МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

|  |  |
| --- | --- |
| Институт (факультет) | Институт информационных технологий |
| Кафедра | МПО ЭВМ |

КУРСОВАЯ РАБОТА

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

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

|  |  |
| --- | --- |
|  | Выполнил студент группы 1ПИб-02-2оп-22 |
|  | *группа* |
|  | Направления подготовки (специальности) |
|  | 09.03.04 Программная инженерия |
|  | *шифр, наименование* |
|  | Цветков Антон Сергеевич |
|  | *фамилия, имя, отчество* |
|  |  |
|  | Руководитель |
|  | Пышницкий Константин Михайлович |
|  | *фамилия, имя, отчество* |
|  | Старший преподаватель |
|  | *должность* |
|  | Дата представления работы |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2023 г. |
|  |  |
|  | Заключение о допуске к защите |
|  |  |
|  |  |
|  | Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | *количество баллов* |
|  | Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

2023 год

Оглавление

[Введение 2](#_Toc137006744)

[Описание предметной области 3](#_Toc137006745)

[Описание классов Graphics, Pen и Brush 3](#_Toc137006746)

[1. Класс Graphics 3](#_Toc137006747)

[2. Класс Pen 4](#_Toc137006748)

[3. Класс Brush 5](#_Toc137006749)

[Описание созданного приложения 5](#_Toc137006750)

[1. Постановка задачи 5](#_Toc137006751)

[2. Логическое проектирование – алгоритм работы программы 5](#_Toc137006752)

[3. Физическое проектирование – структуры данных и спецификация функций 7](#_Toc137006753)

[4. Тестирование 7](#_Toc137006754)

[5. Результаты работы 8](#_Toc137006755)

[Заключение 8](#_Toc137006756)

[Источники 9](#_Toc137006757)

[Приложение 1. Техническое задание 10](#_Toc137006758)

[Приложение 2. Руководство пользователя 15](#_Toc137006759)

[Приложение 3. Программный код 19](#_Toc137006760)

Введение

Высокоуровневый язык программирования – средство записи компьютерных программ, обеспечивающее высокую скорость и удобство работы. Его отличительной чертой является абстракция. Другими словами, высокоуровневый язык программирования обеспечивает возможность введения смысловых конструкций, способных коротко описать форматы данных и операции с ними в тех случаях, когда описания на низкоуровневом языке (например, на машинном коде) будут сложными для восприятия и очень длинными. Основным достоинством машинно-независимых языков программирования являются их простота и универсальность. Как следствие, значительно сокращается продолжительность написания кода и отладки. Одна и та же программа может быть выполнена на компьютерах разной архитектуры [1].

Представителем высокоуровневых языков программирования является язык С++.

Язык программирования C++ является мощным инструментом, широко применяемым для разработки различных приложений, включая графические программы. Благодаря своей эффективности, гибкости и высокой производительности, C++ стал предпочтительным выбором для создания программ, работающих с графикой, пользовательскими интерфейсами и другими сложными задачами [2].

Для более подробного ознакомления с темой «Программирование на языке высокого уровня», напишем программу на языке С++, реализующую построение орнамента окружности по окружности.

Описание предметной области

Главная цель данной работы – изучение программирования на высокоуровневом языке С++, в частности, работа с графикой в Windows Forms.

Windows Forms - это набор управляемых библиотек в .NET Framework, предназначенный для разработки многофункциональных клиентских приложений. Это графический API для отображения данных и управления пользовательскими взаимодействиями с более простым развертыванием и повышенной безопасностью в клиентских приложениях [3].

Предметной областью является орнамент окружности по окружности (рис.1).

Окружности строятся на определенном расстоянии друг от друга, их центры находятся на другой окружности.

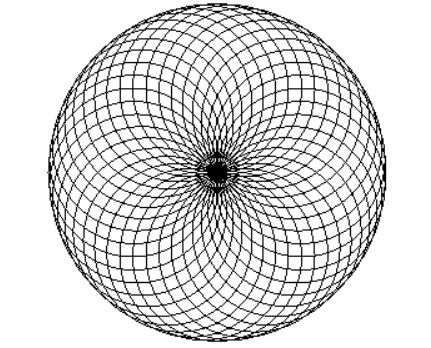


Рис.1. Орнамент окружности по окружности

Описание классов Graphics, Pen и Brush

Классы Graphics, Pen и Brush относятся к библиотеке System.Drawing. Пространство имен System.Drawing обеспечивает доступ к функциональным возможностям графического интерфейса GDI+ [4].

1. Класс Graphics

Данный класс входит в пространство имен System.Drawing, как и большинство классов для работы с графикой. Класс Graphics определяет набор методов для вывода текста, изображений и геометрических фигур [4].

Методов в этом классе огромное количество, поэтому рассмотрим некоторую часть из них в табл. 1:

Таблица 1

Методы класса Graphics

|  |  |
| --- | --- |
| Имя | Описание |
| AddMetafileComment | Добавляет комментарий к текущему объекту Metafile. |
| BeginContainer() | Сохраняет графический контейнер, содержащий текущее состояние данного объекта Graphics, а затем открывает и использует новый графический контейнер. |
| Clear | Очищает всю поверхность рисования и выполняет заливку поверхности указанным цветом фона. |
| CopyFromScreen(Int32, Int32, Int32, Int32, Size) | Выполняет передачу данных о цвете, соответствующих прямоугольной области пикселей, блоками битов с экрана на поверхность рисования объекта Graphics. |
| Dispose | Освобождает все ресурсы, используемые данным объектом Graphics. |
| DrawArc(Pen, Int32, Int32, Int32, Int32, Int32, Int32) | Рисует дугу, которая является частью эллипса, заданного парой координат, шириной и высотой. |
| DrawBezier(Pen, Point, Point, Point, Point) | Рисует кривую Безье, определяемую четырьмя структурами Point. |
| DrawLine(Pen, Int32, Int32, Int32, Int32) | Проводит линию, соединяющую две точки, задаваемые парами координат. |
| FillClosedCurve(Brush, Point[], FillMode) | Заполняет внутреннюю часть замкнутой фундаментальной сплайновой кривой, определяемой массивом структур Point, используя указанный режим заливки. |
| FillEllipse(Brush, Int32, Int32, Int32, Int32) | Заполняет внутреннюю часть эллипса, определяемого ограничивающим прямоугольником, заданным с помощью пары координат, ширины и высоты. |
| FillPath | Заполняет внутреннюю часть объекта GraphicsPath. |
| FillPie(Brush, Rectangle, Single, Single) | Заполняет внутреннюю часть сектора, определяемого эллипсом, который задан структурой RectangleF, и двумя радиальными линиями. |
| FillPie(Brush, Int32, Int32, Int32, Int32, Int32, Int32) | Заполняет внутреннюю часть сектора, определяемого эллипсом, который задан парой координат, шириной, высотой и двумя радиальными линиями. |
| FillPolygon(Brush, Point[]) | Заполняет внутреннюю часть многоугольника, определяемого массивом точек, заданных структурами Point. |
| FillRectangle(Brush, Rectangle) | Заполняет внутреннюю часть прямоугольника, определяемого структурой Rectangle. |

2. Класс Pen

Определяет объект, используемый для рисования прямых линий и кривых. Этот класс не наследуется [4].

Методы представлены в табл. 2:

Таблица 2

Методы класса Pen

|  |  |
| --- | --- |
| Имя | Описание |
| Pen(Color) | Инициализирует новый экземпляр класса Pen с указанным цветом. |
| Pen(Color, Single) | Инициализирует новый экземпляр класса Pen с указанными свойствами Color и Width. (Width - устанавливает ширину пера Pen, в единицах объекта Graphics, используемого для рисования) |

3. Класс Brush

Определяет объекты, которые используются для заливки внутри графических фигур, таких как прямоугольники, эллипсы, круги, многоугольники и дорожки.

Это абстрактный базовый класс, который не может быть реализован. Для создания объекта "кисть" используются классы, производные от Brush, такие как SolidBrush, TextureBrush и LinearGradientBrush [4].

Описание созданного приложения

В данном разделе описаны основные этапы разработки приложения для построения орнамента окружности по окружности.

1. Постановка задачи

Для корректной работы программы требуется выполнить следующие задачи:

* Программа должна получать радиус и количество окружностей;
* программа должна позволять менять масштаб и цвет орнамента;
* программа должна создавать графическое изображение орнамента при нажатии соответствующей кнопки;
* программа должна позволять сохранять в файловом виде изображение при нажатии соответствующей кнопки;
* программа должна иметь возможность повторно создать графическое изображение при изменении параметров.

2. Логическое проектирование – алгоритм работы программы

На вход программа получает радиус и количество окружностей, цвет и масштаб орнамента. Проверяется корректность введенных данных радиуса и количества окружностей. Расчет координат центра изображения. Расчет значения масштабированного радиуса окружностей. Расчет угла построения окружностей, в зависимости от их количества. Итерация для каждой окружности, в которой происходит: расчет координат окружности (на основе центра, радиуса и угла построения), построение окружности, увеличение угла построения.

Формулы координат центра окружностей:

x = centerX + radius \* cos(angle \* Math::PI / 180.0) \* scale

y = centerY + radius \* sin(angle \* Math::PI / 180.0) \* scale

Где: x, y – координаты центра окружностей; centerX, centerY – координаты центра изображения, radius – радиус окружностей, angle – угол для рисования каждой окружности, scale – масштаб.

Алгоритм работы программы представлен на блок-схеме (рис. 2):



Рис. 2. Алгоритм работы программы в блок-схеме

3. Физическое проектирование – структуры данных и спецификация функций

Переменные, примененные для реализации программы, представлены в табл. 3:

Таблица 3

Переменные

|  |  |  |
| --- | --- | --- |
| Наименование | Обозначение | Тип данных |
| Значение радиуса окружностей | radius | float |
| Количество окружностей | numCircles | int |
| Масштаб изображения | scale | float |
| Координата центра изображения по ширине | centerX | float |
| Координата центра изображения по высоте | centerY | float |
| Масштабированный радиус окружностей | scaleRadius | float |
| Текущий угол для построения окружностей | angle | float |
| Приращение угла для построения окружностей | angleIncrement | float |
| Координата центра окружности по ширине | x | float |
| Координата центра окружности по высоте | y | float |

Спецификация функций представлена в табл. 4:

Таблица 4

Спецификация функций

|  |  |  |  |
| --- | --- | --- | --- |
| Имя модуля | Заголовок процедуры или функции | Формальные параметры | Выполняемое действие |
| Project2.sln | button1\_Click | System::Object^ sender, System::EventArgs^ e | Функция считывает параметры, введенные пользователем, определяет корректность введенных данных, создает и отображает на pictureBox1 орнамент |
|  | button2\_Click | System::Object^ sender, System::EventArgs^ e | Функция сохраняет полученное изображение орнамента в файл с именем “ornament.png”. |

4. Тестирование

Результаты тестирования представлены в табл. 5:

Таблица 5

Результаты тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата тестирования | Тестируемый модуль | Кто проводил тестирование | Описание теста | Результат тестирования |
| 15.05.2023 | Project2.sln | Цветков А.С. | Ввод значений радиуса и количества окружностей | Успех |
| 16.05.2023 | Project2.sln | Цветков А.С. | Построение орнамента | Успех |
| 18.05.2023 | Project2.sln | Цветков А.С. | Изменение цвета | Успех |
| 19.05.2023 | Project2.sln | Цветков А.С. | Масштабирование | Успех |
| 23.05.2023 | Project2.sln | Цветков А.С. | Проверка корректности введенных данных | Успех |
| 30.05.2023 | Project2.sln | Цветков А.С. | Проверка всех функций работы программы | Успех |

5. Результаты работы

В результате работы было создано приложение для построения орнамента окружности по окружности в Windows Forms на языке С++. Орнамент строится на основе данных, введенных пользователем.

Программа корректно при вводе данных, не соответствующих предполагаемому формату. Сохраняет изображение орнамента.

Интерфейс не перегружен и понятен (рис. 3).

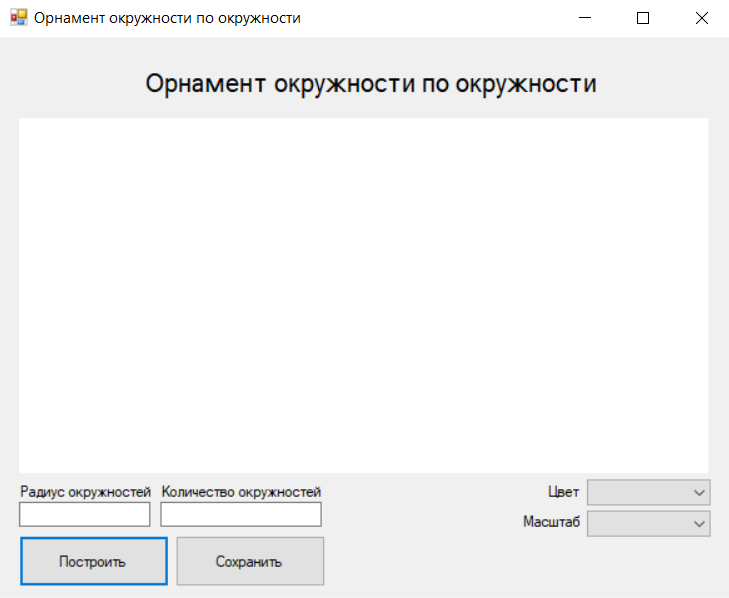


Рис. 3. Интерфейс программы

Заключение

В рамках данного проекта была разработана программа в Windows Forms на языке С++, реализующая графическое изображение орнамента окружности по окружности.

В ходе работы, были получены навыки в работе с Windows Forms, освоены возможности классов Graphics, Pen и Brush.

Поставленные задачи были выполнены в полном объеме.

Источники

* 1. Высокоуровневые языки программирования: <https://gb.ru/blog/vysokourovnevyj-yazyk-programmirovaniya/>. Дата доступа: 14.05.2023
  2. Язык программирования С++: [https://blog.skillfactory.ru/cplus-komu-i-dlya-chego-nuzhen/](https://blog.skillfactory.ru/cplus-komu-i-dlya-chego-nuzhen/%20) Дата доступа: 16.05.2023
  3. Windows Forms: [https://ru.theastrologypage.com/windows-forms](https://ru.theastrologypage.com/windows-forms%20) Дата доступа: 26.05.2023
  4. Основы библиотеки System.Drawing: <https://goo.su/FUUV2HZ>. Дата доступа: 28.05.2023

Приложение 1. Техническое задание

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

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

УТВЕРЖДАЮ

Зав. кафедрой\_\_\_\_\_\_\_\_\_\_\_\_

д.т.н., профессор\_\_\_Ершов Е.В.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_2023г.

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ

Техническое задание на курсовую работу

Листов 5

Руководитель: старший преподаватель

Пышницкий К.М.

Исполнитель: студент гр. 1ПИб-02-2оп-22

Цветков А.С.

2023 год

Введение

Программа для построения орнамента окружности по окружности в Windows Forms рисует изображение по данным, введенным пользователем и сохраняет полученный результат.

Данный проект предназначен для формирования знаний и навыков по дисциплине «Структурное программирование».

1. Основания для разработки

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

Наименование темы разработки: программирование на языке высокого уровня.

2. Назначение разработки

Назначение разработки этой программы состоит в создании приложения для генерации и визуализации орнамента окружности по окружности. Программа предоставляет пользователю возможность вводить значения параметров через графический интерфейс, после чего она создает орнамент и отображает его на экране. Кроме того, программа позволяет сохранить полученный орнамент в файл для дальнейшего использования.

3. Требования к программе

3.1. Требования к функциональным характеристикам

Требования к функциональным характеристикам программы:

* Генерация орнамента: Программа должна позволять пользователю генерировать орнаменты на основе заданных параметров, таких как радиус и количество окружностей, цвет и масштаб.
* Визуализация орнамента: Сгенерированный орнамент должен быть наглядно отображен на экране, чтобы пользователь мог визуально оценить результат.
* Изменение параметров орнамента: Пользователь должен иметь возможность изменять параметры орнамента, такие как радиус и количество окружностей, цвет и масштаб, и видеть соответствующие изменения визуализации.
* Сохранение орнамента: Пользователь должен иметь возможность сохранить сгенерированный орнамент в файле, чтобы иметь возможность использовать его в дальнейшем.
* Обработка ошибок: Программа должна обрабатывать возможные ошибки пользовательского ввода, например, некорректные значения радиуса или количества окружностей, и информировать пользователя об этих ошибках.
* Понятный интерфейс: Интерфейс программы должен быть интуитивно понятным и удобным в использовании, чтобы пользователь мог легко настраивать параметры орнамента и видеть результаты своих действий.
* Поддержка различных цветов и масштабов: Программа должна предоставлять пользователю возможность выбирать цвет орнамента из предустановленного списка. Также пользователь должен иметь возможность выбирать масштаб орнамента из предустановленного списка.

3.2. Требования к надежности

Программа должна проверять пользовательский ввод на предмет некорректных данных, предусмотреть реакции на этот случай.

3.3. Условия эксплуатации

Условия эксплуатации программы зависят от условий эксплуатации персонального компьютера.

3.4. Требования к составу и параметрам технических средств

Минимальные требования к техническим характеристикам компьютера для работы программы:

* процессор с тактовой частотой 1 ГГц;
* оперативная память объемом не менее 1 ГБ;
* жесткий диск с объемом не менее 10 ГБ;
* монитор;
* мышь;
* клавиатура.

3.5. Требования к информационной и программной совместимости

Для обеспечения программной совместимости необходим Visual Studio 2019 ли новее на ОС Windows 7 или выше.

3.6. Требования к маркировке и упаковке

Требования к маркировке и упаковке программы, как правило, не являются применимыми, так как программа представляет собой цифровой продукт, который распространяется в электронном формате.

3.7. Требования к транспортированию и хранению

Нет требований к транспортированию, так как программа не является материальным объектом, который нужно перемещать или хранить в физическом пространстве. Необходимо сохранить программу на внешний носитель, чтобы избежать потери информации.

3.8. Специальные требования

Ограничение возрастного порога с 10 лет.

4. Требования к программной документации

4.1. Содержание расчетно-пояснительной записки

Программная документация должна содержать расчётно-пояснительную записку с содержанием:

Титульный лист

Оглавление

Введение

Описание предметной области

Описание классов Graphics, Pen и Brush

Описание созданного приложения

1. Постановка задачи

2. Логическое проектирование – алгоритм работы программы

3. Физическое проектирование – структуры данных и спецификация функций

4. Тестирование

5. Результаты работы

Заключение

Источники

Приложения

4.2. Требования к оформлению

Программная документация должна удовлетворять следующему оформлению (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.  Файлы предъявляются на компакт-диске: РПЗ с ТЗ; программный код.  Листы и диск в конверте вложены в пластиковую папку скоросшивателя. |
| 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 – выровнены по левому краю, числа, даты – по правому. |

5. Технико-экономические показатели

Технико-экономические показатели к данной программе не предъявляются.

6. Стадии и этапы разработки

Стадии и этапы разработки представлены в табл. П1.2:

Таблица П1.2

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки | Сроки разработки | Результат выполнения | Отметка о выполнении |
| Разработка ТЗ | до 10.05.2023 | Разработанное ТЗ |  |
| Разработка программы | 25.04.2023-06.06.2023 | Разработанный web-сайт |  |
| Разработка руководства пользователя | 01.05.2023-25.05.2023 | Разработанное руководство пользователя |  |
| Разработка РПЗ | 16.04.2023-06.06.2023 | Разработанная РПЗ |  |

7. Порядок контроля и приемки

Порядок контроля и приемки представлен в табл. П1.3:

Таблица П1.3

Порядок контроля и приемки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование  контрольного этапа выполнения курсовой работы | Сроки  контроля | Результат выполнения | Отметка о приемке  результата  контрольного этапа |
| Оформление ТЗ | 10.05.2023 | Оформленное ТЗ |  |
| Разработка программы | 23.05.2023 | Неконечная версия программы |  |
| Оформление руководство пользователя | 25.05.2023 | Оформленное руководство пользователя |  |
| Доработка программы | 06.06.2023 | Конечная версия программы |  |
| Оформление РПЗ | 06.06.2023 | Оформленная РПЗ |  |
| Сдача РПЗ | 07.06.2023 | Оценка за курсовую работу |  |

Приложение 2. Руководство пользователя

1. Общие сведения о программе

Программа предназначена для графического изображения орнамента окружности по окружности (рис. П2.1).

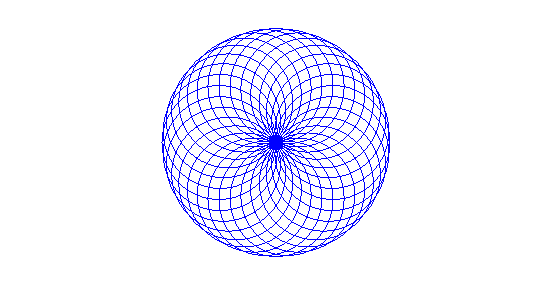


Рис. П2.1. Орнамент окружности по окружности

Программа создает орнамент по параметрам: радиус и количество окружностей, цвет и масштаб изображения.

2. Описание установки

Установка программы не требуется. Чтобы запустить программу, нужно иметь доступ к файлу программы и Visual Studio 2019 или выше.

3. Описание запуска

1. Открыть папку, в которой находится файл программы.

2. Открыть программу.

3. Появляется окно программы, в котором можно работать (рис. П2.2).

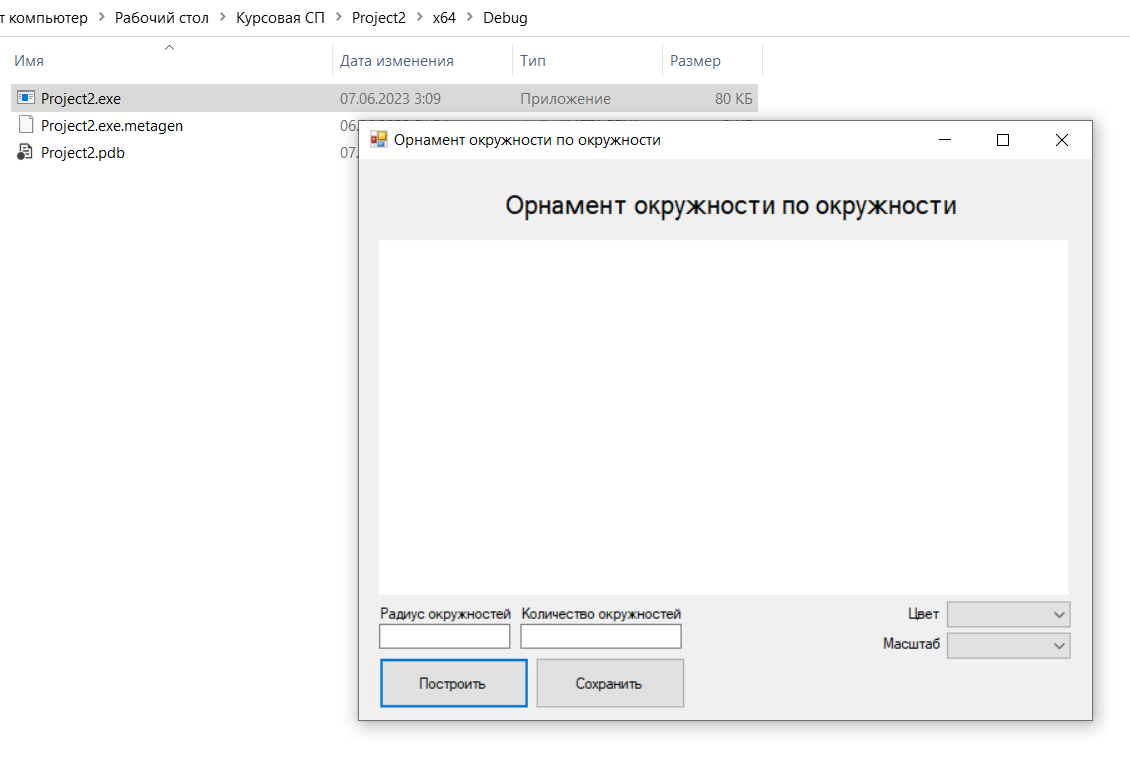


Рис. П2.2. Окно программы

4. Инструкция по работе

Параметры построения орнамента окружности по окружности задаются с помощью элементов интерфейса управления (рис. П2.3).

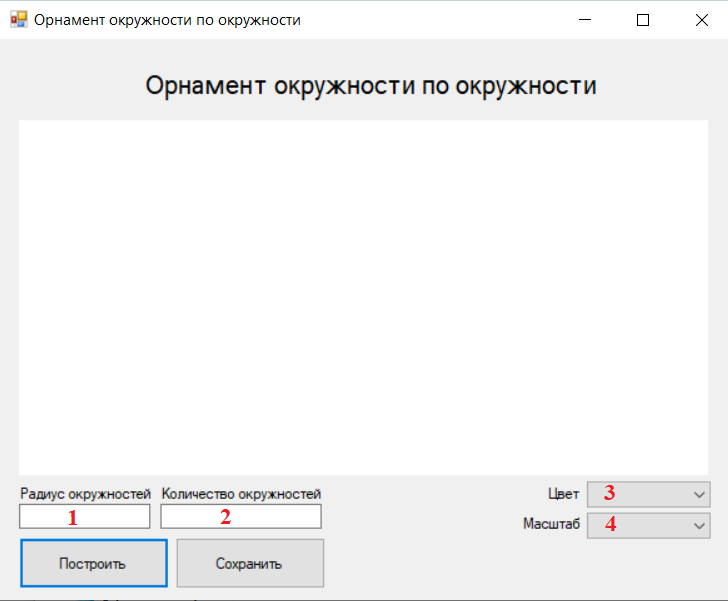


Рис. П2.3. Элементы управления

1. Ввод радиуса окружности. Ввести можно целые или действительные числа от 0.
2. Ввод количества окружностей. Ввести можно только целые числа от 1.
3. Выбор цвета графика. Для выбора цвета, нужно кликнуть на треугольник в правой части поля и выбрать нужный цвет из доступных.
4. Выбор масштаба. Для выбора масштаба, нужно кликнуть на треугольник в правой части поля и выбрать нужный масштаб из доступных.
5. Построение орнамента. Для построения орнамента нужно нажать на кнопку «Построить».
6. Сохранение изображения. Для сохранения изображения нужно нажать на кнопу «Сохранить». Изображения сохранится в формате под именем «ornament.png» в папке, в которой находится приложение.

После заполнения всех элементов приложения построит орнамент (рис. П2.4) и сохранит его изображение (рис. П2.5).

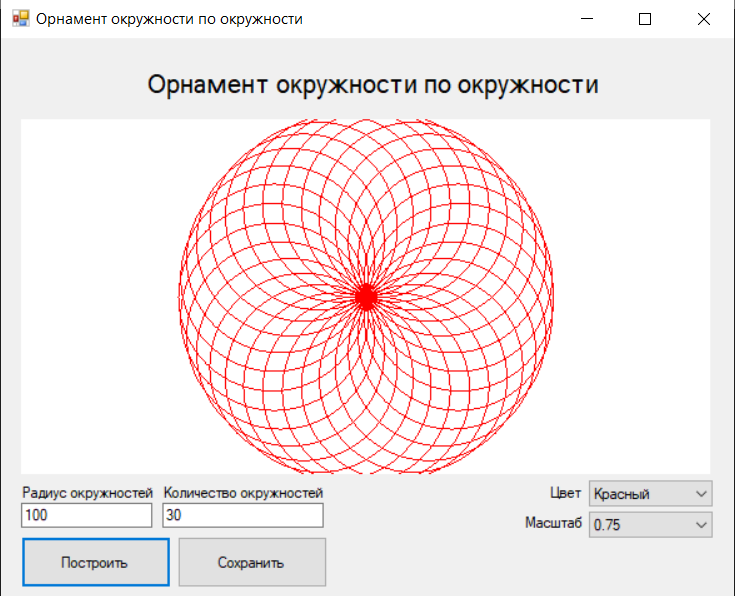


Рис. П2.4. Построенный орнамент

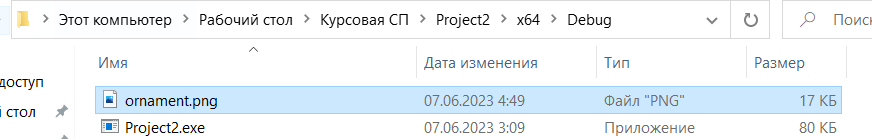


Рис. П2.5. Сохраненное изображение

5. Сообщения пользователю

При некорректном вводе данных в поле радиуса окружностей выводится уведомление, представленное на рис. П2.6:

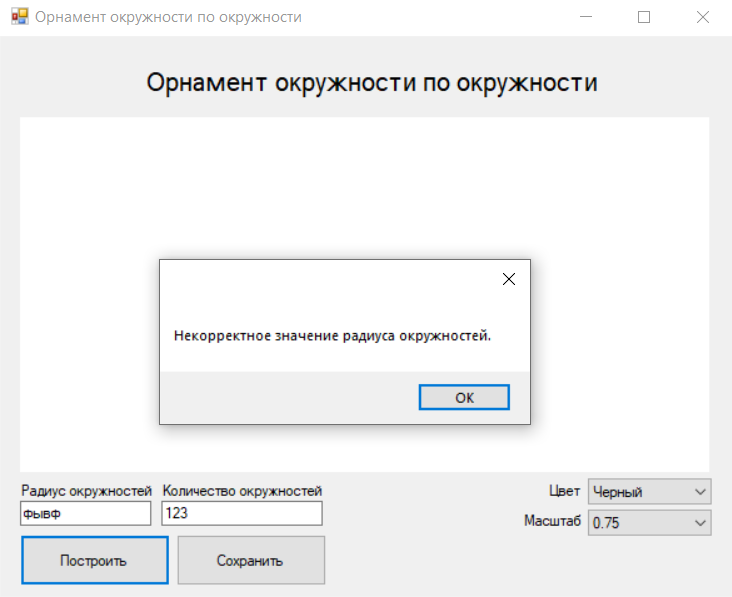


Рис. П2.6. Некорректное значение радиуса окружностей

При некорректном вводе данных в поле количества окружностей выводится уведомление, представленное на рис. П2.7:

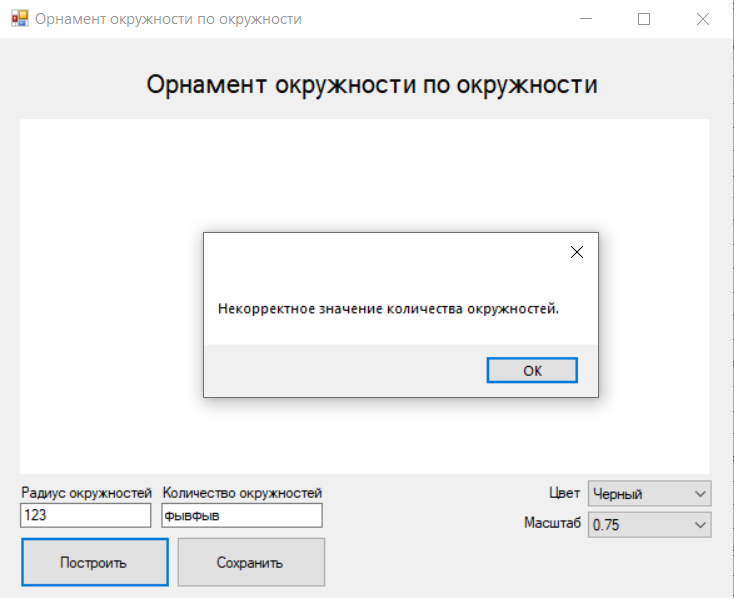


Рис. П2.7. Некорректное значение количества окружностей

При нажатии на кнопку «Сохранить», выводится уведомление, представленное на рис. П2.8:

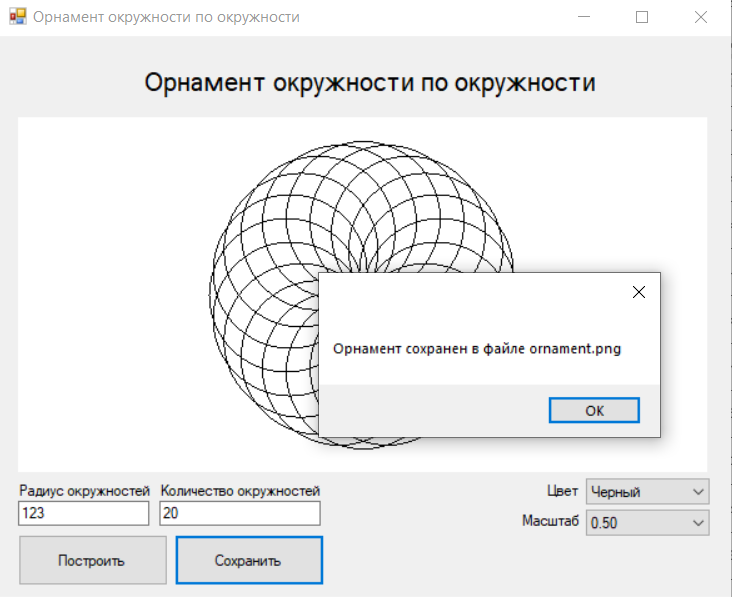


Рис. П2.8. Уведомление о сохранении

Приложение 3. Программный код

Код MyForm.cpp:

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

int main(array<String^>^ args)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Project2::MyForm form;

Application::Run(% form);

}

Код Project2:

#include <Windows.h>

#include <cmath>

namespace Project2 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

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

{

public:

MyForm(void)

{

InitializeComponent();

}

protected:

~MyForm()

{

if (components)

{

delete components;

}

}

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

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

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

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

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

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

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

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::TextBox^ textBox2;

private: System::Windows::Forms::ComboBox^ comboBox1;

private: System::Windows::Forms::ComboBox^ comboBox2;

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

private:

System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code

void InitializeComponent(void)

{

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

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

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

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

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

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

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

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->textBox2 = (gcnew System::Windows::Forms::TextBox());

this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());

this->comboBox2 = (gcnew System::Windows::Forms::ComboBox());

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

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

this->SuspendLayout();

//

// pictureBox1

//

this->pictureBox1->BackColor = System::Drawing::SystemColors::ControlLightLight;

this->pictureBox1->Location = System::Drawing::Point(22, 80);

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

this->pictureBox1->Size = System::Drawing::Size(735, 350);

this->pictureBox1->TabIndex = 0;

this->pictureBox1->TabStop = false;

//

// button1

//

this->button1->Location = System::Drawing::Point(22, 491);

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

this->button1->Size = System::Drawing::Size(160, 50);

this->button1->TabIndex = 1;

this->button1->Text = L"Построить";

this->button1->UseVisualStyleBackColor = true;

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

//

// button2

//

this->button2->Location = System::Drawing::Point(188, 491);

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

this->button2->Size = System::Drawing::Size(160, 50);

this->button2->TabIndex = 2;

this->button2->Text = L"Сохранить";

this->button2->UseVisualStyleBackColor = true;

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

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(19, 439);

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

this->label1->Size = System::Drawing::Size(144, 16);

this->label1->TabIndex = 3;

this->label1->Text = L"Радиус окружностей";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(169, 439);

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

this->label2->Size = System::Drawing::Size(174, 16);

this->label2->TabIndex = 4;

this->label2->Text = L"Количество окружностей";

//

// label3

//

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(581, 439);

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

this->label3->Size = System::Drawing::Size(39, 16);

this->label3->TabIndex = 5;

this->label3->Text = L"Цвет";

//

// label4

//

this->label4->AutoSize = true;

this->label4->Location = System::Drawing::Point(555, 469);

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

this->label4->Size = System::Drawing::Size(65, 16);

this->label4->TabIndex = 6;

this->label4->Text = L"Масштаб";

//

// textBox1

//

this->textBox1->Location = System::Drawing::Point(22, 458);

this->textBox1->Name = L"textBox1";

this->textBox1->Size = System::Drawing::Size(139, 22);

this->textBox1->TabIndex = 7;

//

// textBox2

//

this->textBox2->Location = System::Drawing::Point(172, 458);

this->textBox2->Name = L"textBox2";

this->textBox2->Size = System::Drawing::Size(171, 22);

this->textBox2->TabIndex = 8;

//

// comboBox1

//

this->comboBox1->DropDownStyle = System::Windows::Forms::ComboBoxStyle::DropDownList;

this->comboBox1->FormattingEnabled = true;

this->comboBox1->Items->AddRange(gcnew cli::array< System::Object^ >(4) { L"Черный", L"Красный", L"Зеленый", L"Синий" });

this->comboBox1->Location = System::Drawing::Point(626, 436);

this->comboBox1->Name = L"comboBox1";

this->comboBox1->Size = System::Drawing::Size(131, 24);

this->comboBox1->TabIndex = 9;

//

// comboBox2

//

this->comboBox2->DropDownStyle = System::Windows::Forms::ComboBoxStyle::DropDownList;

this->comboBox2->FormattingEnabled = true;

this->comboBox2->Items->AddRange(gcnew cli::array< System::Object^ >(8) {

L"0.25", L"0.50", L"0.75", L"1.00", L"1.25", L"1.50",

L"1.75", L"2.00"

});

this->comboBox2->Location = System::Drawing::Point(626, 466);

this->comboBox2->Name = L"comboBox2";

this->comboBox2->Size = System::Drawing::Size(131, 24);

this->comboBox2->TabIndex = 10;

//

// label5

//

this->label5->AutoSize = true;

this->label5->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 15));

this->label5->Location = System::Drawing::Point(150, 30);

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

this->label5->Size = System::Drawing::Size(456, 29);

this->label5->TabIndex = 11;

this->label5->Text = L"Орнамент окружности по окружности";

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);

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

this->ClientSize = System::Drawing::Size(782, 553);

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

this->Controls->Add(this->comboBox2);

this->Controls->Add(this->comboBox1);

this->Controls->Add(this->textBox2);

this->Controls->Add(this->textBox1);

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

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

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

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

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

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

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

this->Name = L"MyForm";

this->Text = L"Орнамент окружности по окружности";

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

this->ResumeLayout(false);

this->PerformLayout();

}

private:

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

{

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

float radius;

int numCircles;

if (!Single::TryParse(textBox1->Text, radius))

{

MessageBox::Show("Некорректное значение радиуса окружностей.");

return;

}

if (!Int32::TryParse(textBox2->Text, numCircles))

{

MessageBox::Show("Некорректное значение количества окружностей.");

return;

}

radius = System::Convert::ToSingle(textBox1->Text);

numCircles = System::Convert::ToInt32(textBox2->Text);

Color color;

switch (comboBox1->SelectedIndex)

{

case 0:

color = Color::Black;

break;

case 1:

color = Color::Red;

break;

case 2:

color = Color::Green;

break;

case 3:

color = Color::Blue;

break;

default:

color = Color::Black;

break;

}

float scale;

switch (comboBox2->SelectedIndex)

{

case 0:

scale = 0.25;

break;

case 1:

scale = 0.50;

break;

case 2:

scale = 0.75;

break;

case 3:

scale = 1.00;

break;

case 4:

scale = 1.25;

break;

case 5:

scale = 1.50;

break;

case 6:

scale = 1.75;

break;

case 7:

scale = 2.00;

break;

default:

scale = 1.00;

break;

}

Graphics^ g = Graphics::FromImage(bitmap);

g->Clear(Color::White);

float centerX = pictureBox1->Width / 2.0;

float centerY = pictureBox1->Height / 2.0;

float scaledRadius = radius \* scale;

float angle = 0.0;

float angleIncrement = 360.0 / numCircles;

float x, y;

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

{

x = centerX + radius \* cos(angle \* Math::PI / 180.0) \* scale;

y = centerY + radius \* sin(angle \* Math::PI / 180.0) \* scale;

g->DrawEllipse(gcnew Pen(color), x - scaledRadius, y - scaledRadius, scaledRadius \* 2, scaledRadius \* 2);

angle += angleIncrement;

}

pictureBox1->Image = bitmap;

}

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

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

pictureBox1->DrawToBitmap(bitmap, pictureBox1->ClientRectangle);

bitmap->Save("ornament.png");

MessageBox::Show("Орнамент сохранен в файле ornament.png");

}

};

}