Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Тульский государственный педагогический университет им. Л.Н. Толстого»
ФГБОУ ВО «ТГПУ им. Л.Н. Толстого»

УТВЕРЖДЕНА

на заседании Ученого совета университета

протокол № 2 от 11.02.2016 г.

Рабочая программа дисциплины «Объектно-ориентированное программирование»

Трудоемкость: 3 зачетные единицы

Квалификация (степень) выпускника: бакалавр

Форма обучения: очная

Год начала подготовки: 2014

Рассмотрена на заседании кафедры информатики и информационных технологий
протокол № 03 от «18» ноября 2015 г.

Заведующий кафедрой Якушин А.В.

Одобрена на заседании Ученого совета факультета математики, физики и информатики
протокол № 5 от 17.12.2015 г.

Декан факультета Реброва И.Ю

СОДЕРЖАНИЕ

1. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы ... 3
2. Место дисциплины в структуре ООП бакалавриата ... 3
3. Объем дисциплины и виды учебной работы ... 4
4. Содержание дисциплины, структурированное по темам (разделам) с указанием отведенного на них количества академических или астрономических часов и видов учебных занятий ... 4
5. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине ... 6
6. Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине ... 6
6.1. Перечень компетенций с указанием этапов их формирования в процессе освоения образовательной программы ... 6
6.2. Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания ... 6
6.3. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы ... 7
6.4. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций ... 22
7. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины ... 23
7.1. Основная литература ... 23
7.2. Дополнительная литература ... 23
8. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины ... 23
9. Методические указания для обучающихся по освоению дисциплины ... 23
10. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень программного обеспечения и информационных справочных систем ... 25
11. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине ... 26
12. Аннотация рабочей программы дисциплины. ... 26
13. Лист регистрации изменений к рабочей программе дисциплины ... 28

1. ПЕРЕЧЕНЬ ПЛАНИРУЕМЫХ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ, СООТНЕСЕННЫХ С ПЛАНИРУЕМЫМИ РЕЗУЛЬТАТАМИ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Достижение планируемых результатов обучения, соотнесенных с общими целями и задачами ОПОП, является целью освоения дисциплины.

Планируемые результаты освоения образовательной программы (код и название компетенции)	Планируемые результаты обучения	Этапы формирования компетенции в процессе освоения образовательной программы
способность использовать знания основных концептуальных положений функционального, логического, объектно-ориентированного и визуального направлений программирования, методов, способов и средств разработки программ в рамках этих направлений (ОПК-7)	**Выпускник знает:** о стилях программирования, об объектно-ориентированном программировании; о методах проектирования объектно-ориентированных программ; об объектно-ориентированных языках программирования и визуальном программировании; понятие класса и объекта, основные принципы объектно-ориентированного программирования; принципы построения классов, критерии проверки правильности построения классовосновные тенденции в области развития технологий объектно-ориентированного программирования. **Умеет:** уметь использовать современные методы объектно-ориентированного программирования при кодировании программных систем разного уровня сложности. иметь опыт работы со средой визуального программирования MS Visual Studio, и языком программирования высокого уровня C#. **Владеет** технологией объектно-ориентированной разработки программного обеспечения	2 этап из 2 (4 семестр)

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ООП БАКАЛАВРИАТА

Дисциплина «Объектно-ориентированное программирование» относится к дисциплинам по выбору вариативной части образовательной программы. Изучение данной дисциплины осуществляется в 4 семестре.

Изучение данной дисциплины базируется на освоении студентами дисциплин «Основы программирования», «Методы программирования».

К началу изучения дисциплины студенты должны владеть:

знаниями в области программирования, построения и реализации алгоритмических моделей системных процессов и задач;

умениями решать типовые задачи по программированию с использованием базовых алгоритмических конструкций и подпрограмм

навыками владения методами декомпозиции сложных задач на независимые подзадачи

Дисциплина «Объектно-ориентированное программирование» является базовой для дисциплин «Менеджмент проектов», «Параллельное программирование».

3. ОБЪЕМ ДИСЦИПЛИНЫ И ВИДЫ УЧЕБНОЙ РАБОТЫ

Вид учебной работы	Объем зачетных единиц / часов по формам обучения
Максимальная учебная нагрузка (всего)	108/3
Контактная работа обучающихся с преподавателем (всего)	44
в том числе:	
лекции	16
практические занятия	26
контрольные работы	
другие виды контактной работы (КСРС)	2
Самостоятельная работа студента (всего)	64
в том числе:	
внеаудиторная самостоятельная работа при подготовке к семинарским и/или практическим занятиям	30
выполнение заданий для самостоятельной работы в системе управления обучением MOODLE	30
подготовка к зачету	4
Промежуточная аттестация в форме зачета	

4. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ, СТРУКТУРИРОВАННОЕ ПО ТЕМАМ (РАЗДЕЛАМ) С УКАЗАНИЕМ ОТВЕДЕННОГО НА НИХ КОЛИЧЕСТВА АКАДЕМИЧЕСКИХ ИЛИ АСТРОНОМИЧЕСКИХ ЧАСОВ И ВИДОВ УЧЕБНЫХ ЗАНЯТИЙ

Очная форма обучения

Наименование тем (разделов).	Количество академических или астрономических часов по видам учебных занятий			
	Занятия лекционного типа	Практические занятия	Другие виды учебных занятий	Самостоятельная работа обучающихся
Раздел 1. Объектно-ориентированное визуальное программирование				
Тема 1.1. Теоретические основы разработки классов в среде объектного программирования	2	4		10
Тема 1.2. Проекты C#	2	4		10
Тема 1.3. Компоненты C#	2	4		10
Тема 1.4. Язык C#.	4	4		10
Раздел 2. Разработка приложений для Windows				10
Тема 2.1. Требования к интерфейсу пользователя приложений для Windows	4	6		10
Тема 2.2. Моделирование реальных природных процессов	2	4		10
Контроль самостоятельной работы студентов			2	
Индивидуальные консультации				
Подготовка к зачету				4
Групповые консультации				
ИТОГО	16	26	2	64

Раздел 1. Объектно-ориентированное визуальное программирование

Тема 1.1. Теоретические основы разработки классов в среде объектного программирования Объектно-ориентированное и процедурное программирование. Действие и объект. Программы, сообщения, объекты. Объекты, свойства, методы, события. Классы и наследование, инкапсуляция, полиморфизм. Конструкторы и деструкторы. Основы программирования в среде визуального интерфейса. C# — система быстрой разработки приложений. Интегрированная среда C#. Общий вид окна среды. Главное меню. Структура меню C#. Инспектор объектов, страницы свойств и событий. Различные способы задания свойств (значение, список, диалог). Окно формы. Окно редактора кода. Палитра компонентов. Быстрые кнопки. Управление конфигурациями окон среды. Структура кода модуля.

Тема 1.2. Проекты C# Виды проектов в C#. Организация проекта в C#, основные файлы проектов. Основные проектные операции. Создание и сохранение проекта. Менеджер проекта. Оформление завершенного проекта. Включение в проект новой формы. Создание отдельной копии формы. Просмотр форм и модулей без включения их в проект. Размещение компонентов на форме. Работа с группой компонентов. Инструментальные средства поддержки разработки кода. Применение Code Insight—Знатока Кода. Исследователь кода Code Explorer. Просмотр иерархии классов, модулей и глобальных символов — Object Browser. Получение информации о классах, свойствах, методах из окна Редактора Кода. Навигация в коде, закладки и дополнительные окна редактирования. Отладка приложений. Настройка инструментальной панели. Настройка палитры компонентов. Настройка Редактора Кода. Настройка Code Insight. Настройка Code Explorer и Object Browser. Настройка отладчика.

Тема 1.3.Компоненты C#. Организация библиотеки компонентов. Страницы библиотеки компонентов. Программный доступ к свойствам и методам объектов. Настройка палитры компонентов. Создание и запись в библиотеку шаблонов компонентов и групп компонентов. Компоненты ввода и отображения текстовой, цифровой и иерархической информации. Компоненты выбора из списков. Таблица строк—компонент StringGrid. Ввод и отображение чисел, дат и времени. Секционированное отображение текстов. Компоненты отображения графической информации. Компонент Shape. Построение графиков и диаграмм. Мультимедиа и анимация. Универсальный проигрыватель Media Player Воспроизведение видеоклипов. Кнопки, индикаторы, управляющие элементы. Компоненты — меню. Главное меню — компонент MainMenu. Контекстное всплывающее меню - компонент PopupMenu. Панели и компоненты внешнего оформления. Инструментальные панели. Перестраиваемые панели. Системные диалоги. Диалоги открытия и сохранения файлов. Диалог выбора шрифта. Диалог выбора цвета. Диалоги печати и установки принтера. Диалоги поиска и замены текста. Компоненты организации управления приложением.

Тема 1.4. Язык C#. Синтаксис языка. Компилятор. Файлы проекта C#. Области видимости и время жизни. Константы, переменные, типизированные константы. Переменные. Процедуры и функции. Операции. Операторы. Условные операторы выбора if. Условный оператор множественного выбора case . Операторы цикла for, repeat, while. Прерывание цикла: оператор break, процедуры continue, exit и abort. Исключения. Типы данных в языке C++. Классы. Свойства. Методы и их наследование, полиморфизм. Конструкторы и деструкторы. События. Процедуры и функции C++. Строка описания формата и функция Format. Математические функции. Процедуры и функции преобразования дат и времени. Процедуры и функции файлового ввода/вывода и управления файлами. Процедуры и функции вызова диалоговых окон, воспроизведения звуков.

Раздел 2. Разработка приложений для Windows

Тема 2.1. Требования к интерфейсу пользователя приложений для Windows. Общие рекомендации по разработке графического интерфейса. Многооконные приложения. Меню. Методика проектирования меню и инструментальной панели. Компоновка. Печать из приложения текстов и изображений. Проектирование окон с изменяемыми размерами. Масштабирование компонентов. Обработка событий клавиатуры и мыши. Перетаскивание объектов. Буксировка компонентов в окне приложения. Управление формами. Модальные формы. Графика и муль-

тимедиа Построение графических изображений. Установка и настройка приложения: работа с системным реестром. Автономные приложения и пакеты.

Тема 2.2. Моделирование реальных природных процессов. Молекула газа в закрытом сосуде Броуновское движение .«Равновесие» (второе начало термодинамики). «Равновесие» (второе начало термодинамики). Графические эффекты. Моделирование спрайтов. Организация скроллинга изображения, реализация эффектов затухания, составления изображения из отдельных точек, красивой смены фонового изображения.

5. ПЕРЕЧЕНЬ УЧЕБНО-МЕТОДИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

Преподавание дисциплины предполагает использование следующего учебно-методического обеспечения.

Комплекта мультимедийных презентаций для лекционных занятий.

Теоретического курса и информационных приложений, размещенных в электронной образовательной среде MOODLe.

Комплекса тестовых заданий и заданий для практических занятий, размещенных в электронной образовательной среде MOODLe.

Виды самостоятельной работы обучающихся: выполнение заданий на практические занятия, выполнение индивидуального проектного задания.

При подготовке к занятиям и выполнении самостоятельной работы студентам доступны следующие учебно-методические ресурсы, перечисленные в п.7 рабочей программы, а также электронный учебный ресурс размещенный в среде электронного обучения ТГПУ им. Л.Н. Толстого (http://moodle.tsput.ru)

6. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

6.1. Перечень компетенций с указанием этапов их формирования в процессе освоения образовательной программы

Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы представлен в таблице пункта 1 рабочей программы.

Формирование компетенции «способность использовать знания основных концептуальных положений функционального, логического, объектно-ориентированного и визуального направлений программирования, методов, способов и средств разработки программ в рамках этих направлений» (ОПК-7) осуществляется в течение двух этапов освоения основной образовательной программы.

Первый этап формирования компетенции осуществляется в процессе освоения дисциплины «Дискретная математика».

Второй этап формирования компетенции осуществляется в процессе освоения одной из дисциплин по выбору «Математика в банковской сфере» или «Объектно-ориентированное программирование».

6.2. Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания

Дескриптор компетен-	Показатели оценивания	Критерии оценивания

ций		
Знания	о стилях программирования, об объектно-ориентированном программировании; о методах проектирования объектно-ориентированных программ; об объектно-ориентированных языках программирования и визуальном программировании; понятия класса и объекта, об основных принципах объектно-ориентированного программирования; о принципах построения классов, критериев проверки правильности построения классов основные тенденции в области развития технологий объектно-ориентированного программирования;	Отметка «зачтено» выставляется, если студент в целом за семестр набрал от 41 до 100 баллов (с учетом баллов, набранных на промежуточной аттестации (зачете)). Отметка «не зачтено» выставляется, если студент в целом за семестр набрал менее 41 балла (с учетом баллов, набранных на промежуточной аттестации (зачете)).
Умения	использовать современные методы объектно-ориентированного программирования при кодировании программных систем разного уровня сложности; работать со средой визуального программирования MS Visual Studio и языком программирования высокого уровня C#;	
Навыки и опыт деятельности	владения технологией объектно-ориентированной разработки программного обеспечения.	

Критерии оценивания компетенций формируются на основе балльно-рейтинговой системы с помощью всего комплекса методических материалов, определяющих процедуры оценивания знаний, умений, навыков и опыта деятельности, характеризующих данный этап формирования компетенций.

Баллы, набранные студентом в течение семестра	Баллы за промежуточную аттестацию (зачет)	Общая сумма баллов за модуль в семестр	Отметка
21 – 80	20 – 30	41-100	Зачтено
0 – 20	0 – 20	0 – 40	Не зачтено

Оценка «*зачтено*» ставится, если студент освоил программный материал всех разделов, последователен в изложении программного материала, достаточно последовательно и логически стройно его излагает, умеет увязывать теорию с практикой, успешно прошел текущий контроль успеваемости по дисциплине, продемонстрировал индивидуальные знания, умениями и навыки практической работы.

Оценка «*не зачтено*» ставится, если студент не знает значительной части программного материала, допускает существенные ошибки, непоследователен в его изложении, не прошел текущий контроль успеваемости, не в полной мере владеет необходимыми знаниями, умениями и навыками при выполнении практических заданий, то есть студент не может продолжить обучение без дополнительной подготовки по соответствующей дисциплине.

6.3. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы

Образцы заданий к практическим занятиям:

Цель работы: Изучение контейнеров WPF для управления расположением элементов.

В WPF вводится понятие макет, в котором координаты находятся на втором месте, а расположение макета становится главным. Это дает возможность создавать гибкие интерфейсы, которые не зависят от разрешения монитора или размера окна приложения.

Основная идея окна WPF в том, что оно может содержать только один элемент. Поэтому, чтобы создать окно со сложным интерфейсом из нескольких элементов, нужно поместить в окно контейнер и добавить все нужные элементы в этот контейнер.

Контейнерами WPF макетов панели могут быть:

- **StackPanel** - Места элементов в горизонтальном или вертикальном стеке.
- **WrapPanel** - Места элементов в серии перенесенных строк.
- **DockPanel** - Выравнивание элементов против всего края контейнера.
- **Grid** - Располагает элементы в строках и столбцах в соответствии с невидимой таблицей.

Начнем с создания нового проекта WPF Application. Назовем его WPFLesson1.По умолчанию Visual Studio создает новое окно с названием Window1.xaml и с содержанием:

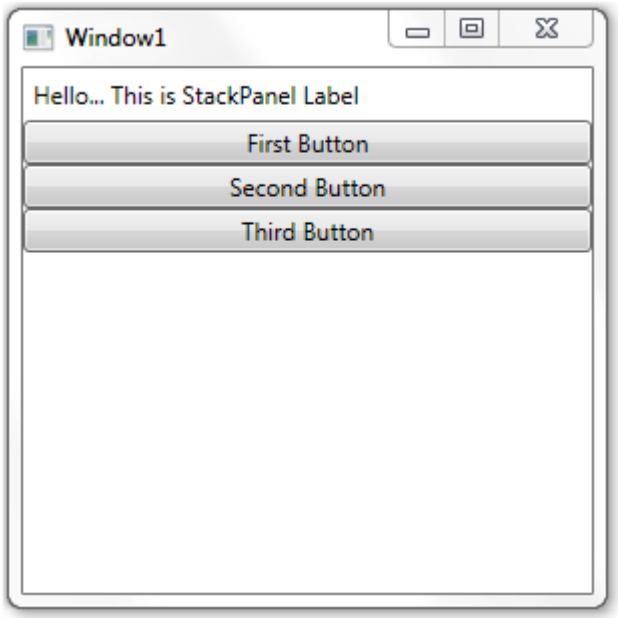
```
<Window x:Class="WPFLesson1.Window1"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   Title="Window1" Height="300" Width="300">
   <Grid>

   </ Grid >
</ Window >
```

Как видим, **Grid** является контейнером «по умолчанию». Изучим возможности каждого контейнера.

Контейнер StackPanel. Начнем с StackPanel. Удалите <Grid> </ Grid> и добавьте StackPanel с одной надписью и тремя кнопками:

```
<Window x:Class="WPFLesson1.Window1"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   Title="Window1" Height="300" Width="300">
   <StackPanel>
      <Label>Hello... This is StackPanel Label</Label>
      <Button>First Button </Button>
      <Button>Second Button </Button>
      <Button>Third Button </Button>
   </StackPanel>
</Window>
```

Запустите проект, он должен выглядеть следующим образом:

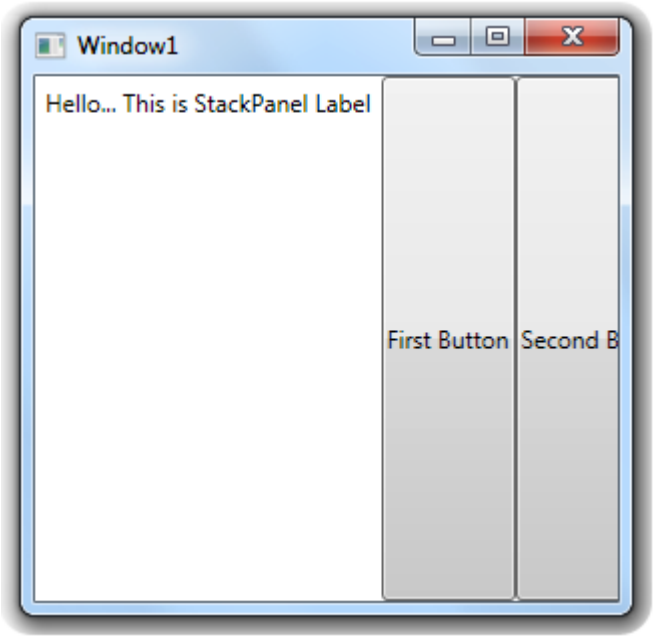


Такой код, размещает надпись и кнопки сверху вниз. Все эти элементы растягиваются на всю ширину панели, а их высота устанавливается автоматически, достаточно для отображения текст внутри них.

Если требуется, чтобы элементы отображались горизонтально, то надо изменить <StackPanel> следующим образом:

```
<StackPanel Orientation="Horizontal">
```

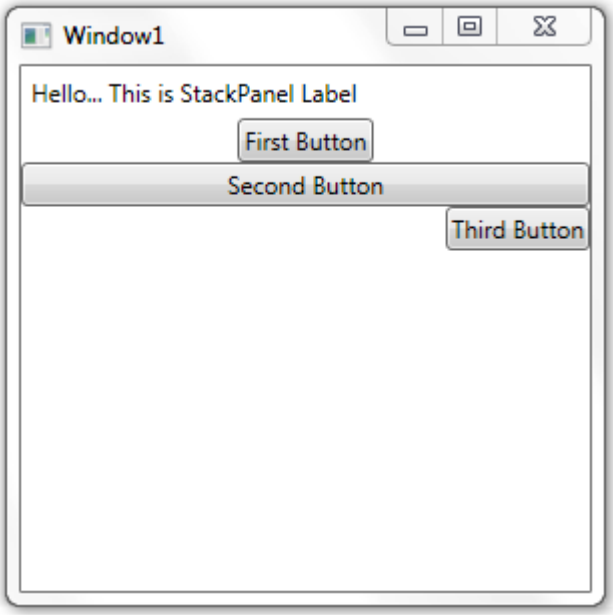
Запустите проект и посмотрите. Это выглядит так:



Как вы видите, высота элемента растягивается на всю панель, а ширина установлена для размещения текста.

Вернемся к тому, как было раньше, и попытаемся сделать некоторые выравнивания. Сделаем подпись слева, первую кнопку- в центре и последнюю кнопку- справа. Используем HorizontalAlignment для этих элементов.

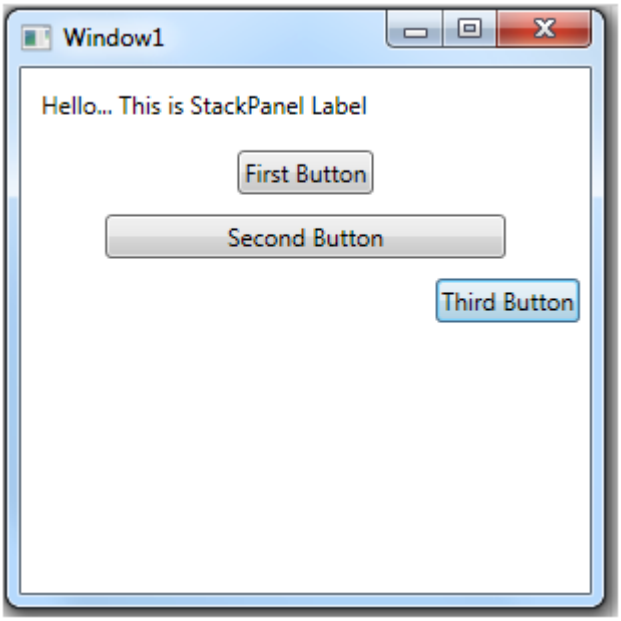
```
<StackPanel>
    <Label HorizontalAlignment="Left">Hello... This is StackPanel Label</Label>
    <Button HorizontalAlignment="Center">First Button </Button>
    <Button>Second Button </Button>
    <Button HorizontalAlignment="Right">Third Button </Button>
</StackPanel>
```



Вы можете видеть результат выше. Важно: у элементов, которые вы выровняли, при изменении размеров формы размеры не изменяются.

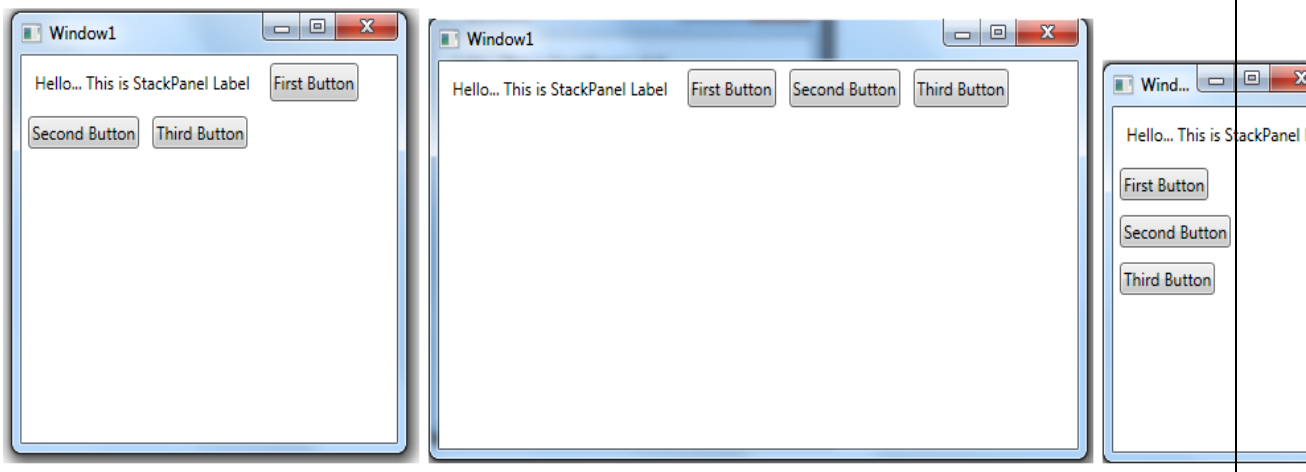
Есть еще две важные вещи: поля и минимальный / максимальный размера. Чтобы обеспечить некоторое пространство между элементами и вокруг них, используют Margin. Установка явных размеров не рекомендуется, но рекомендуется установить диапазон размеров.

```
<StackPanel>
    <Label Margin="5" MaxWidth="400" HorizontalAlignment="Left">Hello... This is StackPanel Label</Label>
    <Button Margin="5" MaxWidth="400" HorizontalAlignment="Center">First Button </Button>
    <Button Margin="5" MaxWidth="200" >Second Button </Button>
    <Button Margin="5" MaxWidth="400" HorizontalAlignment="Right">Third Button </Button>
</StackPanel>
```



Контейнер WrapPanel - организует размещение элементов в доступное пространство (в одну строку или колонку).

```
<WrapPanel>
    <Label Margin="5" MaxWidth="400" HorizontalAlignment="Left">Hello... This is StackPanel Label</Label>
    <Button Margin="5" MaxWidth="400" HorizontalAlignment="Center">First Button </Button>
    <Button Margin="5" MaxWidth="200" >Second Button </Button>
    <Button Margin="5" MaxWidth="400" HorizontalAlignment="Right">Third Button </Button>
</WrapPanel>
```

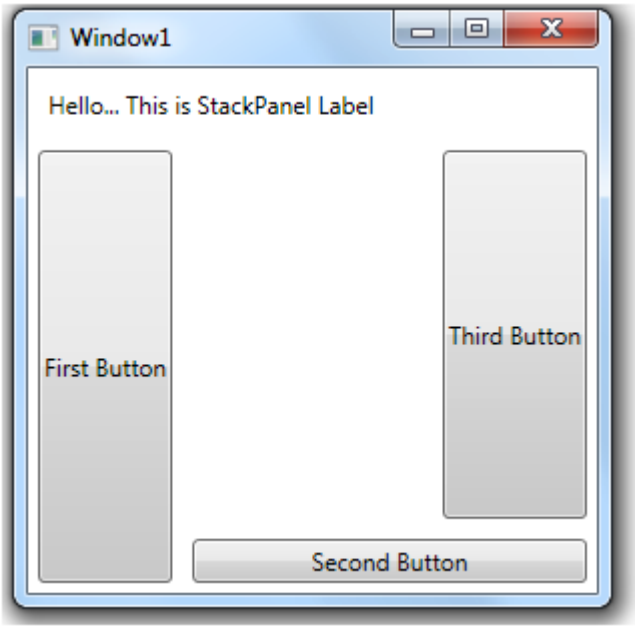


Измените размер окна, чтобы увидеть, как WrapPanel перемещает элементы.

Контейнер DockPanel – используется, если требуется «привязать» элемент к одной стороне панели. Значение свойства привязки (DockPanel.Dock) для каждого элемента можно указывать отдельно.

```
<DockPanel>
```

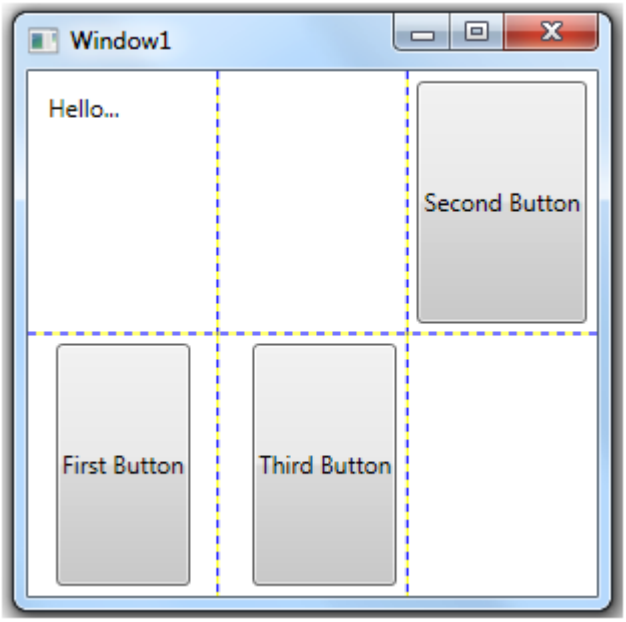
```
        <Label DockPanel.Dock="Top" Margin="5" MaxWidth="400"
HorizontalAlignment="Left">Hello... This is StackPanel Label</Label>
        <Button DockPanel.Dock="Left" Margin="5" MaxWidth="400"
HorizontalAlignment="Center">First Button </Button>
        <Button DockPanel.Dock="Bottom" Margin="5" MaxWidth="200" >Second Button
</Button>
        <Button DockPanel.Dock="Right" Margin="5" MaxWidth="400"
HorizontalAlignment="Right">Third Button </Button>
      </DockPanel>
```



Контейнер Grid - размещает элементы на сетке из строк и столбцов. С его помощью можно сделать почти все, что делает с помощью других контейнеров . При построении подразумевается два этапа: сначала выбрали число строк и столбцов, а затем присвоили каждый элемент в соответствующие строки и столбцы. Обычно линии сетки не видны, но сейчас для наглядности их включим.

```
    <Grid ShowGridLines="True">
        <Grid.RowDefinitions>
          <RowDefinition></RowDefinition>
          <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
          <ColumnDefinition></ColumnDefinition>
          <ColumnDefinition></ColumnDefinition>
          <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Label Grid.Row="0" Grid.Column="0" Margin="5" MaxWidth="400"
HorizontalAlignment="Left">Hello... </Label>
        <Button Grid.Row="1" Grid.Column="0" Margin="5" MaxWidth="400"
HorizontalAlignment="Center">First Button </Button>
        <Button Grid.Row="0" Grid.Column="3" Margin="5" MaxWidth="200" >Second
Button </Button>
        <Button Grid.Row="1" Grid.Column="1" Margin="5" MaxWidth="400"
HorizontalAlignment="Right">Third Button </Button>
```

```
</Grid>
```



Рассмотренные контейнеры можно использовать совместно.

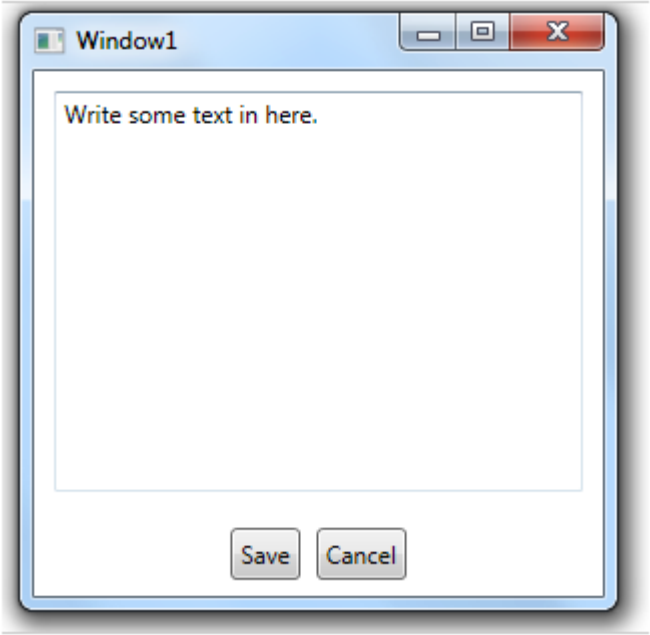
Первый пример: совместное использование **DockPanel** и **StockPanel**

Окна будут организовано следующим образом: большое текстовое поле вверху и две кнопки (Сохранить и Отмена) в нижней части окна. Для кнопок мы будем использовать **StockPanel**, которая будет встроена внутрь **DockPanel**.

Установите для свойства **LastChildFil**l панели **DockPanel** значение «Истина», что бы использовать остальную часть окна.

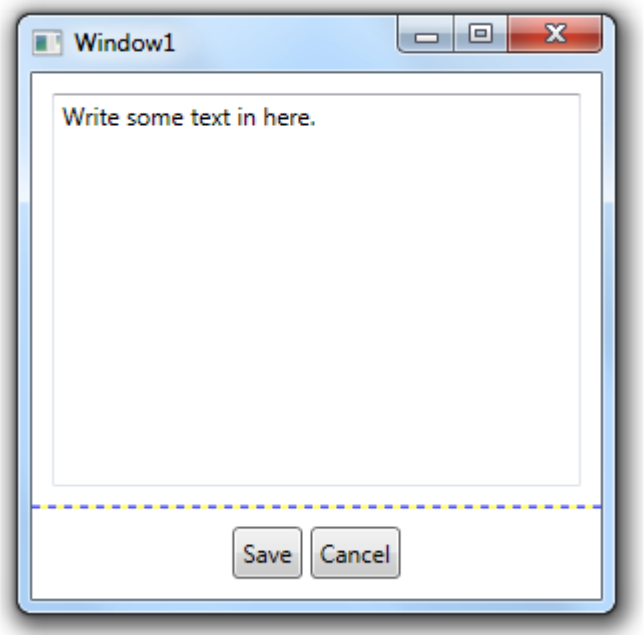
```
<DockPanel LastChildFill="True">
    <StackPanel DockPanel.Dock="Bottom" HorizontalAlignment="Center" Orientation="Horizontal">
        <Button Margin="8,8,4,8" Padding="3">Save</Button>
        <Button Margin="4,8,8,8" Padding="3">Cancel</Button>
    </StackPanel>
    <TextBox DockPanel.Dock="Top" Margin="10">Write some text in here.</TextBox>
</DockPanel>
```

Запустите проект.



Второй пример: с помощью **Grid.** В первой строке сетки разместим элемент TextBox для вывода текста, во второй - StackPanel с двумя кнопками.

```
<Grid ShowGridLines="True">
    <Grid.RowDefinitions>
      <RowDefinition Height="*"></RowDefinition>
      <RowDefinition Height="Auto"></RowDefinition>
    </Grid.RowDefinitions>
    <TextBox Margin="10" Grid.Row="0">Write some text in here.</TextBox>
    <StackPanel Grid.Row="1" HorizontalAlignment="Center" Orientation="Horizontal">
      <Button Margin="10,10,2,10" Padding="3">Save</Button>
      <Button Margin="2,10,10,10" Padding="3">Cancel</Button>
    </StackPanel>
  </Grid>
```



Вопросы к зачету

1. Описание класса. Экземпляр класса.
2. Реализация класса в ООП.
3. Поля, методы, свойства класса.
4. Наследование инкапсуляция, полиморфизм.
5. Перечислите и охарактеризуйте основные принципы ООП.
6. Основные структурные части ИСР C#.
7. Основные проектные операции C#.
8. Назначение основных элементов знатока кода C#.
9. Постройте иерархию базовых классов VCL?
10. Перечислить и описать компоненты отображения информации.
11. Опишите особенности работы с канвой. Укажите компоненты, обладающие канвой. Перечислите основные методы канвы.
12. Перечислите компоненты VCL, реализующие работу системных диалогов различного назначения. Укажите их основные свойства и методы.
13. Операции с классами as и is.Приведите примеры их использования.
14. Укажите основные принципы работы с формами, модальными формами. Перечислите основные события форм.
15. Язык C++. Назначение класса исключений.
16. Язык C++. Классификация стандартных функций.
17. Приложение стандарта Windows. MDI и SDI.
18. Приложение стандарта Windows. Установка и настройка приложения.

Индивидуальное проектное задание заключается в разработке программы, удовлетворяющего системе требований.

Требования к индивидуальному проектному заданию

1. Проанализировать предметную область, определить необходимую функциональность разрабатываемой программы.
2. Исследовать и разработать алгоритм решения задачи, разработать математическую модель и структуру данных.
3. Определить набор входных и выходных данных, хранение которых желательно осуществлять с использованием различных типов данных.
4. Разработать графический интерфейс программы с использованием элементов управления из различных групп. Обще количество элементов управления должно быть не менее 15.

Графический интерфейс обязательно должен содержать следующие элементы управления:

a. Button
b. Label
c. TextBox
d. CheckBox/ RadioButton
e. ComboBox
f. GroupBox
g. Panel

h. MenuStrip

Желательно использование не менее двух элементов управления, созданных динамически.

5. Реализовать работу с файловой системой (загружать входные и/или выгружать выходные данные, используя файл).
6. Разработать систему справки и поддержки, информация о разработчике, должна быть доступна конечному пользователю.
7. Разработать не менее 1-2 классов, если задание позволяет, то с использованием наследования. Обязательным требованием является использование основных принципов ООП.

Разработать тестовые примеры (не менее 3).

Примеры индивидуальных проектных заданий:

1. Применение графических фильтров к изображениям

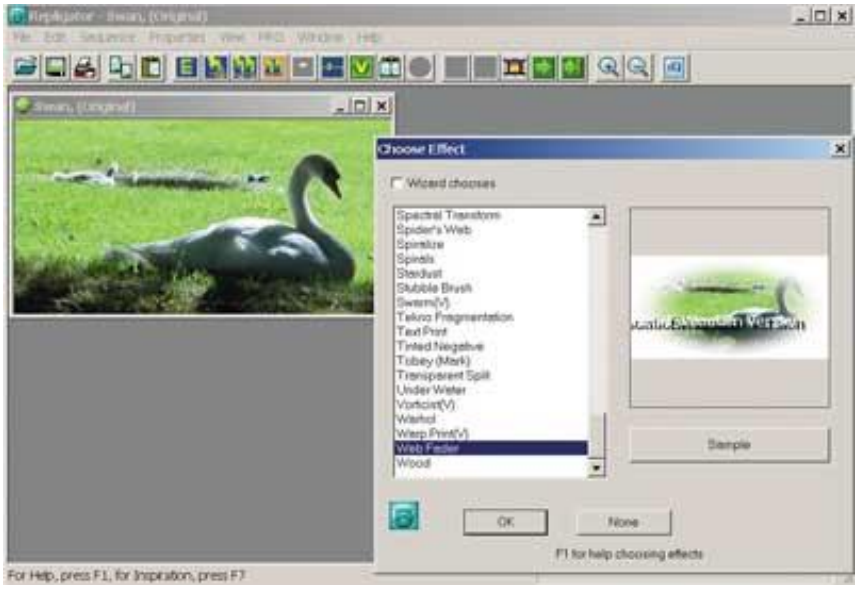
Над любым изображением можно производить различные преобразования, позволяющие изменять исходную картинку. Основной целью такого преобразования является усиление или уменьшение каких либо свойств исходного изображения. Наиболее простыми преобразованиями являются локальные преобразования, затрагивающие вместе с определенным пикселем лишь его непосредственную окрестность (это значит, изменить цвет пикселя в соответствии с цветом его ближайших соседей) с целью достижения некоторого эффекта. Такие преобразования называются фильтрами. С исходным изображением можно работать, как с обычной матрицей, выполняя над ним различные численные преобразования. Интенсивность каждого пикселя изображения - результат действия фильтра вычисляется с помощью воздействия фильтра на соответствующий пиксель исходного изображения и его окрестность.

Спектр применения графических фильтров очень велик, начиная с коррекции цифровых фотографий и заканчивая созданием специальных эффектов на исходных изображениях.

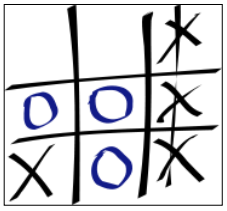
Требуется написать программу (отдельное приложение), которая будет накладывать разнообразные эффекты на исходные изображения или генерировать новые, применяя тот или иной набор фильтров.

Преобразования, которые программа должна делать с изображением:

- Отражения и повороты изображения на угол, кратный 90°;
- Размытие и наведение резкости на изображении;
- Инверсия цветов;
- Удаление шума;
- Отказ от всех преобразований;



2. Игра «крестики-нолики»



Крестики-нолики — логическая игра между двумя противниками на квадратном поле 3 на 3 клетки или большего размера (вплоть до «бесконечного поля»). Один из игроков играет «крестиками», второй — «ноликами».

Классический вариант

Игроки по очереди ставят на свободные клетки поля 3х3 знаки (один всегда крестики, другой всегда нолики). Первый, выстроивший в ряд 3 своих фигур по вертикали, горизонтали или диагонали, выигрывает. Первый ход делает игрок, ставящий крестики.

Обычно по завершении партии выигравшая сторона зачёркивает чертой свои три знака (нолика или крестика), составляющих сплошной ряд.

Более длинные линии

Можно рассматривать игру, в которой победителем считается игрок, первым построивший $n \geq 3$ одинаковых знаков на достаточно большом для этого прямоугольном поле. При этом можно ограничить поле каким-нибудь размером (начиная с $n \times n$), либо вовсе не ограничивать (в этом случае говорят о «бесконечном» поле)

Игра до 4 одинаковых знаков на бесконечном поле неинтересна, ибо начинающий довольно быстро строит «вилку» и выигрывает. Игра до $n \geq 6$ также неинтересна из-за «ничейной смерти». Существуют стратегии, не дающие противнику построить нужную линию никогда. Однако при n = 5 игра становится намного содержательнее. Такой вариант имеет специальное название — гомоку.

Основной победной тактикой при игре на бесконечном поле считается построение пересечений («вилок»), которые не дают противнику возможности блокировать все возможные пути построения пятёрки. Чтобы не проиграть, необходимо своевременно прерывать линии противника длиной в три фигуры.

Практика показала, что при равных правилах для игроков тот, кто делает первый ход, имеет преимущество, позволяющее при достаточно квалифицированной игре одержать победу. Чтобы сохранить интерес к игре, в гомоку были введены ограничения для играющего чёрными — ему запрещено строить ряд структур, полученная игра стала называться рэндзю и по ней до сих пор проводятся международные турниры.

Модификация поля

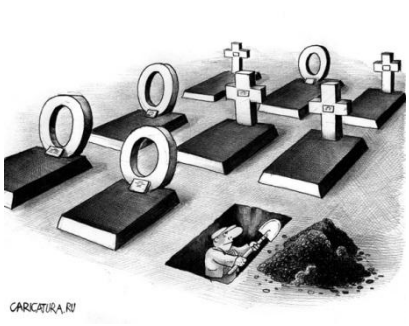
Вариантом изменения топологии поля является склеивание противоположных сторон поля, образуя при этом либо поверхность цилиндра или тора, либо проективную плоскость.

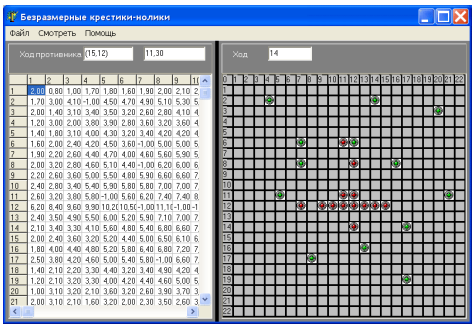
Обмен значков

Можно отменить правило, указывающее игрокам ставить только свой вид значков. Например, вариантом игры может быть: игроки ставят крестик или нолик (что захотят), первый выигрывает, если кто-нибудь построит линию нужной длины из одинаковых значков, второй - если до заполнения поля этого не произойдёт. К этому варианту игры относятся крестики-нолики Силвермэна

Изменение условия выигрыша

Вместо того, чтобы заканчивать игру построением первой линии нужной длины, можно на этом не останавливаться и продолжить до полного заполнения поля. Например, на любом поле можно играть на то, кто больше построит «четвёрок» из своих знаков.





Требуется написать программу, которая реализует пользовательский интерфейс игры крестики-нолики для $3 \le n \le 30$. Основной режим игры пользователь-пользователь, для случая $n = 3$ программа должна функционировать и в режиме пользователь-компьютер. Для отображения крестика и нолика можно добавить выбор изображений как из предложенных вариантов так и из файлов пользователя.

Обязательным является возможность сохранения и возобновление текущей игры!

3. Игра «Жизнь»

Идея клеточных автоматов появилась в конце сороковых годов 20 века. Она была задумана и сформулирована Джоном фон Нейманом и Конрадом Цусе независимо друг от друга как универсальная вычислительная среда для построения, анализа и сравнения характеристик алгоритмов.

Клеточный автомат представляет собой совокупность пространства, поделенного на клетки, в каждой из которых содержится некое значение, и правил, задающих закон преобразования при совершении шага. Это пространство может быть как плоскостью, так и любым N-мерным пространством.

Представим себе некоторое однородное пространство, состоящее из клеток. Каждая клетка обладает несколькими состояниями, которые могут зависеть от следующих параметров:

- "Соседи" клетки - клетки, располагающиеся в непосредственной близости от данной;
- количество ходов, прошедших от начала игры;
- свойства самой клетки и т.д.

Правила игры достаточно просты:

Полем для игры является бесконечная плоскость, разбитая на клетки. Каждая клетка может находиться в двух состояниях: "живом" и "мертвом". Правила перехода клетки в новое состояние выбирается из следующих критериев: минимальное, максимальное количество соседей для "рождения" клетки, минимальное, максимальное количество соседей для "вымирания" клетки.

Конвей много экспериментировал с этими параметрами и остановился на следующем:

1. Соседями клетки являются все восемь клеток, имеющих либо общую сторону, либо общую вершину.
2. Клетка рождается, если количество соседей равно 3.
3. Клетка умирает, если количество соседей либо больше 3 (перенаселение), либо меньше 2 (одиночество). (Модель 3323)

В процессе игры популяция непрестанно претерпевает необычные, нередко очень красивые и всегда неожиданные изменения. Иногда первоначальная колония организмов вымирает, но, в большинстве своем, исходные конфигурации либо переходят в устойчивые и перестают изменяться, либо переходят в колебательный режим. Наблюдается тенденция к приобретению симметрии конфигурациями, которых на первый взгляд, никак не назвать симметричными.

Для начала условимся классифицировать конфигурации по следующим параметрам:

- По количеству клеток в комбинации: единичная клетка, дуплет, триплет и т.д.

- По перспективе развития: развивающиеся, стабильные, вымирающие.
- По расположению клеток относительно друг друга.

Очевидно, что конфигурация, состоящая из одной клетки, как и любая пара клеток, расположенных по горизонтали, вертикали, или диагонали, погибает на первом же ходу, поэтому рассмотрим простейшие конфигурации, состоящие из 3 клеток (тримино), и проследим их эволюцию в течение нескольких ходов:

Первые три конфигурации на втором ходу погибают. Относительно третей конфигурации заметим, что любой диагональный ряд клеток с каждым ходом с каждого конца по клетке. Четвертая конфигурация на втором ходу переходит в устойчивую конфигурацию "Блок". Пятая конфигурация является циклической - она повторяет себя каждые 2 хода - "Мигалка"

Остальные триплеты (без учета ориентации) исчезают на первом же ходу. Пойдем дальше и возьмем в качестве основы конфигурацию, состоящую из четырех клеток (тетрамино):

Простые конфигурации являются устойчивыми. Ниже перечислена большая часть устойчивых конфигураций.

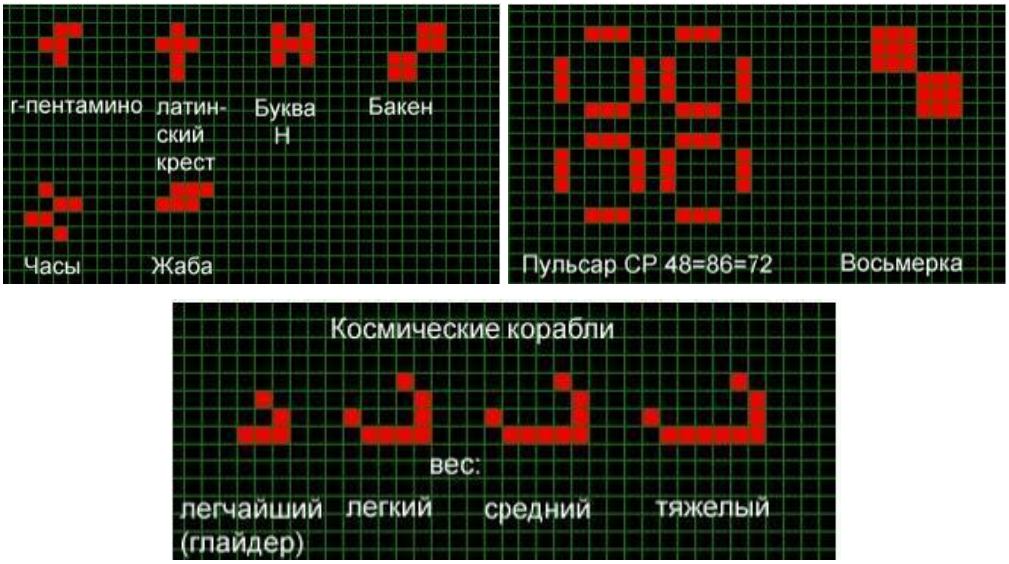
Вторая и третья конфигурации после второго хода переходят в устойчивую конфигурацию, называемую "Ульем". Четвертая конфигурация переходит в "Улей" на третьем ходу. Пятая конфигурация после девяти ходов переходит в устойчивую конфигурацию, называемую "Навигационные огни". "Навигационные огни" представляют комбинацию из четырех "Мигалок", вертящихся в одинаковой фазе. Подобные конфигурации в процессе игры возникают довольно часто.

Если увеличить число клеток в конфигурации до 5, то можно найти очень интересную конфигурацию, которую сам Конвей назвал "глайдером". Дело в том, что эта конфигурация является периодической, она восстанавливает себе каждые четыре хода, и при этом сдвигается по диагонали!

Для изучения подобного рода конфигураций, Конвей ввел понятие скорости света в своей игре. Скорость, с которой передвигается шахматный король, и была названа скоростью света.

Конвей открыл много передвигающихся циклических конфигураций, которых он назвал "космическими кораблями". В этой терминологии глайдер является "космическим кораблем легчайшего веса". Он также показал, что можно продлить корпус "космического корабля" на сколь угодное количество клеток, но если у "корабля" длина корпуса больше шести клеток, в "полете" возникают "искры", которые мешают полету. Для поглощения искр, требуется эскорт из "космических кораблей" меньшего размера. Так, например "космическому кораблю" с корпусом в 100 клеток требуется целая флотилия, состоящая из 33 "кораблей" меньших размеров. Тем, кто хочет подробнее узнать об игре "Жизнь" можно рекомендовать книги М. Гарднера "Математические досуги" и "Крестики-нолики".

Стандартные конфигурации клеток



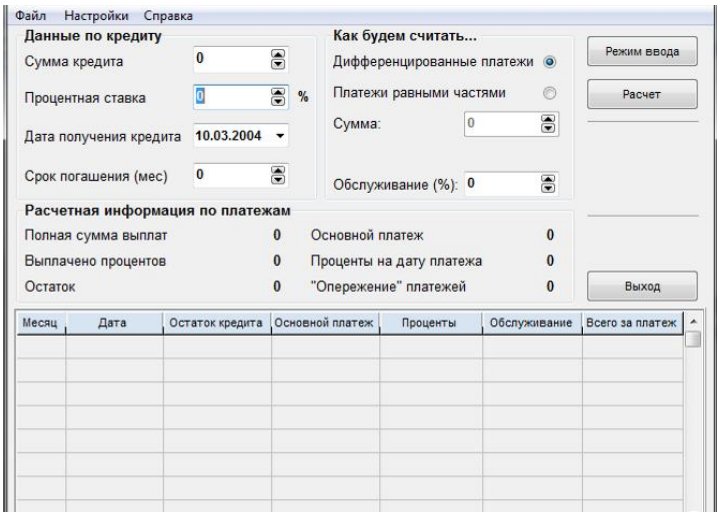
Требуется написать программу, которая отображает состояние системы для произвольного n. Необходимо реализовать настройку стиля отображения и составить базу стандартных конфигураций клеток, для последующего использования. Программа должна сохранять все состояния системы в графические файлы (все картинки должны относиться к одному блоку изображений и содержать в названии временные характеристики, например Example001.jpeg, Example002.jpeg…), а так же приветствуется возможность создания видео файла, отображающего динамику развития системы.

4. Кредитный калькулятор

Требуется написать программу, которая может быть использована для примерного расчета ежемесячных платежей по кредитам. Расчет производится на основании следующих факторов: сумма покупки, первый взнос за покупку в процентах, срок кредита, годовой процент и т.д.

Кредитные калькулятор должен имеет следующие возможности:

- Расчет дифференцированных и аннуитетных платежей по кредиту;
- Дробная процентная ставка;
- Учет обслуживания счета (в процентах от суммы кредита или от остатка кредита, может быть дробной);
- Возможность расчета срока погашения кредита при платежах по кредиту равными суммами (не аннуитет);
- Подсчет полной суммы выплат по кредиту, суммы выплаченных процентов.
- Возможность учета платежей по кредиту, планирования платежей;
- Сохранение и загрузка истории платежей;
- Подсчет процентов по кредиту в зависимости от количества дней между платежами, количества дней в году;
- Быстрый показ процентов на дату платежа;
- Возможность прогнозирования окончания платежей;
- Учет просрочки;
- Отображение полной суммы выплат, суммы выплаченных процентов, "опережения платежей", остатка.



5. Калькулятор

Требуется написать программу, которая реализовывает стековый калькулятор, работающий с вещественными числами. Калькулятор должен выполнять следующие арифметические действия:

- +
- -
- *
- /
- %
- 1/x

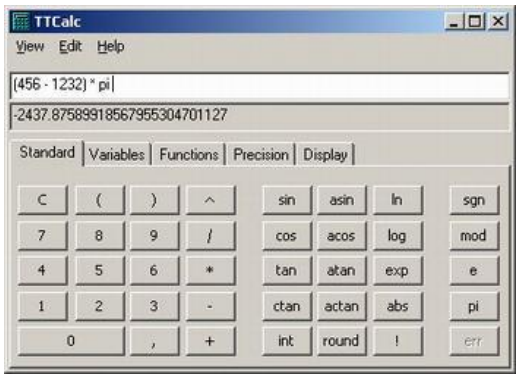
вычислять стандартные математические функции:

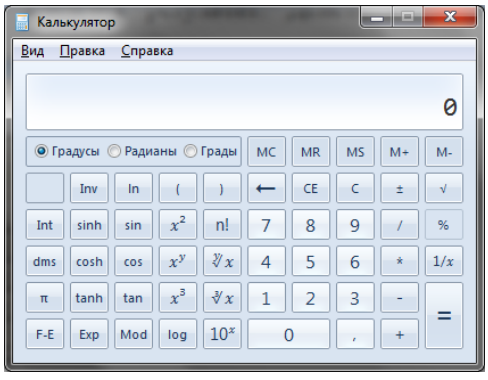
- Sin, Cos, Tan
- Sinh, Cosh, Tanh
- Sqrt
- x^2, x^y, x^3
- exp, 10^x, log, ln
- abs

и использовать стандартные математические константы π, e

Калькулятор должен иметь возможность помещать в память и извлекать из памяти отложенные значения, исправлять введенное значение, очищать память от всех предыдущих вычислений, а также оперировать с тремя типами измерений градусы, радианы и грады.

Действия выполняются над элементами в вершине стека, эти элементы удаляются из стека, а на их место кладется результат. Действиям соответствуют клавиши калькулятора.





Критерии оценки проектов

Составляющие проекта	Критерии для оценивания	Максимальное количество балов
Постановка проблемы и ее обоснованность, формулирование целей и задач	• общественная значимость и актуальность выдвинутых проблем; • соответствие темы, цели и задач проекта; разумность масштаба работ.	10
Содержание проекта/ проектной разработки	• логичность, взаимосвязь и последовательность этапов проекта; • адекватность предлагаемых мероприятий решению поставленных задач; • корректность используемых методов работы; • четкость определения целевой группы и обоснованность её участия при реализации проекта; • соответствие теоретической, эмпирической и проектной частей, их связь с практикой и выбранным видом профессиональной деятельности; • соблюдение заявленных временных рамок реализации проекта; самостоятельность и активность участника проекта.	10
Результат выполнения прикладного проекта	• соответствие ожиданий от проекта / планируемого результата полученному продукту; • степень решения заявленной проблемы; • успешность преодоления трудностей в реализации проекта; • оценка участников целевой группы; • перспективы развития проекта после завершения проекта; возможность тиражирования проекта.	10
Презентация результатов работы над прикладным проектом	• ясность, логичность, профессионализм изложения доклада; • наглядность и структурированность материала презентации; умение корректно использовать профессиональную лексику и понятийно-категориальный аппарат.	10
Ответы на вопросы	• степень владения темой; • ясность аргументации взглядов студента, презентующего результаты выполнения проекта; четкость и лаконичность ответов на вопросы.	10

6.4. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Описание балльно-рейтинговой системы по дисциплине.

Итоговая рейтинговая оценка по дисциплине складывается из следующих составляющих:

1) В течении семестра за выполнение заданий по курсу студент может максимально получить 70 баллов.;

2) Обязательной формой текущей аттестации знаний является выполнение индивидуаль-

ного проектного задания 20 баллов.

3) На зачёте ответ студента может быть максимально оценен в 30 баллов.

При этом, для получения положительной итоговой оценки на зачете необходимо получить не менее 60% по каждой составляющей и выполнить все задания для практических занятий. Шкала перевода баллов в оценку: до 40 - «не зачтено»; 41 - 100 - «зачтено».

№ п/п	Критерии оценивания	Максимальное количество баллов	Баллы, полученные студентом
1.	Выполнение заданий:	80	
1.1.	Практические занятия	60	
1.2.	Индивидуальное проектное задание	20	
3.	Зачет	20	
	ИТОГО:	100	

7. ПЕРЕЧЕНЬ ОСНОВНОЙ И ДОПОЛНИТЕЛЬНОЙ УЧЕБНОЙ ЛИТЕРАТУРЫ, НЕОБХОДИМОЙ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

7.1. Основная литература

1. Жданов, С. А. Информационные системы : учебник / С.А. Жданов, М.Л. Соболева, А.С. Алфимова. - М. : Прометей, 2015. - 302 с. - ISBN 978-5-9906-2644-7 : Б. ц. URL: http://biblioclub.ru/index.php?page=book_view&book_id=426722
2. Мартынюк, Ю. М. Методы программирования [Текст] : учебное пособие / Ю. М. Мартынюк, С. С. Гербут, В. С. Ванькова ; рец.: Е. Г. Торина, Е. А. Снижко ; ФГБОУ ВПО "Тульский государственный педагогический университет им. Л. Н. Толстого". - Тула : Изд-во ТГПУ им. Л. Н. Толстого, 2013. - 70 с.

7.2. Дополнительная литература

1. Объектно-ориентированное программирование на С++ [Текст] / Айра Пол. - 2-е изд. - М. : Бином, 1999. - 462 с. : ил. - ISBN 5798901408
2. Практикум по объектно-ориентированному программированию [Текст] : учебное пособие / И. Бабушкина. - [Б. м.] : Бином, 2004. - 366 с. : ил. - ISBN 5947741296
3. Объектно-ориентированное программирование [Текст] : учебник для студ.вузов / Г. С. Иванова. - М. : Изд-во МГТУ им. Н. Баумана, 2003. - 368 с. : ил. - ISBN 5703822807
4. С++.Объектно-ориентированное программирование.Задачи и упражнения [Текст] : учебное пособие для студ.вузов / В. В. Лаптев. - С П б. : Питер, 2007. - 288 с. : ил. - ISBN 9785469014379

8. ПЕРЕЧЕНЬ РЕСУРСОВ ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННОЙ СЕТИ «ИНТЕРНЕТ», НЕОБХОДИМЫХ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

1. ИКТ [Электронный ресурс] : федеральный образовательный портал / ФГАУ ГНИИ ИТТ "Информатика". - М. : [б. и.], 2003. - Загл. с титул. экрана.
   URL: http://www.ict.edu.ru
2. CITForum.ru [Электронный ресурс] : образовательный портал / "ЦИТ Форум". - [Б. м. : б. и.], 1997. - Загл. с титул. экрана.
   URL: http://citforum.ru/

3. Виртуальный компьютерный музей [Электронный ресурс] : сайт / Э. Пройдаков. - М. : [б. и.], 1997. - Загл. с титул. экрана. - Б. ц.
URL:http://www.computer-museum.ru

9. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Приступая к изучению новой учебной дисциплины, студенты должны ознакомиться с учебной программой, учебной, научной и методической литературой, имеющейся в библиотеке университета, встретиться с преподавателем, ведущим дисциплину, получить в библиотеке рекомендованные учебники и учебно-методические пособия, осуществить запись на соответствующий курс в среде электронного обучения университета.

Глубина усвоения дисциплины зависит от активной и систематической работы студента на лекциях и практических занятиях, а также в ходе самостоятельной работы, по изучению рекомендованной литературы.

На лекциях важно сосредоточить внимание на ее содержании. Это поможет лучше воспринимать учебный материал и уяснить взаимосвязь проблем по всей дисциплине. Основное содержание лекции целесообразнее записывать в тетради в виде ключевых фраз, понятий, тезисов, обобщений, схем, опорных выводов. Необходимо обращать внимание на термины, формулировки, раскрывающие содержание тех или иных явлений и процессов, научные выводы и практические рекомендации. Желательно оставлять в конспектах поля, на которых делать пометки из рекомендованной литературы, дополняющей материал прослушанной лекции, а также подчеркивающие особую важность тех или иных теоретических положений. С целью уяснения теоретических положений, разрешения спорных ситуаций необходимо задавать преподавателю уточняющие вопросы. Для закрепления содержания лекции в памяти, необходимо во время самостоятельной работы внимательно прочесть свой конспект и дополнить его записями из учебников и рекомендованной литературы. Конспектирование читаемых лекций и их последующая доработка способствует более глубокому усвоению знаний, и поэтому являются важной формой учебной деятельности студентов.

Прочное усвоение и долговременное закрепление учебного материала невозможно без продуманной самостоятельной работы. Такая работа требует от студента значительных усилий, творчества и высокой организованности. В ходе самостоятельной работы студенты выполняют следующие задачи: дорабатывают лекции, изучают рекомендованную литературу, готовятся к практическим занятиям, к коллоквиуму, контрольным работам по отдельным темам дисциплины. При этом эффективность учебной деятельности студента во многом зависит от того, как он распорядился выделенным для самостоятельной работы бюджетом времени.

Результатом самостоятельной работы является прочное усвоение материалов по предмету согласно программы дисциплины. В итоге этой работы формируются профессиональные умения и компетенции, развивается творческий подход к решению возникших в ходе учебной деятельности проблемных задач, появляется самостоятельности мышления.

Целью практических занятий по данной дисциплине является закрепление теоретических знаний, полученных при изучении дисциплины.

При подготовке к практическому занятию целесообразно выполнить следующие рекомендации: изучить основную литературу; ознакомиться с дополнительной литературой, новыми публикациями в периодических изданиях: журналах, газетах и т. д.; при необходимости доработать конспект лекций. При этом учесть рекомендации преподавателя и требования учебной программы.

При выполнении практических занятий основным методом обучения является самостоятельная работа студента под управлением преподавателя. На них пополняются теоретические знания студентов, их умение творчески мыслить, анализировать, обобщать изученный материал, проверяется отношение студентов к будущей профессиональной деятельности.

Оценка выполненной работы осуществляется преподавателем комплексно: по результатам выполнения заданий, устному сообщению и оформлению работы. После подведения итогов занятия студент обязан устранить недостатки, отмеченные преподавателем при оценке его работы.

Преподавание дисциплины должно включать в себя следующие образовательные технологии:

1) Проведение лекций с использованием презентаций на основе мультимедийных технологий;
2) Обеспечение студентов сопутствующими материалами, размещенными среде Moodle;

Примерная тематика практических занятий по дисциплине.

Полные варианты практических занятий размещены в в системе управления обучением MOODLE.

Описание практических занятий по дисциплине

Полные варианты практических занятий размещены в в системе управления обучением MOODLE.

№	Наименование практических занятий	Объем в часах
1	Создание простейшего WPF приложения	4
2	Управление расположением элементов	4
3	Управление основными объектами	4
4	Разработка приложения «Калькулятор».	4
5	Простой редактор градиентов. Стили, триггеры и анимация	4
6	Создание классов	6
	Итого	26

Типовые задания для самостоятельной работы по дисциплине

Задание1 . Добавьте в созданный класс методы для вычисления площади и периметра треугольника, сделайте соответствующие изменения в интерфейсе приложения (поля для хранения площади и периметра также должны быть закрытыми).

Задание 2. Создайте новый проект, с разработкой класса для работы с комплексными числами. В приложении организуйте ввод двух комплексных чисел, вычисление их суммы, произведения, модуля.

10. ПЕРЕЧЕНЬ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, ИСПОЛЬЗУЕМЫХ ПРИ ОСУЩЕСТВЛЕНИИ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ, ВКЛЮЧАЯ ПЕРЕЧЕНЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ИНФОРМАЦИОННЫХ СПРАВОЧНЫХ СИСТЕМ

1. Подписка Microsoft DreamSpark Premium - Сублицензионный договор № S-2042626/M18 от 04.06.2013:
   1.1. Средства для разработки и проектирования Visual Studio 2008, 2010, 2012 и 2013 Professional Editions;
   1.2. Интегрированная среда разработки Visual Studio Express;
   1.3. Операционная система Windows Server 2008 Standard Edition 32-bit;
   1.4. Операционная система Windows 8.1 Pro;
   1.5. Отдельные программы из Office 2007, Office 2010, Office 2013;
2. Операционная система Microsoft Windows XP Professional Russian – Лицензия № 16698685 от 08.08.2003 г.;
3. Программное обеспечение Microsoft Office XP Professional Win32 Russian– Лицензия № 16698685 от 08.08.2003 г.;

4. Веб-браузеры.
5. Доступ студентов через личные кабинеты к электронным библиотечным системам.
6. Возможность работы студентов на удаленном рабочем столе кафедры информатики и информационных технологий.

11. ОПИСАНИЕ МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЙ БАЗЫ, НЕОБХОДИМОЙ ДЛЯ ОСУЩЕСТВЛЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

- компьютерный класс для проведения лабораторных занятий по дисциплине, оснащённый компьютерами с процессорами классов Pentium или Core Duo (количество компьютеров – не менее 10 укомплектованных компьютерами рабочих мест);
- видеопроектор и в качестве средства поддержки лекционных занятий;
- интерактивная доска в качестве средства поддержки лекционных занятий;
- Интернет-доступ, позволяющий осуществлять подбор материалов для выполнения заданий, подготовки информационного проекта, научных сообщений, реферата;
- аудитории для самостоятельной работы студентов, оснащенные компьютерной техникой, имеющей доступ к информационно-телекоммуникационной сети «Интернет», электронной информационно-образовательной среде ТГПУ им. Л.Н. Толстого, внутривузовскому сетевому окружению;
- наличие прав доступа к перечисленному выше программному обеспечению

12. АННОТАЦИЯ РАБОЧЕЙ ПРОГРАММЫ ДИСЦИПЛИНЫ.

1.Планируемые результаты обучения при освоении дисциплины, соотнесенные с планируемыми результатами освоения образовательной программы.

В результате освоения дисциплины у студента должна быть сформирована следующая компетенция: способность использовать знания основных концептуальных положений функционального, логического, объектно-ориентированного и визуального направлений программирования, методов, способов и средств разработки программ в рамках этих направлений (ОПК-7).

В результате освоения дисциплины студент должен приобрести:

знания о стилях программирования, об объектно-ориентированном программировании; о методах проектирования объектно-ориентированных программ; об объектно-ориентированных языках программирования и визуальном программировании; понятия класса и объекта, об основных принципах объектно-ориентированного программирования; о принципах построения классов, критериев проверки правильности построения классов основные тенденции в области развития технологий объектно-ориентированного программирования;

умения использовать современные методы объектно-ориентированного программирования при кодировании программных систем разного уровня сложности; работать со средой визуального программирования MS Visual Studio и языком программирования высокого уровня C#;

навыки владения технологией объектно-ориентированной разработки программного обеспечения.

2.Место дисциплины в структуре ОПОП.

Дисциплина «Объектно-ориентированное программирование» относится к дисциплинам по выбору вариативной части образовательной программы. Изучение данной дисциплины осуществляется в 4 семестре.

3.Объем дисциплины: 3 зачетные единицы.

4.Образовательный процесс осуществляется на русском языке.

5.Разработчик: Якушин А.В., к.п.н., доцент, зав. кафедрой ИиИТ.

13. ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ К РАБОЧЕЙ ПРОГРАММЕ ДИСЦИПЛИНЫ

Внесены изменения в п.7 «Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины».

Обновлен п.10 «Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень лицензионного программного обеспечения и информационных справочных систем» на основании действующих лицензионных соглашений.

Заведующий кафедрой ИиИТ _____А.В. Якушин

«29» августа 2016 г.

Программа составлена в соответствии с требованиями ФГОС ВО.

Разработчик (и):

Фамилия, имя, отчество	Учёная степень	Учёное звание	Должность	Дата разработки	Подпись
Якушин Алексей Валериевич	к.п.н.	Доц.	зав. кафедры информатики и информационных технологий	17.11.2015