минобрнауки россии

федеральное государственное бюджетное образовательное учреждение

высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |  |  |
| --- | --- | --- |
| Институт (факультет) | | Институт информационных технологий |
| Кафедра | Математического и программного обеспечения ЭВМ | |

КУРСОВАЯ РАБОТА

|  |  |
| --- | --- |
| по дисциплине | Структурное программирование |

|  |  |
| --- | --- |
| на тему | Программирование на языке высокого уровня |

|  |
| --- |
| Выполнил студент группы |
| 1ПИб-02-3оп-23 |
| направление подготовки (специальности) |
| 09.03.04., Программная инженерия |
| шифр, наименование |
| Богданов Ренат Алексеевич |
| фамилия, имя, отчество |

|  |
| --- |
| Руководитель |
| Журавлева Юлия Михайловна |
| фамилия, имя, отчество |
| Ведущий специалист по учебно-методической работе |
| должность |

|  |
| --- |
| Дата представления работы |
| « » \_\_\_\_\_\_ 2024 г. |
| Заключение о допуске к защите |
|  |
|  |
|  |
| Оценка \_\_\_\_\_, \_\_\_\_\_\_ количество баллов |
|  |
| Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Череповец, 2024

*год*

Оглавление

[Введение 3](#_Toc169214365)

[1. Описание предметной области 4](#_Toc169214366)

[2. Описание классов 6](#_Toc169214367)

[2.1 Класс Graphics 6](#_Toc169214368)

[3. Описание созданного приложения 9](#_Toc169214369)

[3.1 Постановка задачи 9](#_Toc169214370)

[3.2 Логическое проектирование 10](#_Toc169214371)

[3.3 Физическое проектирование 13](#_Toc169214372)

[3.4 Разработка пользовательского интерфейса 15](#_Toc169214373)

[3.5 Тестирование 16](#_Toc169214374)

[3.6. Результаты работы 18](#_Toc169214375)

[Заключение 19](#_Toc169214376)

[Список источников 20](#_Toc169214377)

[Приложение 1. Техническое задание 21](#_Toc169214378)

[Приложение 2. Руководство пользователя 26](#_Toc169214379)

[Приложение 3. Программный код 28](#_Toc169214380)

# Введение

C++ (произносится «си плюс плюс») [1] — компилируемый язык программирования общего назначения со статической типизацией. Он поддерживает различные парадигмы программирования и сочетает в себе возможности как высокоуровневых, так и низкоуровневых языков. По сравнению со своим предшественником, языком C, основное внимание в C++ уделяется поддержке объектно-ориентированного и обобщенного программирования. Название «C++» происходит от языка C, где оператор ++ используется для инкремента переменной.

C++ является одним из самых популярных языков программирования и широко применяется для разработки программного обеспечения. Он используется в операционных системах, приложениях, драйверах устройств, встроенных системах, серверах и развлекательных приложениях (например, видеоиграх). Существует несколько реализаций языка C++, как бесплатных, так и коммерческих, от таких поставщиков, как GNU Project, Microsoft, Intel и Embarcadero (Borland).

Windows Forms [2] — это платформа пользовательского интерфейса для создания классических приложений Windows. Она обеспечивает один из самых эффективных способов создания классических приложений с помощью визуального конструктора в Visual Studio. Такие функции, как размещение визуальных элементов управления путем перетаскивания, упрощают создание классических приложений.

В Windows Forms можно разрабатывать графически сложные приложения, которые просто развертывать, обновлять, и с которыми удобно работать как в автономном режиме, так и в сети. Приложения Windows Forms могут получать доступ к локальному оборудованию и файловой системе компьютера, на котором работает приложение.

Цель курсовой работы заключается в разработке графического приложения, создающего изображение орнамента из шестиугольников, с возможностью менять его цвет, размер, ширину линий и реализующего сохранение получившегося изображения в растровом формате.

# Описание предметной области

Геометрический орнамент [3] представляет собой узор (шаблон, повторяющуюся последовательность), состоящий из простых геометрических фигур (точек, линий, треугольников, квадратов, окружностей и т. д.).

Из определения следует, что в основе построения геометрического орнамента лежат математические принципы. Это придает ему четкую и структурированную форму. Вместе с тем, конечный результат должен представлять собой единую, пластичную и непрерывную композицию, состоящую из отдельных элементов. Таким элементом является мотив.

Мотив — это основной элемент орнамента. Он может быть простым, состоящим из одной фигуры, или включать в себя несколько форм, образуя сложное орнаментальное целое.

Раппорт — это наименьшая повторяющаяся область рисунка, которая включает в себя мотивы и расстояние до следующих мотивов. Существуют различные типы раппортов (см. табл. 1.1).

Табл. 1.1

Типы раппортов.

|  |  |
| --- | --- |
| Тип | Пример |
| Ленточный раппорт — это тип повторяющегося рисунка, где мотив помещается в четырехугольную форму. Он применяется для создания декоративных элементов, таких как обрамления, полосы, фризы, тяги и филёнки. |  |

Продолжение табл. 1.1

|  |  |
| --- | --- |
| Сплошной. Мотив вписан в четырёхугольную сетку. При комплексных мотивах часто используют арабеску. Используется для оформления тканей, ковров, мозаик |  |
| Центральный лучевой розетчатый раппорт. Мотив вписан в треугольник |  |
| Обратный или геральдический раппорт. Данный вид раппорта более комплексный поскольку может включать все вышеперечисленные типы. изображения и такая сторона будет являться осью симметрии |  |

# Описание классов

# Класс Graphics

Graphics находится в пространстве имен System.Drawing и считается основным для работы с графикой.

Класс Graphics реализует поверхность рисования и все методы рисования GDI+. GDI расшифровывается как Graphic Device Interface (интерфейс графического устройства), и это понятие хорошо отражает его суть, потому что GDI реализует методы для рисования не только на дисплее, но и на других графических устройствах, например на принтерах. GDI+ стал продолжением первой версии графического интерфейса.

Табл. 2.1

Методы класса Graphics

|  |  |
| --- | --- |
| Метод | Описание работы |
| Clear(Color) | Очищает всю поверхность рисования и выполняет заливку поверхности указанным цветом фона. |
| DrawArc(Pen, Int32, Int32, Int32, Int32, Int32, Int32) | Рисует дугу, которая является частью эллипса, заданного парой координат, шириной и высотой. |
| Flush() | Вызывает принудительное выполнение всех отложенных графических операций и немедленно возвращается, не дожидаясь их окончания. |
| DrawEllipse(Pen, Int32, Int32, Int32, Int32) | Рисует эллипс, определяемый ограничивающим прямоугольником, заданным с помощью координат верхнего левого угла прямоугольника, высоты и ширины. |
| DrawPolygon(Pen, Point[]) | Рисует многоугольник, определяемый массивом структур Point. |
| DrawRectangle(Pen, Int32, Int32, Int32, Int32) | Рисует прямоугольник, определяемый парой координат, шириной и высотой. |
| DrawString(String, Font, Brush, PointF, StringFormat) | Создает заданную текстовую строку в указанном месте с помощью заданных объектов Brush и Font, используя атрибуты форматирования заданного формата StringFormat. |

2.2 Класс Pen

Класс Pen используется для создания пера, при помощи которого проводятся прямые и кривые линии. Рисует Pen линию указанной ширины и стиля. Линия, рисуемая элементом, Pen может быть заполнена различными стилями заливки, включая сплошные цвета и текстуры. Стиль заливки зависит от кисти или текстуры, используемой в качестве объекта заливки.

Табл. 2.2.

Методы класса Pen

|  |  |
| --- | --- |
| Метод | Описание работы |
| Clone() | Создает точную копию данного объекта Pen. |
| Dispose() | Освобождает все ресурсы, используемые этим объектом Pen. |
| ToString() | Возвращает строку, представляющую текущий объект. |
| SetLineCap(LineCap, LineCap, DashCap) | Устанавливает значения, определяющие стиль завершения, используемый в конце линий, нарисованных при помощи объекта Pen. |
| TranslateTransform(Single, Single) | Выполняет перевод локального геометрического преобразования на заданные размеры. Этот метод вставляет перевод перед преобразованием. |
| ResetTransform() | Возвращает матрице геометрического преобразования объекта Pen единичное значение. |
| MemberwiseClone() | Создает неполную копию текущего объекта Object. |
| Finalize() | Позволяет объекту попытаться освободить ресурсы и выполнить другие операции очистки, перед тем как он будет уничтожен во время сборки мусора. |

2.3 Класс Brush

Класс Brush входит в пространство имён «System.Drawing». Определяет объекты, которые используются для заливки внутри графических фигур, таких как прямоугольники, эллипсы, круги, многоугольники и дорожки [4].

Табл. 2.3.

Методы класса Brush

|  |  |
| --- | --- |
| Метод | Описание работы |
| Clone() | При переопределении в производном классе создает точную копию этого объекта Brush. |
| Dispose() | Освобождает все ресурсы, используемые этим объектом Brush. |
| Dispose(Boolean) | Освобождает неуправляемые ресурсы, используемые объектом Brush, а при необходимости освобождает также управляемые ресурсы. |
| Finalize() | Позволяет объекту попытаться освободить ресурсы и выполнить другие операции очистки, перед тем как он будет уничтожен во время сборки мусора. |
| SetNativeBrush(IntPtr) | в производном классе задает ссылку на объект GDI+ brush. |
| GetType() | Возвращает объект Type для текущего экземпляра. |
| ToString() | Возвращает строку, представляющую текущий объект. |

# Описание созданного приложения

# Постановка задачи

Программа должна:

* строить орнамент, состоящий из шестиугольников и линий внутри них;
* иметь возможность изменять цвет фона и линий через диалоговое окно выбора цвета;
* иметь возможность изменять масштаб фигур;
* иметь возможность изменять толщину линий;
* иметь возможность закрасить участки (шестиугольники) случайным образом;
* иметь возможность настраивать частоту закрашивания случайных участков;
* иметь возможность изменять цвет заливки случайных участков;
* иметь возможность сохранять орнамент в формате растрового изображения.

Для отрисовки был выбран следующий орнамент (рис. 3.1):

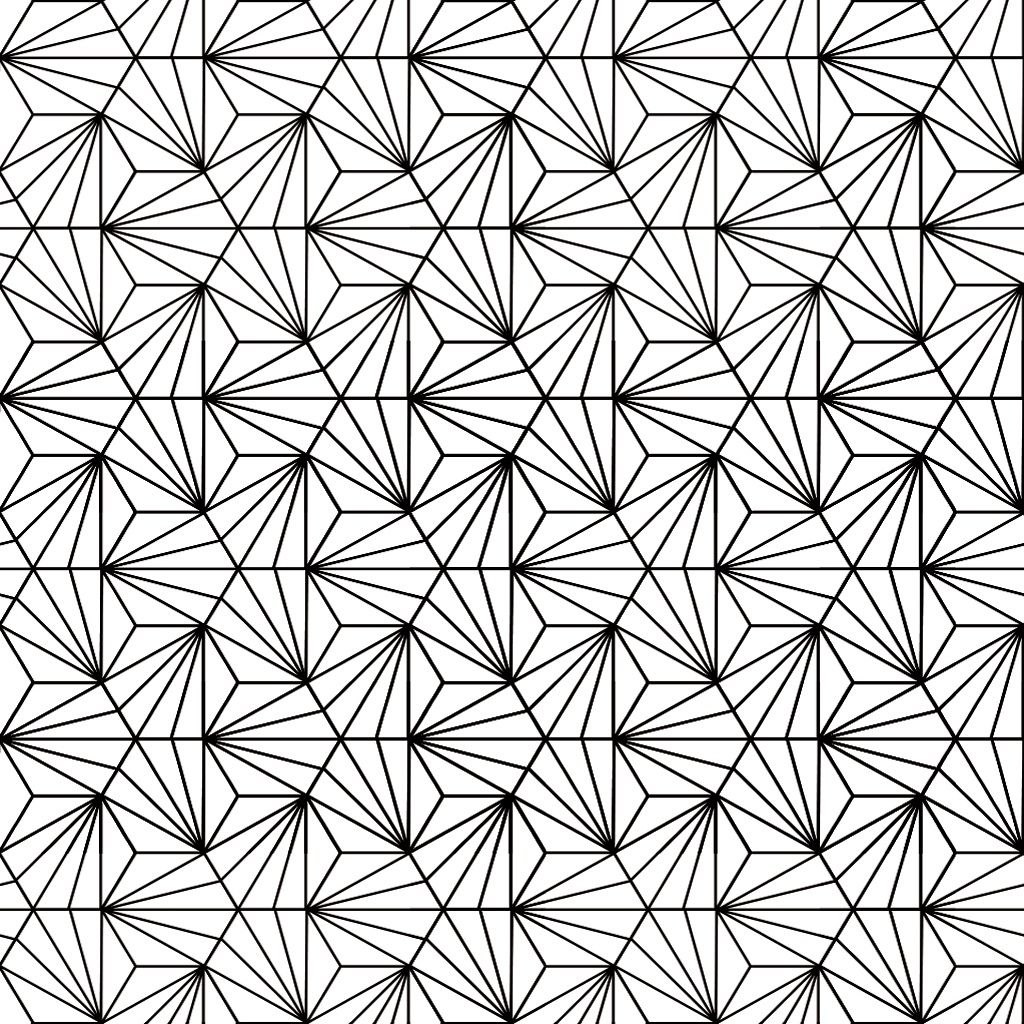


Рис. 3.1. Выбранный орнамент

# Логическое проектирование

Программа получает на вход следующие параметры:

* размер шестиугольника;
* цвет фона;
* цвет контура;
* цвет фигуры;
* ширину контура;
* необходимость отрисовки сетки;
* необходимость отрисовки линий внутри фигуры;
* необходимость заливки случайно выбранных фигур;
* вероятность заливки случайно выбранных фигур.

Из выбранного орнамента можно выделить три ряда, после которых начинается повторение. Направление узора внутри фигур меняется с каждым рядом (см. рис 3.2).

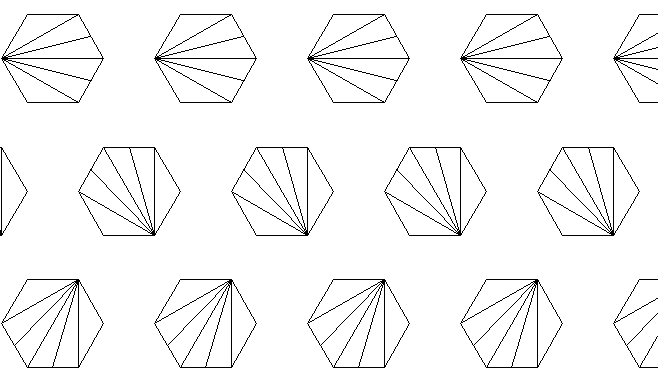


Рис. 3.2. Закономерность орнамента

Для построения выделяем длину и ширину шестиугольника. Задаём начальные координаты орнамента и вводим переменные для хранения текущих координат. Условный размер фигуры равен длине линии, проведённой перпендикулярно к одной из сторон. Ширина фигуры равна двум размерам, а высота - √3 от размера (см. рис 3.3). Расстояние между начальной точкой построения фигуры и началом следующей справа фигура равняется 1.5 от ширины фигуры (см. рис 3.4). Расстояние по y между рядами равно половине высоты. Смещение каждого второго ряда по x равно 0.75 от ширины.

Шестиугольник реализован с помощью создания полигона из 6 точек.

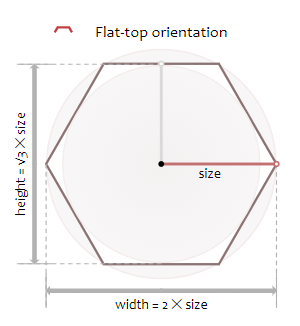


Рис. 3.3. Размер фигуры

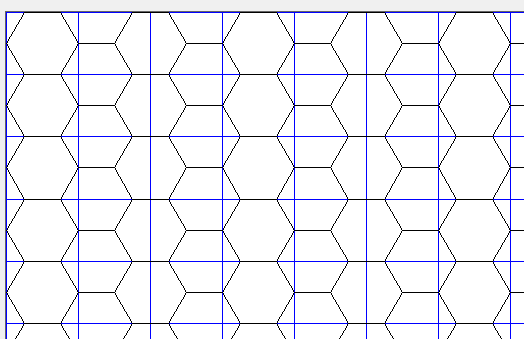


Рис. 3.4. Ряды и координатная сетка (ширина по x и высота по y)

Алгоритм работы программы:

1. закрашиваем фон в выбранный цвет (по умолчанию: в белый);
2. получаем координаты точек шестиугольника;
3. закрашиваем случайно выбранные шестиугольники в выбранный цвет (если включён данный параметр). Вероятность закрашивания выбирается пользователем;
4. рисуем прямоугольную сетку (если включён данный параметр);
5. рисуем сетку шестиугольников. Ширина линий и цвет заранее определяются пользователем.

Блок-схема алгоритма представлена на рисунке 3.5.

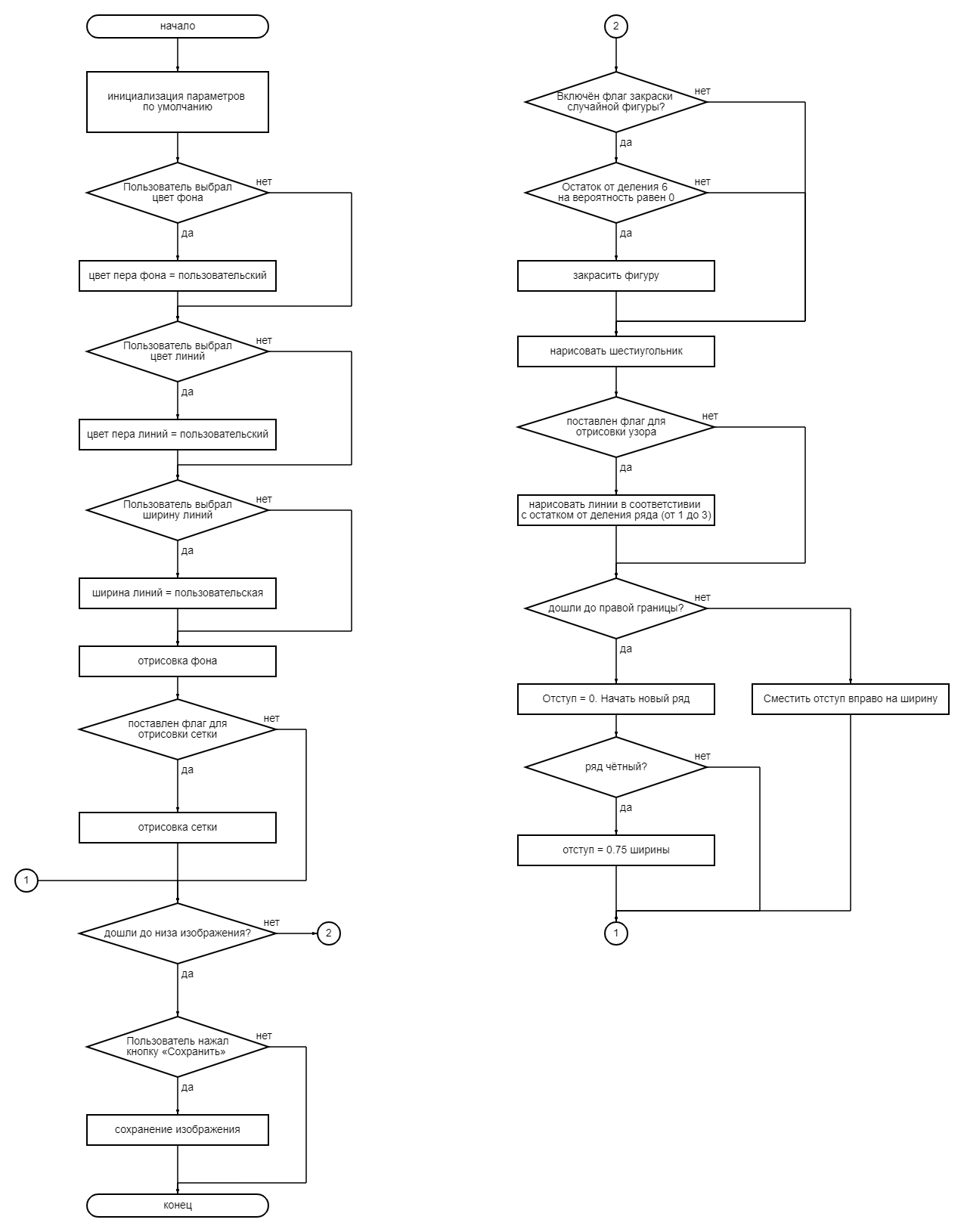


Рисунок 3.5. Блок-схема

# Физическое проектирование

Табл. 3.1.

Переменные

|  |  |  |
| --- | --- | --- |
| Наименование | Обозначение | Тип данных |
| Размер фигуры | size | double |
| Отрисовка сетки | draw\_grid | bool |
| Отрисовка линий | draw\_lines |  |
| Случайная заливка | random\_fill |  |
| Толщина линий | pen\_size | float |
| Частота заливки | fill\_frequency | int |
| Цвет линий | pattern\_color | Color |
| Цвет фона | background\_color |  |
| Цвет заливки | fill\_color |  |
| Цвет для замены | temp\_color |  |
| Холст | bmp | Bitmap |
| Поверхность для рисования | g | Graphics |
| Перо для сетки | pen\_grid | Pen |
| Перо для линий | pen\_lines |  |
| Кисть для заливки | brush\_fill | Brush |
| Высота | height | Double |
| Ширина | width |  |
| Нулевая точка по x | zero\_point\_x |  |
| Нулевая точка по y | zero\_point\_y |  |
| Шаг по x | step\_x |  |
| Шаг по y | step\_y |  |
| Смещение ряда по y | modifier\_i | int |
| Координаты вершин фигуры | p1, p2, p3, p4, p5, p6 | Point |

Табл. 3.2.

Функции программы

|  |  |  |
| --- | --- | --- |
| Наименование | Параметры | Выполняемое действие |
| draw\_pattern |  | Отрисовка орнамента |
| button1\_Click | System::Object^ sender, System::EventArgs^ e | Вызов функции draw\_pattern |
| checkBox1\_CheckedChanged | System::Object^ sender, System::EventArgs^ e | Изменение параметра отрисовки сетки |
| button3\_Click | System::Object^ sender, System::EventArgs^ e | Изменение цвета линий |
| button4\_Click | System::Object^ sender, System::EventArgs^ e | Изменение цвета фона |
| button2\_Click | System::Object^ sender, System::EventArgs^ e | Сохранение изображения |
| trackBar1\_Scroll | System::Object^ sender, System::EventArgs^ e | Изменение размера фигуры |
| checkBox2\_CheckedChanged | System::Object^ sender, System::EventArgs^ e | Изменение параметра отрисовки линий |
| button5\_Click | System::Object^ sender, System::EventArgs^ e | Смена цветов линий и фона между собой |
| trackBar2\_Scroll | System::Object^ sender, System::EventArgs^ e | Изменение ширины линий |
| checkBox3\_CheckedChanged | System::Object^ sender, System::EventArgs^ e | Изменение параметра случайной заливки |
| button6\_Click | System::Object^ sender, System::EventArgs^ e | Изменение цвета заливки |
| trackBar3\_Scroll | System::Object^ sender, System::EventArgs^ e | Изменение частоты заливки |

# Разработка пользовательского интерфейса

Интерфейс приложения (см. рис. 3.6) поделён на две части. Левая часть – меню параметров, правая – область для изображения.

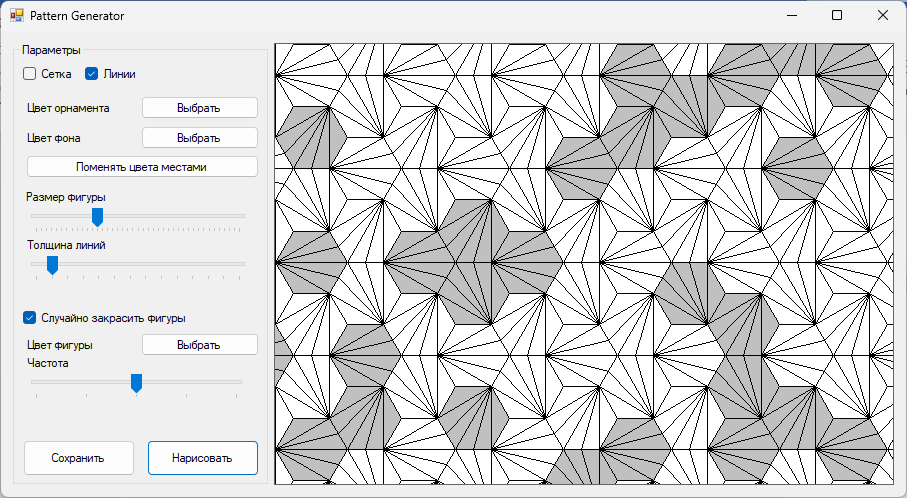


Рис. 3.6. Интерфейс приложения

Описание элементов интерфейса:

* checkBox1 – флажок «Сетка», управляющий параметром отрисовки сетки;
* checkBox2 – флажок «Линии», управляющий параметром отрисовки линий внутри фигур;
* button3 – кнопка «Выбрать» (цвет орнамента), при нажатии открывающая диалоговое окно выбора цвета для линий;
* button4 – кнопка «Выбрать» (цвет фона), при нажатии открывающая диалоговое окно выбора цвета для фона;
* button5 – кнопка «Поменять цвета местами», при нажатии меняющая цвет орнамента и цвет фона между собой;
* trackBar1 – ползунок «Размер фигуры», при перемещение изменяющий размер шестиугольников;
* trackBar2 – ползунок «Толщина линий», при перемещение изменяющий толщину линий;
* checkBox3 – флажок «Случайно закрасить фигуры», управляющий параметром заливки цветом случайных фигур;
* button6 – кнопка «Цвет фигуры», при нажатии открывающая диалоговое окно выбора цвета закраски случайной фигуры;
* trackBar3 – ползунок «Частота», при перемещение изменяющий вероятность закраски фигуры.

# Тестирование

Этапы и результаты тестирования представлены в таблице 3.3.

Табл. 3.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата проведения тестирования | Исходные данные | Тестируемая функция | Результат теста | Результат тестирования |
| 15.05.2024 | Пользователь запускает программу, нажимает кнопку «Нарисовать» | Функция рисования орнамента | В PictureBox появляется изображение орнамента с оригинальными параметрами | Успех |
| 18.05.2024 | Пользователь запускает программу, нажимает кнопку «Выбрать» (цвет орнамента), выбирает цвет в диалоговом окне, нажимает кнопку «Нарисовать» | Функция смены цвета линий | В PictureBox появляется изображение орнамента с заданными параметрами | Успех |
| 18.05.2024 | Пользователь запускает программу, нажимает кнопку «Выбрать» (цвет фона), выбирает цвет в диалоговом окне, нажимает кнопку «Нарисовать» | Функция смены цвета фона | В PictureBox появляется изображение орнамента с заданными параметрами | Успех |

Продолжение табл. 3.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 19.05.2024 | Пользователь запускает программу, перетаскивает ползунок «Размер фигуры» на нужное значение, нажимает кнопку «Нарисовать» | Функция изменения размера орнамента | В PictureBox появляется изображение орнамента с заданными параметрами | Успех |
| 21.05.2024 | Пользователь запускает программу, ставит флажок «Случайно закрасить фигуры», нажимает кнопку «Нарисовать» | Функция закраски случайных фигур | В PictureBox появляется изображение орнамента с заданными параметрами | Успех |
| 22.05.2024 | Пользователь запускает программу, перетаскивает ползунок «Частота» на нужное значение, нажимает кнопку «Нарисовать» | Функция изменения частоты закраски случайных фигур | В PictureBox появляется изображение орнамента с заданными параметрами | Успех |
| 23.05.2024 | Пользователь запускает программу, нажимает кнопку «Выбрать» (цвет фигуры), выбирает цвет в диалоговом окне, нажимает кнопку «Нарисовать»» | Функция изменения цвета закраски фигур | В PictureBox появляется изображение орнамента с заданными параметрами | Успех |

# 3.6. Результаты работы

По результатам работы было создано приложение, реализующее геометрический орнамент, выбранный в сети Интернет. Программа создаёт соответствующее графическое изображение, позволяет менять его масштаб (размеры) и цвета. Пример работы программы приведён на рисунке 3.7. Также программа способна сохранять изображение в растровых форматах (пример сохранённого изображения на рис. 3.8).

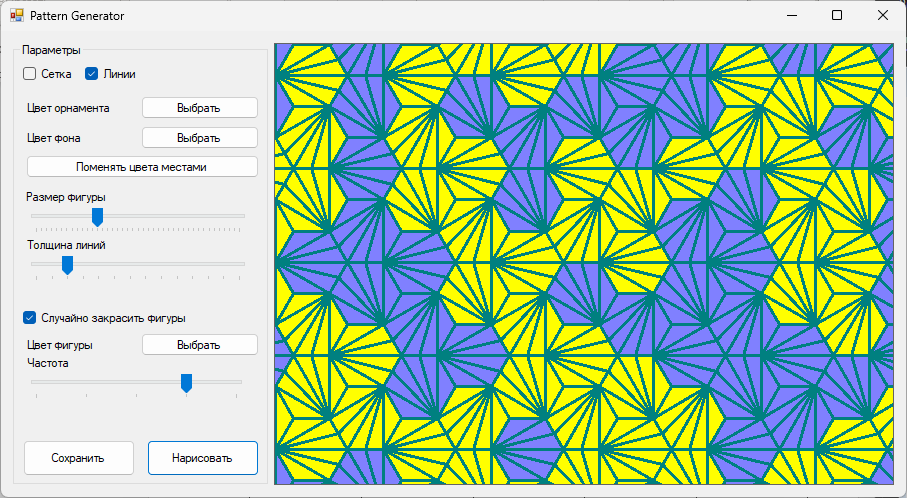


Рис. 3.7. Работа приложения

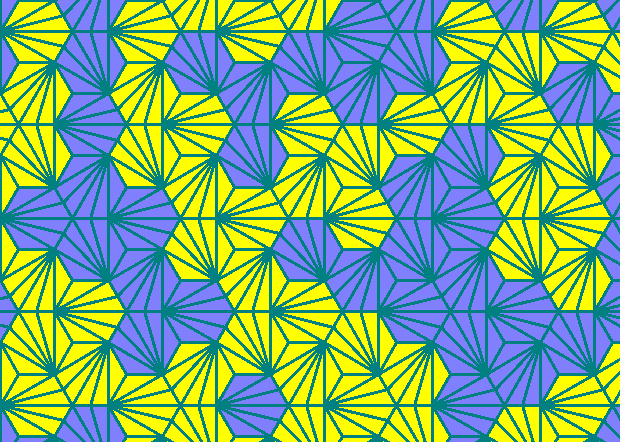


Рис. 3.8. Сохранённое изображение

# Заключение

В ходе выполнения курсовой работы были освоенны принципы работы с приложением Windows Forms и получен практический опыт программирования на языке C++. В результате была разработана программа для построения геометрических орнаментов с широкими возможностями настройки: изменение цветов фона, линий и заливки фигур; изменение ширины линий, размеров фигур; изменение частоты закраски случайно выбранных фигур. В программе предусмотрена возможность сохранения полученного орнамента в формате растрового изображения.

Разработанное приложение демонстрирует потенциал использования языка C++ для создания интерактивных программ с графическим интерфейсом. Полученный опыт и навыки программирования на C++ будут полезны при разработке будущих проектов, требующих визуализации и работы с графикой.

# Список источников

1. Язык программирования C++. [электронный ресурс] https://programmers.fandom.com/ru/wiki/C%2B%2B. Дата обращения: 12.06.2024
2. Что такое Windows Forms – Windows Forms .NET [электронный ресурс] https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-8.0. Дата обращения: 12.06.24
3. Кинева Л. А. История и теория орнамента: учебное пособие / Л. А. Кинева; научный редактор О. И. Ган; Министерство образования и науки Российской Федерации, Уральский федеральный университет имени первого Президента России Б.Н. Ельцина. — Екатеринбург: Издательство Уральского университета, 2018. — 131, [1] с. — ISBN 978-5-7996-2387-6.
4. Класс Graphics [электронный ресурс] https://docs.microsoft.com/ru-ru/dotnet/api/system.drawing.graphics?view=netframework-4.5. Дата обращения: 12.06.2024
5. Класс Pen [электронный ресурс] https://docs.microsoft.com/ru-ru/dotnet/api/system.drawing.pen?view=dotnet-plat-ext-6.0. Дата обращения: 12.06.2024
6. Класс Brush [электронный ресурс] https://docs.microsoft.com/ru-ru/dotnet/api/system.drawing.brush?view=dotnet-plat-ext-6.0. Дата обращения: 12.06.2024
7. Сетки гексагонов [электронный ресурс] https://www.redblobgames.com/grids/hexagons/. Дата обращения: 8.06.2024

# Приложение 1. Техническое задание

МИНОБРАНАУКИ РОССИИ

федеральное государственное бюджетное

образовательное учреждение высшего образования

ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

|  |  |
| --- | --- |
| Институт информационных технологий   |  | | --- | | наименование института (факультета) | |
| Кафедра математического и программного обеспечения |
| Наименование кафедры |
| Структурное программирование |
| Наименование дисциплины в соответствии с учебным планом |
|  |

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ

д. т.н. \_\_\_\_\_\_\_\_\_ Ершов Е.В.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2024 г.

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ

Техническое задание на курсовую работу

Листов 5

Руководитель: ведущий специалист по учебно-методической работе Журавлева Ю.М.

Исполнитель: студент гр. 1ПИб-02-3оп-23

Богданов Р.А.

2024 год

Введение

Данный курсовой проект посвящен разработке графического приложения, визуализирующего орнамент с заданными параметрами.

1. Основания для разработки

Основанием для разработки сайта является задание на курсовую работу по дисциплине "Структурное программирование", выданное на кафедре МПО ЭВМ ИИТ ЧГУ.

Дата утверждения: 1 марта 2024 года.

Наименование темы разработки: “Программирование на языке высокого уровня”.

2. Назначение разработки

Требуется написать программу, реализующую один из геометрических орнаментов. Программа должна:

* строить орнамент, состоящий из шестиугольников и линий внутри них;
* иметь возможность изменять цвет фона и линий через диалоговое окно выбора цвета;
* иметь возможность изменять масштаб фигур;
* иметь возможность изменять толщину линий;
* иметь возможность закрасить участки (шестиугольники) случайным образом;
* иметь возможность настраивать частоту закрашивания случайных участков;
* иметь возможность изменять цвет заливки случайных участков;
* иметь возможность сохранять орнамент в формате растрового изображения.

3.1 Требования к функциональным характеристикам

Приложение должно корректно изображать орнамент на основе введённых пользователем данных, а также сохранять в виде растрового изображения при желании пользователя.

3.2. Требования к надёжности

Приложение должно:

* не позволять пользователю вводить некорректные данные, приводящие к сбою;
* корректно работать при удовлетворяющих данных.

3.3. Условия эксплуатации

Приложение должно исполняться на исправном ПК с установленными драйверами Microsoft Visual C++.

3.4. Требования к составу и параметрам технических средств

Минимальные системные требования:

• Оперативная память: 100 Мб и более;

• Архитектура процессора: x64;

• Наличие компьютерной мыши, клавиатуры и монитора;

• 10 Мб свободного места на диске;

3.5. Требования к информационной и программной совместимости

ПК с операционной системой Windows и с установленными драйверами Microsoft Visual C++

Требования к системе:

Windows 7, Windows 8, Windows 8.1, Windows 10 или более поздней версии.

Примечание. Также поддерживаются следующие серверные ОС: Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2 или Windows Server 2016.

3.6. Требования к маркировке и упаковке

Отсутствуют.

3.7. Требования к транспортированию и хранению

Отсутствуют.

3.8. Специальные требования

Отсутствуют.

4. Требования к программной документации

4.1 Содержание расчётно-пояснительной записки:

Программная документация должна содержать расчётно-пояснительную записку с содержанием:

1. титульный лист;

1. оглавление;
2. введение;
3. описание предметной области;
4. описание классов;
5. описание созданного приложения;
6. заключение;
7. источники;
8. техническое задание;
9. руководство пользователя;
10. программный код.

4.2. Требования к оформлению

Требования к оформлению, установленные ГОСТ, должны быть выполнены на протяжении всей работы без каких-либо изменений (в табл. П1.1).

Таблица П1.1

Требования к оформлению

|  |  |
| --- | --- |
| Документ | Печать на отдельных листах формата А4 (210х297 мм); оборотная сторона не заполняется; листы нумеруются. Печать возможна ч/б. |
| Страницы | Ориентация — книжная; отдельные страницы, при необходимости, альбомная. Поля: верхнее, нижнее — по 2 см, левое — 3 см, правое — 1 см. |
| Абзацы | Межстрочный интервал — 1, перед и после абзаца — 0. |
| Шрифты | Кегль — 14. В таблицах шрифт 12. Шрифт листинга — 10 (возможно в 2 колонки). |
| Рисунки | Подписывается под ним по центру: «Рис.Х. Название В» приложениях: «Рис.П1.3. Название» |
| Таблицы | Подписывается: над таблицей, выравнивание по правому: «Таблица Х». В следующей строке по центру Название Надписи в «шапке» (имена столбцов, полей) — по центру. В теле таблицы (записи) текстовые значения — выравнены по левому краю, числа, даты — по правому. |

5. Стадии и этапы разработки

Стадии и этапы разработки представлены в таблице П1.2.

Таблица П1.2

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки ПО | Сроки разработки | Результат выполнения | Отметка о выполнении |
| Описание алгоритма программы | 20.05.2024 | Составленный алгоритм работы приложения |  |
| Написание программы | 25.05.2024 | Доработанная программа |  |
| Сборка готового приложения из программного кода | 10.06.2024 | Готовое приложение |  |

6. Порядок контроля и приёмки

Порядок контроля и приёма представлены в таблице П1.3.

Таблица П1.3

Порядок контроля и приёма

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование контрольного этапа выполнения курсовой работы | Сроки контроля | Результат выполнения | Отметка о приёмке результата контрольного этапа |
| Оформление Техническое задание. | 15.05.2024 | Выполненное оформление ТЗ |  |
| Оформление сопроводительной документации – РПЗ. | 11.06.2024 | Выполненное оформление РПЗ |  |
| Сдача курсовой работы | 14.06.2024 | Получение оценки за выполненную работу |  |

# Приложение 2. Руководство пользователя

1. Общие сведения о программе

Приложение «Pattern Generator» предназначено для изучения орнамента, состоящего из шестиугольников путем экспериментального изменения его характеристик.

1. Описание установки

Разместить файл «Pattern Generator.exe» в удобном и легкодоступном месте.

1. Описание запуска

Для запуска приложения необходимо нажать, двойным кликом курсора по значку приложения или правой кнопкой мыши по значку и в выпадающем меню нажать кнопку “Открыть”.

1. Инструкция по работе

Элементы интерфейса изображены на рисунке П2.1. Все элементы пронумерованы:

1. флаг «Сетка», определяющий состояние параметра отрисовки сетки;
2. флаг «Линии», определяющий состояние параметра отрисовки линий (узора);
3. кнопка выбора цвета орнамента (линий);
4. кнопка определения цвета фона;
5. кнопка смены цветов друг с другом (цвет линий и цвет фона);
6. ползунок, определяющий размер фигуры;
7. ползунок, определяющий толщину линий;
8. флаг «Случайно закрасить фигуры», определяющий соответствующий параметр;
9. кнопка выбора цвета заливки случайных фигур;
10. ползунок определяющий вероятность заливки фигуры;
11. кнопка сохранения изображения;
12. кнопка отрисовки изображения на холсте;
13. холст, на котором рисуется орнамент.

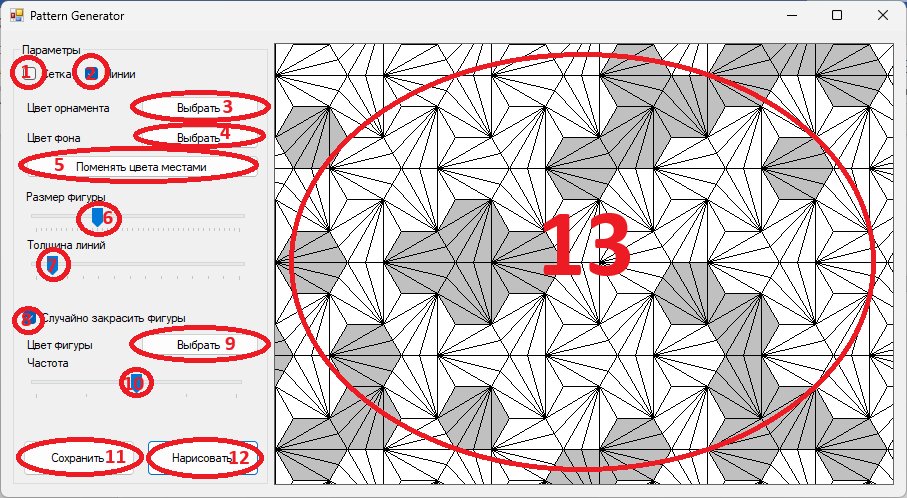


Рис. П2.1. Интерфейс приложения

# Приложение 3. Программный код

Файл “CppCLR\_WinFormsProject.cpp”

#include "pch.h"

#include <cmath>

#include <time.h>

#include "Form1.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

int main()

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Application::Run(gcnew CppCLRWinFormsProject::Form1());

return 0;

}

Файл “Form1.h”

#pragma once

namespace CppCLRWinFormsProject {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for Form1

/// </summary>

public ref class Form1 : public System::Windows::Forms::Form

{

public:

Form1(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~Form1()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::GroupBox^ groupBox1;

private: System::Windows::Forms::PictureBox^ pictureBox1;

private: System::Windows::Forms::Panel^ panel2;

private: System::Windows::Forms::Panel^ panel1;

private: System::Windows::Forms::Button^ button2;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::CheckBox^ checkBox1;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::Button^ button4;

private: System::Windows::Forms::Button^ button3;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::ColorDialog^ colorDialog1;

private: System::Windows::Forms::SaveFileDialog^ saveFileDialog1;

private: System::Windows::Forms::TrackBar^ trackBar1;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::CheckBox^ checkBox2;

private: System::Windows::Forms::Button^ button5;

private: System::Windows::Forms::TrackBar^ trackBar2;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::CheckBox^ checkBox3;

private: System::Windows::Forms::TrackBar^ trackBar3;

private: System::Windows::Forms::Label^ label6;

private: System::Windows::Forms::Button^ button6;

private: System::Windows::Forms::Label^ label5;

protected:

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

void InitializeComponent(void)

{

this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());

this->panel2 = (gcnew System::Windows::Forms::Panel());

this->trackBar3 = (gcnew System::Windows::Forms::TrackBar());

this->label6 = (gcnew System::Windows::Forms::Label());

this->button6 = (gcnew System::Windows::Forms::Button());

this->label5 = (gcnew System::Windows::Forms::Label());

this->checkBox3 = (gcnew System::Windows::Forms::CheckBox());

this->trackBar2 = (gcnew System::Windows::Forms::TrackBar());

this->label4 = (gcnew System::Windows::Forms::Label());

this->button5 = (gcnew System::Windows::Forms::Button());

this->checkBox2 = (gcnew System::Windows::Forms::CheckBox());

this->label3 = (gcnew System::Windows::Forms::Label());

this->trackBar1 = (gcnew System::Windows::Forms::TrackBar());

this->label2 = (gcnew System::Windows::Forms::Label());

this->label1 = (gcnew System::Windows::Forms::Label());

this->button4 = (gcnew System::Windows::Forms::Button());

this->button3 = (gcnew System::Windows::Forms::Button());

this->checkBox1 = (gcnew System::Windows::Forms::CheckBox());

this->panel1 = (gcnew System::Windows::Forms::Panel());

this->button2 = (gcnew System::Windows::Forms::Button());

this->button1 = (gcnew System::Windows::Forms::Button());

this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());

this->colorDialog1 = (gcnew System::Windows::Forms::ColorDialog());

this->saveFileDialog1 = (gcnew System::Windows::Forms::SaveFileDialog());

this->groupBox1->SuspendLayout();

this->panel2->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar3))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar2))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar1))->BeginInit();

this->panel1->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->BeginInit();

this->SuspendLayout();

//

// groupBox1

//

this->groupBox1->Controls->Add(this->panel2);

this->groupBox1->Controls->Add(this->panel1);

this->groupBox1->Location = System::Drawing::Point(12, 12);

this->groupBox1->Name = L"groupBox1";

this->groupBox1->Size = System::Drawing::Size(255, 442);

this->groupBox1->TabIndex = 0;

this->groupBox1->TabStop = false;

this->groupBox1->Text = L"Параметры";

//

// panel2

//

this->panel2->Controls->Add(this->trackBar3);

this->panel2->Controls->Add(this->label6);

this->panel2->Controls->Add(this->button6);

this->panel2->Controls->Add(this->label5);

this->panel2->Controls->Add(this->checkBox3);

this->panel2->Controls->Add(this->trackBar2);

this->panel2->Controls->Add(this->label4);

this->panel2->Controls->Add(this->button5);

this->panel2->Controls->Add(this->checkBox2);

this->panel2->Controls->Add(this->label3);

this->panel2->Controls->Add(this->trackBar1);

this->panel2->Controls->Add(this->label2);

this->panel2->Controls->Add(this->label1);

this->panel2->Controls->Add(this->button4);

this->panel2->Controls->Add(this->button3);

this->panel2->Controls->Add(this->checkBox1);

this->panel2->Location = System::Drawing::Point(7, 20);

this->panel2->Name = L"panel2";

this->panel2->Size = System::Drawing::Size(242, 357);

this->panel2->TabIndex = 1;

//

// trackBar3

//

this->trackBar3->Location = System::Drawing::Point(3, 309);

this->trackBar3->Maximum = 5;

this->trackBar3->Minimum = 1;

this->trackBar3->Name = L"trackBar3";

this->trackBar3->Size = System::Drawing::Size(227, 45);

this->trackBar3->TabIndex = 15;

this->trackBar3->Value = 3;

this->trackBar3->Scroll += gcnew System::EventHandler(this, &Form1::trackBar3\_Scroll);

//

// label6

//

this->label6->AutoSize = true;

this->label6->Location = System::Drawing::Point(4, 293);

this->label6->Name = L"label6";

this->label6->Size = System::Drawing::Size(49, 13);

this->label6->TabIndex = 14;

this->label6->Text = L"Частота";

//

// button6

//

this->button6->Location = System::Drawing::Point(121, 270);

this->button6->Name = L"button6";

this->button6->Size = System::Drawing::Size(118, 23);

this->button6->TabIndex = 13;

this->button6->Text = L"Выбрать";

this->button6->UseVisualStyleBackColor = true;

this->button6->Click += gcnew System::EventHandler(this, &Form1::button6\_Click);

//

// label5

//

this->label5->AutoSize = true;

this->label5->Location = System::Drawing::Point(4, 275);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(73, 13);

this->label5->TabIndex = 12;

this->label5->Text = L"Цвет фигуры";

//

// checkBox3

//

this->checkBox3->AutoSize = true;

this->checkBox3->Location = System::Drawing::Point(3, 247);

this->checkBox3->Name = L"checkBox3";

this->checkBox3->Size = System::Drawing::Size(170, 17);

this->checkBox3->TabIndex = 11;

this->checkBox3->Text = L"Случайно закрасить фигуры";

this->checkBox3->UseVisualStyleBackColor = true;

this->checkBox3->CheckedChanged += gcnew System::EventHandler(this, &Form1::checkBox3\_CheckedChanged);

//

// trackBar2

//

this->trackBar2->Location = System::Drawing::Point(3, 191);

this->trackBar2->Maximum = 14;

this->trackBar2->Minimum = 1;

this->trackBar2->Name = L"trackBar2";

this->trackBar2->Size = System::Drawing::Size(230, 45);

this->trackBar2->TabIndex = 10;

this->trackBar2->Value = 2;

this->trackBar2->Scroll += gcnew System::EventHandler(this, &Form1::trackBar2\_Scroll);

//

// label4

//

this->label4->AutoSize = true;

this->label4->Location = System::Drawing::Point(4, 175);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(86, 13);

this->label4->TabIndex = 9;

this->label4->Text = L"Толщина линий";

//

// button5

//

this->button5->Location = System::Drawing::Point(6, 92);

this->button5->Name = L"button5";

this->button5->Size = System::Drawing::Size(233, 23);

this->button5->TabIndex = 8;

this->button5->Text = L"Поменять цвета местами";

this->button5->UseVisualStyleBackColor = true;

this->button5->Click += gcnew System::EventHandler(this, &Form1::button5\_Click);

//

// checkBox2

//

this->checkBox2->AutoSize = true;

this->checkBox2->Checked = true;

this->checkBox2->CheckState = System::Windows::Forms::CheckState::Checked;

this->checkBox2->Location = System::Drawing::Point(65, 3);

this->checkBox2->Name = L"checkBox2";

this->checkBox2->Size = System::Drawing::Size(58, 17);

this->checkBox2->TabIndex = 7;

this->checkBox2->Text = L"Линии";

this->checkBox2->UseVisualStyleBackColor = true;

this->checkBox2->CheckedChanged += gcnew System::EventHandler(this, &Form1::checkBox2\_CheckedChanged);

//

// label3

//

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(3, 127);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(87, 13);

this->label3->TabIndex = 6;

this->label3->Text = L"Размер фигуры";

//

// trackBar1

//

this->trackBar1->Location = System::Drawing::Point(3, 143);

this->trackBar1->Maximum = 64;

this->trackBar1->Minimum = 24;

this->trackBar1->Name = L"trackBar1";

this->trackBar1->Size = System::Drawing::Size(230, 45);

this->trackBar1->TabIndex = 5;

this->trackBar1->Value = 36;

this->trackBar1->Scroll += gcnew System::EventHandler(this, &Form1::trackBar1\_Scroll);

//

// label2

//

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(4, 68);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(61, 13);

this->label2->TabIndex = 4;

this->label2->Text = L"Цвет фона";

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(4, 38);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(90, 13);

this->label1->TabIndex = 3;

this->label1->Text = L"Цвет орнамента";

//

// button4

//

this->button4->Location = System::Drawing::Point(121, 63);

this->button4->Name = L"button4";

this->button4->Size = System::Drawing::Size(118, 23);

this->button4->TabIndex = 2;

this->button4->Text = L"Выбрать";

this->button4->UseVisualStyleBackColor = true;

this->button4->Click += gcnew System::EventHandler(this, &Form1::button4\_Click);

//

// button3

//

this->button3->Location = System::Drawing::Point(121, 33);

this->button3->Name = L"button3";

this->button3->Size = System::Drawing::Size(118, 23);

this->button3->TabIndex = 1;

this->button3->Text = L"Выбрать";

this->button3->UseVisualStyleBackColor = true;

this->button3->Click += gcnew System::EventHandler(this, &Form1::button3\_Click);

//

// checkBox1

//

this->checkBox1->AutoSize = true;

this->checkBox1->Location = System::Drawing::Point(3, 3);

this->checkBox1->Name = L"checkBox1";

this->checkBox1->Size = System::Drawing::Size(56, 17);

this->checkBox1->TabIndex = 0;

this->checkBox1->Text = L"Сетка";

this->checkBox1->UseVisualStyleBackColor = true;

this->checkBox1->CheckedChanged += gcnew System::EventHandler(this, &Form1::checkBox1\_CheckedChanged);

//

// panel1

//

this->panel1->Controls->Add(this->button2);

this->panel1->Controls->Add(this->button1);

this->panel1->Location = System::Drawing::Point(7, 383);

this->panel1->Name = L"panel1";

this->panel1->Size = System::Drawing::Size(242, 53);

this->panel1->TabIndex = 0;

//

// button2

//

this->button2->Location = System::Drawing::Point(3, 14);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(112, 36);

this->button2->TabIndex = 1;

this->button2->Text = L"Сохранить";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &Form1::button2\_Click);

//

// button1

//

this->button1->Location = System::Drawing::Point(127, 14);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(112, 36);

this->button1->TabIndex = 0;

this->button1->Text = L"Нарисовать";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &Form1::button1\_Click);

//

// pictureBox1

//

this->pictureBox1->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((((System::Windows::Forms::AnchorStyles::Top | System::Windows::Forms::AnchorStyles::Bottom)

| System::Windows::Forms::AnchorStyles::Left)

| System::Windows::Forms::AnchorStyles::Right));

this->pictureBox1->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->pictureBox1->Location = System::Drawing::Point(273, 12);

this->pictureBox1->Name = L"pictureBox1";

this->pictureBox1->Size = System::Drawing::Size(620, 442);

this->pictureBox1->TabIndex = 1;

this->pictureBox1->TabStop = false;

//

// Form1

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(905, 466);

this->Controls->Add(this->pictureBox1);

this->Controls->Add(this->groupBox1);

this->Name = L"Form1";

this->Text = L"Pattern Generator";

this->groupBox1->ResumeLayout(false);

this->panel2->ResumeLayout(false);

this->panel2->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar3))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar2))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar1))->EndInit();

this->panel1->ResumeLayout(false);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->EndInit();

this->ResumeLayout(false);

}

#pragma endregion

double size = 36;

bool draw\_grid = false,

draw\_lines = true,

random\_fill = false;

float pen\_size = 1.0f;

int fill\_frequency = 2;

Color pattern\_color = Color::Black,

background\_color = Color::White,

fill\_color = Color::Green;

private: void draw\_pattern() {

Bitmap ^bmp = gcnew Bitmap(pictureBox1->Width, pictureBox1->Height);

Graphics ^g = pictureBox1->CreateGraphics();

Pen ^pen\_grid = gcnew Pen(Color::Blue, 1.0f),

^pen\_lines = gcnew Pen(pattern\_color, pen\_size);

Brush^ brush\_fill = gcnew SolidBrush(fill\_color);

double height = size \* pow(3, 0.5),

width = size \* 2,

zero\_point\_x = -1.5 \* width,

zero\_point\_y = -3 \* height,

step\_x = zero\_point\_x,

step\_y = zero\_point\_y;

int modifier\_i = 0;

g = Graphics::FromImage(bmp);

pictureBox1->Image = bmp;

g->Clear(background\_color);

srand(time(0));

if (draw\_grid) {

for (int i = 0; i <= 10; i++) {

g->DrawLine(pen\_grid, 0, height \* i, 1024, height \* i);

g->DrawLine(pen\_grid, width \* i, 0, width \* i, 1024);

}

}

for (int y = 0; y <= 50; y++) {

for (int x = 0; x <= 25; x++) {

Point p1 = Point(0.25 \* width + step\_x, step\_y),

p2 = Point((0.75 \* width + step\_x), step\_y),

p3 = Point((width + step\_x), 0.5 \* height + step\_y),

p4 = Point((0.75 \* width + step\_x), height + step\_y),

p5 = Point((0.25 \* width + step\_x), height + step\_y),

p6 = Point((step\_x), 0.5 \* height + step\_y);

array <Point>^ points = { p1,p2,p3,p4,p5,p6 };

if ((random\_fill) && (rand() % (6-fill\_frequency) == 0)) {

g->FillPolygon(brush\_fill, points);

}

g->DrawPolygon(pen\_lines, points);

if (draw\_lines) {

if (y % 3 == 0) {

g->DrawLine(pen\_lines, Point(step\_x, 0.5 \* height + step\_y), Point(step\_x + width, 0.5 \* height + step\_y));

g->DrawLine(pen\_lines, Point(step\_x, 0.5 \* height + step\_y), Point(step\_x + 0.75 \* width, step\_y));

g->DrawLine(pen\_lines, Point(step\_x, 0.5 \* height + step\_y), Point(step\_x + 0.75 \* width, height + step\_y));

g->DrawLine(pen\_lines, Point(step\_x, 0.5 \* height + step\_y), Point(step\_x + 1.75 \* width / 2, step\_y + 0.25 \* height));

g->DrawLine(pen\_lines, Point(step\_x, 0.5 \* height + step\_y), Point(step\_x + 1.75 \* width / 2, step\_y + 0.75 \* height));

}

else if (y % 3 == 2) {

g->DrawLine(pen\_lines, Point(step\_x + 0.75 \* width, step\_y), Point(step\_x + 0.25 \* width, step\_y + height));

g->DrawLine(pen\_lines, Point(step\_x + 0.75 \* width, step\_y), Point(step\_x + 0.75 \* width, step\_y + height));

g->DrawLine(pen\_lines, Point(step\_x + 0.75 \* width, step\_y), Point(step\_x, 0.5 \* height + step\_y));

g->DrawLine(pen\_lines, Point(step\_x + 0.75 \* width, step\_y), Point(step\_x + 0.5 \* width, step\_y + height));

g->DrawLine(pen\_lines, Point(step\_x + 0.75 \* width, step\_y), Point(step\_x + 0.25 \* width / 2, 0.75 \* height + step\_y));

}

else {

g->DrawLine(pen\_lines, Point(step\_x + 0.75 \* width, step\_y + height), Point(step\_x + 0.25 \* width, step\_y));

g->DrawLine(pen\_lines, Point(step\_x + 0.75 \* width, step\_y + height), Point(step\_x + 0.75 \* width, step\_y));

g->DrawLine(pen\_lines, Point(step\_x + 0.75 \* width, step\_y + height), Point(step\_x, step\_y + 0.5 \* height));

g->DrawLine(pen\_lines, Point(step\_x + 0.75 \* width, step\_y + height), Point(step\_x + 0.5 \* width, step\_y));

g->DrawLine(pen\_lines, Point(step\_x + 0.75 \* width, step\_y + height), Point(step\_x + 0.25 \* width / 2, step\_y + 0.25 \* height));

}

}

step\_x += 1.5 \* width;

}

modifier\_i ^= 1;

step\_x = zero\_point\_x;

step\_x += modifier\_i \* width \* 0.75;

step\_y += 0.5 \* height;

}

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

draw\_pattern();

}

private: System::Void checkBox1\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

draw\_grid ^= 1;

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

ColorDialog^ colorDialog = gcnew ColorDialog();

if (colorDialog->ShowDialog() == System::Windows::Forms::DialogResult::OK)

{

Color selectedColor = colorDialog->Color;

pattern\_color = selectedColor;

}

}

private: System::Void button4\_Click(System::Object^ sender, System::EventArgs^ e) {

ColorDialog^ colorDialog = gcnew ColorDialog();

if (colorDialog->ShowDialog() == System::Windows::Forms::DialogResult::OK)

{

Color selectedColor = colorDialog->Color;

background\_color = selectedColor;

}

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

SaveFileDialog^ saveFileDialog1 = gcnew SaveFileDialog();

saveFileDialog1->Filter = "JPEG Image (\*.jpg)|\*.jpg|PNG Image (\*.png)|\*.png|BMP Image (\*.bmp)|\*.bmp";

saveFileDialog1->DefaultExt = "jpg";

if (saveFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

{

pictureBox1->Image->Save(saveFileDialog1->FileName, System::Drawing::Imaging::ImageFormat::Bmp);

}

}

private: System::Void trackBar1\_Scroll(System::Object^ sender, System::EventArgs^ e) {

size = trackBar1->Value;

}

private: System::Void checkBox2\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

draw\_lines ^= 1;

}

private: System::Void button5\_Click(System::Object^ sender, System::EventArgs^ e) {

Color temp\_color = pattern\_color;

pattern\_color = background\_color;

background\_color = temp\_color;

}

private: System::Void trackBar2\_Scroll(System::Object^ sender, System::EventArgs^ e) {

pen\_size = trackBar2->Value;

}

private: System::Void checkBox3\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

random\_fill ^= 1;

}

private: System::Void button6\_Click(System::Object^ sender, System::EventArgs^ e) {

ColorDialog^ colorDialog = gcnew ColorDialog();

if (colorDialog->ShowDialog() == System::Windows::Forms::DialogResult::OK)

{

Color selectedColor = colorDialog->Color;

fill\_color = selectedColor;

}

}

private: System::Void trackBar3\_Scroll(System::Object^ sender, System::EventArgs^ e) {

fill\_frequency = trackBar3->Value;

}

};

}