МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

ФГБОУ ВО «БАЙКАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра математических методов и цифровых технологий

**Разработка приложений для мобильных устройств**

Отчёт по первой лабораторной работе

|  |  |  |
| --- | --- | --- |
| студента бакалавриата группы | БИ-17-1 | Точилова Е.А. Фамилия И.О. |
| проверил | к.т.н., доцент кафедры математических методов и цифровых технологий  ученое звание, должность | Родионов А.В.  Фамилия И.О |

Иркутск, 2020

# Задание . Основные элементы компоновки XAML

1. Изучить и описать назначение, свойства и события следующих XAML элементов (на примере UWP):
   * Grid
   * VariableSizedWrapGrid
   * StackPanel
   * SplitView
   * RelativePanel
2. Провести параллели с элементами XAML из главного подпроекта, исследовать их сходство и различия.
3. Изучите создание элементов как с использованием XAML разметки, так и генерации их из кода приложения.
4. Результат выполнения задания оформить в виде отчета в Word.

# Grid

Элемент Grid представляет контейнер в виде обычной таблицы. Он содержит столбцы и строки, а в отдельных ячейках этой таблицы размещаются элементы управления. То есть Grid служит не просто контейнером, но и управляет позиционированием вложенных элементов.

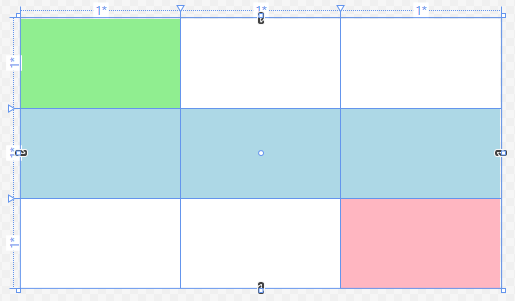
При создании нового проекта по шаблону Blank App разметка страницы в xaml уже содержит элемент Grid. Однако по умолчанию он содержит одну ячейку (или одну строку и один столбец), в которой размещаются все элементы управления. Для определения строк используется свойство RowDefinitions, а для определения столбцов - свойство ColumnDefinitions:

|  |  |
| --- | --- |
|  | <Grid.RowDefinitions>      <RowDefinition></RowDefinition>      <RowDefinition></RowDefinition>      <RowDefinition></RowDefinition>  </Grid.RowDefinitions>  <Grid.ColumnDefinitions>      <ColumnDefinition></ColumnDefinition>      <ColumnDefinition></ColumnDefinition>      <ColumnDefinition></ColumnDefinition>  </Grid.ColumnDefinitions> |

Каждая строка задается с помощью вложенного элемента RowDefinition, то есть в данном случае в гриде определено 3 строки. А каждый столбец объявляется с помощью элемента ColumnDefinition, то есть здесь у нас три столбца. В итоге в данном случае у нас получается таблица 3х3.

Чтобы поместить элемент управления в определенную ячейку Grid, в разметке элемента управления надо указать свойства Grid.Column и Grid.Row. Они соответственно определяют столбец и строку, на пересечении которых будет размещаться элемент управления. Кроме того, если мы хотим растянуть элемент управления на несколько строк или столбцов, то можно указать свойства Grid.ColumnSpan и Grid.RowSpan, как в следующем примере:

|  |  |
| --- | --- |
|  | <Page      x:Class="LayoutApp.MainPage"      xmlns="<http://schemas.microsoft.com/winfx/2006/xaml/presentation>"      xmlns:x="<http://schemas.microsoft.com/winfx/2006/xaml>"      xmlns:local="using:LayoutApp"      xmlns:d="<http://schemas.microsoft.com/expression/blend/2008>"      xmlns:mc="<http://schemas.openxmlformats.org/markup-compatibility/2006>"      mc:Ignorable="d">        <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">          <Grid.RowDefinitions>              <RowDefinition />              <RowDefinition />              <RowDefinition />          </Grid.RowDefinitions>          <Grid.ColumnDefinitions>              <ColumnDefinition />              <ColumnDefinition />              <ColumnDefinition />          </Grid.ColumnDefinitions>          <Rectangle Grid.Column="0" Grid.Row="0" Fill="LightGreen"/>          <Rectangle Grid.Column="0" Grid.Row="1" Grid.ColumnSpan="3" Fill="LightBlue"  />          <Rectangle Grid.Column="2" Grid.Row="2" Fill="LightPink" />      </Grid>  </Page> |



В данном случае в ячейках грида размещены элементы Rectangle - прямоугольники с цветовой заливкой.

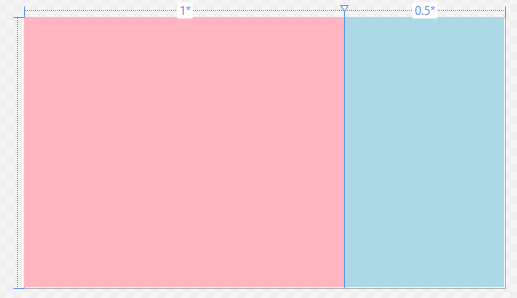
Если у элементов грида явным образом не указан номер строки (или столбца), то по умолчанию элемент помещается в самую первую строку (или столбец).

# Размеры строк и столбцов

# Пропорциональные размеры

В примере выше строки и столбцы определены без указания их размеров. Поэтому они по умолчанию принимают пропорциональные размеры. Мы можем явным образом задать пропорциональные размеры, используя символ \* (звездочка):

|  |  |
| --- | --- |
|  | <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">      <Grid.ColumnDefinitions>          <ColumnDefinition Width="\*" />          <ColumnDefinition Width="0.5\*" />      </Grid.ColumnDefinitions>      <Rectangle Grid.Column="0" Fill="LightPink"/>      <Rectangle Grid.Column="1" Fill="LightBlue"  />  </Grid> |



Здесь два столбца с пропорциональными размерами. Только второй столбец имеет половину от ширины первого столбца.

Если столбец имеет ширину, равную \*, то данный столбец будет занимать все оставшееся место. Если у нас есть несколько столбцов, ширина которых равна \*, то все доступное место делится поровну между всеми такими сроками и столбцами. Использование коэффициентов (0.5\*) позволяет уменьшить или увеличить выделенное место на данный коэффициент. При этом все коэффициенты складываются (коэффициент \* аналогичен 1\*) и затем все пространство делится на сумму коэффициентов. То же самое касается высоты строк.

Так, в вышеприведенном примере сумма коэффициентов столбцов равна 1\* + 0.5\* = 1.5\*. Если у нас грид имеет ширину 300 единиц, то для коэффициент 1\* будет соответствовать пространству 300 / 1.5 = 200 единиц. Поэтому первый столбец будет иметь ширину в 200 единиц, второй - 200 \* 0.5 = 100 единиц.

# Автоматические размеры

В этом случае столбец или строка занимает то место, которое им нужно. Для указания подобных размеров используется значение Auto:

|  |  |
| --- | --- |
|  | <ColumnDefinition Width="Auto" />  <RowDefinition Height="Auto" /> |

# Абсолютные размеры

В данном случае высота и ширина указываются в единицах, независимых от устройства

|  |  |
| --- | --- |
|  | <ColumnDefinition Width="150" />  <RowDefinition Height="200" /> |

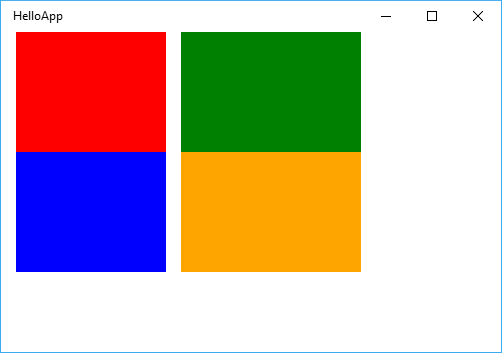
Можно комбинировать все типы размеров. В этом случае от ширины/высоты грида отнимается ширина/высота столбцов/строк с абсолютными или автоматическими размерами, и затем оставшееся место распределяется между столбцами/строками с пропорциональными размерами.

# VariableSizedWrapGrid

VariableSizedWrapGrid также представляет контейнер табличного типа, однако позиционирование внутри контейнера в нем задают два свойства: MaximumRowsOrColumns и Orientation. MaximumRowsOrColumns указывает на максимальное количество строк или столбцов в гриде. А Orientation задает вертикальную или горизонтальную ориентацию. При вертикальной ориентации MaximumRowsOrColumns указывает на максимальное количество строк, а при горизонтальной - столбцов. Если количество элементов в строке (столбце) превышает максимальное, то для вмещения элементов создается новая строка (столбец).

Также с помощью свойств ItemWidth и ItemHeight можно задать ширину и высоту ячейки грида соответственно. Если эти свойства явным образом не заданы, то ширина и высота ячеек вычисляется на основе размеров элементов. Например:

|  |  |
| --- | --- |
|  | <Page      x:Class="LayoutApp.MainPage"      xmlns="<http://schemas.microsoft.com/winfx/2006/xaml/presentation>"      xmlns:x="<http://schemas.microsoft.com/winfx/2006/xaml>"      xmlns:local="using:LayoutApp"      xmlns:d="<http://schemas.microsoft.com/expression/blend/2008>"      xmlns:mc="<http://schemas.openxmlformats.org/markup-compatibility/2006>"      mc:Ignorable="d">      <VariableSizedWrapGrid MaximumRowsOrColumns="2" ItemHeight="120" ItemWidth="180">          <Rectangle Fill="Red" Width="150" Height="120" />          <Rectangle Fill="Blue" Width="150" Height="120" />          <Rectangle Fill="Green" Width="180" Height="120" />          <Rectangle Fill="Orange" Width="180" Height="120" />      </VariableSizedWrapGrid>  </Page> |



По умолчанию VariableSizedWrapGrid имеет вертикальную ориентацию, поэтому значение MaximumRowsOrColumns="2" в данном случае указывает на количество строк - не больше 2. Даже если на форме еще будет место для дополнительных строк, их будет не больше 2.

При горизонтальной ориентации ограничение было бы на количество столбцов:

|  |  |
| --- | --- |
|  | <VariableSizedWrapGrid Background="White" MaximumRowsOrColumns="2" Orientation="Horizontal"> |

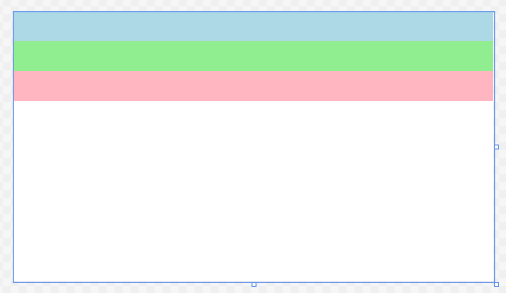
С помощью прикрепляемых свойств VariableSizedWrapGrid.ColumnSpan и VariableSizedWrapGrid.RowSpan можно растянуть элемент на несколько столбцов и строк:

|  |  |
| --- | --- |
|  | <VariableSizedWrapGrid MaximumRowsOrColumns="3" ItemHeight="100" ItemWidth="100">      <Rectangle Fill="Red" />      <Rectangle Fill="Blue" VariableSizedWrapGrid.RowSpan="2"/>      <Rectangle Fill="Green" VariableSizedWrapGrid.RowSpan="2"          VariableSizedWrapGrid.ColumnSpan="2"/>      <Rectangle Fill="Yellow" VariableSizedWrapGrid.ColumnSpan="2"/>  </VariableSizedWrapGrid> |

# StackPanel

StackPanel располагает все элементы в ряд либо по горизонтали, либо по вертикали в зависимости от ориентации. Например, расположим элементы по вертикали:

|  |  |
| --- | --- |
|  | <Page      x:Class="LayoutApp.MainPage"      xmlns="<http://schemas.microsoft.com/winfx/2006/xaml/presentation>"      xmlns:x="<http://schemas.microsoft.com/winfx/2006/xaml>"      xmlns:local="using:LayoutApp"      xmlns:d="<http://schemas.microsoft.com/expression/blend/2008>"      xmlns:mc="<http://schemas.openxmlformats.org/markup-compatibility/2006>"      mc:Ignorable="d">        <StackPanel>          <Rectangle Fill="LightBlue" Height="40" />          <Rectangle Fill="LightGreen" Height="40" />          <Rectangle Fill="LightPink" Height="40" />      </StackPanel>  </Page> |



По умолчанию свойство Orientation элемента StackPanel использует значение Vertical, то есть создается вертикальный ряд, в который помещаются все вложенные элементы сверху вниз. Мы также можем задать горизонтальный стек. Для этого нам надо указать свойство Orientation="Horizontal":

|  |  |
| --- | --- |
|  | <Page      x:Class="LayoutApp.MainPage"      xmlns="<http://schemas.microsoft.com/winfx/2006/xaml/presentation>"      xmlns:x="<http://schemas.microsoft.com/winfx/2006/xaml>"      xmlns:local="using:LayoutApp"      xmlns:d="<http://schemas.microsoft.com/expression/blend/2008>"      xmlns:mc="<http://schemas.openxmlformats.org/markup-compatibility/2006>"      mc:Ignorable="d">        <StackPanel Orientation="Horizontal">          <Rectangle Fill="LightBlue" Width="60" />          <Rectangle Fill="LightGreen" Width="60" />          <Rectangle Fill="LightPink" Width="60" />      </StackPanel>  </Page> |



При горизонтальной ориентации все вложенные элементы располагаются слева направо. Если мы хотим, чтобы наполнение стека начиналось справа налево, то необходимо задать свойство FlowDirection="RightToLeft":

|  |  |
| --- | --- |
|  | <StackPanel Orientation="Horizontal" FlowDirection="RightToLeft">      <Rectangle Fill="LightBlue" Width="60" />      <Rectangle Fill="LightGreen" Width="60" />      <Rectangle Fill="LightPink" Width="60" />  </StackPanel> |



По умолчанию это свойство имеет значение LeftToRight - то есть слева направо.

Нередко бывает, что StackPanel с элементами не вмещается в область графического окна. И в этом случае StackPanel помещается в ScrollView, который создает полосы прокрутки.

# SplitView

Элемент SplitView представляет контейнер, который состоит из двух панелей: Pane и Content. Панель Content представляет собой основное содержимое контейнера. Она является открытой. А панель Pane может быть скрыта, при необходимости ее можно раскрывать во всю ширину и скрывать обратно:

|  |  |
| --- | --- |
|  | <Page      x:Class="LayoutApp.MainPage"      xmlns="<http://schemas.microsoft.com/winfx/2006/xaml/presentation>"      xmlns:x="<http://schemas.microsoft.com/winfx/2006/xaml>"      xmlns:local="using:LayoutApp"      xmlns:d="<http://schemas.microsoft.com/expression/blend/2008>"      xmlns:mc="<http://schemas.openxmlformats.org/markup-compatibility/2006>"      mc:Ignorable="d">      <SplitView x:Name="splitView" Background="White" PaneBackground="LightBlue">          <SplitView.Pane>              <StackPanel>                  <TextBlock Text="Выдвижная панель" />              </StackPanel>          </SplitView.Pane>          <SplitView.Content>              <Grid>                  <Button Content="Переключить вид" HorizontalAlignment="Center" Click="Button\_Click" />              </Grid>          </SplitView.Content>      </SplitView>  </Page> |

С помощью свойств SplitView.Pane и SplitView.Content устанавливается содержимое панелей Pane и Content. Через свойство PaneBackground задается цвет панели Pane (стандартное свойство Background относится к панели Content). Чтобы разместить на этих панелях какое-либо содержимое, как правило, в них помещается еще один контейнер. Так, в панели Content размещен контейнер Grid, в котором есть кнопка. У этой кнопки задан обработчик нажатия: Click="Button\_Click". Поэтому перейдем к файлу кода и поместим в него следующий метод:

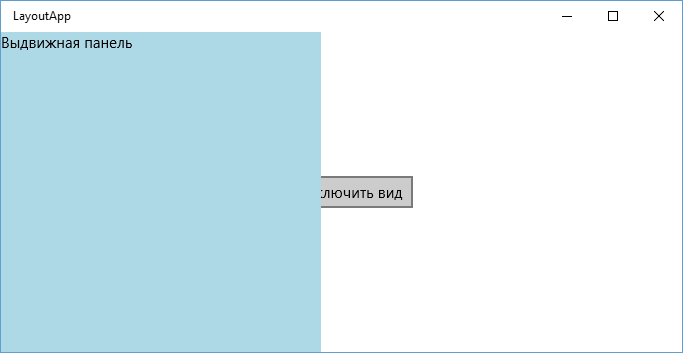
|  |  |
| --- | --- |
|  | private void Button\_Click(object sender, RoutedEventArgs e)  {      splitView.IsPaneOpen = !splitView.IsPaneOpen;  } |

Свойство IsPaneOpen элемента SplitView управляет видимостью панели Pane. Если оно равно true, то панель Pane находится в раскрытом состоянии. И если нам вдруг потребуется, чтобы эта панель была по умолчанию раскрыта, то в разметке XAML у SplitView мы могли бы использовать атрибут IsPaneOpen="True".

Теперь если мы запустим приложения, то мы увидим только содержимое панели Content:



Нажмем на кнопку. У нас сработает обработчик, который переключит состояние панели Pane и тем самым раскроет ее:



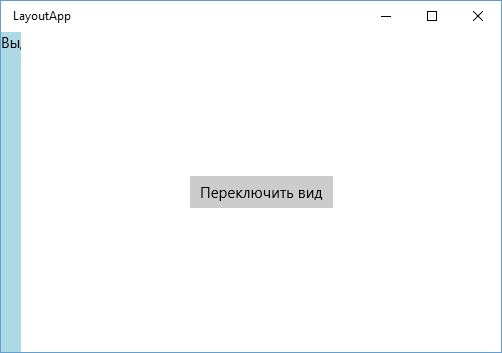
Обратите внимание, что панель Pane частично перекрывает панель Content и кнопку. Это поведение панели Pane по умолчанию, но его можно переопределить, установив у SplitView нужное значение для атрибута DisplayMode:

* Overlay: панель Pane перекрывает панель Content (по умолчанию). Закрыть панель Pane можно, нажав на любое место на панели Content
* Inline: панель Pane сдвигает панель Content. При этом панель Pane нельзя закрыть, нажав на любое место на панели Content, как в случае с Overlay
* CompactOverlay: начальная ширина панели Pane задается с помощью свойства CompactPaneLength, а остальная часть скрывается. При раскрытии панель Pane действует также, как и при режиме Overlay
* CompactInline: начальная ширина панели Pane задается с помощью свойства CompactPaneLength, а остальная часть скрывается. При раскрытии панель Pane действует также, как и при режиме Inline

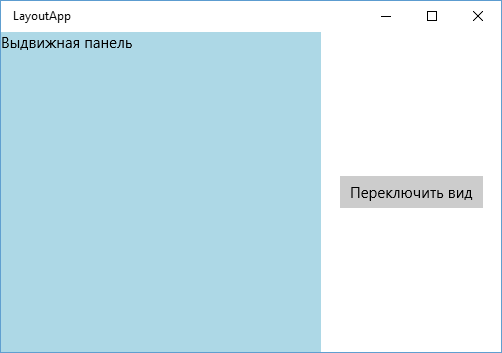
Например, используем режим CompactInline:

|  |  |
| --- | --- |
|  | <SplitView x:Name="splitView" DisplayMode="CompactInline" CompactPaneLength="20"              Background="White" PaneBackground="LightBlue" >  <!-- остальное содержимое -->  </SplitView> |

В скрытом состоянии:



В раскрытом:



Из остальных свойств SplitView следует отметить свойство OpenPaneLength, которое задает ширину панели Pane в раскрытом состоянии.

# RelativePanel

RelativePanel представляет контейнер, в котором можно задать позиционирование элемента относительно другого элемента. Для этого во всех вложенных элементах можно использовать ряд свойств.

Установка позиции относительно другого элемента:

* RelativePanel.Above: размещение над другим элементом
* RelativePanel.Below: размещение под другим элементом
* RelativePanel.LeftOf: размещение слева от другого элемента
* RelativePanel.RightOf: размещение справа от другого элемента

Вторая группа свойств управляет выравниванием одного элемента относительно другого:

* RelativePanel.AlignBottomWith: выравнивает относительно другого элемента по нижнему краю
* RelativePanel.AlignHorizontalCenterWith: выравнивает относительно другого элемента по центру по горизонтали
* RelativePanel.AlignLeftWith: выравнивает относительно другого элемента слева
* RelativePanel.AlignRightWith: выравнивает относительно другого элемента справа
* RelativePanel.AlignTopWith: выравнивает относительно другого элемента сверху
* RelativePanel.AlignVerticalCenterWith: выравнивает относительно другого элемента по центру по вертикали

Обе эти группы свойств принимают в качестве значения имя другого элемента, относительно которого надо размещать текущий элемент.

И третья группа свойств выравнивает элемент относительно границ самого контейнера RelativePanel:

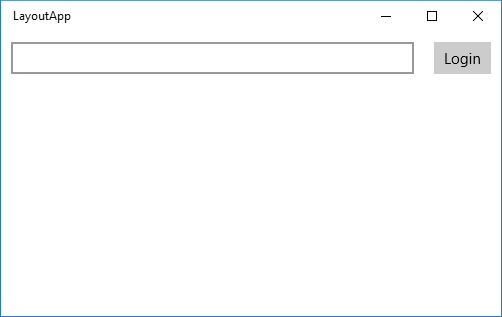
* RelativePanel.AlignBottomWithPanel: при значении true выравнивает элемент по вертикали по нижнему краю панели
* RelativePanel.AlignHorizontalCenterWithPanel: при значении true выравнивает элемент по горизонтали по центру панели
* RelativePanel.AlignLeftWithPanel: при значении true выравнивает элемент по левому краю панели
* RelativePanel.AlignRightWithPanel: при значении true выравнивает элемент по правому краю панели
* RelativePanel.AlignTopWithPanel: при значении true выравнивает элемент по верхнему краю панели
* RelativePanel.AlignVerticalCenterWithPanel: при значении true выравнивает элемент по вертикали по центру панели

Эти свойства принимают логическое значение true или false.

Для достижения нужного эффекта эти свойства можно комбинировать. Например, определим текстовое поле и справа от него кнопку:

|  |  |
| --- | --- |
|  | <Page      x:Class="LayoutApp.MainPage"      xmlns="<http://schemas.microsoft.com/winfx/2006/xaml/presentation>"      xmlns:x="<http://schemas.microsoft.com/winfx/2006/xaml>"      xmlns:local="using:LayoutApp"      xmlns:d="<http://schemas.microsoft.com/expression/blend/2008>"      xmlns:mc="<http://schemas.openxmlformats.org/markup-compatibility/2006>"      mc:Ignorable="d">        <RelativePanel Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">            <Button x:Name="loginButton" Content="Login" Margin="10"                  RelativePanel.AlignTopWithPanel="True"                  RelativePanel.AlignRightWithPanel="True"/>          <TextBox x:Name="loginBox" Margin="10"                   RelativePanel.LeftOf="loginButton"                   RelativePanel.AlignLeftWithPanel="True"          />      </RelativePanel>  </Page> |

Здесь кнопка прижата к верхнему и правому краю контейнера RelativePanel. А текстовое поле прижато к левой границе контейнера и размещается слева от кнопки.



**Задание лабораторной работы**

Вариант: 20

В одномерном массиве, состоящем из n целых элементов, вычислить:

1) минимальный по модулю элемент массива;

2) сумму модулей элементов массива, расположенных после первого элемента, равного нулю.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в четных позициях, а во второй половине – элементы, стоявшие в нечетных позициях.

**Скриншоты лабораторной работы**

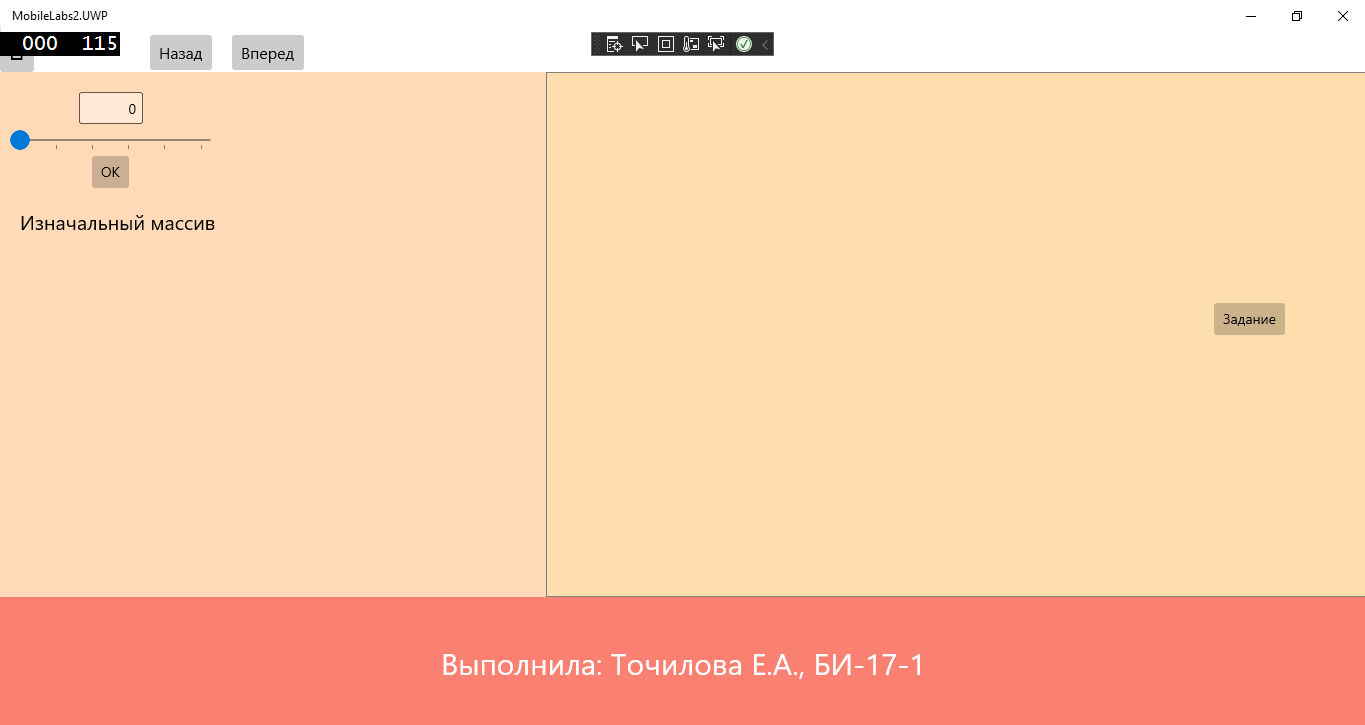


Рисунок . Работа программы 1

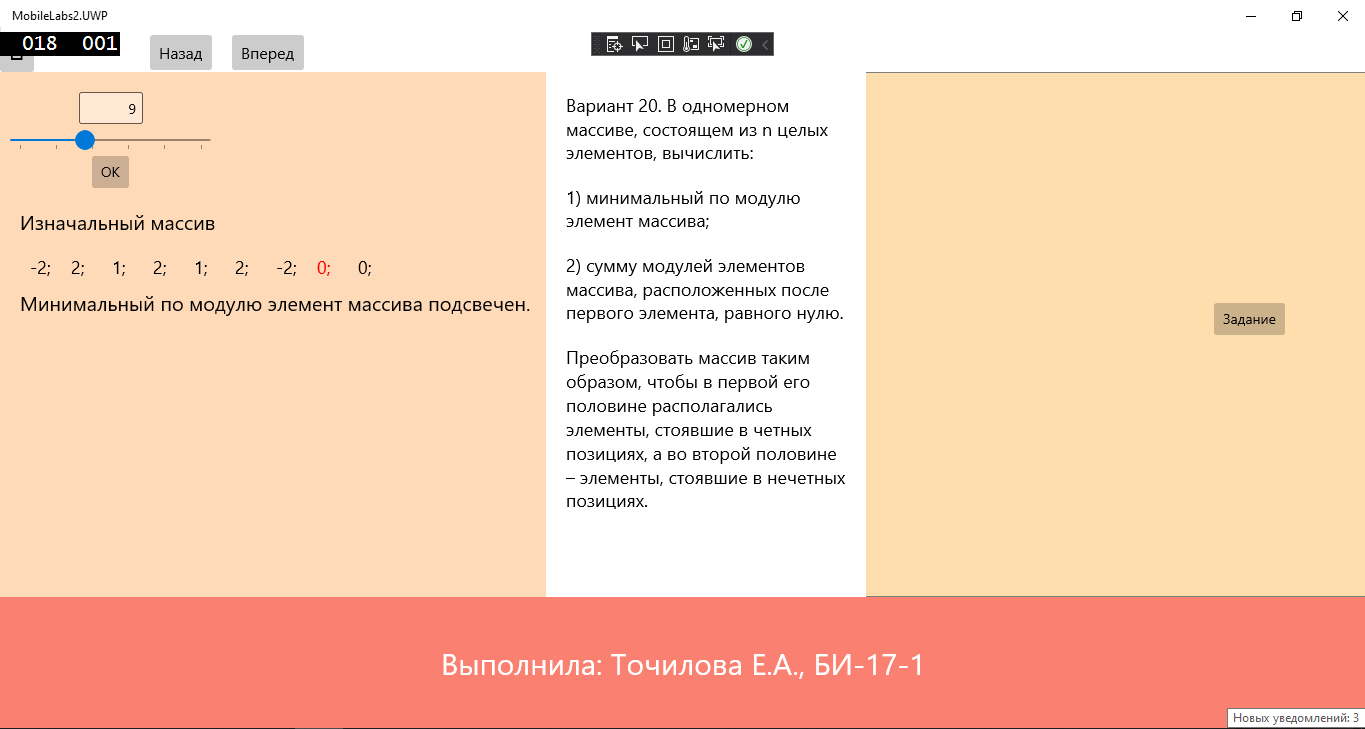


Рисунок . Работа программы 2

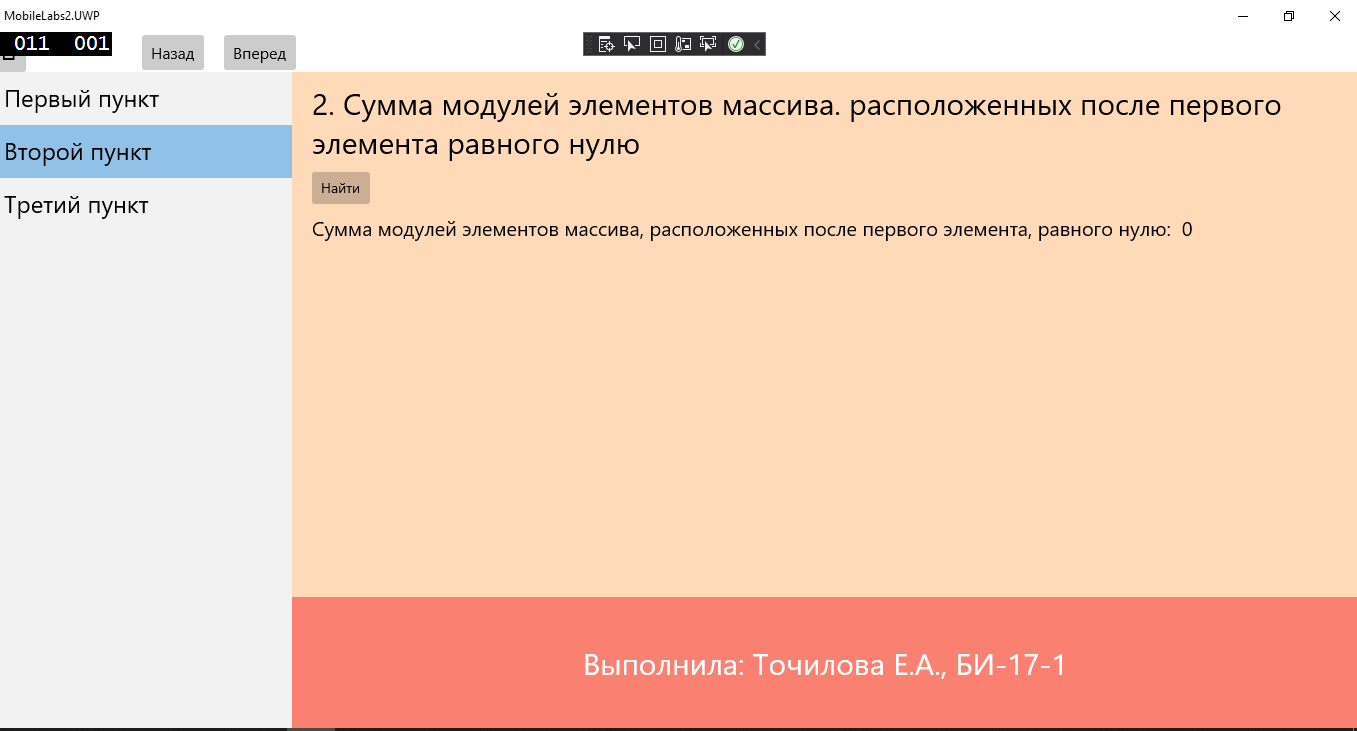


Рисунок . Работа программы 3

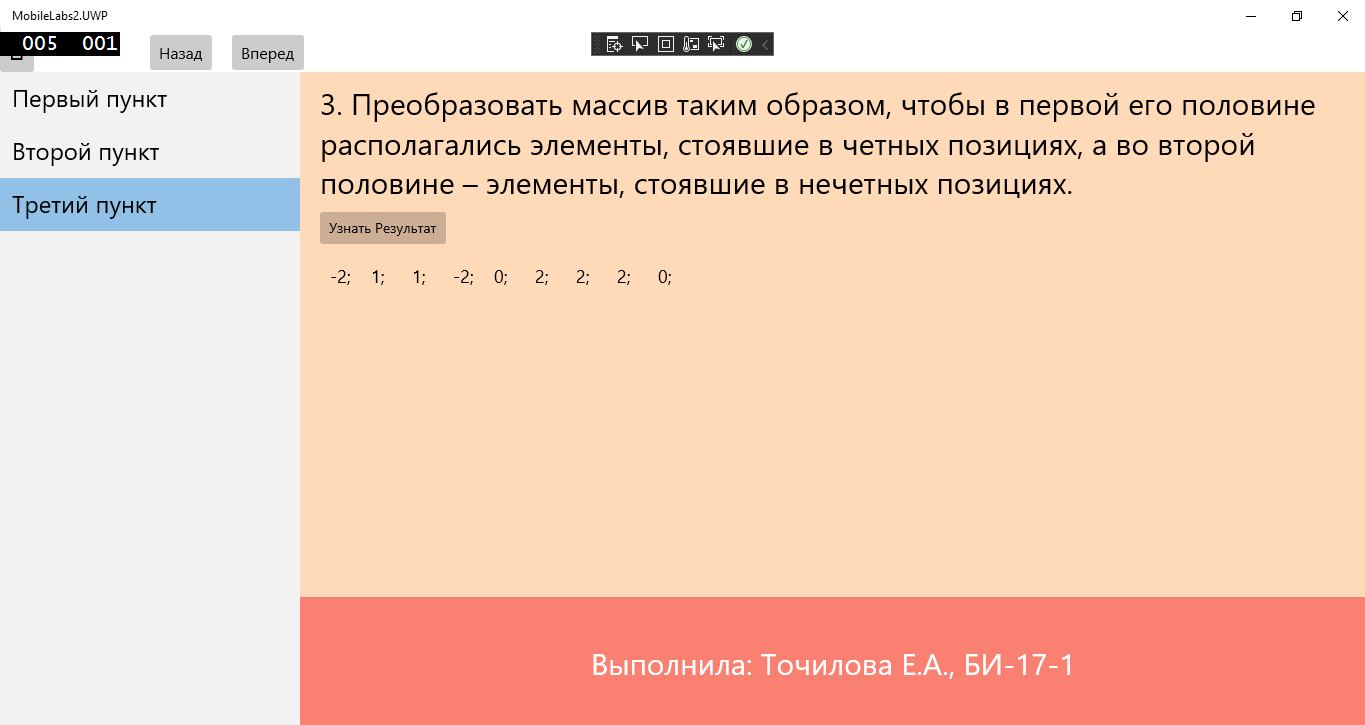


Рисунок . Работа программы 4

**Код программы MainPage.xaml.cs**

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

namespace MobileLabs2.UWP

{

public sealed partial class MainPage

{

internal ObservableCollection<StartList> stList1 { get; set; }

public MainPage()

{

this.InitializeComponent();

stList1 = new ObservableCollection<StartList>();

frx.Navigate(typeof(Page1), stList1);

}

private void ListBox\_SelectionChanged\_1(object sender, SelectionChangedEventArgs e)

{

var t = (ListBoxItem)e.AddedItems[0];

if (t.Name == "p1")

{

frx.Navigate(typeof(Page1), stList1);

}

if (t.Name == "p2")

{

frx.Navigate(typeof(Page2), stList1);

}

if (t.Name == "p3")

{

frx.Navigate(typeof(Page3), stList1);

}

}

private void Btn1\_Click(object sender, RoutedEventArgs e)

{

spw1.IsPaneOpen = !spw1.IsPaneOpen;//Код обработчика событий кнопки

}

private void BtnBack\_Click(object sender, RoutedEventArgs e)

{

if (frx.CanGoBack) frx.GoBack();//возврат на предыдущую страницу

}

private void BtnNext\_Click(object sender, RoutedEventArgs e)

{

if (frx.CanGoForward) frx.GoForward();//переход на следующую страницу

}

}

}

**Код программы MainPage.xaml**

<forms:WindowsPage

x:Class="MobileLabs2.UWP.MainPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:forms="using:Xamarin.Forms.Platform.UWP"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="using:MobileLabs2.UWP"

Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="40"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="40"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Button Grid.Column="0" Grid.Row="0" x:Name="btn1" FontFamily="Segoe MDL2

Assets" FontSize="24" Content="&#xE700;" Click="Btn1\_Click"/>

<StackPanel Orientation="Horizontal" Grid.Column="1"

Grid.Row="0" Margin="100 0 0 0">

<Button x:Name="btnBack" FontSize="16" Content="Назад"

Margin="10 0 10 0" Click="BtnBack\_Click"/>

<Button x:Name="btnNext" FontSize="16" Content="Вперед"

Margin="10 0 10 0" Click="BtnNext\_Click"/>

</StackPanel>

<SplitView Name="spw1" OpenPaneLength="300" Grid.Row="1"

Grid.ColumnSpan="2" DisplayMode="Inline">

<SplitView.Pane>

<ListBox SelectionMode="Single" SelectionChanged="ListBox\_SelectionChanged\_1">

<ListBoxItem Name="p1">

<TextBlock FontSize="24" Text="Первый пункт" />

</ListBoxItem>

<ListBoxItem Name="p2">

<TextBlock FontSize="24" Text="Второй пункт" />

</ListBoxItem>

<ListBoxItem Name="p3">

<TextBlock FontSize="24" Text="Третий пункт" />

</ListBoxItem>

</ListBox>

</SplitView.Pane>

<SplitView.Content>

<Frame x:Name="frx"></Frame>

</SplitView.Content>

</SplitView>

</Grid>

</forms:WindowsPage>

**Код программы Page1.xaml.cs**

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

// Документацию по шаблону элемента "Пустая страница" см. по адресу https://go.microsoft.com/fwlink/?LinkId=234238

namespace MobileLabs2.UWP

{

/// <summary>

/// Пустая страница, которую можно использовать саму по себе или для перехода внутри фрейма.

/// </summary>

public sealed partial class Page1 : Page

{

internal ObservableCollection<StartList> stList1 { get; set; }

public Page1()

{

this.InitializeComponent();

stList1 = new ObservableCollection<StartList>();

}

protected override void OnNavigatedTo(NavigationEventArgs e)

{

if (e.Parameter != null)

{

var p = (ObservableCollection<StartList>)e.Parameter;

stList1 = p;

var n = p.Count;

foreach (StartList s in stList1)

{

int nlen = s.StList.Count();

double minElem;

int minind = 0;

minElem = Math.Abs(s.StList[0]); //изначально берем за минимальный - первый элемент массива

for (int i = 0; i < nlen; i++) //ищем минимальный элемент

{

if (Math.Abs(s.StList[i]) < minElem)

{

minElem = Math.Abs(s.StList[i]);

minind = i;

}

}

startArrGrid.Children.Clear();

for (int i = 0; i < nlen; i++) //вывод сгенерированного массива

{

TextBlock startArrText = new TextBlock();

startArrText.Text = Convert.ToString(s.StList[i]) + "; ";

startArrText.FontSize = 18;

startArrText.Margin = new Thickness(10);

if (i == minind)

{

startArrText.Foreground = new SolidColorBrush(Windows.UI.Colors.Red);

}

startArrGrid.Children.Add(startArrText);

}

}

}

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

spw.IsPaneOpen = !spw.IsPaneOpen;

}

private void ResultBtn\_Click(object sender, RoutedEventArgs e)

{

Random random = new Random();

int nlen = Convert.ToInt32(slValue.Value); // длина массива, которую ввёл пользователь (со слайдера)

double minElem;

List<int> StartL = new List<int>();

int[] startArr = new int[nlen];

startArrGrid.Children.Clear();

int minind = 0;

for (int i = 0; i < nlen; i++) //генерируем массив

{

startArr[i] = random.Next(-2, 3);

StartL.Add(startArr[i]);

}

minElem = Math.Abs(startArr[0]); //изначально берем за минимальный - первый элемент массива

for (int i = 0; i < nlen; i++) //ищем минимальный элемент

{

if (Math.Abs(startArr[i]) < minElem)

{

minElem = Math.Abs(startArr[i]);

minind = i;

}

}

for (int i = 0; i < nlen; i++) //вывод сгенерированного массива

{

TextBlock startArrText = new TextBlock();

startArrText.Text = Convert.ToString(startArr[i]) + "; ";

startArrText.FontSize = 18;

startArrText.Margin = new Thickness(10);

if (i == minind)

{

startArrText.Foreground = new SolidColorBrush(Windows.UI.Colors.Red);

}

startArrGrid.Children.Add(startArrText);

}

Answer.Children.Clear();

TextBlock minElemText = new TextBlock();

minElemText.Text = "Минимальный по модулю элемент массива подсвечен.";

minElemText.FontSize = 20;

minElemText.TextWrapping = TextWrapping.Wrap;

Answer.Children.Add(minElemText);

var sttt = new StartList

{

StList = StartL,

};

stList1.Add(sttt);

}

}

}

**Код программы Page1.xaml**

<Page

x:Class="MobileLabs2.UWP.Page1"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="using:MobileLabs2.UWP"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

<ColumnDefinition Width="3\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="4\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Border Background="#FFDAB9"/>

<Border Grid.Column ="1" Background="#FFDEAD"/>

<Border Grid.Row="1" Grid.ColumnSpan="2" Background="#FA8072"/>

<StackPanel Padding="10 0 0 0" Orientation="Vertical" HorizontalAlignment="Left" VerticalAlignment="Top">

<StackPanel Margin="0 20" Grid.Column="0" Grid.Row="0" Orientation="vertical" HorizontalAlignment="Left" Width="201">

<TextBox Text="{Binding ElementName=slValue, Path=Value, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Right" Width="40" />

<Slider Maximum="25" TickPlacement="BottomRight" TickFrequency="5" Name="slValue" />

<Button x:Name="ResultBtn" Content="ОК" HorizontalAlignment="Center" Click="ResultBtn\_Click" />

</StackPanel>

<TextBlock FontSize="20" Margin="10 0 10 10" HorizontalAlignment="Left" Text="Изначальный массив"></TextBlock>

<VariableSizedWrapGrid Margin="10 0 0 0" HorizontalAlignment="Left" x:Name="startArrGrid" Orientation="Horizontal" MaximumRowsOrColumns="15">

</VariableSizedWrapGrid>

<StackPanel x:Name="Answer" Padding="10 0 0 0" Orientation="Vertical" HorizontalAlignment="Left" VerticalAlignment="Top">

</StackPanel>

</StackPanel>

<StackPanel Grid.Row="1" Grid.ColumnSpan="2" Orientation="Horizontal" HorizontalAlignment="Center" VerticalAlignment="Center">

<TextBlock FontSize="30" TextWrapping="Wrap" Foreground="White" Margin="10" Text="Выполнила: Точилова Е.А., БИ-17-1"/>

</StackPanel>

<SplitView x:Name="spw" PaneBackground="White" Grid.Column="1" Grid.Row="0">

<SplitView.Pane>

<StackPanel HorizontalAlignment="Center" VerticalAlignment="Top" Margin="10">

<TextBlock FontSize="18" Margin="10" TextWrapping="Wrap" Text="Вариант 20. В одномерном массиве, состоящем из n целых элементов, вычислить:"/>

<TextBlock FontSize="18" Margin="10" TextWrapping="Wrap" Text="1) минимальный по модулю элемент массива;" />

<TextBlock FontSize="18" Margin="10" TextWrapping="Wrap" Text="2) сумму модулей элементов массива, расположенных после первого

элемента, равного нулю."/>

<TextBlock FontSize="18" Margin="10" TextWrapping="Wrap" Text="Преобразовать массив таким образом, чтобы в первой его половине располагались

элементы, стоявшие в четных позициях, а во второй половине – элементы, стоявшие в нечетных позициях." />

</StackPanel>

</SplitView.Pane>

<SplitView.Content>

<Grid>

<RelativePanel BorderBrush="Gray" BorderThickness="1">

<Button RelativePanel.AlignRightWithPanel="True" x:Name="btn" Content="Задание" Margin="0 230 80 0" HorizontalAlignment="Center" Click="Button\_Click" />

</RelativePanel>

</Grid>

</SplitView.Content>

</SplitView>

</Grid>

</Page>

**Код программы Page2.xaml.cs**

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

// Документацию по шаблону элемента "Пустая страница" см. по адресу https://go.microsoft.com/fwlink/?LinkId=234238

namespace MobileLabs2.UWP

{

/// <summary>

/// Пустая страница, которую можно использовать саму по себе или для перехода внутри фрейма.

/// </summary>

public sealed partial class Page2 : Page

{

internal ObservableCollection<StartList> stList1 { get; set; }

public Page2()

{

this.InitializeComponent();

stList1 = new ObservableCollection<StartList>();

}

protected override void OnNavigatedTo(NavigationEventArgs e)

{

if (e.Parameter != null)

{

var p = (ObservableCollection<StartList>)e.Parameter;

stList1 = p;

var n = p.Count;

}

}

private void ResultBtn\_Click(object sender, RoutedEventArgs e)

{

foreach (StartList s in stList1)

{

Answer.Children.Clear();

int nlen = s.StList.Count();

int summa = 0;

bool flag = false;

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

{

if (s.StList[i] == 0)

{

flag = true;

}

if (flag)

{

summa += s.StList[i];

}

}

TextBlock SummaText = new TextBlock();

if (!flag)

{

SummaText.Text = "В массиве нет элементов, равных нулю.";

SummaText.FontSize = 20;

SummaText.TextWrapping = TextWrapping.Wrap;

}

else

{

SummaText.Text = "Сумма модулей элементов массива, расположенных после первого элемента, равного нулю: " + Convert.ToString(summa);

SummaText.FontSize = 20;

SummaText.TextWrapping = TextWrapping.Wrap;

}

Answer.Children.Add(SummaText);

}

}

}

}

**Код программы Page2.xaml**

<Page

x:Class="MobileLabs2.UWP.Page2"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="using:MobileLabs2.UWP"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="4\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Border Background="#FFDAB9"/>

<Border Grid.Row="1" Grid.ColumnSpan="2" Background="#FA8072"/>

<RelativePanel>

<TextBlock x:Name="t1" TextWrapping="Wrap" FontSize="30" Margin="20 10 0 0" Text="2. Сумма модулей элементов массива. расположенных после первого элемента равного нулю"></TextBlock>

<Button x:Name="ResultBtn" RelativePanel.Below="t1" Content="Найти" HorizontalAlignment="Left" Margin="20 10 0 0" Click="ResultBtn\_Click" />

<StackPanel x:Name="Answer" RelativePanel.Below="ResultBtn" Margin="20 10 0 0" Orientation="Vertical" HorizontalAlignment="Left" VerticalAlignment="Top">

</StackPanel>

</RelativePanel>

<StackPanel Grid.ColumnSpan="1" Grid.Row="1" Orientation="Horizontal" HorizontalAlignment="Center" VerticalAlignment="Center">

<TextBlock FontSize="30" TextWrapping="Wrap" Foreground="White" Margin="10" Text="Выполнила: Точилова Е.А., БИ-17-1"/>

</StackPanel>

</Grid>

</Page>

**Код программы Page3.xaml.cs**

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

// Документацию по шаблону элемента "Пустая страница" см. по адресу https://go.microsoft.com/fwlink/?LinkId=234238

namespace MobileLabs2.UWP

{

/// <summary>

/// Пустая страница, которую можно использовать саму по себе или для перехода внутри фрейма.

/// </summary>

public sealed partial class Page3 : Page

{

internal ObservableCollection<StartList> stList1 { get; set; }

public Page3()

{

this.InitializeComponent();

stList1 = new ObservableCollection<StartList>();

}

protected override void OnNavigatedTo(NavigationEventArgs e)

{

if (e.Parameter != null)

{

var p = (ObservableCollection<StartList>)e.Parameter;

stList1 = p;

var n = p.Count;

}

}

private void ResultBtn\_Click(object sender, RoutedEventArgs e)

{

foreach (StartList s in stList1)

{

int nlen = s.StList.Count();

List<int> A = new List<int>();

List<int> B = new List<int>();

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

{

if (i % 2 == 0) //если элемент четный

{

A.Add(s.StList[i]);//добавляем элемент в лист А

}

else //если элемент нечетный

{

B.Add(s.StList[i]);//добавляем элемент в лист B

}

}

A.AddRange(B);

List<int> ResultList = A; //Добавляем в конец листа А лист B и записываем результат в ResultList

resultArrGrid.Children.Clear();

for (int i = 0; i < nlen; i++)//вывод переделанной матрицы

{

TextBlock resultTxt = new TextBlock();

//resultTxt.Text = ResultList[i].ToString("#.0") + ";";

resultTxt.Text = ResultList[i].ToString() + "; ";

resultTxt.FontSize = 18;

resultTxt.Margin = new Thickness(10);

//resultTxt.Width = Double.NaN;

resultArrGrid.Children.Add(resultTxt);

}

}

}

}

}

**Код программы Page3.xaml**

<Page

x:Class="MobileLabs2.UWP.Page3"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="using:MobileLabs2.UWP"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="4\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Border Background="#FFDAB9"/>

<Border Grid.Row="1" Grid.ColumnSpan="2" Background="#FA8072"/>

<RelativePanel >

<TextBlock x:Name="t1" TextWrapping="Wrap" FontSize="30" Margin="20 10 0 0" Text="3. Преобразовать массив таким образом, чтобы в первой его половине располагались

элементы, стоявшие в четных позициях, а во второй половине – элементы, стоявшие в нечетных позициях."></TextBlock>

<Button x:Name="ResultBtn" RelativePanel.Below="t1" Content="Узнать Результат" Margin="20 10 0 0" HorizontalAlignment="Left" Click="ResultBtn\_Click" />

<VariableSizedWrapGrid Margin="20 10 0 0" RelativePanel.Below="ResultBtn" HorizontalAlignment="Left" x:Name="resultArrGrid" Orientation="Horizontal" MaximumRowsOrColumns="30">

</VariableSizedWrapGrid>

</RelativePanel>

<StackPanel Grid.ColumnSpan="2" Grid.Row="1" Orientation="Horizontal" HorizontalAlignment="Center" VerticalAlignment="Center">

<TextBlock FontSize="30" TextWrapping="Wrap" Foreground="White" Margin="10" Text="Выполнила: Точилова Е.А., БИ-17-1"/>

</StackPanel>

</Grid>

</Page>

**Код программы StartList.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MobileLabs2.UWP

{

class StartList

{

public List<int> StList { get; set; }

}

}