ВВЕДЕНИЕ

Разработка мини-САПР для компоновки РЭС последовательным методом может применяться:

1. В организациях радиоэлектронной промышленности, занимающихся проектированием РЭС;
2. В учреждениях образования различного уровня в которых осуществляется подготовка специалистов в области проектирования и производства РЭС.

1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

* 1. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

Основанием для разработки курсового проекта является задание для курсового проектирования по дисциплине «Системы автоматизированного проектирования», утверждённого председателем цикловой комиссии «Информатики и программирования» Тыбербай С.Г. дата «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2015г., протокол №\_\_\_\_\_\_\_\_\_\_\_\_. Разработка ведётся по теме: «Разработка мини-САПР для компоновки радиоэлектронных средств последовательным методом».

1.2 НАЗНАЧЕНИЕ РАЗРАБОТКИ

Программное изделие предназначено для автоматизации процесса компоновки РЭС с использованием математического моделирования, в основу которой положен метод формирования локальных максимумов разрезания графа, где G(X,U), где X - множество радиоэлектронных компонентов, входящих в состав РЭС, U- связи между радиоэлектронными компонентами в соответствии с учётом выполняемой развязки узлов в подграфах электрических соединений.

1.3 ТРЕБОВАНИЕ К ПРОГРАММЕ ИЛИ ПРОГРАММНОМУ ПРОДУКТУ

1.3.1 Требования к функциональным характеристикам

Программное изделие должно осуществлять выполнение следующих функций:

1. Графический ввод вершин графа с возможностью автоматизации присвоения имён, соответствующих оппозиционным обозначениям радиоэлектронных компонентов;
2. Ввод соединений между вершинами графа;
3. Автоматическое формирование матрицы смежности или матрицы инцидентности между графическими изображениями графа и матрицей должна осуществляться после динамическая связь: изменение в рисунке, изменение вершин.
4. Должно автоматически изменять количество строк и столбцов матрицы смежности и наоборот. Изменение в матрице смежности должны приводить к изменению конфигурации рисунка.
5. При формировании матрицы смежности предусмотреть возможность автоматически задавать rji , rij;
6. Предусмотреть возможность контроля правильности вводимых данных.
   * 1. Требования к надёжности
     2. Высокое качество функционирования ПОСАПР;
     3. Получение проектируемых решений в форме, принятой в соответствующей организации;
     4. Удобство сведения и сопровождения ПОСАПР;
     5. Полноту диагностических сообщений, ориентированных на пользователей САПР;
     6. Защищённость программ и данных от несанкционированного доступа;
     7. Полноту документации.

1.3.3 Условия эксплуатации

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

1.3.4 Требования к составу и параметрам технических средств

В состав технических средств должен входить IВМ-совместимый персональный

компьютер (ПЭВМ), включающий в себя:

1. 32-разрядный или 64-разрядный процессор с тактовой частотой 1 ГГц или выше;

2. 1 ГБ оперативной памяти;

3. HDD с размером свободного места не менее 100 МБ;

4. Графическое устройство поддерживающее OpenGL 2.1.0.12.

1.3.5 Требования к информационной и программной совместимости

Системные программные средства, используемые программой, должны быть представлены локализованной версией операционной системы Windows и соответствующими драйверами оборудования ( принтера, сканера, мыши и клавиатуры). Также на компьютере должен быть установлен Microsoft .NET Framework версии 4.0 (или выше).

1.3.6 Требования к упаковке, маркировки и хранению продукта

Диски с работающими экземплярами программы должны иметь маркировку, состоящую из надписи «рабочая», даты последней перезаписи программы. При транспортировании и хранении программного изделия должна быть предусмотрена защита от попадания пыли и атмосферных осадков.

1.4 ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

Состав программной документации должен включать в себя:

1. Титульный лист ( не нумеруется, но входит в общее число страниц);
2. Лист утверждения (не нумеруется и не входит в общее число страниц);
3. Лист задания;
4. Содержание (перечень разделов);
5. Текст документа, который может сопровождаться рисунками, таблицами;
6. Литература;
7. Приложения.

1.5 СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Стадии разработки программного обеспечения:

* 1. техническое задание;
  2. эскизный проект;
  3. технический проект;
  4. рабочий проект;
  5. внедрение.

На стадии технического задания выполняются следующие виды работ:

1. постановка задачи;
2. сбор исходных материалов;
3. выбор и обоснование критериев эффективности и качества разрабатываемой программы;
4. обоснование необходимости проведения научно-исследовательских работ, целесообразности применения ранее разрабатываемых программ принципиальной возможности применения задачи;
5. предварительный выбор метода решения задачи;
6. определение требований к техническим средствам;
7. определение требований, целей разработки программ, стадий, этапов и сроков разработки программы и документации на неё;
8. технико-экономическое обоснование разработки программы;
9. согласование и утверждение технического задания.

На стадии эскизного проекта выполняются следующие виды работ::

1. внешнее проектирования программного изделия;
2. уточнение методов решения задачи;
3. предварительные проектирования внутренних страниц данных;
4. разработка общего алгоритма решения задачи и укрупнённой структурной схемы программного изделия, а также пояснительной записки.

При разработке эскизного проекта определяется:

1. функции пользователей;
2. тип языка взаимодействия;
3. синтаксис и семантика языка взаимодействия;
4. структура и содержание информационных кадров и шаблонов диалога;
5. структура входных и выходных данных.

На стадии технического проекта выполняются следующие этапы:

1. проектирование архитектуры ПИ;
2. проектирование структуры данных;
3. проектирование модульной структуры изделия;
4. проектирование модулей;
5. разработка пояснительной записки.

На стадии рабочего проекта выполняют:

1. кодирование, тестирование и отладку программ;
2. разработку программных документов в соответствии с требованиями ЕСПД;
3. проведение различных видов приёмосдаточных испытаний;
4. корректировка программ и документации по результатам испытаний.

На стадии внедрения осуществляется подготовка и передача программ и программной документации для сопряжения и изготовления, а также передача программ в фонд алгоритмов.

1.6 ПОРЯДОК КОНТРОЛЯ И ПРИЁМКИ

Проверка документации программы осуществляется самим заказчиком с привлечением сторонних экспертов, способных засвидетельствовать факт соответствия созданного программного продукта всем пунктам технической документации, включая техническое задание и технический проект. Испытания и тестирование программы должны проводиться в процессе создания программы самим разработчиком:

* 1. C использованием контрольных тестов, позволяющих добиться проверки правильности работоспособности и взаимной совместимости максимального числа функций и операторов программы или модуля при минимальных затратах временных и финансовых ресурсов.
  2. Путем пошагового исполнения программы или модуля в соответствии с набором тестовых примеров и сравнения полученных в процессе тестирования значений с контрольными значениями тестовых примеров.
  3. С привлечением сторонних неофициальных бета-тестеров, которые в процессе тестирования программного продукта должны сообщать разработчику все найденные ошибки и неточности в работе программы и проверка документации должна проводиться.

Испытания и тестирование программы должны проводиться после завершения создания программы заказчиком:

1. С использованием проверочных тестов, составляемых заказчиком заблаговременно.
2. В процессе начального этапа внедрения программы, путем тестирования программы в организации заказчика в рабочих условиях на протяжении срока в 1 неделю.
3. В организации заказчика с привлечением сторонних экспертов.

Приёмка программы должна осуществляться заказчиком. Программа должна считаться годной, если она удовлетворяет всем пунктам данного технического задания, что должно быть засвидетельствовано сторонними экспертами.

2. ПОСТАНОВКА ЗАДАЧИ

2.1 ХАРАКТЕРИСТИКА ЗАДАЧИ

Программа «Мини-САПР для компоновки радиоэлектронных средств последовательным методом» предназначена для рабочих и учебных целей.

Данный комплекс задач состоит из ряда подзадач:

* 1. Возможность графического ввода вершин графа с возможностью автоматизации присвоения имён, соответствующих оппозиционным обозначениям радиоэлектронных компонентов;
  2. Возможность ввода соединений между вершинами графа;
  3. Автоматическое формирование матрицы смежности или матрицы инцидентности между графическими изображениями графа и матрицей;
  4. Автоматическое изменение количества строк и столбцов матрицы смежности;

Цель разработки задачи разработки мини-САПР для компоновки радиоэлектронных средств последовательным методом – ускорить и улучшить автоматизацию процесса компоновки РЭС с использованием математического моделирования, в основу которой положен последовательный метод разрезания графа, где G(X,U), где X- множество радиоэлектронных компонентов, входящих в состав РЭС, U- связи между радиоэлектронными компонентами в соответствии с учётом выполняемой развязки узлов в подграфах электрических соединений.

Автоматизированное решение задачи имеет следующие преимущества:

* 1. Быстрота, ожидается многократное увеличение скорости решения задач, ускорение и улучшение процесса работы;
  2. Надёжность, автоматизированное решение задачи практически исключает появления ошибок, а если они и появляются, то имеются средства для их исправления.

2.2 ВХОДНЫЕ ДАННЫЕ

В качестве исходных данных для решения задачи размещения рассматриваются схема соединения элементов, заданная в виде графа G(X,U), где X- множество радиоэлектронных компонентов, входящих в состав РЭС, U- связи между радиоэлектронными компонентами в соответствии с учётом выполняемой развязки узлов в подграфах электрических соединений.

2.3 ВЫХОДНЫЕ ДАННЫЕ

Результатами работы программы «Разработка мини-САПР для компоновки радиоэлектронных средств последовательным методом являются значения: количество внутренних связей в графе, количество внешних связей в графе, коэффициент разрезания графа, имена вершин в кусках.

2.4 ВЫБОР ЯЗЫКА ПРОГРАММИРОВАНИЯ

Для написания программы использовался язык программирования C# и среда разработки Microsoft Visual Studio.

2.5 МЕТОДЫ РЕШЕНИЯ ЗАДАЧИ

2.5.1 Методы поиска информации, применяемые для разрабатываемого программного документа

Изучил алгоритм разрезания графа последовательным методом.

2.5.2 Описание методики решения задачи

Разрезание графа последовательным методом осуществляется по алгоритму.

3. ПРОТОКОЛ ИСПЫТАНИЙ ПРОГРАММНОГО ПРОДУКТА

Комиссия в составе:

председатель цикловой комиссии – Тыбербай С. Г.;

руководитель курсового проекта – Шандриков А.С.;

преподаватель спецдисциплины – Шандриков А.С.;

произвела проверку работоспособности программного продукта, разработанного в порядке выполнения курсового проекта.

|  |  |
| --- | --- |
| **1. Информация о программном продукте** | |
| * 1. Название программного продукта | «Разработка мини-САПР для компоновки радиоэлектронных средств последовательным методом». |
| * 1. Номер версии программного продукта | 1.0 |
| * 1. Дата разработки версии программного продукта | 20.02.2016 |
| * 1. Назначение программного продукта | Программный продукт предназначен для ускорения и улучшения автоматизации процесса компоновки РЭС с использованием математического моделирования. |
| * 1. Функции программного продукта | Программный продукт позволяет графически ввести вершины графа с возможностью автоматизации присвоения имён, соответствующих оппозиционным обозначениям радиоэлектронных компонентов; ввести соединения между вершинами графа; автоматически сформировать матрицу смежности между графическими изображениями графа и матрицей; разрезать граф на куски; рассчитать количество внешних и внутренних связей, а также коэффициент разрезания графа. |
| * 1. Язык программирования | С# |
| * 1. Наличие программы инсталляции | Да |
| * 1. Тип компьютера(ов) |  |
| * 1. Требования к компьютеру   - Оперативная память (ОЗУ)  - Жесткий диск(и)  - Монитор(ы) | 1024 Mb и выше  100 Mb  разрешение 1280х720 и выше |

|  |  |
| --- | --- |
| 1.10. Дополнительное периферийное оборудование: принтеры, мышь, модем и т. п. |  |
| 1.11. Тип операционной системы (систем) | Windows XP и выше |
| 1. **Информация об организации-заказчике** | |
| * 1. Полное официальное наименование | Витебский государственный политехнический колледж |
| * 1. Место нахождения (почтовый адрес) | ул. Черняховского д.14 |
| * 1. Номер контактного телефона(ов) | (8-0212)58-30-91 |
| * 1. Номер факса | (8-0212)58-30-91 |
| * 1. Адрес электронной почты | vgpt@vitebsk.by |
| 1. **Информация о разработчике программного продукта** | |
| * 1. Полное официальное наименование | Витебский государственный политехнический колледж |
| * 1. Место нахождения (почтовый адрес) | Ул. Мира, д.17, кв. 22 |
| * 1. Фамилия, инициалы учащегося | Колебянов В.Р. |
| * 1. Номер контактного телефона(ов) | +375259516911 |
| * 1. Номер факса | (8-0212)29-41-13 |
| * 1. Адрес электронной почты | kolebynov@mail.ru |

Продолжение таблицы

Работоспособность программного продукта проверена на контрольных примерах, предусматривающих все возможные варианты исходных и промежуточных данных.

Программный продукт признан работоспособным, соответствующим техническому заданию и позволяет решать поставленные перед ним производственные задачи.

Председатель цикловой комиссии: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Тыбербай С. Г.

Руководитель проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шандриков А.С.

Преподаватель спецдисциплины: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шандриков А.С.

ЗАКЛЮЧЕНИЕ

В ходе написания курсового проекта улучшил знания языка программирования С# и среды разработки Microsoft Visual Studio. Этот язык можно отметить высокой производительностью, надёжностью и практичностью.

«Разработка мини-САПР для компоновки радиоэлектронных средств последовательным методом» соответствует исходному техническому заданию. На основе выполненных тестов можно сделать вывод, что программа отвечает эксплуатационным, технологическим требованиям.

Разработанный программный продукт с включенными в него функциональными возможностями позволяет ускорить и улучшить автоматизацию процесса компоновки РЭС с использованием математического моделирования.

Возможные варианты доработки в следующих версиях:

- добавление графических возможностей, а именно: изменение цвета фона, вершин и связей между ними;

- изменение положения вершин на экране;

- вывод подробного описание решения данного алгоритма.

ЛИТЕРАТУРА

1. Шандриков А.С. Разрезание графа формированием локальных максимумов [Текст] / А.С.Шандриков // Современные проблемы математики: материалы IV Республиканской научной конференции / Бресткий государственный технический университет. – Брест,2005, — С, 88-92.

ПРИЛОЖЕНИЕ А

БЛОК-СХЕММА АЛГОРИТМА РАБОТЫ ПРОГРАММЫ

Рисование графа

Выполнение алгоритма

Расчет параметров

Вывод отчета

ПРИЛОЖЕНИЕ В

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

1. НАЗНАЧЕНИЕ И УСЛОВИЕ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Данная программа позволяет ускорить и улучшить автоматизацию процесса компоновки РЭС с использованием математического моделирования, в основу которой положен последовательным методом разрезания графа, где G(X,U), где X- множество радиоэлектронных компонентов, входящих в состав РЭС, U- связи между радиоэлектронными компонентами в соответствии с учётом выполняемой развязки узлов в подграфах электрических соединений.

В состав технических средств должен входить IВМ-совместимый персональный

компьютер (ПЭВМ), включающий в себя:

1. 32-разрядный или 64-разрядный процессор с тактовой частотой 1 ГГц или выше;

2. 1 ГБ оперативной памяти;

3. HDD с размером свободного места не менее 100 МБ;

4. Графическое устройство поддерживающее OpenGL 2.1.0.12.

2. ВЫПОЛНЕНИЕ ПРОГРАММЫ

Для того, чтобы запустить выполнение программы достаточно открыть файл SAPR\_kyrs.exe. После его открытия откроется программа (рисунок 1):

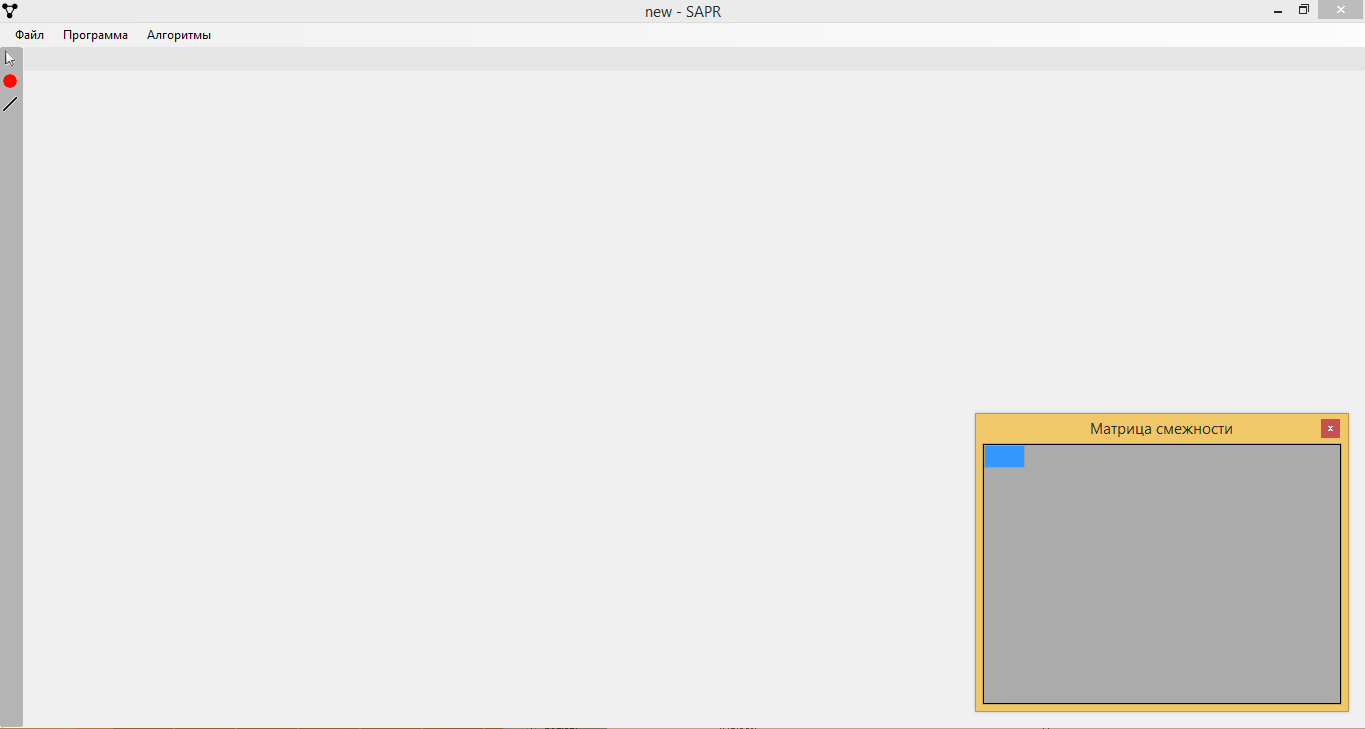


Рисунок 1. Открытие программы.

После того как программа открылось нам необходимо нарисовать граф, а для этого следует воспользоваться панелью инструментов, представленной на рисунке 2:



Рисунок 2. Панель инструментов.

Когда мы нарисовали граф, с помощью панели инструментов, то нужно выбрать вкладку «Алгоритмы» и выбрать последовательный алгоритм, откроется окно для указания параметров разрезания графа( рисунок 3):

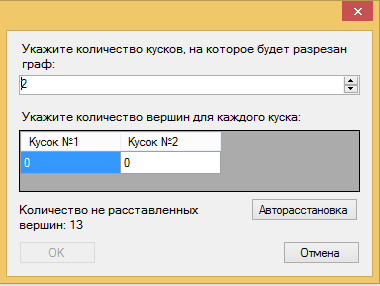


Рисунок 3. Параметры разрезания графа.

При нажатии на кнопку «ОК» производится разрезание графа, расчет внешних, внутренних связей и коэффициента разрезания графа, после чего выводится отчет о разрезании (рисунок 4):

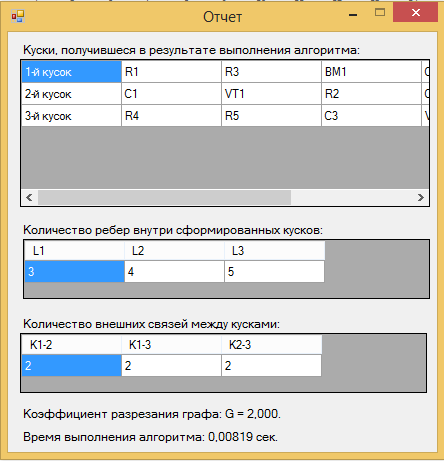


Рисунок 4. Отчет о разрезании.

Также в программе есть меню, в котором можно создать новый файл, открыть старый граф, сохранить граф или сохранить как, выйти из программы (рисунок 5), а также открыть настройки или открыть/скрыть матрицу смежности (рисунок 6):

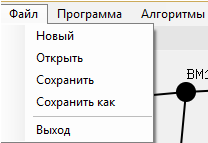
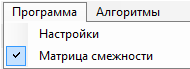
 

Рисунок 5 и 6. Меню программы

ПРИЛОЖЕНИЕ C

ТЕКСТ ПРОГРАММЫ

**AlgorithmInvoker.cs**

using System;

using System.Collections.Generic;

using System.Reflection;

using System.IO;

using System.Runtime.CompilerServices;

namespace SAPR.Classes

{

public class AlgorithmInvoker

{

public List<Algorithm> Algorithms { get { return m\_algorithms; } }

public AlgorithmInvoker(string path)

{

m\_path = path;

m\_algorithms = new List<Algorithm>();

IEnumerable<string> files = null;

try

{

files = Directory.EnumerateFiles(m\_path);

}

catch (DirectoryNotFoundException)

{

return;

}

List<Assembly> testAssemblies = new List<Assembly>();

foreach (string fileName in files)

{

try

{

string fullPath = fileName[1] != ':' ?

Environment.CurrentDirectory + "\\" + fileName :

fileName;

Assembly testAssembly = Assembly.LoadFile(fullPath);

testAssemblies.Add(testAssembly);

}

catch (BadImageFormatException)

{

}

}

foreach (Assembly assembly in testAssemblies)

{

string name = null;

MethodInfo calculateMethod = null;

Type[] types = assembly.GetTypes();

foreach (Type type in types)

{

MethodInfo[] methods = type.GetMethods(BindingFlags.Static | BindingFlags.Public);

bool isCalcMethod = false, isGetNameMethod = false;

foreach (MethodInfo method in methods)

{

if (method.ToString() == m\_methodCalculateName)

{

calculateMethod = method;

if (isGetNameMethod)

break;

isCalcMethod = true;

continue;

}

if (method.ToString() == m\_methodGetName)

{

name = (string)method.Invoke(null, null);

if (isCalcMethod)

break;

isGetNameMethod = true;

}

}

if (name != null && calculateMethod != null)

break;

}

if (name != null && calculateMethod != null)

m\_algorithms.Add(new Algorithm(name, calculateMethod));

}

}

private string m\_path;

private List<Algorithm> m\_algorithms;

private static string m\_methodCalculateName = "Int32[][] Calculate(Int32[,], Int32, Int32[])";

private static string m\_methodGetName = "System.String GetName()";

public class Algorithm

{

public string Name { get { return m\_name; } }

public Algorithm(string name, MethodInfo calculateMethod)

{

m\_name = name;

m\_calculateMethod = calculateMethod;

}

public int[][] Calculate(int[,] adjMatrix, int countPart,

int[] countVertexInParts)

{

return (int[][])m\_calculateMethod.Invoke(null, new object[] {

adjMatrix, countPart, countVertexInParts });

}

private string m\_name;

private MethodInfo m\_calculateMethod;

}

}

}

**ButtonColor.cs**

using System.Windows.Forms;

using System.Drawing;

using System;

namespace SAPR.Classes

{

public partial class ButtonColor : Button

{

public event EventHandler ColorChanged;

public Color Color

{

get { return m\_color; }

set

{

if (m\_color != value)

{

Bitmap bitmap = Image as Bitmap;

for (int y = 0; y < bitmap.Height; y++)

for (int x = 0; x < bitmap.Width; x++)

bitmap.SetPixel(x, y, value);

m\_color = value;

}

}

}

public ButtonColor()

{

Click += (sender, e) =>

{

m\_colorDialog.Color = Color;

m\_colorDialog.ShowDialog();

Color = m\_colorDialog.Color;

if (ColorChanged != null)

ColorChanged(this, new EventArgs());

};

Image = new Bitmap(Size.Width, Size.Height);

Color = m\_colorDialog.Color;

Resize += (sender, e) =>

{

Image.Dispose();

Image = null;

Image = new Bitmap(Size.Width, Size.Height);

};

}

private Color m\_color;

private static ColorDialog m\_colorDialog = new ColorDialog();

}

}

**ExtensionMethods.cs**

using System.Drawing;

using System;

using System.Runtime.CompilerServices;

namespace SAPR.Classes

{

public static class ExtensionMethods

{

public static Point AddByPoint(this Point srcPoint, Point point)

{

return new Point(srcPoint.X + point.X, srcPoint.Y + point.Y);

}

public static Point SubByPoint(this Point srcPoint, Point point)

{

return new Point(srcPoint.X - point.X, srcPoint.Y - point.Y);

}

public static Point DivByNumber(this Point srcPoint, int devider)

{

return new Point(srcPoint.X / devider, srcPoint.Y / devider);

}

public static float GetLength(this Point point)

{

return (float)Math.Sqrt(point.X \* point.X + point.Y \* point.Y);

}

public static int Dot(this Point point1, Point point2)

{

return point1.X \* point2.X + point1.Y \* point2.Y;

}

public static float GetLength(this PointF point)

{

if (point.X < 0)

point.X = Math.Abs(point.X);

if (point.Y < 0)

point.Y = Math.Abs(point.Y);

return (float)Math.Sqrt(point.X \* point.X + point.Y \* point.Y);

}

public static float Dot(this PointF point1, PointF point2)

{

float temp = point1.X \* point2.X + point1.Y \* point2.Y;

if (temp > 1f)

temp = 1f;

return temp;

}

public static PointF Normalize(this PointF point)

{

float length = point.GetLength();

point.X /= length;

point.Y /= length;

return point;

}

public static PointF AddByPoint(this PointF srcPoint, PointF point)

{

return new PointF(srcPoint.X + point.X, srcPoint.Y + point.Y);

}

public static PointF SubByPoint(this PointF srcPoint, PointF point)

{

return new PointF(srcPoint.X - point.X, srcPoint.Y - point.Y);

}

public static Point MaxPoint(Point point1, Point point2)

{

return new Point(point1.X > point2.X ? point1.X : point2.X,

point1.Y > point2.Y ? point1.Y : point2.Y);

}

public static Point MinPoint(Point point1, Point point2)

{

return new Point(point1.X < point2.X ? point1.X : point2.X,

point1.Y < point2.Y ? point1.Y : point2.Y);

}

public static Point MaxPoint(Point[] points)

{

Point temp = new Point(int.MinValue, int.MinValue);

foreach (Point point in points)

{

if (point.X > temp.X)

temp.X = point.X;

if (point.Y > temp.Y)

temp.Y = point.Y;

}

return temp;

}

public static Point MinPoint(Point[] points)

{

Point temp = new Point(int.MaxValue, int.MaxValue);

foreach (Point point in points)

{

if (point.X < temp.X)

temp.X = point.X;

if (point.Y < temp.Y)

temp.Y = point.Y;

}

return temp;

}

}

}

**SettingsObject.cs**

using System.Collections.Generic;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

using System.Runtime.Serialization;

using System;

namespace SAPR.Classes

{

public class SettingsObject

{

public event Action<object, KeyValuePair<string, object>> ValueChanged;

public event EventHandler SettingsSaved;

public string FileName { get; private set; }

public object this[string key]

{

get

{

if (key == null)

return null;

object temp;

if (m\_settingsValues.TryGetValue(key, out temp))

return temp;

else

return null;

}

set

{

object temp;

bool isOk = m\_settingsValues.TryGetValue(key, out temp);

if (isOk && !temp.Equals(value))

{

m\_settingsValues[key] = value;

if (ValueChanged != null)

ValueChanged(this, new KeyValuePair<string, object>(

key, value));

}

}

}

public SettingsObject(string fileName)

{

FileName = fileName;

}

public void AddSetting(string key, object value)

{

m\_settingsValues.Add(key, value);

}

public void RemoveSetting(string key)

{

m\_settingsValues.Remove(key);

}

public void Save()

{

FileStream fileSave = null;

try

{

fileSave = File.Create(FileName);

BinaryFormatter formatter = new BinaryFormatter();

formatter.Serialize(fileSave, m\_settingsValues);

}

catch (Exception)

{

throw new CantSaveSettingsException();

}

finally

{

if (fileSave != null)

fileSave.Close();

}

if (SettingsSaved != null)

SettingsSaved(this, new EventArgs());

}

public IEnumerator<KeyValuePair<string, object>> GetEnumerator()

{

foreach (var pair in m\_settingsValues)

yield return pair;

}

public static SettingsObject LoadFromFile(string fileName)

{

FileStream file = null;

try

{

file = File.Open(fileName, FileMode.Open);

BinaryFormatter formatter = new BinaryFormatter();

Dictionary<string, object> values

= (Dictionary<string, object>)formatter.Deserialize(file);

return new SettingsObject(values, fileName);

}

catch (IOException)

{

throw new NotFoundSettingFileException();

}

catch (SerializationException)

{

throw new BadSettingFileException();

}

catch (InvalidCastException)

{

throw new BadSettingFileException();

}

finally

{

if (file != null)

file.Close();

}

}

private Dictionary<string, object> m\_settingsValues = new Dictionary<string, object>();

private SettingsObject(Dictionary<string, object> values, string fileName)

{

m\_settingsValues = values;

FileName = fileName;

}

}

public class NotFoundSettingFileException : ApplicationException

{ }

public class BadSettingFileException : ApplicationException

{ }

public class CantSaveSettingsException : ApplicationException

{ }

}

**Vertex.cs**

using System;

using System.Drawing;

using System.Windows.Forms;

using System.Collections.Generic;

namespace SAPR.Classes

{

[Serializable()]

public class Vertex : Element

{

public Color TextColor { get; set; } = Color.Black;

public List<Edge> Edges { get { return m\_edges; } }

public Point Position

{

get { return m\_position; }

set

{

m\_position = value;

foreach (Edge edge in m\_edges)

if (AreaElement.Contains(edge.StartPos))

edge.StartPos = m\_position;

else

edge.EndPos = m\_position;

areaElement.Location = new Point(value.X - Radius, value.Y - Radius);

}

}

public override Rectangle AreaElement

{

get

{

if (areaElement.Height / 2 == m\_radius)

return base.AreaElement;

areaElement.X = m\_position.X - m\_radius;

areaElement.Y = m\_position.Y - m\_radius;

areaElement.Width = m\_radius \* 2;

areaElement.Height = m\_radius \* 2;

return base.AreaElement;

}

}

public static int Radius

{

get { return m\_radius; }

set

{

if (value == m\_radius)

return;

m\_radius = value;

}

}

public Vertex(string text, Point position, Color colorSolid)

{

color = colorSolid;

ColorSolid = colorSolid;

Name = text;

m\_text = text;

m\_position = position;

m\_edges = new List<Edge>();

Size size = new Size(Radius \* 2, Radius \* 2);

areaElement = new Rectangle(new Point(position.X - Radius, position.Y - Radius), size);

}

public Vertex(string text, Point position) : this(text, position, Color.Black)

{ }

public Vertex(string text) : this(text, new Point(0, 0))

{ }

public Vertex() : this("")

{ }

public void AddEdge(Edge edge)

{

m\_edges.Add(edge);

}

public override void Dispose()

{

m\_edges.Clear();

m\_text = null;

m\_edges = null;

base.Dispose();

}

private string m\_text;

private Point m\_position;

private List<Edge> m\_edges;

private static int m\_radius = 10;

}

}

**ToolStripButtonColor.cs**

using System;

using System.Windows.Forms;

using System.Drawing;

namespace SAPR.Classes

{

public class ToolStripButtonColor : ToolStripButton

{

public Color Color

{

get { return m\_color; }

set

{

if (m\_color != value)

{

if (Image.Size != Size)

{

Image.Dispose();

Image = new Bitmap(Size.Width, Size.Height);

}

Bitmap bitmap = Image as Bitmap;

for (int y = 0; y < bitmap.Height; y++)

for (int x = 0; x < bitmap.Width; x++)

bitmap.SetPixel(x, y, value);

m\_color = value;

}

}

}

public ToolStripButtonColor()

{

Click += (sender, e) =>

{

m\_colorDialog.Color = Color;

m\_colorDialog.ShowDialog();

Color = m\_colorDialog.Color;

};

Image = new Bitmap(Size.Width, Size.Height);

Color = m\_colorDialog.Color;

}

private Color m\_color;

private static ColorDialog m\_colorDialog = new ColorDialog();

}

}

**CountPartForm.cs**

using System;

using System.Windows.Forms;

using SAPR.Classes;

using System.Diagnostics;

using SAPR.Resources;

using System.Collections.Generic;

using System.Linq;

namespace SAPR.Forms

{

public partial class CountPartForm : Form

{

public CountPartForm(AlgorithmInvoker.Algorithm alg, AdjMatrixGrid adjMatrixGrid,

List<Vertex> vertices)

{

InitializeComponent();

m\_alg = alg;

m\_adjMatrixGrid = adjMatrixGrid;

m\_vertices = vertices;

m\_gridVertexCount.RowCount = 1;

numericUpDown1.Maximum = m\_vertices.Count < 30 ? m\_vertices.Count : 30L;

numericUpDown1\_ValueChanged(this, new EventArgs());

m\_FreeVertex = vertices.Count;

}

private AlgorithmInvoker.Algorithm m\_alg;

private AdjMatrixGrid m\_adjMatrixGrid;

private List<Vertex> m\_vertices;

private int m\_freeVertex = -1;

private bool m\_isFilling = false;

private int m\_FreeVertex

{

get { return m\_freeVertex; }

set

{

if (value == m\_freeVertex)

return;

if (value == 0)

m\_buttonOk.Enabled = true;

else

m\_buttonOk.Enabled = false;

m\_labelFreeVertex.Text = value.ToString();

m\_freeVertex = value;

}

}

private void m\_buttonOk\_Click(object sender, EventArgs e)

{

Close();

Stopwatch timer = new Stopwatch();

int countPart = (int)numericUpDown1.Value;

int[] countVertexInParts = new int[countPart];

for (int i = 0; i < countPart; i++)

countVertexInParts[i] = int.Parse(

(string)m\_gridVertexCount[i, 0].Value);

timer.Start();

int[][] result = m\_alg.Calculate(m\_adjMatrixGrid.Matrix.Values, countPart,

countVertexInParts);

timer.Stop();

string[] vertexNames = m\_vertices.Select(vertex => vertex.Name).ToArray();

ReportData data = new ReportData(vertexNames, result,

timer.Elapsed.TotalSeconds, m\_adjMatrixGrid.Matrix.Values);

ReportForm reportForm = new ReportForm(data);

reportForm.Show();

}

private void m\_buttonCancel\_Click(object sender, EventArgs e)

{

Close();

}

private void numericUpDown1\_ValueChanged(object sender, EventArgs e)

{

m\_gridVertexCount.ColumnCount = (int)numericUpDown1.Value;

for (int i = 0; i < m\_gridVertexCount.ColumnCount; i++)

if (m\_gridVertexCount.Columns[i].Name == string.Empty)

{

m\_isFilling = true;

m\_gridVertexCount.Columns[i].Name = "Кусок №" + (i + 1).ToString();

m\_gridVertexCount[i, 0].Value = "0";

m\_isFilling = false;

}

}

private void m\_gridVertexCount\_CellValidating(object sender, DataGridViewCellValidatingEventArgs e)

{

string oldValue = (string)m\_gridVertexCount[e.ColumnIndex, e.RowIndex].Value;

if ((string)e.FormattedValue == oldValue)

return;

int newNumber;

if (!int.TryParse((string)e.FormattedValue, out newNumber) ||

newNumber < 0)

{

e.Cancel = true;

MessageBox.Show(AppResources.badNumber, AppResources.errorText,

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

private void m\_gridVertexCount\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

if (m\_isFilling)

return;

m\_CalculateFreeVertex();

}

private void m\_gridVertexCount\_ColumnRemoved(object sender, DataGridViewColumnEventArgs e)

{

m\_CalculateFreeVertex();

}

private void button1\_Click(object sender, EventArgs e)

{

m\_isFilling = true;

int countPart = (int)numericUpDown1.Value;

int count = m\_vertices.Count / countPart;

int remain = m\_vertices.Count % countPart;

for (int i = 0; i < countPart; i++)

m\_gridVertexCount[i, 0].Value = (count +

(countPart - i <= remain ? 1 : 0)).ToString();

m\_isFilling = false;

m\_FreeVertex = 0;

}

private void m\_CalculateFreeVertex()

{

int sum = 0;

for (int i = 0; i < m\_gridVertexCount.ColumnCount; i++)

sum += int.Parse((string)m\_gridVertexCount[i, 0].Value);

m\_FreeVertex = m\_vertices.Count - sum;

}

}

}

**MatrixForm.cs**

using SAPR.Classes;

using System.Windows.Forms;

using System.Drawing;

using System;

namespace SAPR.Forms

{

public partial class MatrixForm : Form

{

public AdjMatrixGrid AdjMatrix

{

get { return m\_adjMatrixGrid; }

set

{

Controls.Remove(m\_adjMatrixGrid);

m\_adjMatrixGrid = value;

m\_ResetAdjMatrix(m\_adjMatrixGrid);

Controls.Add(m\_adjMatrixGrid);

}

}

public MatrixForm(AdjMatrixGrid adjMatrixGrid)

{

InitializeComponent();

m\_adjMatrixGrid = adjMatrixGrid;

m\_ResetAdjMatrix(m\_adjMatrixGrid);

Controls.Add(m\_adjMatrixGrid);

}

private AdjMatrixGrid m\_adjMatrixGrid;

private static readonly int COLUMN\_WIDTH = 40;

private void MatrixForm\_Resize(object sender, EventArgs e)

{

m\_adjMatrixGrid.Height = ClientSize.Height;

m\_adjMatrixGrid.Width = ClientSize.Width;

}

private void m\_ResetAdjMatrix(AdjMatrixGrid adjMatrix)

{

adjMatrix.Location = new Point(0, 0);

adjMatrix.Size = new Size(ClientSize.Width, ClientSize.Height);

foreach (DataGridViewColumn column in adjMatrix.Columns)

column.Width = COLUMN\_WIDTH;

}

}

}

**ReportForm.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace SAPR.Forms

{

public partial class ReportForm : Form

{

public ReportForm()

{

InitializeComponent();

}

public ReportForm(ReportData data) : this()

{

int max = data.parts[0].Length;

for (int i = 1; i < data.parts.Length; i++)

if (data.parts[i].Length > max)

max = data.parts[i].Length;

m\_gridPart.ColumnCount = max + 1;

m\_gridPart.RowCount = data.parts.Length;

for (int i = 0; i < data.parts.Length; i++)

{

m\_gridPart[0, i].Value = (i + 1).ToString() + "-й кусок";

for (int j = 0; j < data.parts[i].Length; j++)

m\_gridPart[j + 1, i].Value = data.vertexNames[data.parts[i][j]];

}

int[] innerEdges = m\_CalculateInnerEdges(data.parts, data.adjMatrix);

m\_innerEdgesGrid.ColumnCount = innerEdges.Length;

m\_innerEdgesGrid.RowCount = 1;

for (int i = 0; i < innerEdges.Length; i++)

{

m\_innerEdgesGrid.Columns[i].Name = "L" + (i + 1).ToString();

m\_innerEdgesGrid[i, 0].Value = innerEdges[i];

}

int[][] outerEdges = m\_CalculateOuterEdges(data.parts, data.adjMatrix);

int countPart = data.parts.Length;

m\_outerEdgesGrid.ColumnCount = (countPart \* countPart - countPart) / 2;

m\_outerEdgesGrid.RowCount = 1;

int columnNo = 0;

for (int i = 0; i < outerEdges.Length; i++)

for (int j = 0; j < outerEdges[i].Length; j++, columnNo++)

{

m\_outerEdgesGrid.Columns[columnNo].Name = "K" + (i + 1).ToString() +

"-" + (i + j + 2).ToString();

m\_outerEdgesGrid[columnNo, 0].Value = outerEdges[i][j];

}

m\_labelCutRate.Text += m\_CalculateCatRate(innerEdges, outerEdges).ToString(

"F3") + ".";

m\_labelPerformTime.Text = string.Format("Время выполнения алгоритма: {0:F5} сек.",

data.performTime);

}

private int[] m\_CalculateInnerEdges(int[][] parts, int[,] adjMatrix)

{

int[] temp = new int[parts.Length];

for (int i = 0; i < parts.Length; i++)

{

int inner = 0;

for (int j = 0; j < parts[i].Length; j++)

for (int k = j; k < parts[i].Length; k++)

inner += adjMatrix[parts[i][j], parts[i][k]];

temp[i] = inner;

}

return temp;

}

private int[][] m\_CalculateOuterEdges(int[][] parts, int[,] adjMatrix)

{

int[][] temp = new int[parts.Length - 1][];

for (int i = 0; i < temp.Length; i++)

temp[i] = new int[parts.Length - i - 1];

for (int i = 0; i < parts.Length - 1; i++)

for (int j = i + 1; j < parts.Length; j++)

{

int outer = 0;

for (int k = 0; k < parts[i].Length; k++)

for (int l = 0; l < parts[j].Length; l++)

outer += adjMatrix[parts[i][k], parts[j][l]];

temp[i][j - i - 1] = outer;

}

return temp;

}

private double m\_CalculateCatRate(int[] innerEdges, int[][] outerEdges)

{

int innerSum = innerEdges.Sum();

int outerSum = 0;

foreach (int[] outer in outerEdges)

outerSum += outer.Sum();

return (double)innerSum / outerSum;

}

private void ReportForm\_Resize(object sender, EventArgs e)

{

int newSize = Size.Width - 35;

Size gridSize;

gridSize = m\_gridPart.Size;

gridSize.Width = newSize;

m\_gridPart.Size = gridSize;

gridSize = m\_innerEdgesGrid.Size;

gridSize.Width = newSize;

m\_innerEdgesGrid.Size = gridSize;

gridSize = m\_outerEdgesGrid.Size;

gridSize.Width = newSize;

m\_outerEdgesGrid.Size = gridSize;

}

}

public struct ReportData

{

public string[] vertexNames;

public int[][] parts;

public double performTime;

public int[,] adjMatrix;

public ReportData(string[] vertexNames, int[][] parts, double performTime,

int[,] adjMatrix)

{

this.vertexNames = vertexNames;

this.parts = parts;

this.performTime = performTime;

this.adjMatrix = adjMatrix;

}

}

}

**SettingsForm.cs**

using System;

using System.Windows.Forms;

using System.Drawing;

using SAPR.Classes;

namespace SAPR.Forms

{

public partial class SettingsForm : Form

{

public SettingsForm(SettingsObject settingsObject, Form mainForm)

{

InitializeComponent();

m\_verticesColorButton.ColorChanged += m\_ChangeValueHandler;

m\_edgesColorButton.ColorChanged += m\_ChangeValueHandler;

m\_bgColorButton.ColorChanged += m\_ChangeValueHandler;

m\_textColorButton.ColorChanged += m\_ChangeValueHandler;

m\_settingsObject = settingsObject;

m\_mainForm = mainForm;

m\_LoadSettingsInComponents();

}

private SettingsObject m\_settingsObject;

private bool m\_isUserChanged = true;

private Form m\_mainForm;

private void m\_LoadSettingsInComponents()

{

m\_isUserChanged = false;

foreach (TabPage tab in tabControl.TabPages)

foreach (object obj in tab.Controls)

{

Control element = obj as Control;

object value = m\_settingsObject[(string)element.Tag];

if (value == null)

continue;

switch (element.GetType().Name)

{

case "TextBox":

if (value is string)

((TextBox)element).Text = (string)value;

break;

case "NumericUpDown":

if (value is int)

((NumericUpDown)element).Value = (int)value;

break;

case "ButtonColor":

if (value is Color)

((ButtonColor)element).Color = (Color)value;

break;

}

}

m\_isUserChanged = true;

}

private void m\_buttonSave\_Click(object sender, EventArgs e)

{

try

{

m\_settingsObject.Save();

m\_mainForm.Refresh();

}

catch (CantSaveSettingsException)

{

MessageBox.Show(Resources.AppResources.cantSaveSettings,

Resources.AppResources.errorText, MessageBoxButtons.OK,

MessageBoxIcon.Information);

}

Close();

}

private void m\_buttonCancel\_Click(object sender, EventArgs e)

{

Close();

}

private void m\_ChangeValueHandler(object sender, EventArgs e)

{

if (!m\_isUserChanged)

return;

Control element = sender as Control;

string key = (string)element.Tag;

switch (element.GetType().Name)

{

case "TextBox":

m\_settingsObject[key] = ((TextBox)element).Text;

break;

case "NumericUpDown":

m\_settingsObject[key] = (int)((NumericUpDown)element).Value;

break;

case "ButtonColor":

m\_settingsObject[key] = ((ButtonColor)element).Color;

break;

}

}

private void m\_ChangeAlgFolderClickHandler(object sender, EventArgs e)

{

FolderBrowserDialog folderBrowse = new FolderBrowserDialog();

folderBrowse.SelectedPath = m\_algorithmsFolderText.Text;

if (folderBrowse.ShowDialog() != DialogResult.OK)

return;

m\_algorithmsFolderText.Text = folderBrowse.SelectedPath;

}

}

}