**Задание 16.**

Разработать учебное приложение рабочего стола (*Desktop Application*) с простым графическим пользовательским интерфейсом (*GUI*), который может послужить примером для демонстрации возможностей режима разработки *Windows Forms Application* на уровне обработки событий.

При создании программного обеспечения руководствоваться фрагментами технического задания (ТЗ) на разработку программного обеспечения из перечня, представленного в разделе «Варианты индивидуального задания».

Описание работы взаимосвязанных алгоритмов выполнить при помощи аппарата сетей Петри.

**Примеры событий:**

События можно отличить по пиктограммам в виде молний слева от их наименований в среде разработки (*IDE*).

Экранная форма (*Form*) может быть связана с обработкой событий, перечисленных в Таблице 1.

Таблица 1 – События экранной формы *Windows*

|  |  |  |
| --- | --- | --- |
| *Activated* | *FormClosed* | *MouseDown* |
| *AutoSizeChanged* | *FormClosing* | *MouseEnter* |
| *AutoValidateChanged* | *GiveFeedback* | *MouseHover* |
| *BackColorChanged* | *GotFocus* | *MouseLeave* |
| *BackgroundImageChanged* | *HandleCreated* | *MouseMove* |
| *BackgroundImageLayoutChanged* | *HandleDestroyed* | *MouseUp* |
| *BindingContextChanged* | *HelpButtonClicked* | *MouseWheel* |
| *CausesValidationChanged* | *HelpRequested* | *Move* |
| *ChangeUICues* | *ImeModeChanged* | *PaddingChange* |
| *Click* | *InputLanguageChanged* | *Paint* |
| *ClientSizeChanged* | *InputLanguageChanging* | *ParentChanged* |
| *ContextMenuChanged* | *Invalidated* | *PreviewKeyDown* |
| *ContextMenuStripChanged* | *KeyDown* | *QueryAccessibilityHelp* |
| *ControlAdded* | *KeyPress* | *QueryContinueDrag* |
| *ControlRemoved* | *KeyUp* | *RegionChanged* |
| *CursorChanged* | *Layout* | *Resize* |
| *Deactivate* | *Leave* | *ResizeBegin* |
| *Disposed* | *Load* | *ResizeEnd* |
| *DockChanged* | *LocationChanged* | *RightToLeftChanged* |
| *DoubleClick* | *LostFocus* | *RightToLeftLayoutChanged* |
| *DragDrop* | *MaximizedBoundsChanged* | *Scroll* |
| *DragEnter* | *MdiChildActivate* | *Shown* |
| *DragLeave* | *MenuComplete* | *SizeChanged* |
| *DragOver* | *MenuStart* | *StyleChanged* |
| *EnabledChanged* | *MinimumSizeChanged* | *SystemColorsChanged* |
| *Enter* | *MouseCaptureChanged* | *Validated* |
| *FontChanged* | *MouseClick* | *Validating* |
| *ForeColorChanged* | *MouseDoubleClick* | *VisibleChanged* |

**Варианты индивидуального задания:**

**Вариант 01**. Реализовать смену цвета фона экранной формы (*Form*) по нажатии кнопкой мыши (*Click*) на экранную кнопку (*Button*). Продублировать возможность смены цвета фона по выбранной опции (*RadioButton*). Опции группируются рамкой (*GroupBox*), которая в начальный момент времени после запуска программы невидима (*.Visible = false*). Смена режима управления меняется по логическому признаку (*CheckBox*): если «галочка» снята (*.Checked = false*) – управление производится экранными кнопками (элементы в составе рамки (*GroupBox*) при этом нефункциональны: *.Enabled = false*), если «галочка» выставлена (*.Checked = true*) – управление производится опциями (*RadioButton*). Предусмотреть следующий набор цветов: «Красный», «Зелёный», «Синий», «Фиолетовый», «Оранжевый», «Белый», «Стандартный» («Серый»). Организовать последний в качестве *enum*.

**Вариант 02**. Реализовать функции ввода (*Input*) и очистки (*Clear*) содержимого ярлыка (*Label*) по нажатии кнопкой мыши (*Click*) на соответствующие экранные кнопки (*Buttons*). Ввод содержимого ярлыка (*Label*) осуществляется по следующей схеме: кнопка «Ввод» проявляет (*.Visible = true*) на экранной форме (*Form*) поле для ввода текста (*TextBox*), куда можно ввести произвольный текст. Перенос этого текста на ярлык (*Label*) производится повторным нажатием кнопкой мыши (*Click*) на экранную кнопку (*Button*) «Ввод». При этом для экранной кнопки (*Button*) «Ввод» требуется запретить ввод пустой строки. Очистка ярлыка (*Label*) обеспечивается только нажатием кнопкой мыши (*Click*) на экранную кнопку (*Button*) «Очистить». Организовать вывод стилистически верно оформленных побуждающих сообщений (*MessageBox*) для всех случаев обработки исключительных расчётных ситуаций, а также для случаев некорректных действий пользователя.

**Вариант** **03**. Двойной клик мышью (*Double Click*) по фону экранной формы (*Form*) последовательно заполняет логические признаки (*CheckBox*), размещённые на экранной форме (*Form*) в количестве шести штук. По заполнении всех логических признаков (*CheckBox*) все ранее выставленные «отметки»-«галочки» сбрасываются (*.Checked = false*). Последующие двойные клики мышью (*Double Click*) заново последовательно заполняют логические признаки (*CheckBox*). Количество двойных кликов (*Double Click*) в программе ничем не ограничено. Двойной клик, маскированный клавишей «*Shift*», сбрасывает все логические признаки в исходное состояние. Факт принудительного сброса в исходное состояние логических признаков, а также перехода в исходное состояние через заполнение максимума сопроводить корректно оформленными побуждающими сообщениями (*MessageBox*), чётко указывающими на детали произошедшего события.

**Вариант** **04**. Выпадающий список (*ComboBox*) определяет указанный цвет фона экранной формы (*Form*). Стиль выпадающего содержимого списка (*DropDownStyle*) должен быть определён как выпадающий список (*DropDownList*). Предусмотреть сброс состояния экранной формы (*Form*) к значениям по умолчанию, сопровождённый подтверждением намерения пользователя через диалог побуждающего сообщения (*MessageBox*). Учитывать тот факт, что по умолчанию в комбинированном списке (*ComboBox*) не выбран ни один элемент (*.SelectedIndex = -1*) и цвет фона экранной формы (*Form*) – системный серый. Учесть особенность, что выпадающий список без сопровождения его ярлыком (*Label*), считается недружественным интерфейсным элементом управления.

**Вариант** **05**. Сразу загружаются две экранных формы (*Form* основная и второстепенная) в немодальном режиме (*.Show()*). Второстепенная экранная форма доступна для манипуляций, а элементы управления, расположенные на ней, в начальный момент, – недоступны (*.Enabled = false*). Одна из экранных кнопок (*Button*), расположенных на основной экранной форме, при щелчке по ней левой кнопкой мыши (*Left Click*) делает доступной (*.Enabled = true*) элементы управления второстепенной экранной формы с расположенными на ней двумя экранными кнопками (*Button*), а также две другие экранные кнопки, расположенные на основной экранной форме. Вторая экранная кнопка (*Button*) основной экранной формы призвана запускать второстепенную экранную форму, если та была закрыта пользователем, но, если второстепенная экранная форма открыта, то необходимо уведомить об этом пользователя и не создавать дубликат. Третья экранная кнопка (*Button*) основной экранной формы реализует закрытие приложения, но только в случае, если закрыта второстепенная экранная форма. Если второстепенная экранная форма открыта, то уведомить об этом пользователя и не закрывать приложение. Предусмотренные на второстепенной экранной форме экранные кнопки предназначены для: закрытия второстепенной экранной формы; сброса всех интерфейсных элементов управления в приложении в исходное состояние, соответственно.

**Вариант** **06**. При изменении размеров (*Resize*) экранной формы (*Form*) выводить на ярлык (*Label*) информацию в формате «ширина *х* высота» (*Width x Height*). Ярлык (*Label*) на экранной форме (*Form*) всегда должен быть центрирован и минимальный размер экранной формы (*Form*) не должен становиться меньше ширины (*Width*) ярлыка (*Label*) вместе с содержимым. При попытках пользователя сделать это (сделать форму (*Form*) меньше минимально допустимой), принудительно выставлять значения минимально допустимых границ (*Width & Height*). Предусмотреть также максимальные значения границ по ширине и высоте экранной формы. В случае превышения текущих значений ширины и высоты экранной формы над установленными границами принудительно выставлять граничные значения в качестве текущих и выдавать побуждающие сообщения (*MessageBox*) об ошибках отдельно: как по превышении границ по ширине, так и по превышении границ по высоте.

**Вариант 07**. Маскировать по зажатой клавише «*Shift*» реакцию на нажатие левой кнопкой мыши (*Left Click*) на экранную кнопку (*Button*). Маскированная экранная кнопка (*Button*) реализует смену в циклическом режиме включение / отключение управляющего функционала (*ControlBox*), расположенного в заглавии экранной формы (*Form*). Под управляющим функционалом понимать стандартное сочетание из трёх стандартных экранных кнопок: сворачивание в трей (минимизация), развёртка на весь экран (максимизация), закрытие. По отключении управляющего функционала делать видимыми (*.Visible = true*) авторские экранные кнопки (*Button*): «Убрать», «Развернуть», «Выйти». При отсутствии маскирующего воздействия по нажатии на экранную кнопку выдавать побуждающее сообщение с подробной инструкцией о дальнейших действиях пользователя, который хочет увидеть весь доступный функционал программного обеспечения. Инструкцию подгружать из текстового файла.

**Вариант** **08**. Предусмотреть экранные кнопки (*Buttons*) «1», «2», «3», «4», «5» и однозначно связать их с соответствующими клавишами на клавиатуре. Выводить на ярлык (*Label*) точное значение надписи, которая нанесена на инициировавшую обработку события, нажатую экранной кнопку (*Button*), которая могла быть активирована либо с клавиатуры, либо по нажатии кнопкой мыши (*Click*). После активации конкретной экранной кнопки (*Button*) нельзя активировать её же повторно (*.Enabled = false*). Если была нажата другая экранная кнопка (*Button*), то заблокированная ранее экранная кнопка разблокируется (*.Enabled = true*). Вместе с изменением текста ярлыка менять его шрифт: «1» – 12 пунктов, «2» – 14 пунктов, «3» – 18 пунктов, «4» – 24 пункта, «5» – 32 пункта, «6» – 48 пунктов. В экранном списке (*ListBox*) вести журнал событий, который станет доказательством того, что в программе отсутствуют повторные срабатывания при попытке обращения к заблокированным элементам с клавиатуры.

**Вариант** **09**. Экранный список (*ListBox*) по нажатии левой кнопкой мыши (*Left Click*) по одному из его элементов (*Items*), выводит содержимое выбранного пункта на ярлык (*Label*). Нажатие правой кнопкой мыши (*Right Click*) при зажатой клавише «*Shift*» по любому из элементов (*Items*) экранного списка очищает содержимое ярлыка (*Label*). Нажатие правой кнопкой мыши (*Right Click*) при зажатой клавише «*Ctrl*» по любому из элементов экранного списка убирает этот элемент из перечня, но только при положительном ответе пользователя в побуждающем сообщении, уведомляющем об активации процесса удаления элемента из экранного списка. Нажатие правой кнопкой мыши (*Right Click*) при зажатых клавишах «*Ctrl*» и «*Shift*» по любому из элементов экранного списка очищает весь перечень. Добавление новых элементов выполнить через текстовое поле (*TextBox*), размещённое в контекстном меню (*ContextMenuStrip*), привязанном к экранному списку, или через текстовое поле, размещённое на дополнительной экранной форме (*Form*), вызываемой через пункт контекстного меню, привязанного к экранному списку. Подтверждение ввода нового элемента через поле для ввода текста выполнить по нажатии на клавишу «*Enter*» на клавиатуре.

**Вариант** **10.** Работа с меню (*MenuStrip*): пункт меню (*MenuStripDropDownItem*) определяет, какой именно из интерфейсных элементов управления (*Controls*) отображать (*.Visible = true*). Состояние отображения (*.Visible*) интерфейсного элемента управления (*Controls*) фиксируется / дублируется «галочкой» (логическим признаком – *CheckBox*) напротив пункта меню (*MenuStripDropDownItem*). Предусмотреть следующие интерфейсные элементы управления: поле для ввода текста (*TextBox*), ярлык (*Label*), опцию/радиокнопку (*RadioButton*).

**Вариант** **11**. Двойной щелчок кнопкой мыши (*Double Click*) по свободной от интерфейсных элементов управления области экранной формы (*Form*) последовательно заполняет логические признаки (*CheckBox*), размещённые в количестве пяти штук таким образом, что они соответствуют количеству выполненных двойных нажатий кнопкой мыши (*Double Click*), представленному в двоичном коде (*Binary Code*) из «галочек». После выполнения максимально доступного количества двойных нажатий кнопкой мыши (*Double Click*) реакция экранной формы (*Form*) на них прекращается. При этом активируется (*.Enabled = true*) дополнительно размещённая экранная кнопка (*Button*), предназначенная для сброса в исходное состояние. После нажатия (*Click*) на неё (экранную кнопку – *Button*) счёт количества двойных нажатий (*Double Click*) в приложении может быть начат заново без необходимости его перезапуска, но только по мере подтверждения этого пользователем в рамках побуждающего сообщения. Предусмотреть сценарий, что пользователь может и отказаться от выполнения сброса, который, мог быть инициирован случайно.

**Вариант** **12**. Попытка наведения курсора мыши (*MouseHover*) на экранную кнопку (*Button*) приводит к её перемещению в случайную позицию (*RandomPosition*) на экранной форме (*Form*), но укладывающуюся в область видимости экранной формы (*Form*). Недопустимы ситуации, при которых вся экранная кнопка или её часть скрывается за пределами видимой области экранной формы. Размеры экранной формы могут меняться пользователем в процессе работы приложения. Отключить «убегание» экранной кнопки возможно по нажатии двойным щелчком мыши (*Double Click*) в свободной от интерфейсных элементов управления области экранной формы (*Form*).

**Вариант** **13**. Начала некоторых слов собраны в одном экранном списке (*ListBox*), концы тех же слов – в другом экранном списке (*ListBox*). Каждый раз после потери фокуса одним из экранных списков начала и концы слов появляются в экранных списках (*ListBox*) в случайном порядке (*Random*), но важно, чтобы при выборе из экранного списка (*ListBox*) начала некоторого слова, одновременно подсвечивался в другом экранном списке (*ListBox*) и конец того же слова. Срабатывание функций так же хорошо должно вестись и наоборот: при выборе конца слова в соответствующем экранном списке (*ListBox*) должно подсвечиваться и начало того же слова в другом экранном списке (*ListBox*). Потерю фокуса списками учитывать не при переходе от одного списка к другому, а при переходе только к тестовому полю (*TextBox*). В текстовом поле (*TextBox*) должно прописываться выбранное слово полностью, составленное из фрагментов.

**Вариант** **14.** Выбор даты в специальном интерфейсном элементе управления (*DateTimePicker*) сопровождается отображением её (даты) на ярлыке (*Label*) в указанном опцией (*RadioButton*) формате. Опции (*RadioButtons*) группируются в контейнере (*GroupBox*). Предусмотреть следующие форматы: «дд/мм/гггг», «дд месяц гггг», «дд-е число месяца гггг-го года». Использовать класс *DateTime*. Рассмотреть возможность форматирования при выводе *DateTime* посредством метода *.ToString()*. Например, *.ToString("ddVVyyyy")* для 28 февраля 2023 будет соответствовать выводу "28022023".

**Вариант 15**. Реализовать смену надписи в заглавии экранной формы (*Form*) по нажатии кнопкой мыши (*Click*) на одну из экранных кнопок (*Button*) **ИЛИ** при смене позиции отмеченного логического признака (*CheckBox*). Кнопки группируются рамкой (*GroupBox*), которая в начальный момент времени после запуска программы видима (*.Visible = true*). Логические признаки (их три) группируются рамкой (*GroupBox*), которая в начальный момент времени после запуска программы невидима (*.Visible = false*). Смена режима управления меняется при выборе одной из опций (*RadioButton*), группирующихся ещё одной рамкой (*GroupBox*), которая в начальный момент времени после запуска программы видима (*.Visible = true*):

– если выбрана левая опция (в начальной момент выбрана) (*RadioButton*) – управление ведётся кнопками (*Buttons*) (элементы управления (*Controls*) в составе рамки (*GroupBox*) с логическими признаками (*CheckBox*) нефункциональны: *.Enabled = false*),

– если выбрана правая опция (*RadioButton*) – управление ведётся логическими признаками (*CheckBox*).

Предусмотреть следующие надписи: «Автомат», «Полуавтомат», «Ручное».

**Вариант 16.** Реализовать функции ввода (*Input*) и очистки (*Clear*) содержимого ярлыка (*Label*) по нажатии кнопкой мыши (*Click*) на соответствующие экранные кнопки (*Buttons*). Ввод осуществляется по следующей схеме: кнопка «Ввод» проявляет (*.Visible = true*) на экранной форме (*Form*) комбинированный список (*ComboBox*), из которого можно выбрать одно из наименований цветов («Красный», «Зелёный», «Синий», «Фиолетовый», «Оранжевый», «Белый»). Выбор конкретного цвета переносит его наименование на ярлык (*Label*) и скрывает комбинированный список (*.Visible = false*). При этом в комбинированном списке не существует строки «(пусто)». Очистка ярлыка (*Label*) обеспечивается только нажатием на кнопку «Очистить». Стиль выпадающего содержимого списка (*DropDownStyle*) должен быть определён как выпадающий список (*DropDownList*).

**Вариант 17**. Двойной клик мышью (*Double Click*) по фону экранной формы (*Form*) последовательно перемещает опцию (*RadioButton*) в группе (*GroupBox*), размещённой на экранной форме (*Form*), на следующую позицию (одну из трёх). По проследовании всех позиций опция (*RadioButton*) возвращается на первую позицию. Числовой счётчик (*NumericUpDown*) увеличивает значение на единицу. Последующие двойные клики мышью (*Double Click*) заново последовательно прогоняют опцию (*RadioButton*) по позициям. Количество учитываемых двойных кликов (*Double Click*) в программе ограничено достижением счётчиком значения, равного десяти. Дальнейшие двойные клики (*Double Click*) не меняют позицию опции (*RadioButton*).

**Вариант 18**. Экранный список (*ListBox*) определяет указанный цвет фона экранной формы (*Form*). Предусмотреть сброс состояния экранной формы (*Form*) к значениям по умолчанию. Учитывать тот факт, что по умолчанию в экранном списке (*ListBox*) не выбран ни один элемент и цвет фона экранной формы (*Form*) – системный серый. Размер экранного списка (*ListBox*) на экранной форме (*Form*) подобрать строго по содержимому. Ни один цвет не должен быть скрыт и снизу не должно оставаться избыточно много места. Полоса прокрутки (*Scroll*) экранного списка (*ListBox*) не должна активироваться ни в одном из случаев работы с экранной формой (*Form*).

**Вариант 19**. Двойной щелчок правой кнопкой мыши (*Double Right Click*) по экранной форме (*Form*) делает доступной (*.Enabled = true*) первую из трёх экранную кнопку (*Button*). После этого экранная форма (*Form*) более никак не реагирует на щелчки мыши. После нажатия на ставшую активной кнопку она становится недоступной (*.Enabled = fasle*), но активируется вторая кнопка (*Button*). При нажатии на вторую кнопку становится она становится недоступной (*.Enabled = false*), но активируется третья (*.Enabled = true*). При нажатии на третью экранную кнопку (*Button*) поступает предложение (*MessageBox*) о закрытии экранной формы (*Form*), а вместе с ней и приложения (*Application*). При подтверждении приложение закрывается, при отказе – сбрасывается в исходное состояние.

**Вариант 20**. Маскировать по зажатому сочетанию клавиш «*Shift*» + «*Ctrl*» реакцию на нажатие правой кнопкой мыши (*Right Click*) на экранную кнопку (*Button*). Экранная кнопка (*Button*) реализует смену в циклическом режиме включение / отключение функционала, расположенного в заглавии экранной формы (*Form*). Предусмотреть экранную кнопку (*Button*) «Сброс», которая вне зависимости от маски или нажатия на конкретную кнопку мыши (*Click*) будет реализовывать перевод экранной формы (*Form*) в исходное состояние.

**Вариант 21**. Работа с меню (*MenuStrip*). Предусмотреть следующую схему функционирования: в меню (*MenuStrip*) 5 разделов (*MenuStripItems*). В исходном состоянии отображён (*.Visible = true*) только 1 раздел (*MenuStripItem*), в котором содержится один выпадающий элемент (*MenuStripDropDownItem*) «Раскрыть следующий». Нажатие на выпадающий элемент (*MenuStripDropDownItem*) проявляет (*.Visible = true*) второй раздел (*MenuStripItem*) меню (*MenuStrip*), в котором содержатся два элемента «Скрыть текущий» (*.Visible = false*), «Раскрыть следующий» (*.Visible = true*). Аналогичное содержание в третьем и четвёртом разделах (*MenuStripItems*). Пятый раздел (*MenuStripItem*) содержит только «Скрыть текущий» (*.Visible = false*).

**Вариант 22**. Выставление (*.Checked = true*) логического признака (*CheckBox*) добавляет слово, указанное возле этого логического признака (*CheckBox*), к содержимому поля для ввода текста (*TextBox*) в порядке выставления логического признака (*CheckBox*), снятие логического признака удаляет слово, указанное возле логического признака (*CheckBox*) из поля для ввода текста (*TextBox*) вне зависимости от того, каким по счёту это слово было введено в поле для ввода текста (*TextBox*). Предусмотреть экранную кнопку (*Button*) «Ввод», реализующую перенос составленной строки на ярлык (*Label*). Поле для ввода текста (*TextBox*) не должно реагировать на попытки ввода текста в него с клавиатуры (*KeyPress*).