ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ
БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
**ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ**

Стефанова И. А.
Стефанов М. А.

Технология программирования

Методические указания по выполнению курсовых работ

Самара - 2015

ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ
Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования
«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра программного обеспечения и управления в телекоммуникационных системах

И.А. Стефанова
М.А. Стефанов

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

Методические указания
по выполнению курсовых работ

Самара
2015

УДК 004.42
ББК
С79

Рекомендовано к изданию методическим советом ПГУТИ,
протокол № 5 , от 26.01.2015г.

Стефанова И.А., Стефанов М.А.
С Технология программирования [Текст]: методические указания по выполнению курсовых работ / И.А. Стефанова, М.А. Стефанов. – Самара: ПГУТИ, 2015. – 44 с.

Учебное пособие «Технология программирования: методические указания по выполнению курсовых работ» способствует приобретению базовых знаний в области программирования на языке C# и практических навыков работы в интегрированной среде разработки Microsoft Visual Studio, среде технических расчетов MATLAB (с пакетом расширения для моделирования Simulink). Содержит задания и методические рекомендации по выполнению курсовой работы, а так же примеры выполнения отдельных заданий.

Пособие составлено в соответствии с ФГОС ВПО по направлению подготовки «230200 – Информационные системы и технологии» и предназначено для студентов 1 курса заочного обучения для выполнения курсовой работы.

Оглавление

Введение

Методическая разработка предназначена для использования в качестве пособия при выполнении курсовой работы по дисциплине «Технология программирования» студентами заочного отделения специальности 09.03.02 (230200.62) «Информационные системы и технологии». Работа направлена на приобретение базовых знаний в области программирования и практических навыков работы с программным инструментарием компьютерных информационных технологий.

Рекомендуемая литература:

Основная:

1. Павловская, Т. А. C#. Программирование на языке высокого уровня [Текст] : [учеб. для вузов] / Т. А. Павловская. - СПб. : Питер, 2013. - 432 с. - (Учебник для вузов).
2. Троелсен, Э. Язык программирования C# 2010 и платформа .NET 4, [Текст]: пер. с англ. / Троелсен, Э. - М.: ИД Вильямс, 2011.- 1392 с.
3. Шилдт, Г. C# 4.0: Полное руководство. [Текст]: пер. с англ. / Шилдт, Г. – М.: ИД Вильямс, 2011. – 1056 с.

Дополнительная:

4. Макки, А. Введение в .NET 4.0 и Visual Studio 2010 для профессионалов [Текст]: пер. с англ. / Макки, А. - М.: "Вильямс", 2010.- 412с.
5. Нейгел, К. C# 2008 и платформа .Net 3.5 для профессионалов. [Текст]: – М. Диалектика, 2009, 1392 с.
6. Рихтер, Дж**.** CLR via C#. Программирование на платформе Microsoft. NET Framework 4.5 на языке C# [Текст] / Дж. Рихтер ; пер. Е. Матвеев = CLR via C# / Richter, J. - 4-е изд. - СПб. : Питер, 2014. - 897 с.

1. Цели и задачи курсовой работы

Развить практические навыки работы с современными информационными технологиями. Получить представление о создании табличных и графических объектов на языках высокого уровня.

2. Постановка задачи курсовой работы

Используя современные программные обеспечения создать программу для вычисления конвертируемости рубля России в валюту заданных стран с учетом комиссии и вывода графика полученных валютных значений.

3. Задание на выполнение

Вариант задания курсовой работы определяется последней **M** и предпоследней **N** цифрами номера студенческого билета и выбирается из табл. 1 – 4. В курсовой работе необходимо выполнить следующее:

3.1. В Visual Studio (2010 – 2013) Express на языке программирования C# создать проект, который:

–вычисляет величину конвертируемого рубля России в валюты разных стран,

–вычисляет банковские отчисления,

–определяет выдачу итоговой суммы клиенту в таблице и графически,

–выполняет задачи по типовым операциям с массивами.

Исходными данными для написания программы являются курсы валют заданных пяти стран, процентные ставки банка и атрибуты графиков, представления суммы конверсии. В табл. 1 приведены цифровые коды стран и процентные ставки банков в соответствии с номером варианта, а в табл. 2 – полный список стран и валютных курсов, из которых формируются варианты. Курсы иностранных валют установлены банком РФ на декабрь 2014 г. На момент выполнения работы курс валюты можно уточнить по адресу http://www.cbr.ru/currency_base .

Таблица 1

M	**N**	Наименование	Задания по варианту				
0		Цифровой код	840	036	826	949	392
	0	Процентная ставка	5	3	4	2	6
1		Цифровой код	208	840	980	978	756
	1	Процентная ставка	6	2	5	3	4
2		Цифровой код	826	752	840	124	949
	2	Процентная ставка	2	4	5	3	6
3		Цифровой код	208	978	756	840	980
	3	Процентная ставка	2	5	6	4	3
4		Цифровой код	840	356	826	392	398
	4	Процентная ставка	4	5	3	6	2
5		Цифровой код	398	840	036	978	702
	5	Процентная ставка	6	4	2	3	5
6		Цифровой код	826	124	840	578	208
	6	Процентная ставка	5	4	6	2	3
7		Цифровой код	578	978	702	840	980
	7	Процентная ставка	4	5	2	3	6
8		Цифровой код	840	156	826	756	978
	8	Процентная ставка	6	5	4	3	2
9		Цифровой код	702	840	392	978	124
	9	Процентная ставка	4	3	5	2	6

Таблица 2

Цифровой код	Буквенный код	Единиц	Валюта	Курс* рубля
036	AUD	1	Австралийский доллар	49,54
826	GBP	1	Английский фунт стерлингов	95,03
208	DKK	1	Датских крон	10,01
840	USD	1	Доллар США	60,68
978	EUR	1	Евро	74,57
356	INR	100	Индийских рупий	96,21
398	KZT	100	Казахских тенге	33,32

124	CAD	1	Канадский доллар	52,38
578	NOK	1	Норвежских крон	8,12
156	CNY	10	Китайских юаней	97,55
702	SGD	1	Сингапурский доллар	46,06
949	TRY	1	Турецкая лира	26,13
980	UAH	10	Украинских гривен	38,36
752	SEK	10	Шведских крон	78,58
756	CHF	1	Швейцарский франк	61,84
392	JPY	100	Японских иен	50,83

3.2. Проект должен содержать таблицу для работы с массивами данных (MainForm) и диаграмму, отображающую расчетные показатели (GraphForm). В качестве средства для работы с массивами данных использовать двумерную таблицу, в которую ввести исходные данные: названия валюты, курс рубля, процентные ставки банка, необходимые для подсчета суммы выдачи, а также произвести расчет и вывод результатов вычислений комиссионного сбора банком и суммы выдачи пользователю в соответствующие столбцы таблицы (см. рис. 10).

3.3. По результатам расчета построить диаграмму валютного эквивалента конвертируемой суммы по курсу заданных стран (см. рис. 13). Для построения диаграммы написать программу. Вид диаграммы (гистограмма, линейчатая диаграмма и т.п.) выбрать по своему усмотрению. Исходные данные для диаграммы приведены в табл. 3.

Таблица 3

M	Цвета валютных эквивалентов исходной суммы	Тип и толщина (п.) линии
0	Green, Purple, Navy, Yellow, Maroon	Dash, 3п.
1	Purple, Teal, clFuchsia, Navy, Pink	Dot, 2п.
2	Blue, Red, Navy, Gold, Green	Solid, 3п.
3	Teal, Fuchsia, Purple, Blue, Maroon	DashDot, 2п.
4	Lime, Blue, Yellow, Teal, Coral	DashDotDot, 1п.
5	Fuchsia, Olive, Gold, Purple, Aqua	Dash,2п.
6	Aqua, Purple, Blue, Fuchsia, Navy,	Dot, 3п.
7	Coral, Green, Maroon, Silver, Pink,	Solid, 2п
8	Teal, Lime, Olive, Indigo, Orange	DashDot, 3п.
9	Maroon, SeaGreen, Red, Silver, Yellow	DashDotDot, 2п.

* Отдельные цвета валютных эквивалентов для лучшего дизайна допускается поменять на другие оттенки по Вашему усмотрению.

3.5. На диаграмме отразить валютные эквиваленты заданными цветами, контуры эквивалентов представить заданным типом линии с заданной толщиной, добавить легенду с поясняющими надписями. Для построения диаграммы предусмотреть отдельную форму с компонентами управления.

3.6. Дополнить программу решением задач, приведенных в таблице 4 согласно варианту, предусмотрев для их решения элементы управления и элементы вывода результатов решений. Снабдить элементы интерфейса пояснительными надписями.

3.7. Используя пакет расширений Simulink (математической системы MATLAB), создать модель пересчета денежных сумм, реализующую поставленную задачу (п. 3.1) и вывести статистические показатели по варианту п. 3.6 (1 и 3 задания из табл. 4). **Этот пункт может быть выполнен студентами на практических занятиях.**

Таблица 4

N	Вывести:
0	название валюты, с наибольшим курсом рубля;
	используя поиск, заданную процентную ставку;
	отсортировать массив в порядке возрастания комиссионного сбора;
1	среднюю процентную ставку;
	используя поиск, заданную сумму выдачи;
	отсортировать массив в порядке возрастания курса рубля;
2	наименьший комиссионный сбор банком;
	используя поиск, заданный курс рубля;
	отсортировать массив в порядке возрастания суммы выдачи;
3	наибольшую сумму выдачи по заданным процентным ставкам;
	используя поиск, заданный комиссионный сбор;
	отсортировать массив в порядке возрастания процентных ставок;
4	название валюты, с наименьшим курсом рубля;
	используя поиск, заданный комиссионный сбор;
	отсортировать массив в порядке убывания процентных ставок;
5	наименьшую из заданных процентных ставок;
	используя поиск, заданную сумму выдачи;
	отсортировать массив в порядке убывания курса рубля;
6	среднюю процентную ставку;
	используя поиск, заданный курс рубля;
	отсортировать массив в порядке убывания суммы выдачи;
7	наибольший курс рубля по заданным процентным ставкам;
	используя поиск, заданный курс рубля;
	отсортировать массив в порядке убывания комиссионного сбора;
8	название валюты, с наибольшей суммой выдачи;
	используя поиск, заданный комиссионный сбор;
	отсортировать массив в порядке возрастания процентных ставок;
9	наименьшую из заданных процентных ставок;
	используя поиск, заданный курс рубля;
	отсортировать массив в порядке возрастания суммы выдачи;

3.8. Сравнить полученные результаты и сделать выводы о возможностях среды Visual Studio 2010 Express и системы MATLAB.

4. Требования к оформлению

4.1. Результаты выполнения заданий представляются в электронном варианте на дискете (папка с файлами проекта и файл пояснительной записки, выполненной в текстовом редакторе), а также в виде распечатанной на принтере пояснительной записки. Пояснительная записка сдается на проверку преподавателю, а электронный вариант курсовой работы предоставляется студентом во время ее защиты.

4.2. Пояснительная записка должна содержать:

- титульный лист, с номером зачетной книжки;
- оглавление, созданное с использованием возможностей Office;

- постановку задачи и исходные данные по варианту;

- описание технологии изготовления проекта;

- табличный и графический интерфейс проекта (скомпилированной и запущенной программы);

- исходный код программной реализации поставленных задач (табличная и графическая реализация), снабженный комментариями;

- выводы по работе;

- список используемой литературы.

4.3. Пояснительная записка оформляется в текстовом редакторе на листах формата А4, сброшюрованных в виде папки. При этом:

- обложкой пояснительной записки является титульный лист, форма которого приведена на рис. 1;

- текст пояснительной записки располагается на одной стороне листа и выполняется шрифтом Times New Roman размером 14 пунктов, межстрочный интервал – одинарный. Параметры страниц (поля): слева – 20 мм, справа – 10 мм, сверху и снизу – по 15 мм. Предусмотреть автоматический перенос слов в предложениях, отступ первой строки 1,5 см, выравнивание абзацев по ширине, а заголовков – по центру;

- все страницы, за исключением титульного листа, нумеруются сверху от центра и в следующем порядке: страница 2 – рецензия на работу (оставляется пустой), страница 3 – оглавление с указанием номеров страниц разделов (создается с использованием возможностей Office), страница 4 и т.д. – содержательная часть работы (оформляется с использованием элементов форматирования Office);

- все рисунки и таблицы должны иметь сквозную нумерацию, название и при необходимости пояснительный текст;

- исходный код программы должен содержать необходимый комментарии;

- интерфейсы проекта (табличный и графический) в режимах проектирования и запуска с полученными результатами копируются в пояснительную записку с использованием клавиш Alt+PrtSc.

4.4. Курсовая работа подписывается студентом на титульном листе

ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ
Кафедра ПОУТС

Сдана на проверку
«____»_____201__г.

Допустить к защите
«____»_____201__г.

Защищена с оценкой _____
«____»_____201__г.

Курсовая работа по дисциплине

«Технология программирования»

Табличный и графический способ представления данных на языке высоко уровня C#

Пояснительная записка
на _____ листах

Студент(ка) группы _____ (Ф. И. О.)
(роспись)
Руководитель _____ (Ф. И. О.)
(роспись)
№ зачетной книжки _____

Самара
201__г.

Рис. 1. Образец оформления титульного листа.

5. Вопросы к защите

1. Что такое массив? Способы объявления массивов.
2. Какие операторы языка используют для описания массивов?
3. Как можно произвести инициализацию массива?
4. Сравнить способы ввода элементов массива.
5. Сравнить способы вывода массивов.
6. Охарактеризовать основные свойства компонента dataGridView, используемые в курсовой работе.
7. Перечислить типовые операции с массивами.
8. Пояснить алгоритм поиска максимального/минимального элемента массива.
9. Пояснить алгоритм нахождения суммы и среднего значения.
10. Как осуществляется поиск заданного элемента массива?
11. Пояснить алгоритм сортировки элементов массива.
12. Формат условного оператора if…else.
13. Вложенные условные операторы. Формат записи.
14. Формат оператора перехода goto…
15. Какие действия выполняются операторами условного и безусловного перехода?
16. Итерационный и регулярный циклические процессы.
17. Понятие циклической структуры «до» и «после».
18. Назначение и формат оператора while….
19. Назначение и формат оператора do… while.
20. Преимущества использования операторов цикла в программе.
21. Использование оператора цикла for… инкрементного и декрементного типа.
22. Организация вычисления сумм и произведений.
23. Что такое подпрограмма? Ее назначение.
24. Формальные и фактические параметры.
25. Понятие функций. Правила описания функции и форма вызова.
26. Понятие процедур. Правила описания процедуры и форма вызова.
27. Назначение и структура модуля.
28. Графические возможности C#.
29. Свойства компонентов, используемые при построении графики.
30. Какие объекты используются при построении графики.
31. Как строятся графические изображение в C#?
32. Какие методы используются при программировании графики?
33. Как вывести текстовые надписи на графику?
34. Дать понятие класса, конструктора, интерфейса.
35. Как создать вторую форму и передать управление ей из первой формы?

6. Методические рекомендации

6.1. Создание приложения с графическим интерфейсом

Разработка приложений в Visual C# состоит из двух этапов:

– создание интерфейса приложения (внешний вид формы при выполнении приложения);

– определение функциональности приложения (процедуры, которые выполняются при возникновении определенных событий).

Для создания приложения с графическим интерфейсом для операционной системы **Windows** необходимо выполнить команду Файл ⇒ Создать проект ⇒ Создать **приложение Windows Form** ⇒ ОК**.** Загружается основной экран среды Visual C# Express (рис. 2).

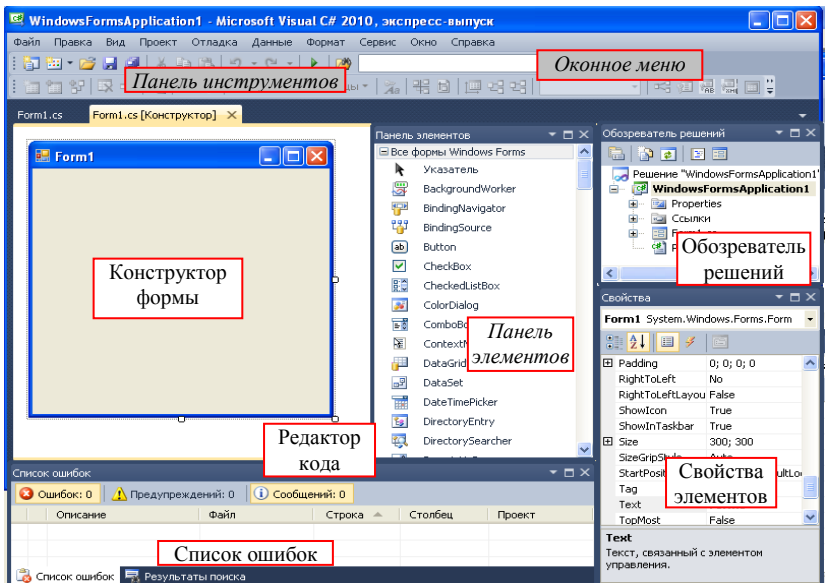


Рис. 2. Основной экран среды Visual C# Express 2010.

Простейшее приложение создается автоматически и представляет собой заготовку, обеспечивающую все необходимое для каждого приложения. Первоначально оно содержит одну форму с заголовком, кнопки управления размерами и закрытием окна. Эта форма ожидает действий пользователя, связанных с мышью и клавиатурой.

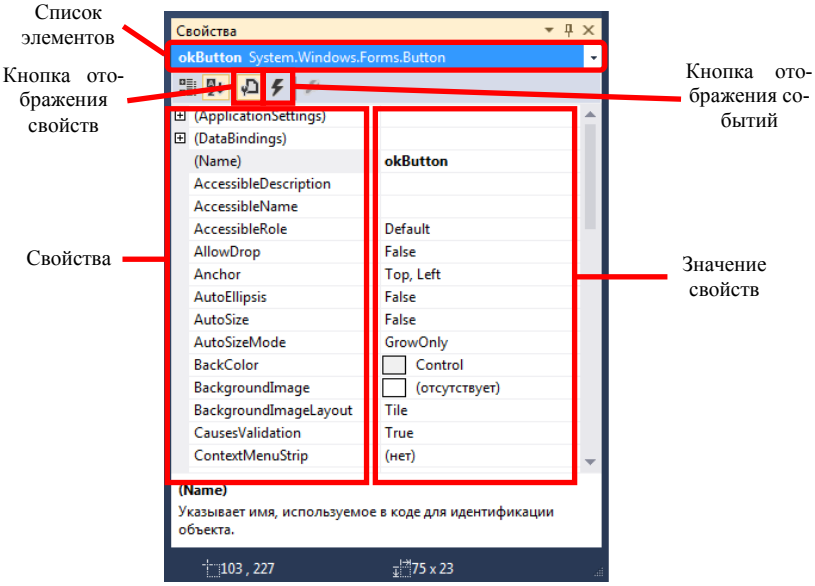
При конструировании приложения разработчик добавляет к простейшему приложению новые формы, управляющие элементы, новые обработчики событий.

Интерфейс пользователя составляют компоненты, которые размещаются на *Панели элементов*. Если по умолчанию эта панель не отображается, необходимо в меню **Вид** выбрать пункт «**Панель элементов**»

Для создания ***интерфейса*** необходимо:

- выбрать нужные объекты на *Панели элементов*;
- определить внешний вид и их функциональные возможности;
- расположить компоненты с помощью мыши на форме.

Для добавления элемента на форму можно либо дважды кликнуть по нему в панели элементов, либо «перетащить на форму» (drug and drop). После чего изменяются их свойства согласно функциональности приложения. Управлять свойствами компонентов можно с помощью *Окна «Свойства»* или программно (если это окно не отображается выберите пункт «**Окно свойств**» в меню **Вид**). Среда Visual C# позволяет легко манипулировать свойствами компонентов, как в режиме проектирования, так и в режиме выполнения программы.



Для изменения параметров элемента необходимо сначала его выбрать. Это можно сделать, кликнув левой кнопкой мыши по элементу в конструкторе формы, или выбрав элемент из списка в окне *Свойства*.

В режиме проектирования для изменения значения свойств выделенного компонента необходимо:

- щелкнуть мышью по имени свойства в левой колонке *окна «Свойства»*;
- выбрать из списка или внести вручную нужное значение в правой колонке *окна свойств* и нажать на клавишу <Enter>.

В режиме выполнения невозможно использовать *Окно свойства элементов*, поэтому все изменения значений свойств компонент осуществляются путем прямой записи операторных строк на языке C#.

Для обеспечения ***функциональности*** приложения необходимо:

- задать в *окне «Свойства»* значения свойств и процедур обработки событий;
- написать программный код для заданных процедур обработки событий.

Реакция на воздействие присуща каждой форме и объектам, размещенным на ней. Каждый компонент имеет стандартный обработчик событий. Список событий для выделенного компонента отображается на вкладке *События окна «Свойства»*.

Для создания процедуры обработки события нужно:

1) выделить на форме объект, для которого создается процедура;
2) перейти на вкладку *События окна «Свойства»*;
3) сделать двойной щелчок по пустому полю в области значения нужного события. Среда автоматически создаст в коде для формы заготовку процедуры-обработчика, и откроет редактор кода;
4) в *Редакторе кода* в месте, где установится курсор, в операторных скобках { } написать код, который будет выполняться при воздействии на этот компонент пользователем.

Например, если вы создали форму и назвали ее *mainForm*, затем в режиме конструктора разместили на ней кнопку и назвали ее *okButton*. Далее выбрали эту кнопку (по-прежнему в режиме конструктора), открыли вкладку «*События*» в окне «*Свойства*» и дважды щелкнули левой кнопкой мыши напротив события «*Click*», то среда автоматически создаст в файле *.cs, содержащем описание класса созданной Вами формы *MainForm*, процедуру обработчик:

```
public partial class MainForm : Form //объявление класса созданной формы
{
  public MainForm() //конструктор класса
  {
     InitializeComponent();
  }
   //обработка события - клика
  private void okButton_Click(object sender, EventArgs e)
  {
    /*здесь будут операторы, выполняемые при нажатии пользователем на эту кнопку*/
  }
}
```

6.2. Представление данных

6.2.1. Функции преобразования.

При вводе и выводе информации в объекты C# как правило используются строки. Для преобразования строк в числа используются либо методы статичного класса *Convert*, либо метод *Parse*(), определенный для всех числовых типов данных. В следующей таблице приведены наиболее часто используемые методы

Функции	Описание
Convert.ToDouble(s)	Вещественное число, изображением которого является строка s
Convert.ToInt16(s)	Целое, изображением которого является строка s
Convert.ToChar(q)	Код символа, изображением которого является символ q
S.ToString()	Строка, являющаяся изображением вещественного s
int.Parse(s)	Целое, изображением которого является строка s
double.Parse(s)	Вещественное, изображением которого является строка s
String.Format(s,x)	Вещественное x, изображением которого является строка s в форматном выводе числа

Примеры использования функций преобразования:

```
int a = int.Parse(quantityTextBox.Text);
/* преобразование текста, введенного в текстовое поле quantityTextBox в целое число */

double b = double.Parse(amountTextBox.Text);
/* преобразование текста, введенного в  amountTextBox в вещественное число */

costTotalLabel.Text = S.ToString()+ " руб.";
/* преобразование числа S в строку с добавлением строки – идентификатора валюты*/

costStaticLabel.Text =String.Format("{0:f 3}",x);
/* преобразование числа x в строку с форматным выводом символов в компонент costStatic-
Label через его свойство Text, где 0 – индекс переменной, f – мантисса, 3 – количество символов после запятой. */

quantityStaticLabel.Text =String.Format("{0,5:f 2}",x);
/* преобразование числа x в строку с форматным выводом символов в компонент quantityStaticLabel через его свойство Text, где 0 – индекс переменной, 5 – количество выводимых символов, f – мантисса, 2 – количество символов после запятой.*/
```

6.2.2. Ввод данных

Ввод данных в C# можно осуществить с помощью полей редактирования однострочного текстового редактора *TextBox.* Ввод данных осуществляется через свойство *Text*.

В качестве примера рассмотрим обработчик события щелчка по кнопке *calculateButton*, реализующий ввод данных в поля редактирования *costTextBox* и *quantityTextBox*, расчет суммы оплаты за приобретенный товар, и вывод результата вычисления в метку *resultLabel*:

```
private void calculateButton_Click(object sender, EventArgs e)
```

```
{
   // преобразование строкового отображения в вещественное
   double cost = double.Parse(costTextBox.Text);
   // преобразование строкового отображения в целое число
   int quantity = int.Parse(quantityTextBox.Text);
   // расчет стоимости и   вывод результата
   double costTotal = cost * quantity;
   costTotalLabel.Text = "S= " +
                         String.Format("{0,8:f2}", S) + " руб.";
}
```

Результат реализации примера приведен на рис. 3.

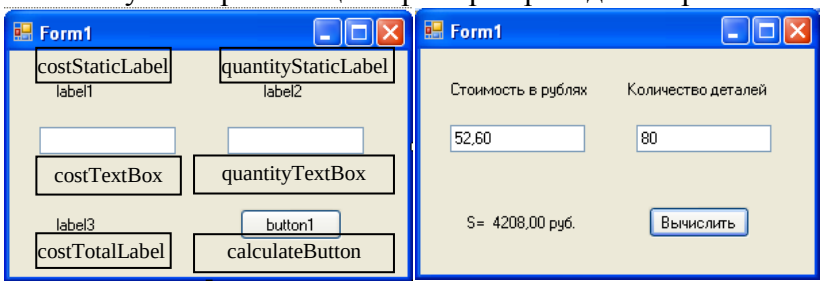


Рис.3. Пример использования полей *costTextBox, quantityTextBox* для ввода данных и метки *costTotalLabel* для вывода данных.

6.2.3. Вывод данных

Наиболее просто можно вывести результат работы в окно сообщения или в поле вывода метку *Label*. При выводе информации в поле метки содержимое вывода определяется значением свойства *Text*, которое можно изменить как во время разработки формы приложения, так и при выполнении программы.

Свойство *Text* символьного типа. Поэтому для вывода в него числового значения, нужно преобразовать число в строку с помощью метода S.ToString(). Интерфейс проекта, реализующий пример вывода результата вычисления, приведен на рис. 3.

Компонент класса Label может так же использоваться для размещения на форме различного рода надписей через свойство Text. С помощью свойства *Font* можно разнообразить вид надписи.

Окна сообщения при выводе информации используется для привлечения внимания пользователя. При этом программа может проинформировать об ошибке в исходных данных или запросить подтверждение выполнения необратимой операции, например удаление файла или вычисления при не корректных данных.

Вывести на экран окно с сообщением можно при помощи функции MessageBox(), которая выводит на экран окно с текстом и командной кнопкой *ОК*.

Кроме того, она позволяет поместить в окно с сообщением:

- один из стандартных значков (x, ?, !, i),
- количество и тип командных кнопок,
- определить, какую из кнопок нажал пользователь.

Формат вызова:

MessageBox.Show (*Сообщение, Заголовок, Кнопки, Тип*),

где *Сообщение* – текст, который будет выведен в окне;

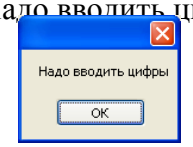
Заголовок – текст в строке заголовка окна;

Кнопки – список кнопок, отображаемых в окне сообщения;

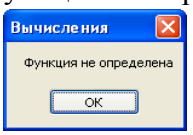
Тип – тип сообщения (информация, предупреждение, ошибка – каждому соответствует свой значок) указывающий какой из значков будет отображен в окне. Тип сообщения задается именной константой, приведенной в таблице. Разные типы сообщения имеют разные значки.

Константа	Тип сообщения	Значок
Warning	Внимание	!
Error	Ошибка	x
Information	Информация	i
Question	Подтверждение	?

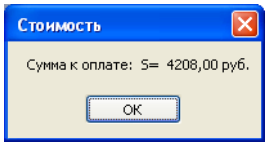
Пример1 (1 аргумента без значка):
MessageBox.Show("Надо вводить цифры");



Пример2 (2 аргумента без значка):
MessageBox.Show("Функция не определена", "Вычисления");



Пример3 (2 аргумента без значка):
MessageBox.Show("Сумма к оплате: " + " S= "+
String.Format("{0,8:f2}", S) + " руб.", "Стоимость ");



Пример4 (4 аргумента):
MessageBox.Show("Функция не определена", "Вычисления",
MessageBoxButtons.OK, MessageBoxIcon.Information);

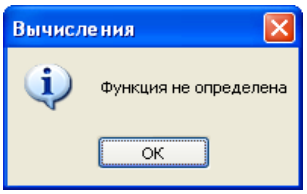


Рис. 4. Примеры использования окна сообщения

6.2.4. Использование компонента dataGridView

При работе с массивами данных удобно использовать компонент dataGridView, предназначенный для создания таблиц, в ячейках которых располагаются произвольные текстовые строки. В таблицах (рис.5) можно производить обработку данных отдельных ячеек, расчет данных во всей таблице, изменение самой таблицы.

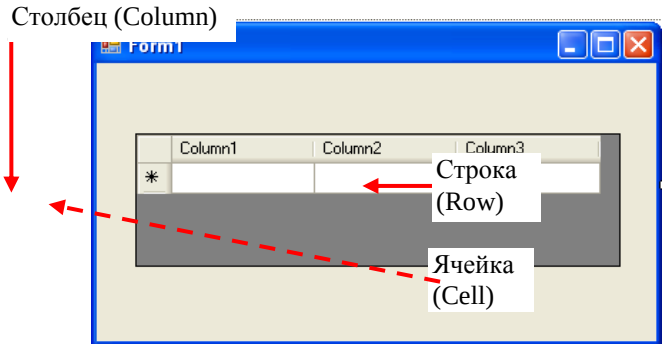


Рис. 5. Структура компонента dataGridView

Таблица делится на две части – фиксированную и рабочую. Фиксированная часть служит для показа заголовков колонок и рядов, а также для ручного управления их размерами. По умолчанию таблица не имеет фиксированных областей, но можно задать их количество. Фиксированные элементы выделяются цветом и при прокрутке информации остаются неподвижными. Рабочая часть – это остальная часть таблицы. Она может содержать произвольное число строк и столбцов. Размеры таблицы определяют свойства ColumnCount и RowCount, задающие, соответственно, число столбцов и строк. По умолчанию оба эти свойства имеют значение 0, (нумерация строк и столбцов начинается с 0).

Для создания колонок необходимо выделить размещенный на форме объект dataGridView1 (имя по умолчанию первого добавленного на форму объекта) и с помощью свойства Columns заполнить коллекцию – из 6 столбцов с соответствующими названиями. При этом использовать поле «Текст заголовка» и кнопку «Добавить» окна «Добавить столбец» (рис. 6).

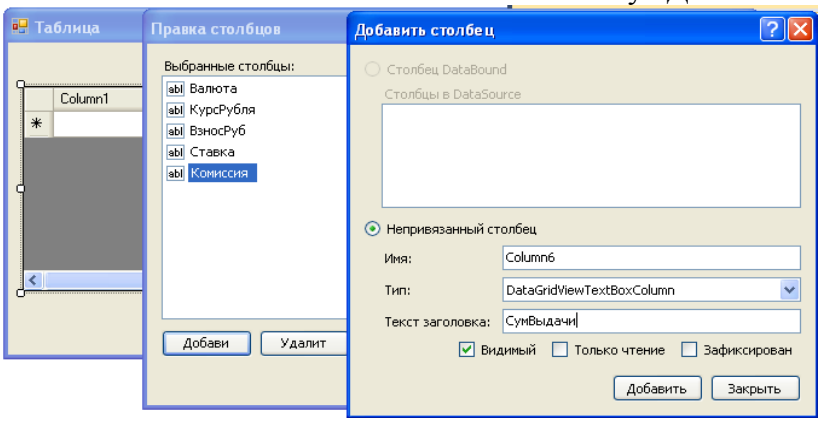


Рис 6. Создание фиксированных столбцов таблицы

После создания колонок интерфейс проекта примет вид, подобный приведенному рисунку 7.

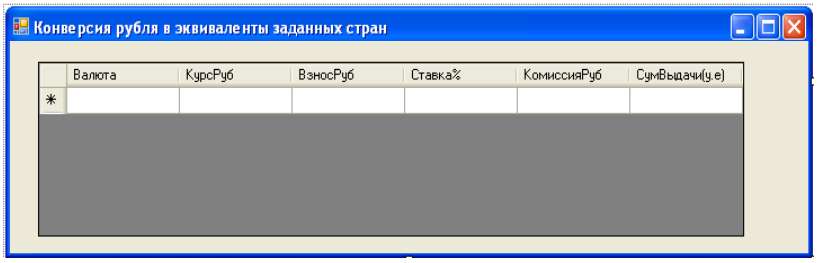


Рис 7. Окно таблицы на этапе проектирования

Чтобы установить несколько строк в таблице, в соответствии с вариантом необходимо их добавить с использованием метода dataGridView.Rows.Add() при открытии класса MainForm.

Центральным свойством таблицы является свойство Rows[n].Cells[k] – двумерный массив ячеек, которое задает ячейку Cell, лежащую на пересечении n-й строки и k-ого столбца. Конкретная ячейка определяется парой чисел – номером строки (ряда) и номером столбца (колонки), на пересечении которых, она находится, (нумерация начинается с нуля). Свойство Rows[n].Cells[k] можно использовать только во время выполнения программы. Оно имеет тип **String**, поэтому программа может легко прочитать или записать содержимое нужной ячейки. Например,

```
/* в ячейку, лежащую на пересечении 1 столбца и 1 строки таблицы заносится строковые
данные "Доллар США" */
dataGridView.Rows[0].Cells[0].Value = "Доллар США";
```

6.2.5. Элементы управления проектом

Компонент Button (кнопка), широко используются для управления программами. Связанный с кнопкой алгоритм управления реализуется в обработчике события щелчка по кнопке. В отличие от других видимых компонентов кнопка класса Button является элементами операционной системы Windows и поэтому не может изменить свой цвет произвольным образом.

При большом количестве используемых кнопок в качестве органов управления весьма удобно располагать их в контейнере типа Panel. За счет использования в нем кромок – компонент имеет средства создания различных эффектов объемности. Свойство *BorderStyle* – определяет стиль рамки: значение *None* – отсутствие рамки, а значения *FixedSingle* и *Fixed3D* позволяют панель по периметру обвести линией или сделать объемной соответственно.

В качестве элемента управления программой можно использовать пункты главного меню формы menuStrip. Компонент класса MenuStrip определяет главное меню формы. Пункты меню создаются путем выделения объекта мышью и ввода с клавиатуры названий пунктов, каждый раз закрепляя надпись очередного пункта меню нажатием клавиши *Enter*. На рис. 8 показан пример создания многоуровневого меню.

Чтобы связать с выбором пункта меню нужное действие, следует определить обработчик его события … ToolStripMenuItem_Click().

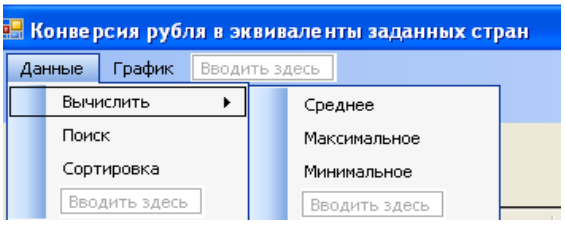


Рис.8. Многоуровневое меню.

С помощью свойства *ShortCut* главного меню устанавливается сочетание клавиш для прямого вызова команд меню. Нажимая эти клавиши, пользователь может активизировать нужный пункт меню.

6.3. Создание табличного интерфейса

Для выполнения задания курсовой работы по проектированию табличного интерфейса представления данных необходимо:

1. Создать новый проект, сохранить его в папке «KR»: под именем *GraphTable.*

2. Создать форму и дать ей имя MainForm. Разместить на форме (рис. 9) компоненты, обеспечивающие требуемые режимы работы приложения. Например:

– строковую таблицу DataGridView – для ввода исходных данных, их редактирования и вывода результатов вычислений;

– панель Panel – для размещения на ней элементов управления;

– кнопки (closeButton, sortButton, searchButton…), (или главное меню MenuStrip) – как средства для запуска расчета, очистки расчетных данных, решения индивидуальных задач и вызова средств построения графика;

–метки (answerStaticLabel, amountOfMoneyStaticLabel, searchStaticLabel…) – для пояснительных надписей к каждому элементу интерфейса приложения и вывода расчетных данных;

– однострочные редакторы (amountOfMoneyTextBox, patternTextBox…) – как средства ввода и редактирования данных,

–DateTimePicker – для вывода текущей даты,

–другие элементы, позволяющие наглядно представлять задание.

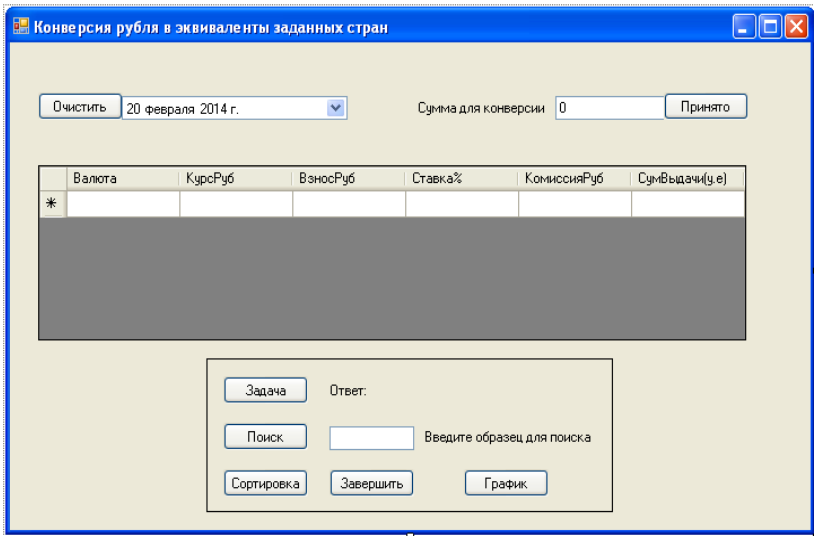


Рис. 9. Вариант интерфейса табличного представления данных.

3. Оформить таблицу столбцами. С помощью свойства Columns заполнить коллекцию – из 6 столбцов соответствующими названиями.

4. Установить количество строк в таблице равным 5, в соответствии с вариантом путем добавления их через метод dataGridView.Rows.Add() при объявлении класса MainForm.

```
public MainForm()
{
   InitializeComponent();
   dataGridView.Rows.Add(5); //добавление 5 строк
}
```

5. Столбцы «Валюта», «КурсРуб» и «Ставка%» могут быть заполнены в процессе запуска проекта вручную (при этом заполнение столбцов надо производить каждый раз при запуске проекта) или программно, путем глобального объявления этих значений в соответствующих массивах с инициализацией заданных стран, курсов и ставок.

6. Для всех объектов, включая саму форму, установить стартовые свойства, соответствующие выполняемым ими функциям.

7. Поскольку тип данных в таблице dataGridView – **String,** то для удобства дальнейших вычислений лучше глобально объявить одномерные массивы с соответствующими типами данных. Занести в каждый из массивов данные соответствующие значениям конкретного столбца таблицы.

8. Создать необходимые процедуры и обработчики событий.

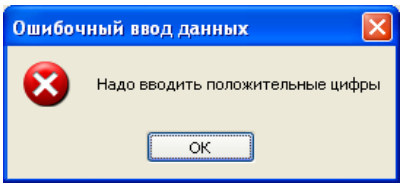
8.1. Для упрощения расчета поставленных задач и многократного обращения к таблице с вычисляемыми столбцами, целесообразно создать процедуру *Calculate()*, которая должна в соответствии с вводимой величиной конвертируемой суммы формировать взнос, вычислять комиссионные и сумму выдачи, а также отображать результаты расчета в соответствующих столбцах таблицы.

При написании процедуры *Calculate()* необходимо сначала опубликовать процедуру и описать ее, например:

```
. . .
public void Calculate()
{
    //тело процедуры
}
```

В теле процедуры предусмотреть следующее:

– ситуацию ввода в окно редактора в качестве конвертируемой суммы нуля или отрицательного значения с соответствующим сообщением о необходимости для конверсии денежной величины ввода положительных значений,



–преобразование вводимых с клавиатуры данных в число через свойство Text для расчета требуемых показателей, например, для объекта textBox, преобразование вида double.Parse(textBox.Text),

– формирование *взноса* для конверсии, а также вычисление *комиссионного сбора* по процентным ставкам и результирующей *суммы выдачи* при конверсии рубля производится по формулам (1) – (3):

Взнос = СуммаКонверсии. (1)

Комиссионные = Взнос * ПроцентСтавка / 100; (2)

СуммаВыдачи = (Взнос – Комиссионные) / КурсРубля; (3)

–расчет в цикле по формулам (1 – 3) требуемых показателей и запись их в соответствующие столбцы с преобразованием полученных при расчете численных значений в их текстовое отображение. Например, строки кода:

```
dataGridView.Rows[i].Cells[4].Value = commissions[i];
dataGridView.Rows[i].Cells[5].Value=String.Format("{0:f2}",sumR[i]);
```

преобразуют в цикле с параметром *i* все элементы массивов commissions [i] и sumR[i], представленные в виде чисел, в строковый формат и заносит их в 5-й и 6-й соответственно столбцы таблицы dataGridView.

8.2. Создать обработчик события щелчка для кнопки «Принято», при выполнении которого вызывается процедура Calculate и столбцы «Взнос», «Комиссионные» и «СуммаВыдачи» заполняются значениями, рассчитанными в процедуре по формулам (1) – (3).

8.3. Создать обработчик события щелчка для кнопки «Очистить», при выполнении которого происходит очистка столбцов таблицы от расчетных данных и очистка однострочных редакторов amountOfMoneyTextBox, patternTextBox от исходных данных. Например, для 5 столбца таблицы операция очистки производится так:

```
dataGridView.Rows[i].Cells[5].Value = " ";
```

9. Сохранить изменения в проекте. Откомпилировать проект.

10. Запустить проект на выполнение. Внести в однострочный редактор сумму для конверсии и проконтролировать результат расчетов в соответствующих ячейках таблицы.

11. Создать адекватные процедуры для кнопок управления решением задач приведенных в табл. 4. Для всех кнопок (кроме *График*), расположенных на панели, создать методы, при осуществлении которых возникают адекватные события: очистки столбцов от расчетных данных, завершения работы, нахождения минимального (максимального или среднего) значений, сортировка элементов массива, поиска элемента в таблице по образцу.

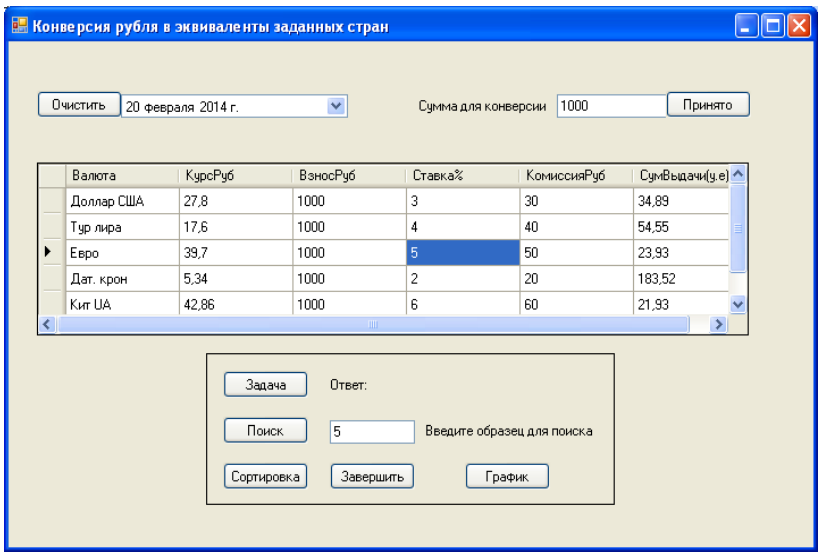


Рис 10. Проект в режиме запуска.

12. Добавить к проекту новую форму Form2 (командой *Проект \ Добавить форму Windows*). Переименовать ее в GraphForm. В окне добавления нового элемента выбрать *Форма Windows Forms* и подтвердить выбор кнопкой <Добавить>. Для кнопки «График» (например, makeGraphButton) создать обработчик события щелчка по кнопке, при выполнении которого, открывается вторая форма, используемая для создания графического интерфейса:

```
private void makeGraphButton_Click(object sender,EventArgs e) {
    GraphForm graphForm = new GraphForm();
    graphForm.Show();
}
```

13. Проверить при запуске подключение к проекту новой формы GraphForm. Сохранить результаты работы.

6.4. Графические возможности среды

Среда MS Visual Studio позволяет снабдить проектируемое приложение:

– готовыми картинками и фотографиями, загрузив их в приложение,
– выводом форматированных графиков в объект chart,
– разработкой своих собственных программ, которые выведут графику на поверхность формы.

6.4.1. Создание рисованных изображений

Рисованные изображения получаются на поверхности формы при выполнении программы с помощью различных инструментов. Изображение при этом представляет собой комбинацию простейших фигур – графических примитивов (точка, линия, круг или прямоугольник).

Форма состоит из отдельных точек – пикселей (рис. 11). *Пиксель* – это наименьший элемент поверхности рисунка, с которым можно манипулировать. Важное свойство пикселя – его цвет. Каждая точка формы имеет координаты X и Y, которые измеряются в пикселях.

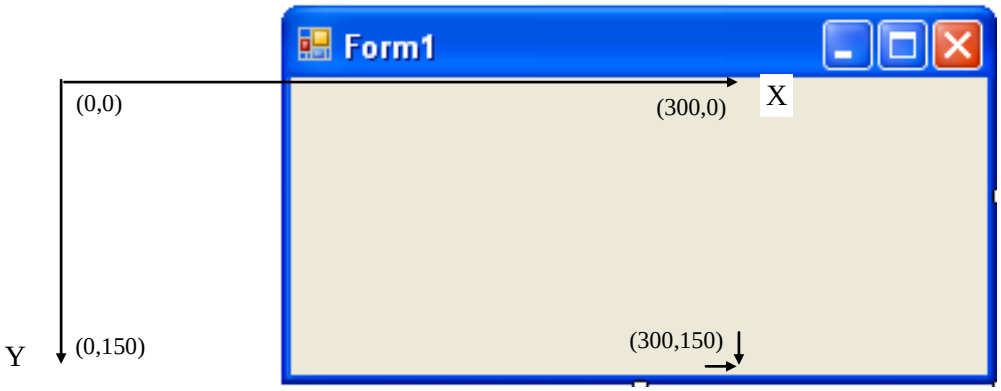


Рис. 11. Координаты точек холста

При выполнении графических операций используется текущий указатель (указатель позиции). Он представляет собой невидимый маркер, определяющий на форме позицию, начиная с которой выполняется следующая графическая операция. Текущая позиция определяется горизонтальной (X) и вертикальной (Y) координатами. По умолчанию начало системы координат (0,0) находится в левом верхнем углу поверхности рисования (рис.11).

Координата X возрастает при перемещении указателя слева на право, а координата Y – при перемещении его сверху вниз. Значения координат правой нижней точки на форме зависят от размера формы. Размер формы можно получить, обратившись к ее свойству Size (w;h).

Объект Graphics

Объект **Graphics** – это по сути поверхность, на которой будет производиться рисование. Пусть мы хотим рисовать на форме Windows. Синтаксис задания ссылки на нее выглядит следующим образом:

```
Graphics g = Graphics.FromHwnd(this.Handle);//с помощью ручки
```

Здесь:
Graphics – тип объекта, g – имя переменной,
Graphics.FromHwnd(this.Handle) – используемый метод FromHwnd из класса Graphics, который задает ссылку Handle на форму Windows.

В C# инструменты рисования определены в пространстве имен System.Drawing. Там находятся классы (помимо всего прочего):

–Pen (перо). Объекты пера используются в методах рисования линий и контуров геометрических фигур.

–Brush (кисть). Объекты кисти используются в методах заливки областей, ограниченных контурами.

6.4.2. Инструменты для рисования

Карандаш и кисть

Объекты пера используются в методах рисования линий и графических фигур. Объекты *Pen* выбираются из класса *Pens* (перья). Класс *Pens* содержит набор объектов для выбора. У них толщина линии (1 пиксель), стиль линии – сплошная. У каждого объекта свой цвет линии, имя которого идентифицирует объект. Такой объект нельзя редактировать, его можно только применять.

Например, создаем объект myPen, совпадающий с шаблоном Pens:

```
Pen myPen = Pens.Black;
```

Поскольку новые объекты *Pen* с изменяемыми свойствами создаются из класса *Pens*, то для них можно устанавливать дополнительные свойства. Основные свойства:
Color – цвет линии;

Brush – ссылка на кисть, используемую в качестве пера;
Width – толщина линии;
DashStyle – стиль штриховой линии,
DotStyle – стиль пунктирной линии
DashDotStyle – штрих пунктир,
DashDotDotStyle – штрих двойной пунктир,
SolidStyle – непрерывная линия (по умолчанию).

Новому объекту myPen зададим новые атрибуты, например цвет:

```
Pen myPen = new Pen(Color.Red);
```

Затем ему можно изменить свойства стиля и толщины линий:

```
myPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Solid;
myPen.PenType = System.Drawing.Drawing2D.PenType.SolidColor;
myPen.Width = 2;
```

Объекты кисти используются в методах заливки графических фигур. Определены кисти разного типа:
Brush – простая кисть, одноцветная заливка.
HatchBrush – кисть со штриховой заливкой (Hatch – штриховка),
LinearGradientBrush – кисть с линейной градиентной заливкой, цвет фрагментов фигуры меняется плавно.
PathGradientBrush – кисть с градиентной заливкой, цвет фрагментов фигуры меняется скачкообразно.

Объекты *Brush* выбираются из класса *Brushes*, который содержит кисти со сплошной заливкой. Класс *Brushes* содержит набор объектов для выбора, у которых по умолчанию определен цвет. У каждого объекта выбора имя – это цвет заливки.

Например, создаем объект myBrush, совпадающий с шаблоном:

```
Brush myBrush = Brushes.Blue;                // Заливка синим
```

Заливка

Объекты *HatchBrushes* выбираются из класса *HatchBrushes*. Класс *HatchBrushes* содержит набор объектов для выбора, у которых по умолчанию определены стиль заливки HatchBrushes, цвет переднего плана ForeColor и цвет фона BackColor (задний план). Определены стили заливки:
сетка (Cross),
диагональная сетка (DiagonalCross),
прямая диагональ (ForwardDiagonal),
обратная диагональ (BackwardDiagonal) и др.

Например, создаем кисть с заливкой сеткой HatchStyle.Cross:

```
HatchBrush brush2 = new HatchBrush(HatchStyle.Cross, ForeColor, BackColor);
```

Вывод текста

Для вывода на форму текста используется метод DrawString. Метод рисует заданную текстовую строку в заданном прямоугольнике с помощью определяемых объектов кисти и шрифта, используя атрибуты форматирования заданного формата. Синтаксис метода:

```
DrawString(String, Font, Brush, RectangleF, StringFormat);
```

String –текстовая строка для рисования.
Font – шрифт текстовой строки.
Brush – кисть. Определяет цвет и текстуру создаваемого текста.
RectangleF – прямоугольник вывода.

StringFormat – формат, определяющий атрибуты форматирования, такие как междустрочный интервал и выравнивание, которые применяются к создаваемому тексту.

Font. Это шрифт текстовой строки. Выбирается с помощью методов класса *Font*. Предоставляет возможность выбора размера и стиля шрифта. Возможны несколько способов вызова, отличающиеся друг от друга числом аргументов и способом задания нового шрифта. Например:

```
font MyFont = new  Font("Arial" , 24 , FontStyle.Bold ) ;
                                    или без учета начертания шрифта
font MyFont = new  Font("Arial Narrow" , 14 ) ;
```

6.3.3. Методы вычерчивания графиков

Рисовать на форме можно разными способами – с помощью линий, окружностей, дуг, секторов, прямоугольников, многоугольников.

В C# определены методы рисования линий и фигур. Все методы выполняются по-разному с разными аргументами.

Рисование с помощью пера Pen

При рисовании можно использовать перо с разными стилями линий LineStyle. Например, Solid (сплошная линия), Dash (штрих), Dot (пунктир), DashDot (штрих-пунктир), DashDotDot (штрих-пунктир-пунктир).

DrawLine. Прямая линия между двумя точками. Синтаксис метода:

```
g.DrawLine(pen, x[1], y[1], x[2], y[2]);
```

Здесь
g – объект, на котором рисуем,
DrawLine – метод рисования линии,
pen – перо,
x[1], y[1], x[2], y[2] – точки границы линии (координаты линии).

DrawRectangle. Прямоугольник. Синтаксис метода:

```
g.DrawRectangle(pen, rect);
```

Здесь:
g – объект, на котором рисуем,
DrawRectangle – метод рисования прямоугольника,
pen – перо,
rect – прямоугольник, свойства которого задаются целыми числами

```
Rectangle rect = new Rectangle(x, y, w, h);.
```

Здесь:
x, y – координаты левого верхнего угла,
w – ширина прямоугольника,
h – высота прямоугольника.

DrawEllipse. Эллипс. Синтаксис метода:

```
g.DrawEllipse(pen, rect);
```

Здесь:
g – объект, на котором рисуем,
DrawEllipse – метод рисования эллипса,
pen – перо,
rect – прямоугольная область, в которую вписывается эллипс.

6.4.4. Методы заливки

В C# определены различные методы заливки фигур. Все методы выполняются по-разному с разными аргументами.

В примерах заливка разных фигур осуществляется **простой** кистью с Brush.Cyan и кистью HatchBrush с **разными** стилями заливки DashStyle.

FillRectangle. Закрашивает прямоугольник. Синтаксис метода:

```
g.FillRectangle (brush, p);
```

Здесь:
g – объект, на котором рисуем,
FillRectangle – метод рисования залитого прямоугольника,
brush – кисть,
p – массив точек.

Получаемый результат зависит от стиля заливки.

FillEllipse. Закрашивает эллипс. Синтаксис метода:

```
g.FillEllipse (brush, rect);
```

Здесь
g – где рисуем,
FillEllipse – метод рисования залитого эллипса,
brush – кисть,
rect – прямоугольная область, в которую вписывается эллипс/

Получаемый результат зависит от стиля заливки.

В качестве примера рассмотрим код обработчика события щелчка для кнопки «*Нарисовать*» (drawButton), при реализации которого, вычерчивается рисунок (рис.12) на поверхности формы.

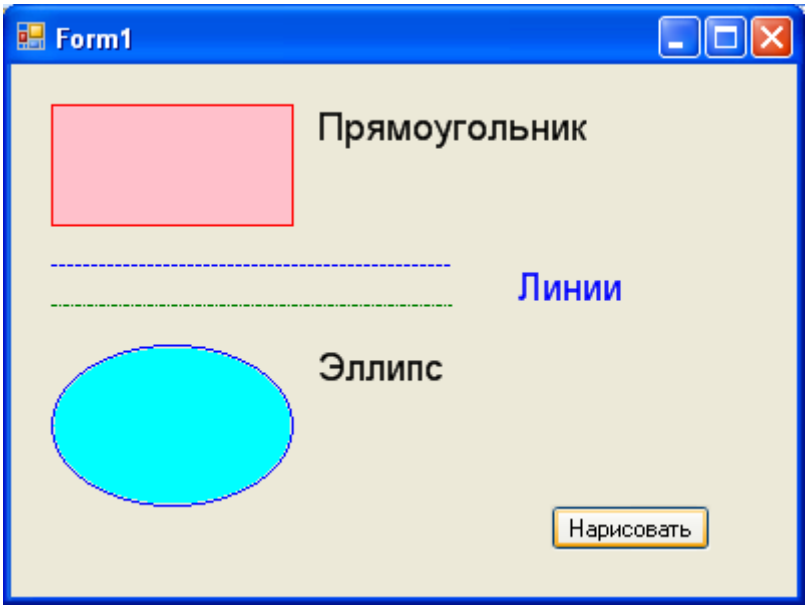


Рис.12. Рисунок, полученный с помощью примитивов.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
```

```
using System.Windows.Forms;
using System.Drawing.Drawing2D;  //подключаем модуль для реализации стилей рисования
namespace DrawFormExample
{
  public partial class MainForm : Form
  {
    public MainForm()
    {
      InitializeComponent();
    }
    private void drawButton_Click(object sender, EventArgs e)
    {
          //рисуем на форме
          Graphics g = Graphics.FromHwnd(this.Handle);
       g.DrawString("Линии", new Font("Arial", 14),                    Brushes.Blue,      250,
100);
          // новый карандаш pen
          Pen pen = new Pen(Color.Blue);
          // стиль линии пунктирный
          pen.DashStyle = DashStyle.Dash;
          // координаты 1 линии
          g.DrawLine(pen, 20, 100, 220, 100);
          // еще один карандаш
          Pen anotherPen = new Pen(Color.Green);
          // стиль линии штрих-пункт.
          anotherPen.DashStyle = DashStyle.DashDot;
          // координаты 2 линии
          g.DrawLine(anotherPen, 20, 120, 220, 120);
          // вывод текста «Прямоугольник»
          g.DrawString("Прямоугольник", new Font("Arial", 14),
                    Brushes.Black, 150, 20);
          // еще один карандаш
          Pen thirdPen = new Pen(Color.Red);
          // стиль линии непрерывная
          thirdPen.DashStyle = DashStyle.Solid;
          //пустой прямоугольник
          g.DrawRectangle(thirdPen, new Rectangle(20,20,120,60));
          //объект brush розового цвета
          Brush brush = Brushes.Pink;
          //заполненный цветом  brush прямоугольник
          g.FillRectangle(brush, 21, 21, 119, 59);
          // вывод текста «Эллипс»
          g.DrawString("Эллипс", new Font("Arial", 14),
                    Brushes.Black, 150, 140);
          // новый карандаш forthPen
          Pen forthPen= new Pen(Color.Blue);
          //стиль линии непрерывная
          forthPen.DashStyle = DashStyle.Solid;
          //пустой эллипс
          g.DrawEllipse(forthPen, new Rectangle(20,140,120,80));
          //объект anotherBrush голубого цвета
          Brush anotherBrush = Brushes.Cyan;
```

```
            // заполненный цветом  anotherBrush
            g.FillEllipse(anotherBrush, 21, 141, 118, 78);
            /*заполненный цветом  brush сектор
            на форме отсутствует*/
            g.FillPie(brush, Rectangle(50, 10 150, 200), 50, 250);
    }
  }
}
```

6.5. Создание графического интерфейса

Для отображения диаграммы валютного эквивалента исходной суммы по курсам заданных стран необходимо:

1. Создать в проекте новую форму GraphForm, описание которой приведено в п.12.раздела 6.2.

2. Разместить на форме GraphForm компоненты, обеспечивающие требуемые режимы работы приложения (см. рис.13). Например, кнопки clearButton и drawButton как средства построения диаграммы и очистки экрана.

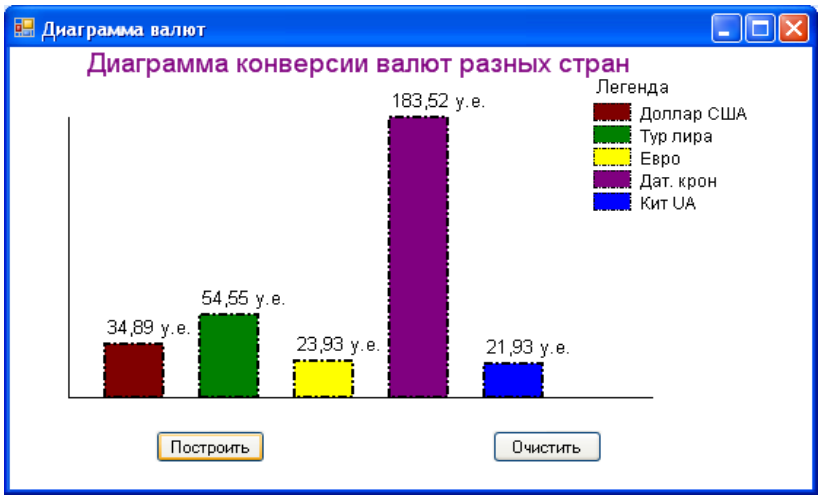


Рис.13. Графический интерфейс в режиме запуска

3. Для всех объектов, включая саму форму, установить свойства, соответствующие выполняемым ими функциям.

4. Создать необходимые обработчики событий.

4.1. Для того, чтобы получить доступ во второй форме к рассчитанным эквивалентам, помещенным в 6-й столбец объекта dataGridView1 первой формы, необходимо при определении класса Form2 объявить атрибут – массив. Указанные эквиваленты помещаются в этот атрибут в конструкторе класса, который принимает один аргумент – массив рассчитанных эквивалентов. Например,

```
public partial class GraphForm : Form
{
   private double[] SumR;  //объявление массива
   public GraphForm()
   {
      InitializeComponent();
       }
      public GraphForm(double[] SumR)
   {
      InitializeComponent();
```

```
        //this.SumR = new double[SumR.GetLength(0)];    //передача управления данными из
столбца 6 таблицы в GraphForm
        this.SumR = SumR;
      }
…
   }
```

4.2. Для реализации стилей рисования объявить об использовании (подключении) модуля Drawing2D:

```
using System.Drawing.Drawing2D;
```

4.3. Для построения диаграммы создать обработчик события щелчка по кнопке «Построить», при выполнении которого на форме GraphForm должна отобразиться диаграмма расчета денежных эквивалентов. При этом необходимо учесть, что величины эквивалентов могут колебаться в весьма значительном диапазоне, и могут составлять как десятки единиц, так и миллионы, а область построения графика фиксирована. В этой связи целесообразно ввести масштабируемый коэффициент, учитывающий отображение больших и малых величин конвертируемой валюты в ограниченных размером формы пределах построения диаграммы.

Как вариант для решения этой проблемы можно выявить максимальное значение эквивалента, приравнять его заданным пределам построения графика на форме, а все остальные значения взять как отношения к максимальному значению. Фактические же значения эквивалентов вывести на график (рис. 13), взяв их из соответствующего столбца таблицы dataGridView (рис. 10).

4.4. При создании обработчика события построения графика предусмотреть в разделе объявлений:

– массив цветов столбиков диаграммы, заданных по варианту, например,

```
Brush[] BarColor = { Brushes.Maroon, Brushes.Green,
                     Brushes.Yellow, Brushes.Purple,
                         Brushes.Blue };
```

–массив, в который будут размещаться нормированные значения по отношению к максимальному, суммы выдачи эквивалентов валюты для построения диаграммы.

–задание ширины столбиков и расстоянием между ними;

В теле процедуры предусмотреть:

–проверку наличия расчетных данных в столбце «СумВыдачи», необходимых для построения диаграммы и выдачи сообщения в окно диалога в случае активизации кнопки «Построить» до того, как произошло заполнение данными таблицы StringGrid;

–определение в цикле максимального денежного эквивалента;

–вычисление в цикле нормированных значений столбиков по отношению к максимальному значению;

–вызов объекта *Graphics*:

Graphics g = Graphics.FromHwnd(this.Handle);

–установку атрибутов пера для рисования осей и их рисование;

–вывод строки заголовка диаграммы;

–установку атрибутов пера для рисования контуров столбиков - прямоугольников по варианту,

–рисование в цикле прямоугольников - столбиков с нормированными координатами с шагом по оси x;

–их заливку цветом, выбираемым из массива цветов BarColor;

– вывод над каждым столбиком поясняющую надпись о фактическом значении «Суммы выдачи» разных категорий валют.

Аналогично осуществить построение легенды. При этом в качестве поясняющих надписей категорий валют целесообразно вывести заданные страны (см. рис.13 и 14). Легенду можно строить в любом месте формы, располагая ее вертикально или горизонтально по отношению к графику.

Цвет, стиль шрифта выбираются по желанию, но с учетом наглядного представления атрибутов надписей на диаграмме.

4.5. Для очистки области построения диаграммы создать обработчик события щелчка по кнопке «Очистить», при выполнении которого поверхность формы освобождается от диаграммы (рис. 14).

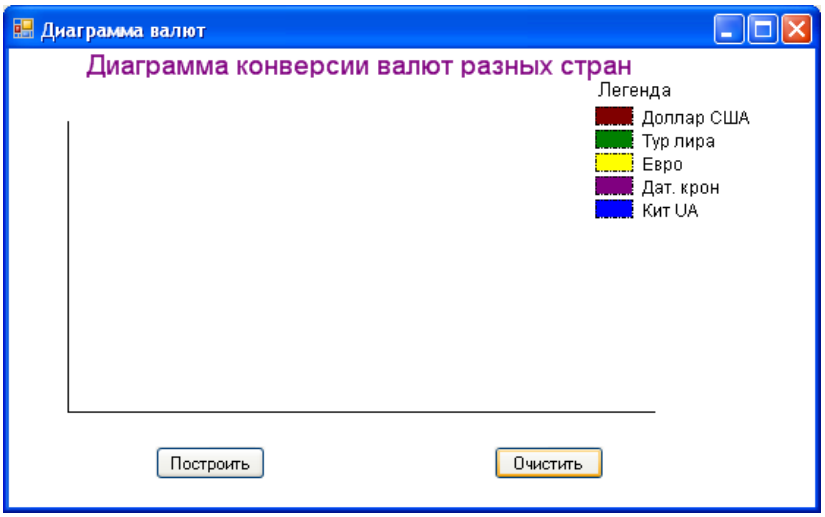


Рис.14. Очистка формы от диаграммы

Для очистки формы от графика можно использовать процедуру заливки прямоугольной области построения графика белым или другим цветом.

4.6. Сохранить изменения и произвести компиляцию проекта.

5. Запустить проект и проверить его работоспособность.

6.6. Запись инструкций программы

При записи операторов и инструкций программы следует руководствоваться следующими рекомендациями:

– программа должна быть легко читаемой. Ее структура должна соответствовать структуре алгоритма решения задачи. В свою очередь, структура алгоритма решения задачи отражается посредством соответствующей структуризации записи операторов:

1) операторы **if** … **else**, **case** … **default** **do**… **while** записываются один под другим,

2) инструкции между **{ . . . }** размещаются одна под другой, но с отступом относительно открывающейся **{** фигурной скобки (см. код рис. 13);

– в пределах операторной строки один оператор может отделяться от другого другими символами, но завершается операторная строка всегда точкой с запятой;

– длинные выражения записывать в несколько сток. Разорвать выражение и перенести оставшуюся часть на следующую строку можно практически в любом месте. Однако нельзя разрывать имена переменных, числовые и строковые константы, а также составные операторы;

– для лучшего восприятия можно вводить «лишние» пробелы и пустые строки, т.к. компилятором они игнорируются;

– для облегчения понимания логики работы программы ее следует сопровождать комментариями;

– имена переменных, процедур и функций должны нести смысловую нагрузку.

В результате программа становится надежной и «дружественной» по отношению к пользователю.

6.7. Компьютерное моделирование в Simulink

Система MATLAB+Simulink представляют удобные и простые средства, в том числе визуального объектно-ориентированного программирования для моделирования систем.

В состав моделей могут включаться:

–источники сигналов,

–регистрирующие приборы,

–графические средства анимации.

Все блоки для моделирования располагаются в соответствующих библиотеках Simulink, доступ к которым возможен через браузер главной библиотеки. На рис. 15 приведено главное окно библиотек.

Порядок создания модели с помощью пакета Simulink:

1) запустить Simulink, нажав на кнопку «Simulink» на панели инструментов системы MATLAB. Появится окно браузера библиотек;

2) создать окно модели командой File => New => Model;

3) методом Drag and Drop скопировать мышью из папок браузера в окно модели нужные блоки и соединить их коннекторами (линиями).

4) установить для каждого блока соответствующие параметры;

5) сохранить модель в виде файла на диске D:\Temp\… командой File => Save as… под именем conversion.mdl;

6) включить симулирование (моделирование) командой Simulation => Start (или кнопкой «Start Simulation» на панели инструментов окна модели). Если полученный результат не совпадает с ожидаемым, то нужно изменить параметры модели.

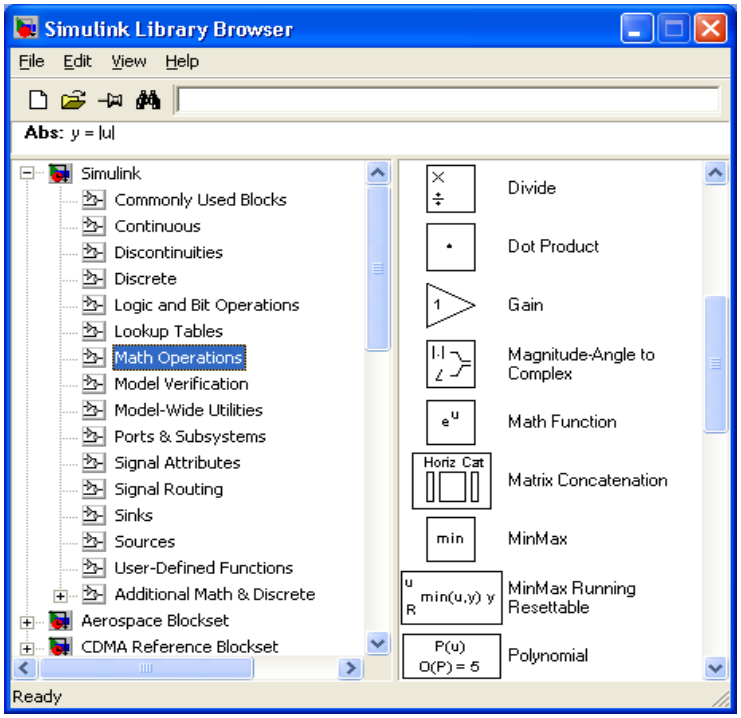


Рис. 15. Окно браузера библиотек Simulink

В качестве примера рассмотрим модель пересчета стоимости аппаратуры, заданной в рублях на эквивалент $ (USA) и € (Euro). Моделирование системы осуществляется с выводом исходных и расчетных значений. Кроме того, в модели предусмотрены блоки вывода самого

дорогого и самого дешевого оборудования. Для наглядности в окно модели помещен рисунок. Описанная модель приведена на рис. 16.

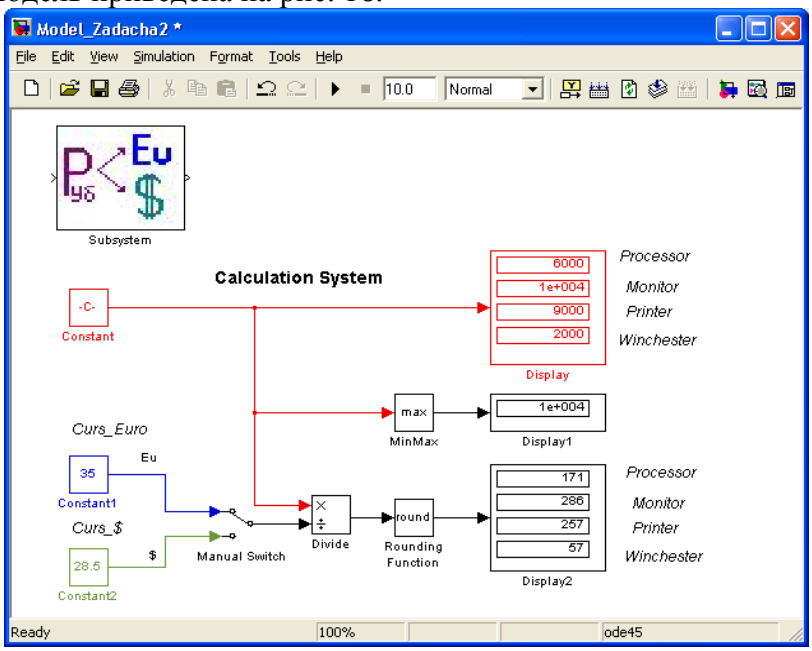


Рис. 16. Модель пересчета стоимости оборудования на валютные эквиваленты.

Модель системы пересчета стоимости оборудования, заданного в рублях на эквиваленты США и Евро содержит:

–средства хранения стоимости оборудования и курсов валют – блоки Constant (1,2), размещенные в браузере Simulink => Sources;

–блоки математических операций – Divide, Rounding Function, MaxMin, размещенные в браузере Simulink => Math Operations;

–переключатель валютных эквивалентов Manual Switch, находящийся в браузере Simulink => Signal Routing;

–средства регистрации исходных данных и результатов вычислений – Display (1,2), размещенные в браузере Simulink => Sinks.

Для блоков Constant (1,2) устанавливаются соответствующие параметры: Constant value [900 600 300 120], Constant1 value 35, Constant2 value 28.5. Переключатель Manual Switch выполняет подключение к одному из каналов двойным щелчком мыши. А блок MaxMin, в зависимости от установленных параметров блока, позволяет выявлять максимальное или минимальное значения.

Для создания рисунка – эмблемы (пиктограммы) модели можно использовать любой графический редактор, например Paint, Image Editor (Delphi), Adobe Photoshop. Сохраняется рисунок в той же папке, что и модель латинскими символами, например Picture.bmp. Для уменьшения размера файла рисунка можно сохранить его с расширением *.jpg.

Порядок отображения готового рисунка на схеме модели:

1) создать пустую подсистему (фрагмент Simulink-модели, оформленный в виде отдельного блока). Для этого: в браузере Simulink => Port & Subsystems найти и перетащить на свободное место уже созданной модели блок Subsystem (см. рис. 16);

2) выполнить ее маскирование (оформить подсистему как отдельный библиотечный блок) с помощью редактора маски Mask Editor. Для запуска редактора необходимо выделить Subsystem и выполнить команду Mask Subsystem…

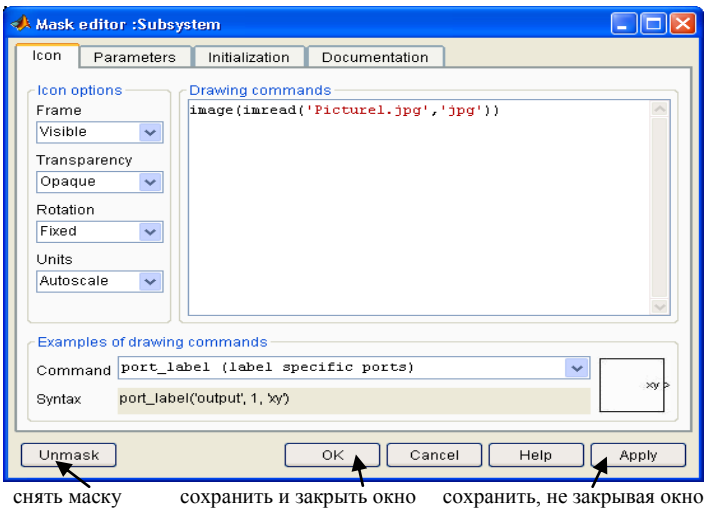


Рис.17. Окно редактора маски

На экран будет выведено окно редактора (рис.17), имеющего 4 вкладки: *Icon* (Пиктограмма), *Parameters* (Параметры), *Initialization* (Инициализация) и *Documentation* (Документация). Первая из вкладок обеспечивает создание пиктограммы подсистемы, вторая – дает возможность создать окно диалога для ввода параметров, третья – задать выражения для инициализации блока и четвертая – позволяет ввести описание блока и создать справку. Открыть саму систему для просмотра и ее редактирования можно двойным щелчком мыши;

3) на вкладке *Icon* в графе *Drawing Commands* ввести команду для считывания из файла и отображения графического образа, например:

image(imread('Picture.bmp', 'bmp')),

где первый параметр – имя файла, а второй – его тип.

Если файл-рисунок сохранен в отдельной папке от модели, то в качестве имени файла указывается полный путь доступа к файлу, например: ' D:\tests\Picture.bmp'. На рисунке 15 показано окно модели с командой загрузки файла с пиктограммой.

Для придания модели законченного вида можно использовать возможности форматирования, команды которого сосредоточены в пункте меню *Format*. Для создания надписи к блокам достаточно два раза щелкнуть мышью в определенных местах модели и ввести с клавиатуры латинские символы. Форматирование символов производится путем их выделения и вызова окна диалога командой *Format => Font…*

Цвет к контурам блоков и коннекторам можно добавить командой *Format => Foreground Color*, а заливку блоков – командой *Format => Background Color*. Цвет фона экрана, в пределах котором размещается модель, можно изменить командой *Format => Screen Color*.

Изменение размера блока (например, Display), нужно выделить объект, установить мышь на угловом маркере выделенного объекта, добиться двунаправленной стрелки и при нажатой левой клавиши мыши растянуть (или уменьшить) блок по диагонали. При этом масштабируется и текстовая надпись.

6.8. Оформление пояснительной записки в текстовом редакторе

При создании пояснительной записки в Office необходимо установить:

– параметры логического листа (*Файл =>Параметры страницы*);

– автоматический перенос слов в словах (*Сервис =>Язык => Расстановка переносов => Автоматическая расстановка переносов*),

– тип шрифта, высоту букв, начертание (*Формат => Шрифт*);

– красную (первую) строку, межстрочный интервал, выравнивание абзацев по ширине (*Формат => Абзац*);

– номера станиц (*Вставка => Номера страниц*). Следует отказаться от возможности установки номера страницы на первом (титульном) листе.

6.9. Создание оглавления

Для создания оглавления в пояснительной записке необходимо:

I. Присвоить стили всем заголовкам, которые должны быть в оглавлении (выделить текст с заголовком и выбрать из списка стилей панели *Форматирования* требуемый).

Пример оформления стилем заголовков пояснительной записки:

1. Цели и задачи курсовой работы (стиль Заголовок 1)
2. Постановка задачи (стиль Заголовок 1)
3. Задание на выполнение (стиль Заголовок 1)
4. Выполнение заданий (стиль Заголовок 1)
   - 4.1. Технология выполнения задания (стиль Заголовок 2)
   - 4.2. Интерфейс проекта в режиме создания (стиль Заголовок 2)
   - 4.3. Код Form1 (стиль Заголовок 2)
   - 4.4. Код Form2 (стиль Заголовок 2)
   - 4.5. Интерфейс проекта в режиме запуска (стиль Заголовок 2)
   - 4.6. Компьютерная модель в Simulink (стиль Заголовок 2)
5. Выводы по работе (стиль Заголовок 1)
6. Литература (стиль Заголовок 1)

II. Организовать дополнительную страницу перед описательной частью с целью размещения на ней будущего оглавления. Для этого:

– поместить курсор перед первым заголовком (1. Цели и задачи курсовой работы);

– выбрать команду *Разрыв…* в меню *Вставка*;

– в диалоговом окне *Разрыв* установить переключатель *Новую страницу* и нажать кнопку *ОК*.

Аналогично организуется дополнительная страница для рецензии.

III. Вставка оглавления:

– установить курсор в место вставки оглавления (в начало новой страницы) и напечатать заголовок «Оглавление»;

– активизировать команду *Вставка\Ссылка\Оглавление и указатели***;**

– в окне *Оглавление и указатели* открыть вкладку *Оглавление;*

– в поле *Форматы* выбрать один из готовых стилей;

– наличие флажка *Показать номера страниц* позволяет для каждого элемента списка отображать номера страниц;

– установка флажка *Номера страниц по правому краю* даёт возможность выровнять номера страниц по правому полю;

– в поле *Уровни* задать количество уровней оглавления (например, 2);

– в поле *Заполнитель* можно выбрать тип линий (точечные, пунктирные или прямые) заполнения пространства между элементами указателя и номерами страниц (выбираются по желанию);

– нажать кнопку ОК.

Изменения в оглавление после его создания можно внести с помощью команды *Обновить поле*, находящейся в контекстном меню оглавления.