СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc185502556)

[1. Постановка задачи и анализ предметной области 4](#_Toc185502557)

[1.1 Постановка задачи 4](#_Toc185502558)

[1.2 Анализ предметной области 4](#_Toc185502559)

[2.Разработка технического задания 5](#_Toc185502560)

[2.1 Основания для разработки приложения 5](#_Toc185502561)

[2.2 Назначение приложения 5](#_Toc185502562)

[2.2.1 Функциональное назначение 5](#_Toc185502563)

[2.2.2 Эксплуатационное назначение 5](#_Toc185502564)

[2.3 требования к приложению 5](#_Toc185502565)

[2.3.1 Требования к функциональным характеристикам 5](#_Toc185502566)

[2.3.2 Требования к надёжности 5](#_Toc185502567)

[2.3.3 Время восстановления после отказа 5](#_Toc185502568)

[2.3.4 Отказы из- за некорректных действий оператора 6](#_Toc185502569)

[2.3.5 Условия эксплуатации 6](#_Toc185502570)

[2.3.6 Требования к видам обслуживания 6](#_Toc185502571)

[2.3.7 Требования к составу и параметрам технических средств 6](#_Toc185502572)

[2.3.8 Требования к информационной и программной совместимости 6](#_Toc185502573)

[2.3.9 Требования к программной документации 7](#_Toc185502574)

[2.3.10 Технико-экономические показатели 7](#_Toc185502575)

[2.4 Паспорт модуля 7](#_Toc185502576)

[2.4.1 Функции программы 7](#_Toc185502577)

[2.4.2 Метрики программы 9](#_Toc185502578)

[3. Проектирование 10](#_Toc185502579)

[3.1 Поведенческие диаграммы 10](#_Toc185502580)

[3.1.1 Диаграмма вариантов использования 10](#_Toc185502581)

[3.1.2 Диаграмма последовательности 10](#_Toc185502582)

[3.1.3 Диаграмма деятельности 11](#_Toc185502583)

[3.1.4 Диаграмма состояний 12](#_Toc185502584)

[3.2 Структурные диаграммы 13](#_Toc185502585)

[3.2.1 Диаграмма классов 13](#_Toc185502586)

[3.2.2 Диаграмма компонентов 14](#_Toc185502587)

[3.2.3 Диаграмма коопераций 14](#_Toc185502588)

[3.2.4 Диаграмма развёртывания 15](#_Toc185502589)

[3.3 Модель С4 15](#_Toc185502590)

[3.3.1 Уровень контекста 15](#_Toc185502591)

[3.3.2 Уровень контейнеров 16](#_Toc185502592)

[3.3.3 Уровень компонентов 17](#_Toc185502593)

[3.3.4 Уровень кода 17](#_Toc185502594)

[4. Тестирование программного обеспечения 19](#_Toc185502595)

[4.1 Объект испытаний 19](#_Toc185502596)

[4.2 Цель испытаний 19](#_Toc185502597)

[4.2.1 Основание для проведения испытаний 19](#_Toc185502598)

[4.2.2 Место и продолжительность испытаний 19](#_Toc185502599)

[4.2.3 Перечень документов, предъявляемых на испытании 19](#_Toc185502600)

[4.2.4 Объём испытаний 19](#_Toc185502601)

[4.3 Требования к программе 19](#_Toc185502602)

[4.3.1 Требования к программной документации 19](#_Toc185502603)

[4.3.2 Технические средства, используемые во время испытаний 20](#_Toc185502604)

[4.4 Результаты предварительных испытаний 20](#_Toc185502605)

[ЗАКЛЮЧЕНИЕ 23](#_Toc185502606)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 24](#_Toc185502607)

[ПРИЛОЖЕНИЕ 25](#_Toc185502608)

# ВВЕДЕНИЕ

Данный отчет отражает результаты прохождения практики. Цель практики заключалась в выборке требований к программному обеспечению, использование методов для получения кода с заданной функциональностью и степенью качества, планирование и проведение тестирования. В отчете подробно описаны выполненные задачи, приобретенные навыки и полученные результаты.

# 1. Постановка задачи и анализ предметной области

# 1.1 Постановка задачи

Цель практики заключается в выборке требований к программному обеспечению, использованию методов для получения кода с заданной функциональностью и степенью качества, планирование и проведение тестирования. Для достижения этих целей необходимо решить конкретные задачи:

- Разработать техническое задание

- Разработать приложение

- Составить диаграммы

- Провести тестирование ПО

# 1.2 Анализ предметной области

Целью данной практики является реализация программы, которое предназначено для вычисления периметра фигур.

Периметр – это сумма длин всех сторон геометрической фигуры. Проще говоря, это расстояние по контуру фигуры. Понятие периметра применяется к различным двумерным фигурам, но расчёт его несколько отличается в зависимости от формы.

Периметр находит широкое применение в различных областях, включая:

- Строительство и архитектура;

- Ландшафтный дизайн;

- Геодезия и землеустройство;

- Математика;

- Картография.

В общем, периметр – это фундаментальное понятие в геометрии и имеет множество практических применений в реальной жизни.

# 2.Разработка технического задания

# 2.1 Основания для разработки приложения

Основание: задание на практику.

# 2.2 Назначение приложения

# 2.2.1 Функциональное назначение

Функциональное назначением программы является инструментом, который может быть полезен как в учебных, так и в профессиональных целях, способствуя более глубокому пониманию геометрии и улучшению практических навыков.

# 2.2.2 Эксплуатационное назначение

Не имеет

# 2.3 требования к приложению

# 2.3.1 Требования к функциональным характеристикам

Программа должна обеспечивать правильное вычисление периметра в зависимости от выбранной фигуры и вводимых значений.

# 2.3.2 Требования к надёжности

Требования к обеспечению надежного функционирования программы являются:

- Защита от некорректного ввода данных;

- обработка отрицательных и 0 значений;

- правильность вычисления.

# 2.3.3 Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать 10 минут при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

# 2.3.4 Отказы из- за некорректных действий оператора

Отказы программы возможны вследствие некорректных действий оператора (пользователя) при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу пользователя без предоставления ему административных привилегий.

# 2.3.5 Условия эксплуатации

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

# 2.3.6 Требования к видам обслуживания

Программа не требует проведения каких-либо обслуживания.

# 2.3.7 Требования к составу и параметрам технических средств

В состав технических средств должен входить персональный компьютер , включающий в себя:

**- Процессор:** Intel Celeron G1610 2.6 ГГц и более

**- ОЗУ:** 2 Гб и более

- **Необходимо место на жестком диске**: 300 Мб

**- Видеоадаптер:** NVIDIA Geforce 9600 GT и лучше

- **Экран:** 1028 x 720

# 2.3.8 Требования к информационной и программной совместимости

Требования к информационным структурам и методам решения

Информационная структура должна включать отображение входных данных и результата на экране

Требования к исходным кодам и языкам программирования

Исходные коды программы должны быть реализованы на языке C#. В качестве интегрированной среды разработки программы должна быть использована среда Microsoft Studio 2022.

Требования к программным средствам, используемым программой

Системные программные средства, используемые программой, должны

быть представлены лицензионной локализованной версией операционной системы Windows 7, 8, 8.1, 10 с платформой .Net 8.0.

Требования к защите информации и программ

Требования к защите информации и программ не предъявляются.

**Специальные требования**

Программа должна обеспечивать взаимодействие с пользователем (оператором) посредством графического пользовательского интерфейса, разработанного согласно рекомендациям лекции.

# 2.3.9 Требования к программной документации

Состав программной документации должен включать в себя:

- Техническое задание на разработку;

- Руководство оператора;

- Программа, методика и результаты испытаний.

# 2.3.10 Технико-экономические показатели

Коммерческое использование не предусмотрено.

# 2.4 Паспорт модуля

# 2.4.1 Функции программы

Данная программа предназначена для вычисления периметра фигуры.

По нажатию кнопки «Решить» происходит проверка правильности ввода значений и если значения введены правильно, то происходит вычисление и вывод периметра данный процесс показан на рисунке 1.

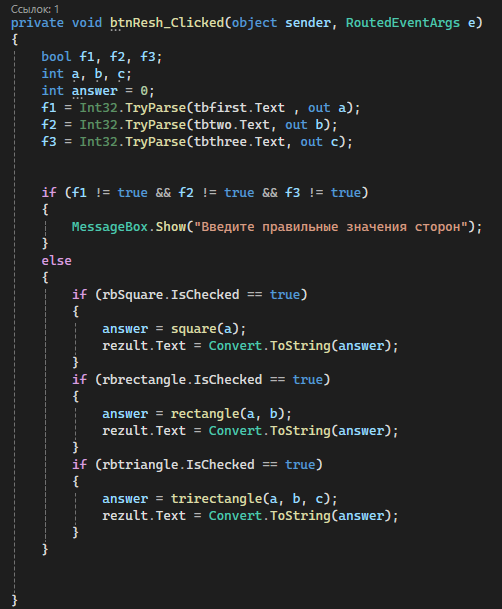


Рисунок 1 – Нажатие кнопки «Решить»

Для вычисления периметра квадрата используется функция «square» показанная на рисунке 2.

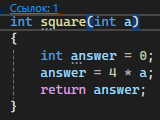


Рисунок 2 – Функция для вычисления периметра квадрата.

Для вычисления периметра прямоугольника используется функция «rectangle» показанная на рисунке 3.

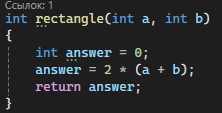


Рисунок 3 – Функция для вычисления периметра прямоугольника.

Для вычисления периметра треугольника используется функция «trirectangle» показанная на рисунке 4.

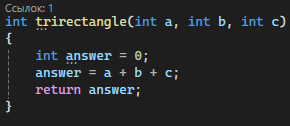


Рисунок 4 – Функция для вычисления периметра треугольника.

При нажатии кнопки «выход» происходит закрытие приложения, это показано на рисунке 5.

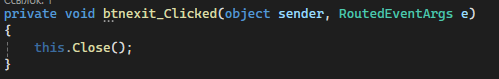


Рисунок 5 – нажатие кнопки «Выход».

При нажатии кнопки «О программе» выводится информация о программе, это показано на рисунке 6.

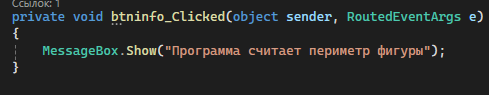


Рисунок 6 – нажатие кнопки «О программе».

# 2.4.2 Метрики программы

Для анализа правильности написания кода на рисунке 7 показаны метрики кода.



Рисунок 7 – Метрики кода.

# 3. Проектирование

# 3.1 Поведенческие диаграммы

# 3.1.1 Диаграмма вариантов использования

ДВИ — это набор событий, которые происходят, когда пользователь использует систему для завершения процесса. Оператор определяется как кто-либо или что-либо, взаимодействующее с системой из-за пределов системы. Таким образом, диаграмма вариантов использования визуально описывает этот набор последовательностей и представляет функциональные требования системы, диаграмма показана на рисунке 8.

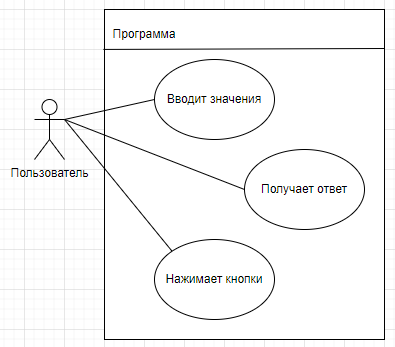


Рисунок 8 – Диаграмма вариантов использования.

# 3.1.2 Диаграмма последовательности

Диаграмма последовательности – это UML-диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта и взаимодействие актеров информационной системы в рамках прецедента, диаграмма показана на рисунке 9.

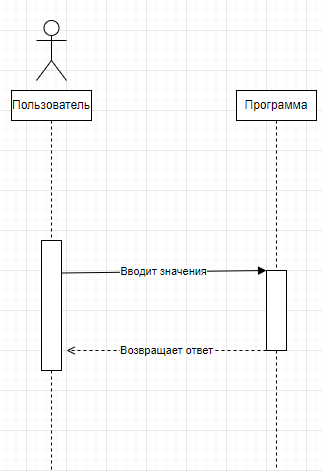


Рисунок 9 – Диаграмма последовательности

# 3.1.3 Диаграмма деятельности

Диаграмма деятельности или активности – это поведенческая диаграмма, позволяющая разделить действия и их последовательность по ролям. Данная графическая структура используется в анализе и оптимизации различных бизнес-процессов, диаграмма показана на рисунке 10.

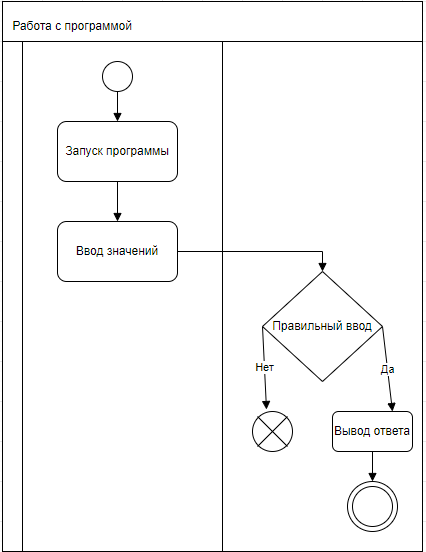


Рисунок 10 – Диаграмма деятельности

# 3.1.4 Диаграмма состояний

Диаграмма состояний – это еще одна поведенческая диаграмма, которая описывает все возможные состояния системы, объекта или объекта внутри системы. Ее главная цель – графически отобразить все этапы различных систем и ее участников, диаграмма показана на рисунке 11.

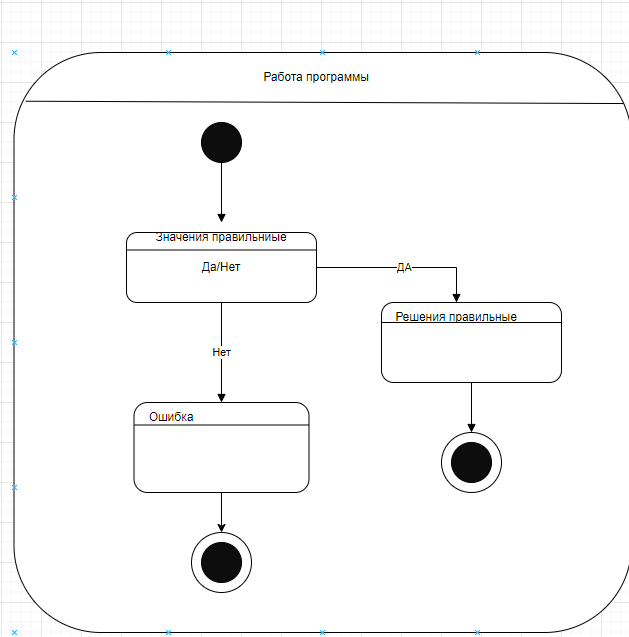


Рисунок 11 – Диаграмма состояний

# 3.2 Структурные диаграммы

# 3.2.1 Диаграмма классов

Диаграмма классов в языке моделирования UML относится к структурному типу диаграмм, используется для визуализации структуры классов в системе, атрибутов, методов, интерфейсов и отношений между ними. Применяется при проектировании архитектуры, документировании системы, уточнении требований, а также для поддержки системы, диаграмма показана на рисунке 12.

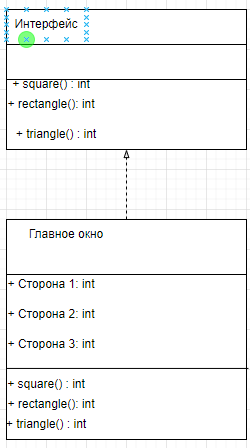
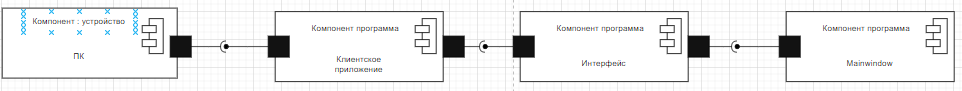


Рисунок 12 – диаграмма классов

# 3.2.2 Диаграмма компонентов

Диаграмма компонентов – это одна из структурных диаграмм, которая описывает структуру системы, отражая взаимосвязи между ее компонентами, диаграмма показана на рисунке 13.

Рисунок 13 – диаграмма компонентов

# 3.2.3 Диаграмма коопераций

Диаграмма кооперации – структурная диаграмма, предназначенная для описания поведения системы на уровне самостоятельных объектов. Эти объекты могут обмениваться между собой сообщениями, чтобы достичь поставленной задачи, диаграмма показана на рисунке 14.

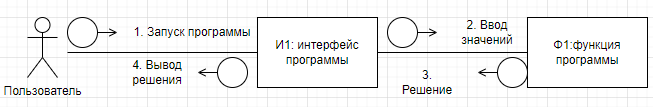


Рисунок 14 – диаграмма коопераций

# 3.2.4 Диаграмма развёртывания

Диаграмма развертывания – это представление архитектуры исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, диаграмма показана на рисунке 15.

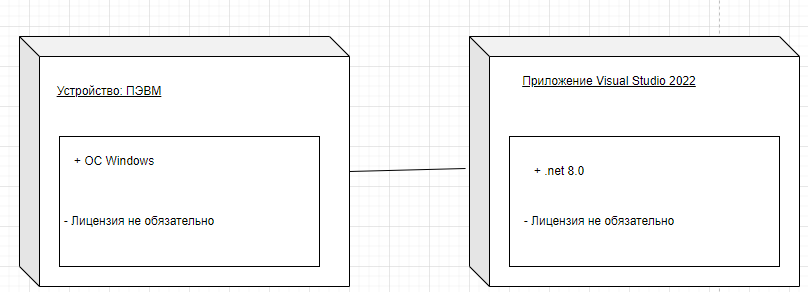


Рисунок 15 – Диаграмма развёртывания

# 3.3 Модель С4

Модель С4 (от англ. С4 model, Context-Container-Component-Code model, модель «контекст-контейнер-компонент-код») - достаточно новая и простая архитектурная модель для проектирования системы. Работает по принципу масштабирования картинки. То есть, увеличиваем масштаб - увеличиваем детализацию схемы.

# 3.3.1 Уровень контекста

Этот этап «поверхностно» описывает систему для пользователя, которому не нужно вдаваться в технические подробности системы, показан на рисунке 16.



Рисунок 16 – Уровень контекста

# 3.3.2 Уровень контейнеров

Следующий уровень необходим для описания системы с точки зрения архитектуры приложения. Техническая часть при этом снова не затрагивается. Данная часть наиболее практична для написания документации, показан на рисунке 17.



Рисунок 17 – уровень контейнеров

# 3.3.3 Уровень компонентов

Целевая аудитория этого уровня абстракции – программисты и архитекторы. Речь пойдет напрямую о технических характеристиках и рисках, показан на рисунке 18.

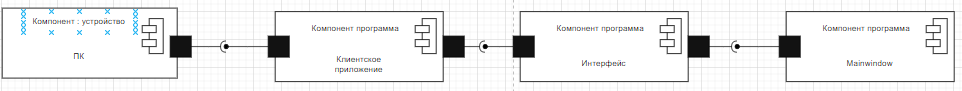


Рисунок 18 – уровень компонентов

# 3.3.4 Уровень кода

Диаграмма кода используется для низкоуровневой детализации системы. На официальной странице модели С4 рекомендуют использовать диаграмму классов, ER-диаграмму или схожие нотации, показан на рисунке 19.

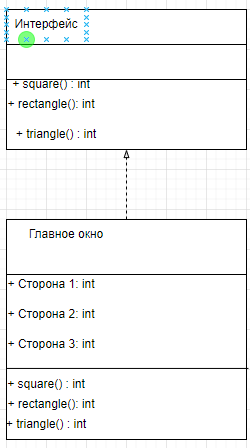


Рисунок 19 – Уровень кода

# 4. Тестирование программного обеспечения

# 4.1 Объект испытаний

Наименование программы «Периметр»

# 4.2 Цель испытаний

Цель проведения испытаний - проверка соответствия характеристик разработанной программы функциональным и отдельным иным видам требований, изложенным в документе Техническое задание.

# 4.2.1 Основание для проведения испытаний

Основание: задание на практику.

# 4.2.2 Место и продолжительность испытаний

Дом, Колледж. Срок 1 неделя.

# 4.2.3 Перечень документов, предъявляемых на испытании

Состав программной документации должен включать в себя:

- Техническое задание;

- Руководство оператора;

- Программа, методика и результаты испытаний.

# 4.2.4 Объём испытаний

Перечень этапов испытаний:

- ознакомительный;

- испытания.

# 4.3 Требования к программе

При проведении испытаний функциональные характеристики программы подлежат проверке на соответствие требованиям, изложенным в п. «Требования к функциональным выполняемых функций» технического задания.

# 4.3.1 Требования к программной документации

Состав программной документации должен включать в себя:

- Техническое задание;

- Руководство оператора;

- Программа, методика и результаты испытаний.

# 4.3.2 Технические средства, используемые во время испытаний

В состав технических средств должен входить персональный компьютер включающий в себя:

**- Процессор:** Intel Celeron G1610 2.6 ГГц и более

**- ОЗУ:** 2 Гб и более

- **Необходимо место на жестком диске**: 300 Мб

**- Видеоадаптер:** NVIDIA Geforce 9600 GT и лучше

- **Экран:** 1028 x 720

# 4.4 Результаты предварительных испытаний

Результаты предварительного тестирования в соответствии с таблицей

1.

Таблица 1 - Результаты тестов.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Тестовая спецификация | | | | | | |
| Разработано: Стрельцов Никита | Начальные установки: нет | | | | | |
| Источник тестовых данных: | Цели: Проверки степени выполнения требований функционального назначения программы | | | | | |
| Test case № | Описание | Шаги теста | Ожидаемые результаты | Реальные результаты | Прошел/ Провалился | Тестер/ Дата |
| 1 | Тестирование вычисления периметра квадрата | Ввести данные и нажать кнопку «Решение» | A = 4, rez =16 | A = 4, rez =16 | Прошел | Владислав Лутов / 19.12.2024 |
| 2 | Тестирование вычисления периметра прямоугольника | Выбрать прямоугольник  Ввести данные и нажать кнопку «Решение» | A = 4, B = 2, rez =12 | A = 4, B = 2, rez =12 | Прошел | Владислав Лутов / 19.12.2024 |
| 3 | Тестирование вычисления периметра треугольника | Выбрать треугольник  Ввести данные и нажать кнопку «Решение» | A = 4, B = 2,  С = 6, rez =12 | A = 4, B = 2,  С=6, rez =12 | Прошел | Владислав Лутов / 19.12.2024 |
| 4 | Тестирование обработки не правильности ввода | Ввести в поле значений «впвапв» | Ошибка | Ошибка | Прошёл | Владислав Лутов / 19.12.2024 |
| 5 | Тестирование кнопки «Выход» | Нажать кнопку «Выход» | Закрытие программы | Закрытие программы | Прошел | Владислав Лутов / 19.12.2024 |

# ЗАКЛЮЧЕНИЕ

При выполнении практики мне удалось достигнуть поставленных целей. Цель практики заключалась в выборке требований к программному обеспечению, использование методов для получения кода с заданной функциональностью и степенью качества, планирование и проведение тестирования.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Шарп, Джон Microsoft Visual C#. Подробное руководство [Текст] : Практическое руководство / Джон Шарп. – 8-е изд., испр. И доп. – СПб.: Издательский дом «Питер», 2017. – 848с.

2 Хейлсберг, А. Язык программирования C#. Классика Computers Science [Текст] : Практическое руководство / А. Хейлсберг [и др.]– 4-е изд., испр. И доп. – СПб.: Издательский дом «Питер», 2014. –816с.

3 Шилдт, Г. C# 4.0 [Текст] : Полное руководство / Г. Шилдт, – М.: ООО

«И.Д. Вильямс», 2015. – 1056с.

4 Ликнесс, Дж. Приложения для Windows 8 на C# и XAML [Текст] : Практическое руководство / Дж. Ликнесс. – СПб.: Издательский дом «Питер», 2013. – 368с.

5 Албахари, Дж C# 7.0. Карманный справочник. Скорая помощь для про- граммистов на C# 7.0 [Текст] : Практическое руководство / Дж. Алба- хари, Б. Албахари – СПб.: Издательство «Альфа-Книга», 2017. – 224с.

# ПРИЛОЖЕНИЕ

HAML Программы

Title="MainWindow" Height="450" Width="800">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition Height="50"/>

<RowDefinition Height="50"/>

<RowDefinition Height="50"/>

<RowDefinition Height="50"/>

<RowDefinition Height="50"/>

<RowDefinition Height="50"/>

<RowDefinition Height="50"/>

<RowDefinition Height="50"/>

</Grid.RowDefinitions>

<Menu VerticalAlignment="Top" Grid.ColumnSpan="6" Height="20">

<MenuItem Header="Выбор фигуры">

<RadioButton x:Name="rbSquare" Content="Квадрат" IsChecked="True" Checked="rblol\_Checked"/>

<RadioButton x:Name="rbrectangle" Content="Прямоугольник" Checked="rblol\_Checked" />

<RadioButton x:Name="rbtriangle" Content="Треугольник" Checked="rblol\_Checked" />

</MenuItem>

<MenuItem Header="О программе" Click="btninfo\_Clicked"/>

<MenuItem Header="Выход" Click="btnexit\_Clicked"/>

</Menu>

<TextBox x:Name="tbfirst" Grid.Row="2" Grid.Column="1" ToolTip="Введите значение 1 стороны"/>

<TextBox x:Name="tbtwo" Grid.Row="3" Grid.Column="1" ToolTip="Введите значение 2 стороны"/>

<TextBox x:Name="tbthree" Grid.Row="4" Grid.Column="1" ToolTip="Введите значение 3 стороны"/>

<TextBox x:Name="rezult" Grid.Row="5" Grid.Column="2" ToolTip="Периметр фигуры" IsReadOnly="True"/>

<Button Grid.Column="1" Grid.Row="7" Content="Решение" Click="btnResh\_Clicked"/>

</Grid>

Код программы

private void btnResh\_Clicked(object sender, RoutedEventArgs e)

{

bool f1, f2, f3;

int a, b, c;

int answer = 0;

f1 = Int32.TryParse(tbfirst.Text , out a);

f2 = Int32.TryParse(tbtwo.Text, out b);

f3 = Int32.TryParse(tbthree.Text, out c);

if (f1 != true && f2 != true && f3 != true)

{

MessageBox.Show("Введите правильные значения сторон");

}

else

{

if (rbSquare.IsChecked == true)

{

answer = square(a);

rezult.Text = Convert.ToString(answer);

}

if (rbrectangle.IsChecked == true)

{

answer = rectangle(a, b);

rezult.Text = Convert.ToString(answer);

}

if (rbtriangle.IsChecked == true)

{

answer = trirectangle(a, b, c);

rezult.Text = Convert.ToString(answer);

}

}

}

int square(int a)

{

int answer = 0;

answer = 4 \* a;

return answer;

}

int rectangle(int a, int b)

{

int answer = 0;

answer = 2 \* (a + b);

return answer;

}

int trirectangle(int a, int b, int c)

{

int answer = 0;

answer = a + b + c;

return answer;

}

private void btninfo\_Clicked(object sender, RoutedEventArgs e)

{

MessageBox.Show("Программа считает периметр фигуры");

}

private void btnexit\_Clicked(object sender, RoutedEventArgs e)

{

this.Close();

}