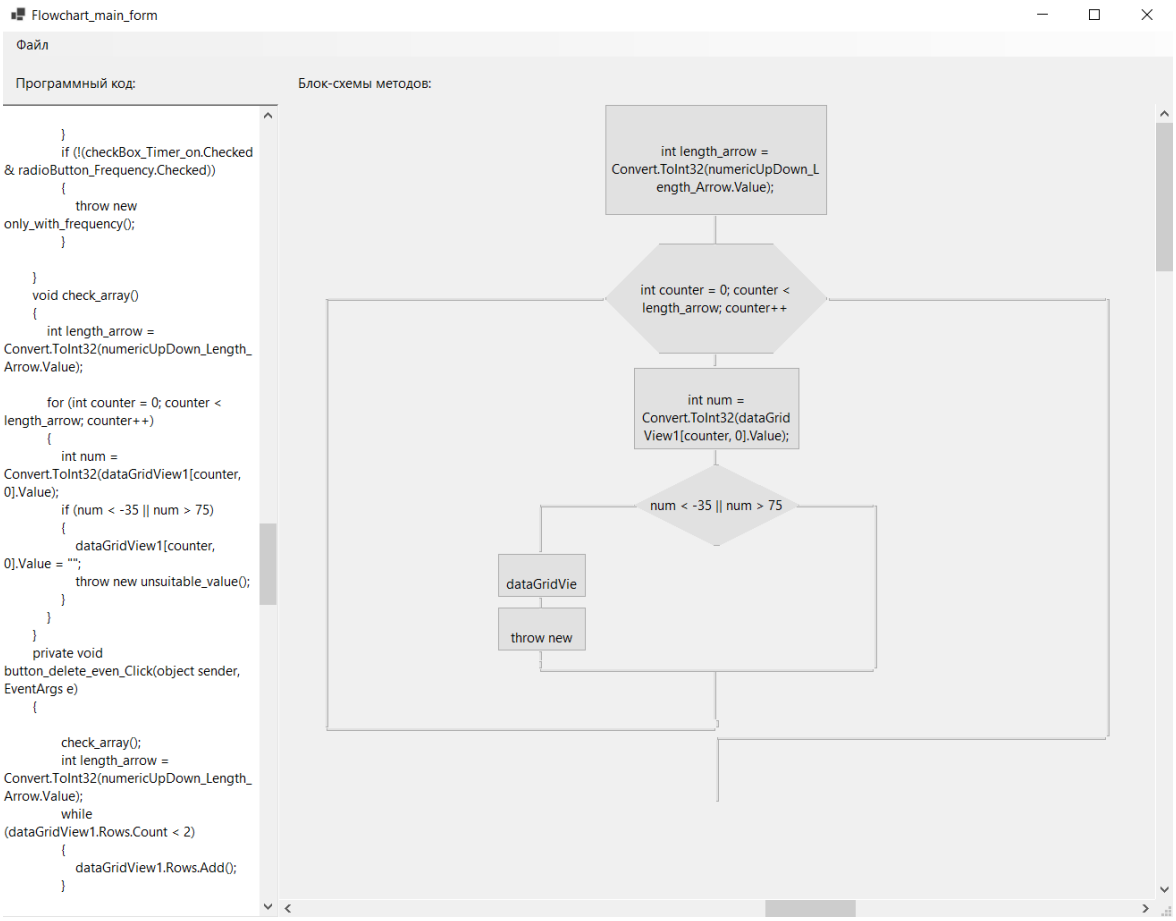


Рис. 11. Отрисовка методов со структурами for и if.

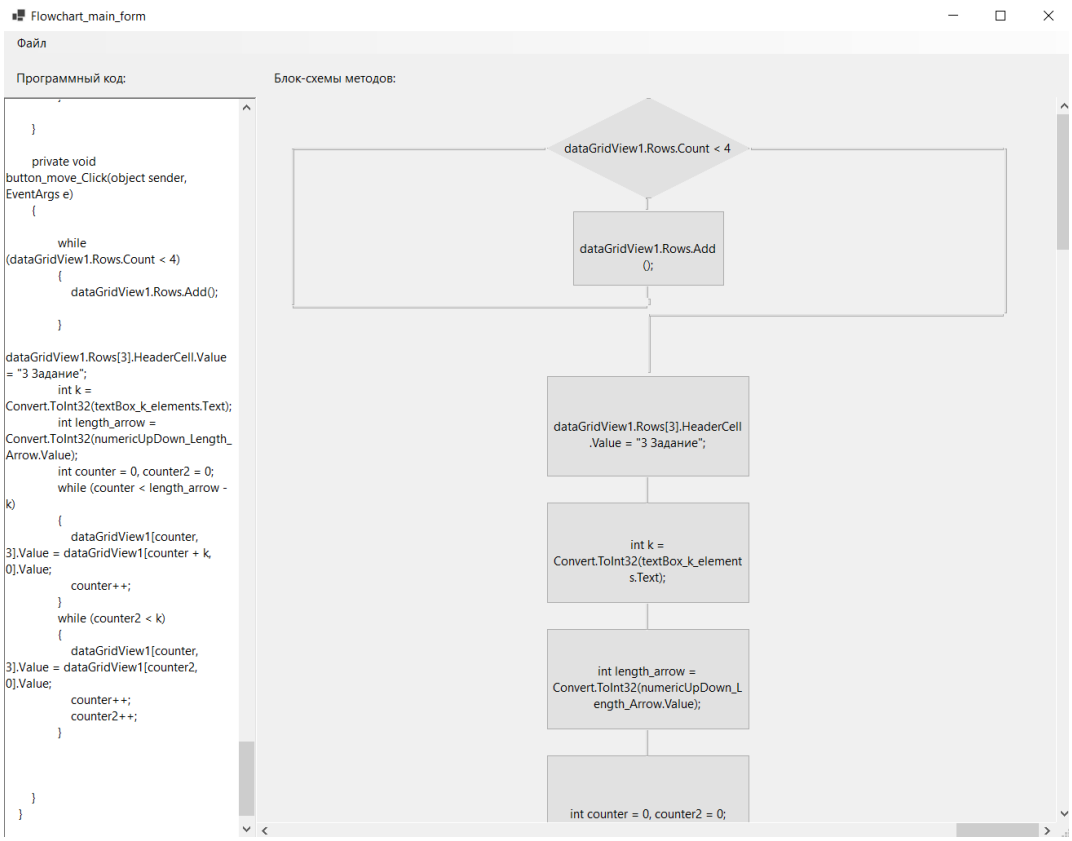


Рис. 12. Отрисовка методов со структурой while.

При нажатии на какой-либо структурный блок открывается отдельное окно с отдельной БСА данного блока. Пример представлен на рис.13.

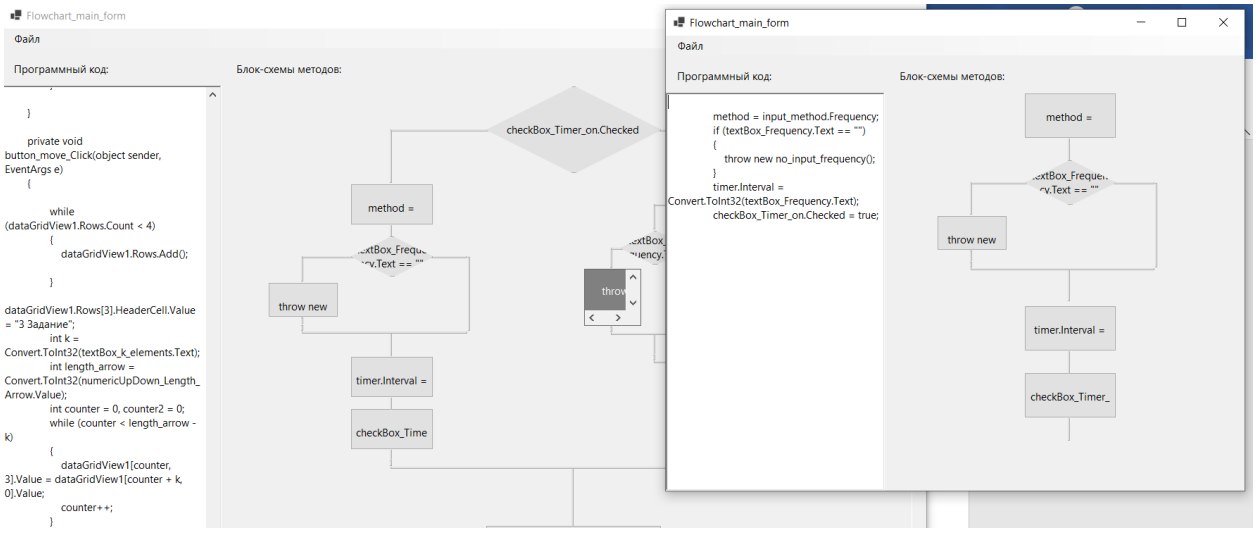


Рис. 13. Пример отдельного окна каждого блока.

При нажатии на какой-либо структурный элемент открывается отдельное окно с описанием его типа. Пример представлен на рис. 14.

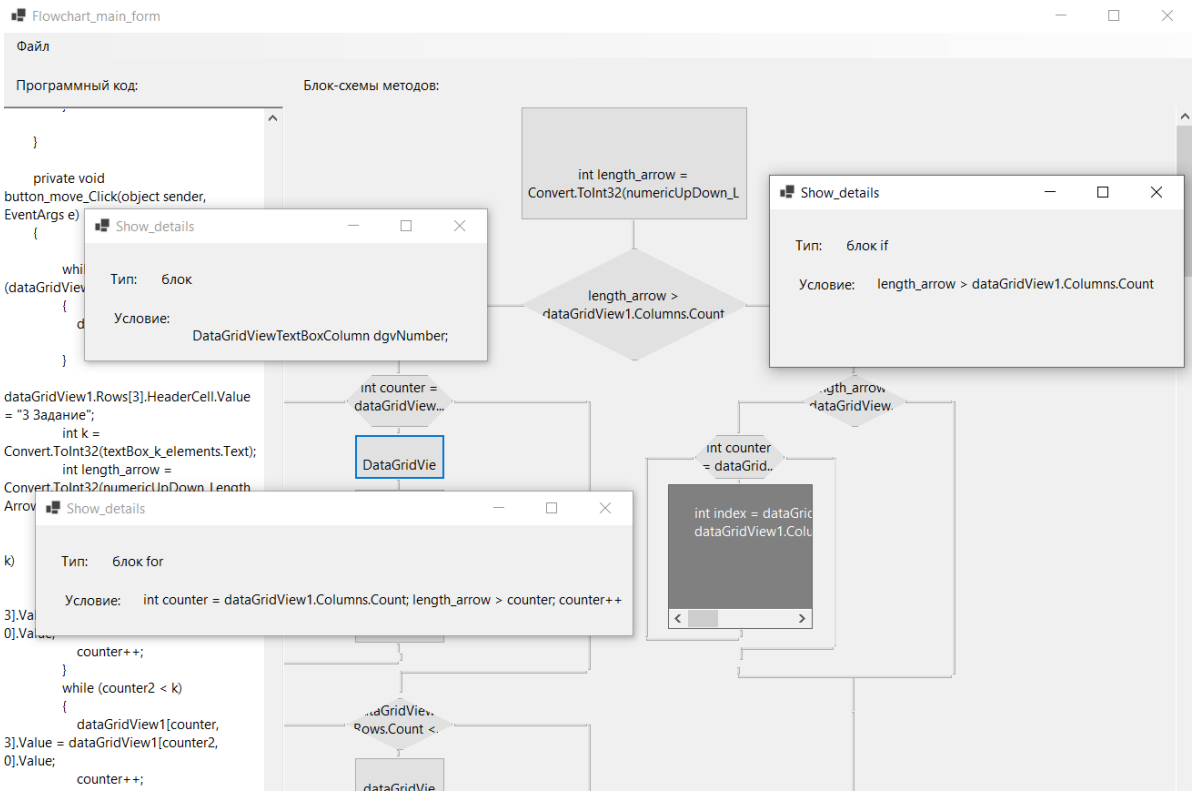


Рис. 14. Пример описания блоков.

Программа не отрисовывает БСА ниже 4 уровня, для него создается отдельная панель более темного цвета, при нажатии на которую БСА отрисуется в отдельном окне. Пример представлен на рис. 15.

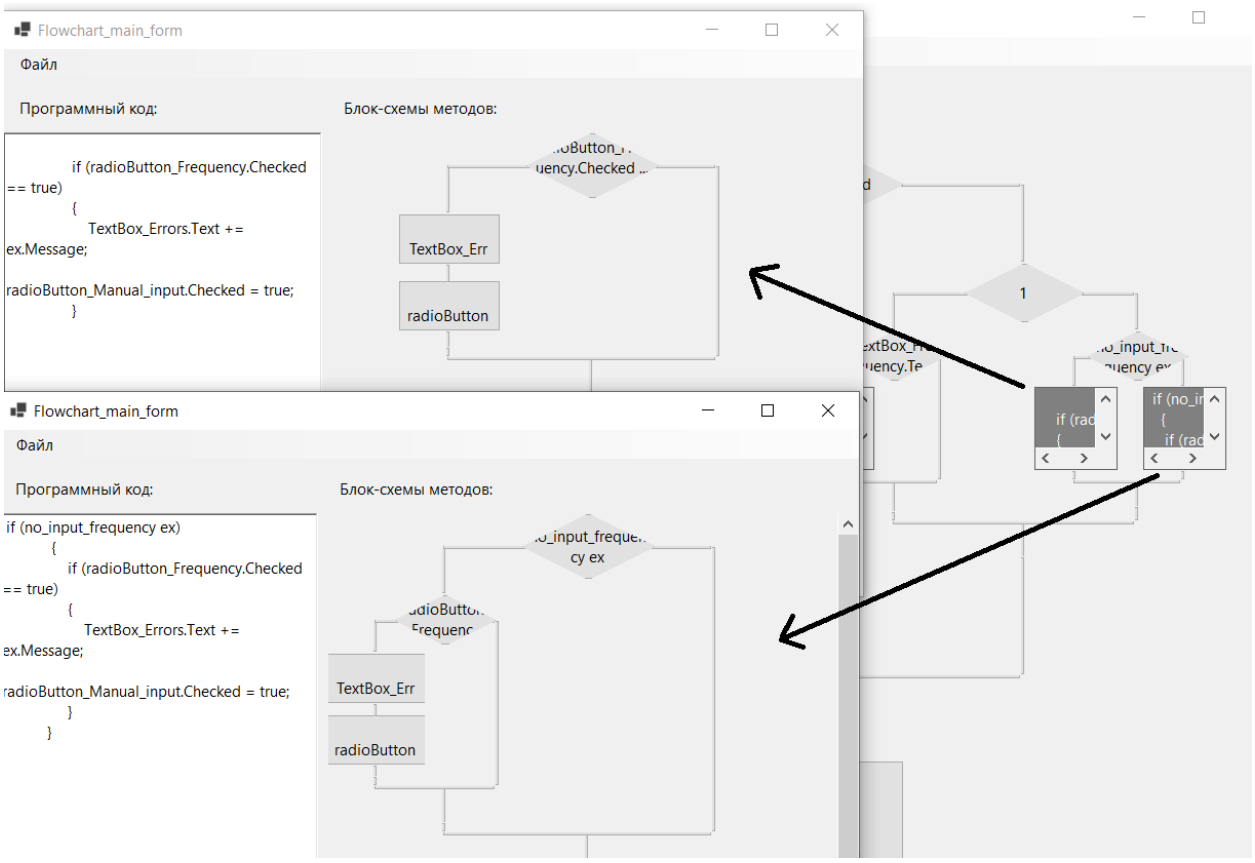


Рис. 15. Отрисовка панелей 4 уровня.

Действие сохранить в БД становится доступным только если есть что сохранять. При сохранении сначала открывается форма для создания названия, если названия еще нет в БД, то код сохраняется в БД. Возможные ошибки при сохранении представлены на рис. 16 и 17.

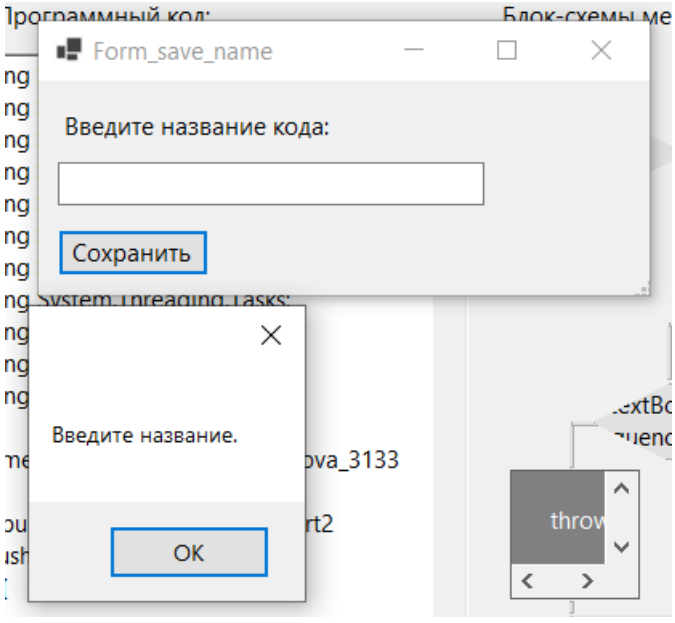


Рис. 16. Вывод ошибки при отсутствии названия.

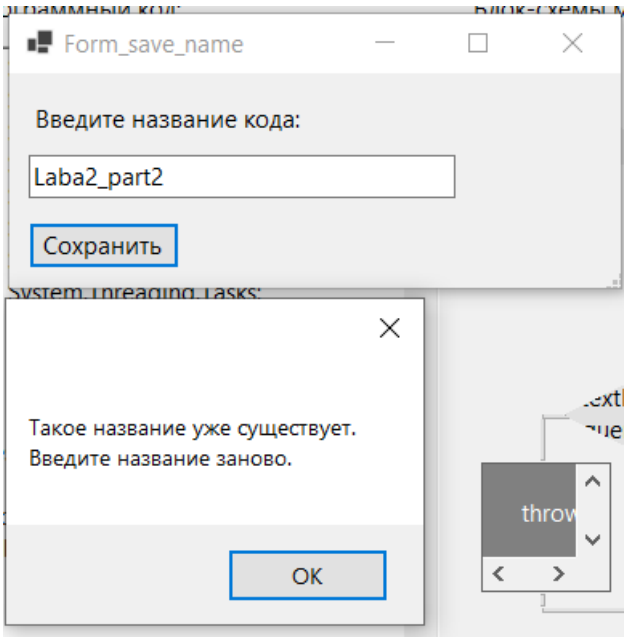


Рис. 17. Вывод ошибки при уже существующем названии.

При открытии сначала открывается форма для выбора по названию кода. Пример ошибки представлен на рис. 18.

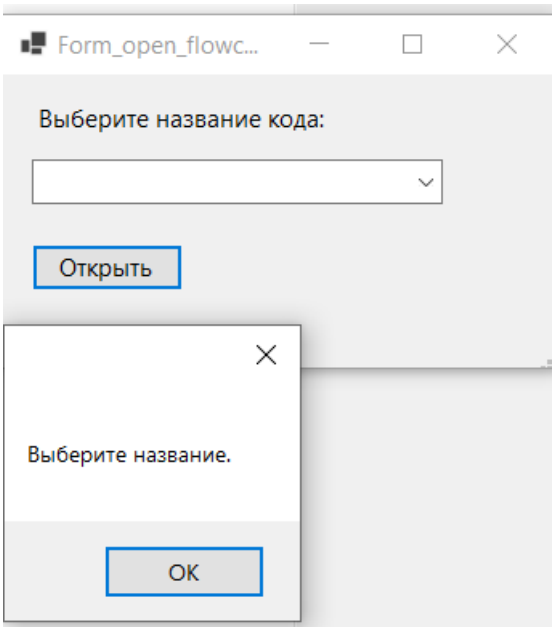


Рис. 18. Пример ошибки при отсутствии выбора.

Таким образом работает разработанная программа, выполняя поставленные задачи.

Для детального разбора результативности программы построим графиков зависимости времени выполнения и объемов потребляемой памяти от размера файла с кодом. Графики представлены на рис. 19 и 20.

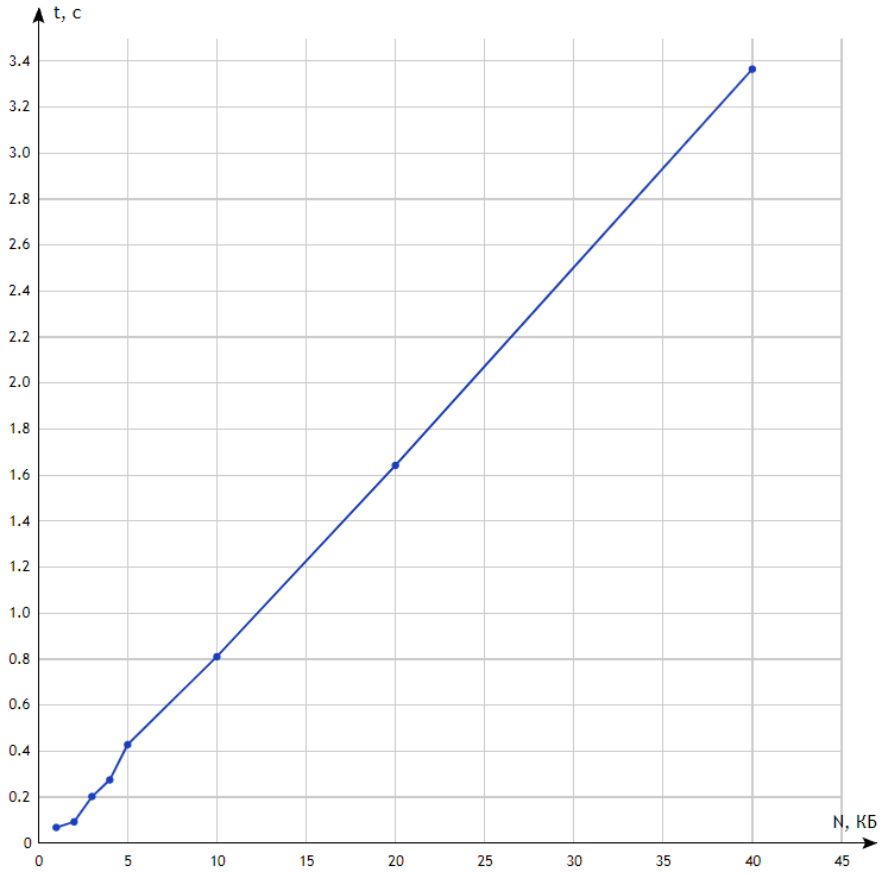


Рис. 19. График зависимости времени выполнения от размера файла.

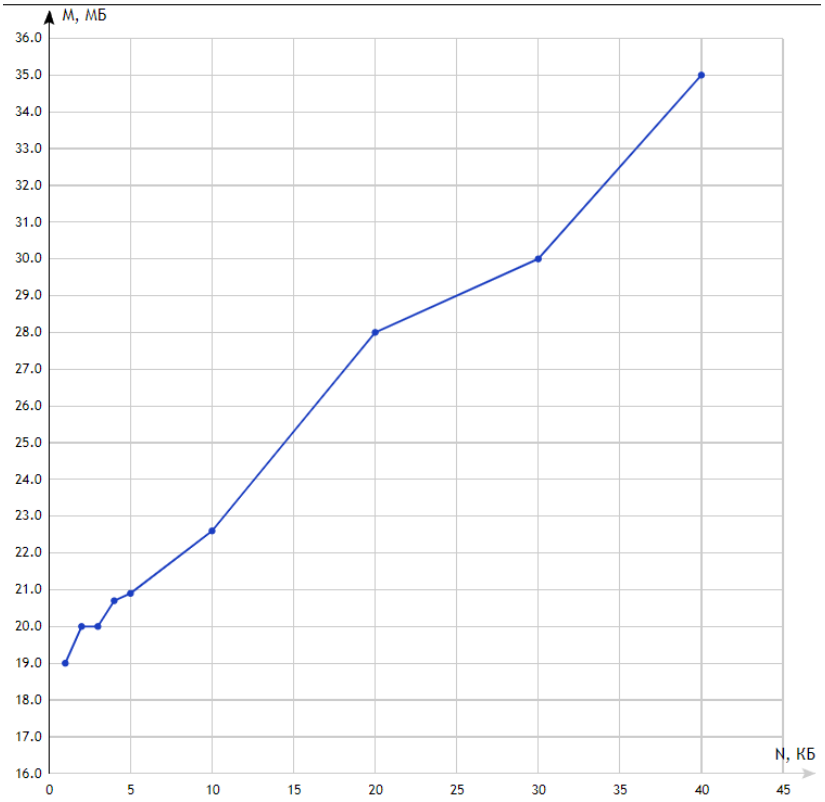


Рис. 20. График зависимости объемов потребляемой памяти от размера файла.

6. Файл – справка

О программе «Построение БСА по исходному программному файлу».

Данная программа позволяет пользователю построить БСА по исходному программному файлу. Кол-во уровней метода – неограничено. Кол-во методов - до 30 на одной форме, при большем кол-ве методов в одном классе создаются дополнительные окна с продолжением отстроенных БСА.

Также есть ограничения на вводимый пользователем программный код:

- Ограничение на использование блока try catch
- Программа отстраивает БСА для следующих структур: if, if else, else if, for, while, switch и их комбинаций
- для корректного подключения к БД в JSON-файле необходимо изменить название сервера

Сценарий работы приложения:

1) После запуска программы пользователь открывает программный файл. Сделать можно двумя способами: либо локальный файл, либо сохраненный в БД. Варианты представлены на рис. 21.

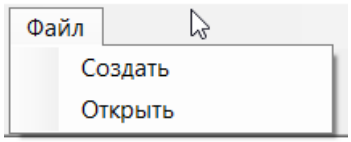


Рис. 21. Изначальный выбор действий.

2) После открытия программа отрисовывает блок-схему введенного алгоритма.

3) Пользователь исследует построенную блок-схему или сохраняет в БД.

ЗАКЛЮЧЕНИЕ

В ходе курсовой работы была разработана система для построения БСА по программному коду. Конечный вес исполняемого файла: 145 КБ. Программа работает под управлением операционной системы Windows. Тестирование проводилось на различных системах ОС Windows 10 изменений не наблюдалось (для корректного подключения к БД в JSON-файле необходимо изменить название сервера).

Временные показатели достаточно хорошие. Отрисовка даже большого кол-ва методов требует не очень много времени, однако больше всего времени требуется на подключение к БД.

Программа имеет широкую область применения. Это приложение позволяет пользователям создавать блок-схемы, которые отображают последовательность шагов, условия и циклы в программе. Приложение для построения блок-схем алгоритмов может быть использовано в различных областях, включая программирование, математику, физику, инженерию и т.д. Это приложение может быть полезно для учащихся, которые изучают основы программирования и алгоритмов, а также программа может пригодиться студентам младших курсов технических вузов, которые выполняют практические работы по программированию.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Регулярные выражения//metanit.com: - [сайт] – URL: https://metanit.com/sharp/tutorial/7.4.php
2. Изменение содержимого строки в C #//microsoft.com – [сайт] – URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/how-to/modify-string-contents#code-try-1
3. Регулярные выражения// regex.sorokin.engineer: - [сайт] – URL: https://regex.sorokin.engineer/ru/latest/regular_expressions.html
4. EventHandler<TEventArgs> Делегат//microsoft.com – [сайт] – URL: https://learn.microsoft.com/ru-ru/dotnet/api/system.eventhandler-1?view=net-7.0
5. Panel Класс// microsoft.com – [сайт] – URL: https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.panel?view=windowsdesktop-7.0
6. Наследование// metanit.com: - [сайт] – URL: https://metanit.com/sharp/tutorial/3.7.php
7. Работа с JSON// metanit.com: - [сайт] – URL: https://metanit.com/sharp/tutorial/6.5.php
8. Обход дерева в глубину//электронный текст - [статья]- URL: https://ilyachalov.livejournal.com/177219.html
9. Бинарные деревья поиска и рекурсия – это просто // habr.com – [статья] – URL: https://habr.com/ru/articles/267855/
10. Обработка исключений// metanit.com: - [сайт] – URL: https://metanit.com/sharp/tutorial/2.14.php
11. Работа со строками// metanit.com: - [сайт] – URL: https://metanit.com/sharp/tutorial/7.1.php
12. Рефлексия// metanit.com: - [сайт] – URL: https://metanit.com/sharp/tutorial/14.1.php
13. Работа с файлами. Классы File и FileInfo// metanit.com: - [сайт] – URL: https://metanit.com/sharp/tutorial/5.3.php

14. Динамическое создание элементов на форме.: Справочник по C#//csharpcoderr.com - [статья] – URL: https://csharpcoderr.com/1208/
15. Элемент управления ToolStripPanel//microsoft.com – [сайт] – URL: https://learn.microsoft.com/ru-ru/ dotnet/ desktop/ winforms/ controls/ toolstrippanel-control ?view=netframeworkdesktop-4.8
16. Размещение элементов управления в формах Windows Forms//microsoft.com – [сайт] – URL: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/controls/putting-controls-on-windows-forms?view=netframeworkdesktop-4.8
17. Практическое руководство. Добавление или удаление элемента в коллекции элементов управления во время выполнения //microsoft.com – [сайт] – URL: https://learn.microsoft.com/ru-ru/dotnet /desktop /winforms/controls/how-to-add-to-or-remove-from-a-collection-of-controls-at-run-time?view=netframeworkdesktop-4.8
18. Разработка пользовательских элементов управления Windows Forms в .NET Framework//microsoft.com – [сайт] – URL: https://learn.microsoft.com/ru -ru/ dotnet/ desktop/ winforms/ controls/ developing-custom-windows-forms-controls?view=netframeworkdesktop-4.8
19. Создание собственных элементов управления//microsoft.com – [сайт] – URL: https://learn.microsoft.com/ru-ru/dotnet/ desktop/ winforms/ controls/ varieties-of-custom-controls?view=netframeworkdesktop-4.8
20. События элементов управления Windows Forms //microsoft.com – [сайт] – URL: https://learn.microsoft.com/ru-ru/dotnet/desktop /winforms/controls/ events-in-windows-forms-controls?view=netframeworkdesktop-4.8
21. Свойства элементов управления Windows Forms //microsoft.com – [сайт] – URL: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/controls/ properties-in-windows-forms-controls?view=netframeworkdesktop-4.8
22. Реализация методов в специализированных элементах управления //microsoft.com – [сайт] – URL: https://learn.microsoft.com/ru-

ru/dotnet/desktop/winforms/controls/method-implementation-in-custom-controls?view=netframeworkdesktop-4.8

23. Функциональная классификация элементов управления Windows Forms //microsoft.com – [сайт] – URL: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/controls/windows-forms-controls-by-function?view=netframeworkdesktop-4.8
24. Пошаговое руководство. Создание составного элемента управления с помощью C# //microsoft.com – [сайт] – URL: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/controls/walkthrough-authoring-a-composite-control-with-visual-csharp?view=netframeworkdesktop-4.8
25. Пошаговое руководство. Наследование от элемента управления Windows Forms с помощью C# //microsoft.com – [сайт] – URL: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/controls/walkthrough-inheriting-from-a-windows-forms-control-with-visual-csharp?view=netframeworkdesktop-4.8

ПРИЛОЖЕНИЕ

Приложение №1. Программный код класса Block

```
public class Block
   {
      Panel panel;
      string code;
      public int whidth_block, height_block=0, gap_block;
      Flowchart_main_form Main;
      //Pen p;
      //Graphics graphics;
      //SolidBrush brush;
      int x, y;//текущее положение в координатах
      int lvl;
      public Block() { }
      public Block(string Code, Flowchart_main_form main,int lv, Panel pan, out Panel pan_well)
{

         Main = main;
         code = Code;
         lvl = lv;
         panel = pan;
         whidth_block = panel.Width / 4;
         if(whidth_block<100) whidth_block=100;
         height_block = whidth_block / 2;
         x = pan.Width / 2; y = height_block/2;// начальные значения на любой панели
         gap_block = height_block +30/lvl;
         //ylast = y0;
         //MessageBox.Show(code);


         if(lvl%4 == 0)
         {
            pan_well = new Panel();
           //pan_well.AutoSize = true;
            pan_well.Width= panel.Width;
            pan_well.Height= panel.Width;
            pan_well.Location = panel.Location;
            pan_well.BackColor = Color.Gray;
            pan_well.BorderStyle = BorderStyle.FixedSingle;
            pan_well.AutoScroll = true;
            Label l = new Label();
            l.Text = code.Replace("    "," ");
            l.AutoSize = true;
            l.ForeColor = Color.White;
            l.Location = new Point(0, 0);
            //l.Dock = DockStyle.Fill;
            //while (l.Width < System.Windows.Forms.TextRenderer.MeasureText(l.Text,
            //    new Font(l.Font.FontFamily, l.Font.Size, l.Font.Style)).Width)
            //{
            //    l.Font = new Font(l.Font.FontFamily, l.Font.Size - 0.5f, l.Font.Style);
```

```
            //}
            pan_well.Controls.Add(l);

            l.Click += click_panel;
        }
        else
        {
            get_action(0);
            //panel.Height += 50;
            panel.Height = y + height_block / 2 + 30 / lvl;
            pan_well = panel;
            if (lvl % 3 == 0 || lvl % 3 == 2)
                pan_well.Click += click_panel;
        }

    }
    //парсит блок по действиям
    int endindex;
    public void get_action(int startindex)
    {

        Regex regex = new Regex(@"(if)|(else)|(while)|(switch)|([^{};]*;)"); //
        // Regex regex = new Regex(@"if\s*\([^{}]*\)\s*{[^{}]*}");
        MatchCollection matches = regex.Matches(code.Substring(startindex));
        //MessageBox.Show(code.Substring(startindex));
        if (matches.Count != 0)
        {
            int y_if=0,x_if=0,y_for=0,height_if_block_true=0, width_if_block_true = 0,
count_if=0;
            bool true_block=false,false_block=false, block_for=false;
            foreach (Match match in matches)
            {
                if (match.Index >= endindex)
                {
                    switch (match.Value)
                    {
                        case "if":
                            int first;
                            int last = get_index_lvl_s(match.Index, out first);//first, last индексы в code
                            string term = code.Substring(first+1, last - first-1);
                            //MessageBox.Show(term);
                            draw_if_block(term.Replace("    ", ""));
                            int startaction;
                            int endaction = get_index_lvl(last, out startaction);// индексы в code
                            string text_new_if_block_true = code.Substring(startaction + 1, endaction -
startaction );//тк длина

                            Panel pan_if_true=new Panel();
                            pan_if_true.Width = panel.Width/ 2-25;
                            pan_if_true.Height = 0;
                            pan_if_true.AutoScroll = false;
                            pan_if_true.Location= new Point(5,y+ gap_block/2);
```

```
                    Block if_block_true = new Block(text_new_if_block_true, Main,lvl+1,pan_if_true, out Panel pan_well_low_if_true) ;

                    panel.Controls.Add(pan_well_low_if_true);
                    panel.Invalidate();
                    get_arrow_horisont(new Point(2 + pan_well_low_if_true.Width / 2, y), new Point(x, y));
                    get_arrow_vert(new Point(2 + pan_well_low_if_true.Width / 2, y),new Point (2 + pan_well_low_if_true.Width / 2, y+ gap_block/2));

                    true_block = true;
                    y_if = y;
                    x_if = x;
                    height_if_block_true = pan_well_low_if_true.Height;
                    width_if_block_true = pan_well_low_if_true.Width;
                    endindex = endaction + startindex;

                    y += pan_well_low_if_true.Height  + gap_block/2;
                    //if (y < 3 * panel.Width)
                    //{
                   //panel.Height += pan_well_low_if_true.Height;
                   // panel.Height = y;
                    //}
                    y += gap_block;
                    break;
                case "else":
                    int startaction_else;
                    int endaction_else = get_index_lvl(match.Index, out startaction_else);
                    string text_new_if_block_false = code.Substring(startaction_else + 1, endaction_else - startaction_else - 1);

                    Panel pan_if_false = new Panel();
                    pan_if_false.Width = panel.Width / 2-25;
                    pan_if_false.Height = 10;
                    pan_if_false.AutoScroll = false;
                    pan_if_false.Location = new Point(panel.Width/2+5, y_if + gap_block/2);
                    Block if_block_false = new Block(text_new_if_block_false, Main,lvl+1, pan_if_false, out Panel pan_well_low_if_false);
                    panel.Controls.Add(pan_well_low_if_false);
                    panel.Invalidate();


                    get_arrow_horisont(new Point(x_if, y_if), new Point(x_if+ 2 + pan_well_low_if_false.Width / 2, y_if));
                    get_arrow_vert(new Point(x_if+2 + pan_well_low_if_false.Width / 2, y_if), new Point(x_if+2 + pan_well_low_if_false.Width / 2, y_if  + gap_block/2));

                    false_block = true;
                    endindex = endaction_else + startindex;
                    //проверка какая ветвь блока if больше
                    if(height_if_block_true < pan_well_low_if_false.Height)
                    {
```

```
                        y =y_if+ pan_well_low_if_false.Height+ gap_block/2;
                        y += gap_block;
                        //if (y < 3 * panel.Width)
                        //{
                        // panel.Height += (pan_well_low_if_false.Height - height_if_block_true);
                        //}
                    }

                    get_arrow_vert(new Point(2 + width_if_block_true / 2, y_if + height_if_block_true +gap_block/2 ),//+ height_block/2 + gap_block
                        new Point(2 + width_if_block_true / 2, y-gap_block+10));
                    get_arrow_vert(new Point(x_if + 2 + width_if_block_true / 2, y_if+ pan_well_low_if_false.Height+gap_block/2),
                        new Point(x_if + 2 + width_if_block_true / 2, y - gap_block + 10));
                    get_arrow_horisont(new Point(2 + width_if_block_true / 2, y - gap_block + 10),
                        new Point(x_if + 2 + width_if_block_true / 2, y - gap_block + 10));
                    get_arrow_vert(new Point(x - 3, y - gap_block + 10), new Point(x - 3, y - height_block / 2 + 3));
                    true_block = false;count_if = 0;

                   // panel.Height = y;
                    break;

                case "while":
                f:
                    int first_for;
                    int last_for = get_index_lvl_s(match.Index, out first_for);//first, last индексы в code
                    string term_for = code.Substring(first_for + 1, last_for - first_for - 1);
                    if (block_for)
                    {
                        block_for = false;
                        draw_for_block(term_for.Replace("    ", ""));
                    }
                    else draw_if_block(term_for.Replace("    ", ""));
                    get_arrow_vert(new Point(x - 3, y + height_block / 2 - 1), new Point(x - 3, y + gap_block/2));

                    int startaction_for;
                    int endaction_for = get_index_lvl(last_for, out startaction_for);// индексы в code
                    string text_new_for_block = code.Substring(startaction_for + 1, endaction_for - startaction_for );//тк длина

                    Panel pan_for = new Panel();
                    pan_for.Width = 3* panel.Width / 4;
                    pan_for.Height = 0;
                    pan_for.AutoScroll = false;
                    pan_for.Location = new Point(x-3* panel.Width / 8, y + gap_block / 2);
```

```
                    Block for_block = new Block(text_new_for_block, Main, lvl + 1, pan_for, out
Panel pan_well_low_for);

                    y_for = y;
                    panel.Controls.Add(pan_well_low_for);
                    panel.Invalidate();
                    endindex = endaction_for + startindex;
                    y += gap_block / 2 + pan_well_low_for.Height;
                    //отрисовка линий
                    get_arrow_vert(new Point(x, y), new Point(x, y + 10));
                    get_arrow_horisont(new Point(x- 3 * panel.Width / 8-whidth_block/4, y+10),
new Point(x, y + 10));
                    get_arrow_vert(new Point(x - 3 * panel.Width / 8 - whidth_block / 4, y_for),
new Point(x - 3 * panel.Width / 8 - whidth_block / 4, y + 10));
                    get_arrow_horisont(new Point(x - 3 * panel.Width / 8 - whidth_block / 4,
y_for), new Point(x-whidth_block/2, y_for));
                    get_arrow_horisont( new Point(x + whidth_block / 2, y_for), new Point(x + 3
* panel.Width / 8 + whidth_block / 4, y_for));
                    get_arrow_vert(new Point(x + 3 * panel.Width / 8 + whidth_block / 4, y_for),
new Point(x + 3 * panel.Width / 8 + whidth_block / 4, y + 20));
                    get_arrow_horisont( new Point(x, y + 20), new Point(x + 3 * panel.Width / 8
+ whidth_block / 4, y + 20));
                    get_arrow_vert(new Point(x, y+20), new Point(x, y +gap_block-
height_block/2));
                    //y += gap_block - height_block / 2;
                    y += gap_block+lvl*2;
                    break;
                //case "switch"://то же самое как и вовторение else if

                //    last = get_index_lvl_s(match.Index, out first);//first, last индексы в code
                //    term = code.Substring(first+1, last - first-1);
                //    //MessageBox.Show(term);
                //    draw_Rectangle(term.Replace("    ", ""));

                //    break;
                default:
                    // проверка на цикл for
                    Regex r_for = new Regex(@"\bfor");
                    MatchCollection match_for = r_for.Matches(match.Value);
                    if (match_for.Count != 0) {block_for = true; goto f; }

                    endindex += match.Length;
                    draw_Rectangle(match.Value.Replace("    ", ""));
                    get_arrow_vert(new Point(x-3, y + height_block / 2-1), new Point(x-3, y +
gap_block-height_block/2+3));
                    y += gap_block;
                    //panel.Height = y;
                    break;

            }
            if (true_block && false_block==false)
            {
```

```
                    count_if++;
                    if (count_if == 2)
                    {
                        get_arrow_horisont(new Point(x_if,y_if),new Point(x_if + 2 +
width_if_block_true / 2, y_if));
                        get_arrow_vert(new Point( 2 + width_if_block_true / 2,
y_if+height_if_block_true + gap_block/2),
                            new Point(2 + width_if_block_true / 2, y_if + height_if_block_true +
gap_block/2+10));
                        get_arrow_vert(new Point(x_if + 2 + width_if_block_true / 2, y_if),
                            new Point(x_if + 2 + width_if_block_true / 2, y_if + height_if_block_true
+gap_block/2+10));
                        get_arrow_horisont(new Point(2 + width_if_block_true / 2, y_if +
height_if_block_true  + gap_block/2+10),
                            new Point(x_if + 2 + width_if_block_true / 2, y_if + height_if_block_true
+ gap_block/2+10 ));
                        get_arrow_vert(new Point(x - 3, y_if + height_if_block_true + gap_block / 2
+ 10), new Point(x - 3, y_if + height_if_block_true + gap_block+10));
                        true_block = false;
                        count_if = 0;
                    }
                }else if (false_block)
                {
                    false_block = false;
                }
            }
        }

        if (true_block )
        {
            get_arrow_horisont(new Point(x_if, y_if), new Point(x_if + 2 + width_if_block_true
/ 2, y_if));
            get_arrow_vert(new Point(2 + width_if_block_true / 2, y_if + height_if_block_true
+ gap_block/2),
                new Point(2 + width_if_block_true / 2, y_if + height_if_block_true +
gap_block/2+10 ));
              get_arrow_vert(new Point(x_if + 2 + width_if_block_true / 2, y_if),
                new Point(x_if + 2 + width_if_block_true / 2, y_if + height_if_block_true +
gap_block/2 +10));
              get_arrow_horisont(new Point(2 + width_if_block_true / 2, y_if +
height_if_block_true  + gap_block/2+10),
                new Point(x_if + 2 + width_if_block_true / 2, y_if + height_if_block_true +
gap_block/2+10 ));
            get_arrow_vert(new Point(x - 3, y_if + height_if_block_true + gap_block / 2 + 10),
new Point(x - 3, y + gap_block - height_block / 2 + 3));
            true_block = false;
              count_if = 0;
            y = y_if + height_if_block_true + gap_block+gap_block/2+5 ;
        }

        y -= gap_block;
        //MessageBox.Show(matches[0].Value);
```

```
            // get_action(endindex);
        }

    }
    public void draw_Rectangle(string action)
    {
        //if (y < 2 * panel.Width)
            //panel.Height += height_block+gap_block;



        Button btn = new Button();
        btn.Tag = "блок";
        btn.Text = action;
        btn.Location = new Point(x - whidth_block / 2, y - height_block / 2);
        btn.Height = height_block;
        btn.Width = whidth_block;
        //размещаем на панели

            btn.Click += click_button;
        panel.Controls.Add(btn);

        panel.Invalidate();
    }
    public void draw_if_block(string action)
    {

        //if (y < 2 * panel.Width)
            //panel.Height += height_block+gap_block;

        Button btn = new Button();
        btn.Tag = "блок if";
        btn.Text =  action;
        btn.Location = new Point(x-whidth_block/2, y-height_block/2);
        btn.Height = height_block;
        btn.Width = whidth_block;
        btn.AutoEllipsis = true;

        // Set a new rectangle to the same size as the button's
        // ClientRectangle property.


        System.Drawing.Drawing2D.GraphicsPath myPath =new
System.Drawing.Drawing2D.GraphicsPath();
        Point[] points = {new Point(0,height_block/2), new Point( whidth_block/2, 0),
            new Point( whidth_block, height_block / 2), new Point( whidth_block/2, height_block)
};

        myPath.AddPolygon(points);

        Region myRegion = new Region(myPath);
```

```
            btn.Region = myRegion;

            btn.Click += click_button;


            //размещаем на панели
            panel.Controls.Add(btn);

            panel.Invalidate();
        }

        private void click_button(object sender, EventArgs e)
        {
            //
            Show_details show_Details = new Show_details();
            Button btn = sender as Button;
            show_Details.label_type_block.Text = (string)btn.Tag;
            show_Details.label_context.Text = btn.Text;
            show_Details.Show();
            //throw new NotImplementedException();
        }

        public void draw_for_block(string action)
        {

            //if (y < 2 * panel.Width)
            //panel.Height += height_block+gap_block;

            Button btn = new Button();
            btn.Tag = "блок for";
            btn.Text =  action;
            btn.Location = new Point(x - whidth_block / 2, y - height_block / 2);
            btn.Height = height_block;
            btn.Width = whidth_block;
            btn.AutoEllipsis = true;

            // Set a new rectangle to the same size as the button's
            // ClientRectangle property.


            System.Drawing.Drawing2D.GraphicsPath myPath = new
System.Drawing.Drawing2D.GraphicsPath();
            Point[] points = {new Point(0,height_block/2), new Point( whidth_block/4, 0),new Point(
whidth_block/2+whidth_block/4, 0),
                new Point( whidth_block, height_block / 2), new Point(
whidth_block/2+whidth_block/4, height_block), new Point( whidth_block/4, height_block) };

            myPath.AddPolygon(points);

            Region myRegion = new Region(myPath);
            btn.Region = myRegion;
            //Event_Args_button e = new Event_Args_button("блок for", action);
```

```
    btn.Click += click_button;

    //размещаем на панели
    panel.Controls.Add(btn);

    panel.Invalidate();
}
//находит индекс парной фигурной скобки
public int get_index_lvl(int startindex, out int first)
{
    int count = 0;
    Regex regex = new Regex(@"{|}");
    // Regex regex = new Regex(@"if\s*\([^{}]*\)\s*{[^{}]*}");
    MatchCollection matches = regex.Matches(code.Substring(startindex));
    first = matches[0].Index + startindex; int last=first;
    foreach (Match match in matches)
    {
        //MessageBox.Show(match.Value);
        if (match.Value == "{") count++;

        else if (match.Value == "}")
        {
            count--;
            if (count == 0)
            {
                last = match.Index + startindex;
                break;

            }
        }
    }
    return last;
}
//находит индекс парной скобки
public int get_index_lvl_s(int startindex, out int first)
{
    int count = 0;
    Regex regex = new Regex(@"\(|\)");
    // Regex regex = new Regex(@"if\s*\([^{}]*\)\s*{[^{}]*}");
    MatchCollection matches = regex.Matches(code.Substring(startindex));
    first = matches[0].Index+startindex;

    int last;
    foreach (Match match in matches)
    {

        if (match.Value == "(") count++;

        else if (match.Value == ")")
        {
            count--;
```

```
            if (count == 0)
            {
               last = match.Index + startindex;
               return last;
            }
         }
      }
      return 0;
   }

   void click_panel(object sender, EventArgs e)
   {
      Flowchart_main_form flowchart_Main_Form = new Flowchart_main_form();

      flowchart_Main_Form.richTextBox_text_program.Text= code;
      Block block_click_panel = new
Block(flowchart_Main_Form.richTextBox_text_program.Text, flowchart_Main_Form, 1,
flowchart_Main_Form.panel_picture_flowchart_main, out
flowchart_Main_Form.panel_picture_flowchart_main);
      flowchart_Main_Form.Show();
   }

   private void get_arrow_vert(Point start,Point end)
   {
      Button btn_arrow = new Button();

      btn_arrow.Location = start;
      btn_arrow.Height = end.Y-start.Y;
      btn_arrow.Width = 3;
      btn_arrow.AutoEllipsis = true;

      panel.Controls.Add(btn_arrow);

      panel.Invalidate();
   }
   private void get_arrow_horisont(Point start,Point end)
   {
      Button btn_arrow = new Button();

      btn_arrow.Location = start;
      btn_arrow.Height = 3;
      btn_arrow.Width = end.X-start.X;
      btn_arrow.AutoEllipsis = true;

      panel.Controls.Add(btn_arrow);

      panel.Invalidate();
   }

   public Block Block1
   {
      get => default;
```

```
        set
        {
        }
    }
}
```

Приложение №2. Программный код класса Flowchart_main_form

```
public partial class Flowchart_main_form : Form
{

    public Flowchart_main_form()
    {
        InitializeComponent();
    }
    public Flowchart_main_form(string text)
    {
        InitializeComponent();
        richTextBox_text_program.Text = text;
        get_methods();
    }

    public void SetSelectionStyle(int startIndex, int lastindex, FontStyle style, Color color)
    {

        richTextBox_text_program.Select(startIndex, lastindex - startIndex);
        richTextBox_text_program.SelectionFont = new
Font(richTextBox_text_program.SelectionFont, richTextBox_text_program.SelectionFont.Style |
style);
        richTextBox_text_program.SelectionColor = color;
    }

    private void ToolStripMenuItem_create_new_program_fail_Click(object sender, EventArgs
e)
    {
        openFileDialog1.Filter = "cs fails(*.cs)|*.cs";
        if (openFileDialog1.ShowDialog() == DialogResult.OK)//Показать диалог
        {
            try
            {

                panel_picture_flowchart_main.Controls.Clear();
                string Filename = openFileDialog1.FileName;
                string filetext = System.IO.File.ReadAllText(Filename);
                richTextBox_text_program.Text = filetext;
                //get_higher_level(1);
                get_code_conversion();
                get_methods();
                richTextBox_text_program.Text = filetext;
            }
            catch (Exception ex)
            {
```

```
                MessageBox.Show(ex.Message);
            }
        }
    }
    public void get_code_conversion()
    {
        Regex regex; MatchCollection matches;


        //преобразование switch
        regex = new Regex(@"\bswitch\s*\([^{}]*\)");
        matches = regex.Matches(richTextBox_text_program.Text);
        for (int i = matches.Count - 1; i >= 0; i--)
        {

            int first;
            int last = get_index_lvl_s(matches[i].Index, out first);
            //MessageBox.Show(richTextBox_text_program.Text.Substring(matches[i].Index));
            string term = richTextBox_text_program.Text.Substring(first + 1, last - first - 1);// запоминаем переменную

            int f_s;
            int l_s = get_index_lvl(last, out f_s);

            // удаляем cwitch с текста

            //MessageBox.Show(richTextBox_text_program.Text);

            Regex r_arg = new Regex(@"(\b\S*){1}:");
            Regex regex_case = new Regex(@"\bcase\s");

            MatchCollection matchCollection = regex_case.Matches(richTextBox_text_program.Text.Substring(f_s, l_s - f_s));// все case у этого switch


            MatchCollection arg = r_arg.Matches(richTextBox_text_program.Text.Substring(f_s + matchCollection[0].Index + matchCollection[0].Length));
            //MessageBox.Show();

            //MessageBox.Show(richTextBox_text_program.Text.Substring(f_s + match_default.Index));
            for (int j = matchCollection.Count - 1; j > 0; j--)
            {
                //MessageBox.Show(richTextBox_text_program.Text.Substring(f_s + matchCollection[j].Index, arg[j].Index + matchCollection[0].Index + matchCollection[0].Length + arg[j].Length + 1 - matchCollection[j].Index));
                richTextBox_text_program.Text = richTextBox_text_program.Text.Replace(richTextBox_text_program.Text.Substring(f_s + matchCollection[j].Index, arg[j].Index + matchCollection[0].Index + matchCollection[0].Length + matchCollection[j].Length + arg[j].Length + 1 - matchCollection[j].Index),
                    "}" + $" else if ({term}=={arg[j].Value.Substring(0, arg[j].Length - 2)}) " + "{");
```

```
            }

//MessageBox.Show(richTextBox_text_program.Text.Substring(matchCollection[0].Index + f_s));
            richTextBox_text_program.Text = richTextBox_text_program.Text.Replace(richTextBox_text_program.Text.Substring(matches[i].Index, f_s + matchCollection[0].Index + 2 * matchCollection[0].Length + arg[0].Index + arg[0].Length - matches[i].Index + 1),
                $"if ({term}=={arg[0].Value.Substring(0, arg[0].Length - 2)}) " + "{");


        }
        Regex regex_default = new Regex(@"\bdefault");
        Match match_default = regex_default.Match(richTextBox_text_program.Text);
        if (match_default.Success)
            richTextBox_text_program.Text = //richTextBox_text_program.Text.Substring(0, f_s + match_default.Index) + "} else { " + richTextBox_text_program.Text.Substring(f_s + match_default.Index + match_default.Length + 2);
             richTextBox_text_program.Text.Replace(match_default.Value, "} else { ");//.Substring(first + match_default.Index)

        //MessageBox.Show(richTextBox_text_program.Text);


        //преобразование if где нет {}
        regex = new Regex(@"\bif\s*\([^{}]*\)");
        matches = regex.Matches(richTextBox_text_program.Text);
        for (int i = matches.Count - 1; i >= 0; i--)
        {
            Match match = matches[i];
            Regex regex_h = new Regex(@"\S");
            Match match_h = regex_h.Match(richTextBox_text_program.Text.Substring(match.Index + match.Length));
            if (match_h.Value != "{")
            {
                richTextBox_text_program.Text = richTextBox_text_program.Text.Insert(match.Index + match.Length + 1, "{");
                Regex r = new Regex(@";");
                Match m = r.Match(richTextBox_text_program.Text.Substring(match.Index + match.Length));
                richTextBox_text_program.Text = richTextBox_text_program.Text.Insert(match.Index + match.Length + m.Index + 1, "}");
            }
        }

        //MessageBox.Show(richTextBox_text_program.Text);
        //преобразование else if

        regex = new Regex(@"\belse\s*if\b");//вхождение else if
                //MatchCollection matches = regex.Matches(richTextBox_text_program.Text);
```

```
            matches = regex.Matches(richTextBox_text_program.Text);
            //MessageBox.Show(richTextBox_text_program.Text);
            for (int i = matches.Count - 1; i >= 0; i--)// обход с конца
            {
                Match match = matches[i];
                int f;
                int first_if = match.Index + 4;
                //MessageBox.Show(richTextBox_text_program.Text.Substring(first_if));
                int last_if = get_index_lvl(first_if, out f);

                Regex regex_else = new Regex(@"\b\S*\b");//первое вхождение else, если есть

                Match match_else =
regex_else.Match(richTextBox_text_program.Text.Substring(last_if));

                if (match_else.Value == "else")
                {

                    last_if = get_index_lvl(last_if + 1, out f) + 1;
                }

                richTextBox_text_program.Text = richTextBox_text_program.Text.Insert(first_if, "{");
                richTextBox_text_program.Text = richTextBox_text_program.Text.Insert(last_if, "}");
                // get_code_conversion();

            }
            //MessageBox.Show(richTextBox_text_program.Text);
        }
        //находит индекс парной фигурной скобки
        public int get_index_lvl(int startindex, out int first)
        {

            int count = 0;
            Regex regex = new Regex(@"{|}");
            // Regex regex = new Regex(@"if\s*\([^{}]*\)\s*{[^{}]*}");
            MatchCollection matches =
regex.Matches(richTextBox_text_program.Text.Substring(startindex));
            first = matches[0].Index + startindex;
            int last = startindex;
            foreach (Match match in matches)
            {
                //MessageBox.Show(match.Value);
                if (match.Value == "{") count++;

                else if (match.Value == "}")
                {
                    count--;
                    if (count == 0)
                    {
                        last = match.Index + startindex;
                        break;
```

```
            }
        }
    }
    return last;
}

public int get_index_lvl_s(int startindex, out int first)
{
    int count = 0;
    Regex regex = new Regex(@"\(|\)");
    // Regex regex = new Regex(@"if\s*\([^{}]*\)\s*{[^{}]*}");
    MatchCollection matches =
regex.Matches(richTextBox_text_program.Text.Substring(startindex));
    first = matches[0].Index + startindex;

    int last;
    foreach (Match match in matches)
    {

        if (match.Value == "(") count++;

        else if (match.Value == ")")
        {
            count--;
            if (count == 0)
            {
                last = match.Index + startindex;
                return last;
            }
        }
    }
    return 0;
}

string block_json;
public void get_methods()
{
    int x = 0;
    int count = 0;
    int count_methhods = 0;

    Regex regex = new Regex(@"{|}");
    // Regex regex = new Regex(@"if\s*\([^{}]*\)\s*{[^{}]*}");
    MatchCollection matches = regex.Matches(richTextBox_text_program.Text);
    int first, last;
    if (matches.Count > 3)
     first = matches[2].Index;
    else first = 0;

    foreach (Match match in matches)
    {
```

```
            if (match.Value == "{")
            {
                count++;
                if (count == 3)
                {
                    first = match.Index;

                }
            }
            else if (match.Value == "}")
            {
                count--;
                if (count == 2 && count_methhods<30)
                {
                    last = match.Index;
                    SetSelectionStyle(first + 1, last, FontStyle.Bold, Color.Purple);

                    Panel panel_methode = new Panel();
                    panel_methode.Width = 1000;
                    panel_methode.Location = new Point(x, 0);
                    panel_methode.AutoScroll = false;
                    //MessageBox.Show(richTextBox_text_program.Text.Substring(first + 1, last -
first - 1));
                    Block method = new Block(richTextBox_text_program.Text.Substring(first + 1,
last - first - 1), this, 1, panel_methode, out panel_methode);

                    panel_picture_flowchart_main.Controls.Add(panel_methode);
                    panel_picture_flowchart_main.Invalidate();


                    x += 1010;
                    count_methhods++;

                }else if (count == 2 )
                {

                    Flowchart_main_form newform = new
Flowchart_main_form("{{"+richTextBox_text_program.Text.Substring(first));
                    newform.Show();
                    break;
                }

            }
        }
        if (count_methhods == 0)
        {
            MessageBox.Show("Нет методов. \n Откройте файл заного.");
            richTextBox_text_program.Text = "";
            ToolStripMenuItem_save_into_BD.Visible = false;
        }
        else
        ToolStripMenuItem_save_into_BD.Visible = true;
```

```
    //block_json = JsonSerializer.Serialize(panel_picture_flowchart_main.Controls);
    //MessageBox.Show(block_json.ToString());
}
public string code_name;
private void ToolStripMenuItem_save_into_BD_Click(object sender, EventArgs e)
{
    try
    {
        Form_save_name name = new Form_save_name(this);
        name.ShowDialog();
        using (ApplicationContext db = new ApplicationContext())
        {
            FlowChart flowChart = new FlowChart
            {
                Name = code_name,
                Jsonstring = richTextBox_text_program.Text
            };
            // добавляем их в бд
            db.FlowChart.Add(flowChart);

            db.SaveChanges();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }


}

private void ToolStripMenuItem_open_Click(object sender, EventArgs e)
{
    try
    {
        Form_open_flowchart form = new Form_open_flowchart(this);
        form.ShowDialog();
        panel_picture_flowchart_main.Controls.Clear();
        get_code_conversion();
        get_methods();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

public Block Block
{
    get => default;
    set
    {
```

```
        }
    }

    public Form_save_name Form_save_name
    {
        get => default;
        set
        {
        }
    }

    internal Show_details Show_details
    {
        get => default;
        set
        {
        }
    }

    public Form_open_flowchart Form_open_flowchart
    {
        get => default;
        set
        {
        }
    }
}
```

Приложение №3. Программный код класса Form_open_flowchart

```
public partial class Form_open_flowchart : Form
{
    Flowchart_main_form Form;
    public Form_open_flowchart(Flowchart_main_form form)
    {
        InitializeComponent();
        Form = form;
        using (ApplicationContext db = new ApplicationContext())
        {
            var flowcharts = (from flowchart in db.FlowChart select flowchart).ToList();
            foreach (var flowchart in flowcharts)
            {
                comboBox_names_code.Items.Add(flowchart.Name);
            }
        }
    }

    public ApplicationContext ApplicationContext
    {
        get => default;
        set
        {
```

```
        }
    }

    private void button_open_Click(object sender, EventArgs e)
    {
        if (comboBox_names_code.Text != "")
        {
            using (ApplicationContext db = new ApplicationContext())
            {
                var flowcharts = (from flowchart in db.FlowChart where flowchart.Name == comboBox_names_code.Text select flowchart).ToList();

                foreach (var flowchart in flowcharts)
                {
                    Form.richTextBox_text_program.Text = flowchart.Jsonstring;
                }
                Close();
            }
        }
        else MessageBox.Show("Выберите название.");
    }
}
```

Приложение №4. Программный код класса Form_save_name.

```
public partial class Form_save_name : Form
{
    Flowchart_main_form flowchart_main_Form;
    public Form_save_name(Flowchart_main_form flowchart_Main_Form)
    {
        InitializeComponent();
        flowchart_main_Form = flowchart_Main_Form;
    }

    public ApplicationContext ApplicationContext
    {
        get => default;
        set
        {
        }
    }

    private void button_save_Click(object sender, EventArgs e)
    {
        if (textBox_name_code.Text != "")
        {
            bool name_ok=true;
            using (ApplicationContext db = new ApplicationContext())
            {
                var flowcharts =  (from flowchart in db.FlowChart select flowchart).ToList();

                foreach (var flowchart in flowcharts)
                {
```

```
                if (textBox_name_code.Text== flowchart.Name)
                {
                    MessageBox.Show("Такое название уже существует.\nВведите название заново.");
                    textBox_name_code.Text = "";
                    name_ok= false;
                    break;
                }
            }
        }
        if (name_ok)
        {
            flowchart_main_Form.code_name = textBox_name_code.Text;
            Close();
        }
    }
    else MessageBox.Show("Введите название.");

  }
}
```