СВОЙСТВА		
Имя	Назначение	Параметры/значения/пример
Внешний вид		

Text	текстовое содержимое ЭУ, позволяет получить или установить текст, который отображается внутри RichTextBox	// Устанавливаем текст в RichTextBox
		richTextBox1.Text = "Пример текста в RichTextBox";
		// Получение текста из RichTextBox
		string textFromRichTextBox = richTextBox1.Text;

BackColor	цвет фона ЭУ	Синтаксис свойства: public Color BackColor { get; set; }
		Color – представляет цвет, который определяется с помощью структуры System.Drawing.Color
		// Установим цвет фона RichTextBox на красный
		richTextBox1.BackColor = Color.Red;
		//С помощью RGB:
		int redValue = 0;
		int greenValue = 128;
		int blueValue = 255;
		richTextBox1.BackColor = Color.FromArgb(redValue, greenValue, blueValue);

BorderStyle	установка стиля границы (рамки)	Синтаксис свойства: public BorderStyle BorderStyle { get; set; } Значения перечисления BorderStyle: • None – нет границы • FixedSingle – одиночная линия толщиной в пиксель • Fixed3D – выпуклая 3D-граница
Cursor	используется для установки или поулчения курсора, который будет отображаться при наведении на ЭУ	Синтаксис свойства: public Cursor Cursor { get; set; } Значение Cursor представляет курсор и определяется с помощью класса System.Windows.Forms.Cursor. Можно установить значение свойства Cursor на предопределенные курсоры, такие как Cursors.Default, Cursors.Hand, Cursors.IBeam, и другие. Также возможно определить свой собственный курсор с помощью конструктора класса Cursor. • Default: Обычная стрелка (стандартный курсор по умолчанию). • Arrow: Обычная стрелка. • Cross: Крестик. • Hand: Рука (обычно используется для ссылок и кликабельных элементов). • Help: Вопросительный знак с курсором. • IBeam: Текстовый курсор в форме вертикальной линии, используемый для ввода текста. • No: Курсор с перечеркнутым кругом, указывающий на недопустимое действие. • SizeAll: Курсор для перемещения по всем направлениям. • SizeNESW: Курсор для изменения размера в направлении северо-восток/юго-запад. • SizeNWSE: Курсор для изменения размера в направлении северо-запад/юго-восток. • SizeWE: Курсор для изменения размера по горизонтали. • SizeNS: Курсор для изменения размера по вертикали. • WaitCursor: Курсор ожидания (чacglass), который указывает на операцию, выполняющуюся в фоновом режиме.

		// Устанавливаем курсор в виде стрелки (Cursors.Default)
		richTextBox1.Cursor = Cursors.Default;

Font	установка и получение штрифта текста	Синтаксис: public Font Font { get; set; }
		Значение Font представляет шрифт текста и определяется с помощью класса System.Drawing.Font
		// Устанавливаем шрифт для RichTextBox (Arial, 12pt, обычный стиль)
		richTextBox1.Font = new Font("Arial", 12, FontStyle.Regular);

ForeColor	установка или получение цвета переднего плана (цвет текста)	Синтаксис свойства: public Color ForeColor { get; set; }
		// Устанавливаем цвет текста в RichTextBox (синий цвет)
		richTextBox1.ForeColor = Color.Blue;

Lines	массив строк, который содержит текст, отображаемый в ЭУ. Каждый элемент массива представляет одну строку текста	Синтаксис свойства: public string[] Lines { get; set; }
		// Устанавливаем несколько строк текста в RichTextBox
		richTextBox1.Lines = new string[]
		{
		"Пример текста в RichTextBox",
		"Это вторая строка.",
		"И это третья строка."
		};

RightToLeft	направление текста	Синтаксис: public RightToLeft RightToLeft { get; set; }
		Значение перечисления RightToLeft:
		• No: Текст отображается слева направо (стандартное значение).
		• Yes: Текст отображается справа налево. Это может быть полезно, когда вы работаете с языками, которые пишутся справа налево, например, арабским или ивритом.
		• Inherit: Направление текста будет определяться автоматически на основе текущего языка или настройки контрола.
		// Устанавливаем направление текста справа налево
		richTextBox1.RightToLeft = RightToLeft.Yes;

ScrollBars	используется для управления отображением полос прокрутки в ЭУ(какие полосы прокрутки будут отображася, если текст не помещается в видимой области контрола»	Синтаксис свойства:
		public RichTextBoxScrollBars ScrollBars { get; set; }
		Значение перечисления RichTextBoxScrollBars:
		• None: Не отображает полосы прокрутки ни по горизонтали, ни по вертикали. Текст в RichTextBox будет обрезаться, если он не помещается в видимую область контрола.
		• Horizontal: Отображает только горизонтальную полосу прокрутки, если текст выходит за пределы видимости по горизонтали.
		• Vertical: Отображает только вертикальную полосу прокрутки, если текст выходит за пределы видимости по вертикали.
		• Both: Отображает как горизонтальную, так и вертикальную полосы прокрутки, если текст выходит за пределы видимости.
		// Отображаем только вертикальную полосу прокрутки
		richTextBox1.ScrollBars= RichTextBoxScrollBars.Vertical;

UseWaitCursor	флаг, который позволяет управлять использованием курсора ожидания (часы) в ЭУ	• true – курсор будет изменен на курсор ожидания (часы), указывая на то, что приложение выполняет операцию, требующую времени; • false Синтаксис свойства: public bool UseWaitCursor { get; set; }

<table>
<tr><td colspan="3">ПОВЕДЕНЧЕСКИЕ:</td></tr>
<tr><td>AcceptsTab</td><td>определяет, буде тли ЭУ обрабатывать символ табуляции (Tab) как обычный текст или как запрос на вставку табуляции</td><td>Синтаксис: public bool AcceptsTab { get; set; }
Значение:
•true: ЭУ будет обрабатывать символ табуляции как запрос на вставку табуляции (при нажатии Tab в тексте курсор сместится на следующую позицию табуляции).
•false (По умолчанию): Элемент RichTextBox будет обрабатывать символ табуляции как обычный текст (при нажатии Tab будет создаваться символ табуляции в тексте).</td></tr>
<tr><td>AutoWordSelection</td><td>управляет поведением выделения при перемещения курсора и выделении текста мышью, не влияет на работу методов копирования, вырезания, вставки текста</td><td>•true (по умолчанию) – автоматически выделяется текст по границам слов («прилипает к словам»)
false – выделяет текст по символам</td></tr>
<tr><td>BulletIndent</td><td>отступ для маркированного списка в ЭУ РичБокс, определяет расстояние между левым краем контрола и началом маркированного списка. Значение – в пикселях, допускаются отрицательные значения (для создания висячего отступа перед маркером списка)</td><td><pre>RichTextBox richTextBox1 = new RichTextBox();
richTextBox1.Text = "Маркированный список: \n\n";
richTextBox1.Text += "\u2022 Первый эл\n";
richTextBox1.Text += "\u2022 Второй эл\n";
richTextBox1.Text += "\u2022 Третий эл\n";

richTextBox1.BulletIndent = 20; // Устанавливаем отступ для маркера списка
this.Controls.Add(richTextBox1);</pre></td></tr>
<tr><td>ContextMenuStrip</td><td>представляет контекстное меню, которое будет отображаться, когда пользователь правой кнопкой мыши щелкает в ЭУ или вызывает контекстное меню с клавиши «Меню» на клавиатуре</td><td>Синтаксис:
public ContextMenuStrip ContextMenuStrip { get; set; }

Значение ContextMenuStrip: объект класса ContextMenuStrip, который представляет контекстное меню. Содержит пункты меню, которые предоставляют опции или команды, специфичные для текущего контекста (то есть, определенного элемента или области приложения).
<pre>// Создание контекстного меню
ContextMenuStrip contextMenuStrip = new ContextMenuStrip();
contextMenuStrip.Items.Add("Опция 1");
contextMenuStrip.Items.Add("Опция 2");
contextMenuStrip.Items.Add("Опция 3");

// Установка контекстного меню для RichTextBox
richTextBox1.ContextMenuStrip = contextMenuStrip;</pre></td></tr>
<tr><td>DetectUrls</td><td>определяет, будет ли ЭУ автоматически обнаруживать и преобразовывать URL в кликабельные гиперссылки</td><td><pre>• true – автоматическое преобразование в кликабельную ссылку
• false – отображение ссылки как обычного текста
richTextBox1.DetectUrls = true; // Включаем обнаружение URL-адресов</pre></td></tr>
<tr><td>EnableAutoDragDrop</td><td>указывает, будет ли ЭУ автоматически</td><td>Синтаксис:
public bool EnableAutoDragDrop { get; set; }</td></tr>
</table>

	обрабатывать операции перетаскивания (drag-and-drop) текста или файлов	• true: RichTextBox автоматически поддерживает операции перетаскивания, и вы можете перетаскивать текст или файлы внутрь RichTextBox или из него.
		• false: Операции перетаскивания не обрабатываются автоматически.
		// Включаем автоматическую обработку операций перетаскивания
		richTextBox1.EnableAutoDragDrop = true;
		EnableAutoDragDrop предназначено специально для RichTextBox

AlowDrop	разрешает/запрещает перетаскивание	• true-разрешает • false (по умолчанию) AllowDrop может применяться к различным элементам управления и позволяет им принимать перетаскиваемые элементы с дополнительным кодом для обработки этих операций. Для обработки операции перетаскивания, нужно определить обработчики событий, такие как DragEnter и DragDrop, и в них реализовать необходимую логику для обработки перетаскиваемых данных.
Enabled	Определяет, разрешено ли пользователю взаимодействовать с этим ЭУ	Синтаксис: public bool Enabled { get; set; } Значение: • true (по умолчанию): Элемент RichTextBox активен, и пользователь может взаимодействовать с ним (по умолчанию). • false: Элемент RichTextBox неактивен, и пользователь не может взаимодействовать с ним.
HideSelection	определяет, будет ли выделеннй текст скрыт, когда ЭУ теряет фоус.	Синтаксис: public bool HideSelection { get; set; } Значение bool: • true (по умолчанию): Выделенный текст останется видимым, даже если элемент управления RichTextBox не в фокусе. • false: Выделенный текст будет скрыт, когда элемент управления RichTextBox теряет фокус.
ImeMode	режим ввода (Input Method Editor, IME) при вводе текста в ЭУ. IME – инструмент, который позволяет вводить текст на языках со сложной структурой (китайский, японский, корейский), используя клавиатуру с алфавитом	Синтаксис: public ImeMode ImeMode { get; set; } Возможные значения: • Inherit: Режим ввода наследуется от родительского элемента управления. • NoControl: Отключает IME, и ввод текста будет осуществляться только с помощью стандартной клавиатуры. • On: Включает IME, и при необходимости пользователь может вводить текст с использованием IME. • Off: Отключает IME и не позволяет пользователю использовать IME для ввода текста. • Disable: Отключает IME, и ввод текста будет осуществляться только с помощью стандартной клавиатуры.

		• Close: Закрывает IME, если он был открыт.
		• Hiragana: Включает режим Hiragana IME для японского языка.
		• Katakana: Включает режим Katakana IME для японского языка.
		• KatakanaHalf: Включает режим KatakanaHalf IME для японского языка.
		// Устанавливаем режим ввода IME для элемента управления RichTextBox
		richTextBox1.ImeMode = ImeMode.On;

MaxLength	определяет максимальное количество символов, которое может быть введено пользователем. Применяется только к вводу текста пользователем и не влияет на установку текста программно	Синтаксис: public int MaxLength { get; set; }
		По умолчанию установлено в 0, что означает отсутствие ограничения на длину текста
		// Установка максимальной длины текста в 100 символов
		richTextBox1.MaxLength = 100;

Multiline	указывает, будет ли ЭУ поддерживать многострочный ввод текста	Синтаксис: public bool Multiline { get; set; } • true: RichTextBox поддерживает многострочный ввод текста. • false: RichTextBox поддерживает только однострочный ввод текста (по умолчанию).
ReadOnly	определяет, можно ли редактировать текст	• true: RichTextBox становится только для чтения, и пользователь не может редактировать его содержимое. • false: RichTextBox становится редактируемым (по умолчанию).

RightMargin	определяет отступ справа в пикселях от правой границы текста. Позволят установить ограничение на ширину строки текста, чтобы текст автоматически переносился на новую строку, когда достигается отступа	Синтаксис: public int RightMargin { get; set; }
		// Устанавливаем отступ справа равный 50 пикселей
		richTextBox1.RightMargin = 50;
		
		// Устанавливаем свойство Multiline в true, чтобы текст переносился на новую строку
		richTextBox1.Multiline = true;

ShortcutsEnable	определяет, будут ли поддерживаться стандартные сочетания клавиш (горячие клавишы)	Синтаксис: public bool ShortcutsEnabled { get; set; } • true (по умолчанию): RichTextBox поддерживает стандартные сочетания клавиш (горячие клавиши, такие как Ctrl+C для копирования, Ctrl+V для вставки, Ctrl+Z для отмены и т.д.) • False: RichTextBox не поддерживает стандартные сочетания клавиш.
ShowSelection Margin	определяет, будет ли отображаться вертикальная полоса выделения слева от выделенного текста. Позволяет визуально выделить область выделения текста, помещая	• true: Вертикальная полоса выделения будет отображаться слева от выделенного текста. • false: Вертикальная полоса выделения не будет отображаться (по умолчанию).

	вертикальную полосу рядом	

TabIndex	определяет порядок перехода фокуса при нажатии клавиши Tab. Когда пользователь нажимает клавишу Tab, фокус переходит на ЭУ со следующим значением TabIndex	Синтаксис: public int TabIndex { get; set; }
		Чем меньше значение TabIndex, тем раньше элемент получит фокус при нажатии клавиши Tab.
		Private void Form1_Load(object sender, EventArgs e)
		{
		RichTextBox richTextBox1 = new RichTextBox();
		richTextBox1.Text = «Текст в RichTextBox»;
		richTextBox1.TabIndex = 1; // Устанавливаем значение TabIndex равным 1
		
		TextBox textBox1 = new TextBox();
		textBox1.Text = «Текст в TextBox»;
		textBox1.TabIndex = 2; // Устанавливаем значение TabIndex равным 2
		
		this.Controls.Add(richTextBox1);
		this.Controls.Add(textBox1);
		}

TabStop	Определяет, уде тли ЭУ получать фокус при нажатии клавиши Tab	• true (по умолчанию): Элемент управления участвует в цикле фокуса при нажатии клавиши Tab. • false: Элемент управления не участвует в цикле фокуса при нажатии клавиши Tab.
Visible	Определяет, будет ли ЭУ видимым или скрытым	• true (по умолчанию): Элемент управления видим на форме и будет отображаться. • false: Элемент управления скрыт и не будет отображаться.
WordWrap	определяет, должен ли текст автоматически переноситься на новую строку при достижении правой границы элемента управления	• true: Текст автоматически переносится на новую строку при достижении правой границы элемента управления (по умолчанию). • false: Текст отображается в одну строку без автоматического переноса.

ZoomFactor	позволяет увеличивать или уменьшать масштаб отображения текста в ЭУ. Позволяет контролировать уровень масштабирования текста относительно исходного размера	Синтаксис: public float ZoomFactor { get; set; }
		Значение float: Уровень масштабирования текста в элементе RichTextBox. Значение 1.0 соответствует исходному масштабу (100%), значения больше 1.0 увеличивают масштаб, а значения меньше 1.0 уменьшают его.
		// Увеличиваем масштаб текста в 150% (1.5 раза)
		richTextBox1.ZoomFactor = 1.5f;

AutoSize	определяет, будет ли размер контрола автоматически изменяться в соответствии с его содержимым	•true-включено автоизменение •false (по умолчанию)
BackgroundImage	установка фонового изображения	richTextBox1.BackgroundImage = Image.FromFile ("путь.jpg");
BackgroundImageLayout	задает способ расположения (масштабирование, выравнивание) фонового	•None – не будет масштабироваться, отображается в исходном размере и позиции. Если больше размера ЭУ, будет обрезано •Tile – будет повторяться вдоль обоих направлений

	изображения, установленного с помощью BackgroundImage	(горгизонтальном/вертикальном), пока не заполнит весь РичБокс • Center – центрируется относительно РичБокса, но отображается в исходном размере • Stretch – растягивается, заполняя весь РичБокс (деформируясь) • Zoom – масштабируется, чтобы занять всю площадь, но сохраняет пропорции richTextBox1.BackgroundImageLayout = ImageLayout.Zoom;
CanRedo	логическое значение, которое показывает, доступла ли операция «Повторить» (Redo). Используется для выполнения обратных действий, котоыре были отменены с помощью Ctrl+Z, чтобы откатить отмену (вернуть все как было) используется операция «Повторить» (Ctrl+Y)	• true – разрешает «Повторить» • false – запрещает (по умолчанию)
CreateParams	представляет структуру CreateParams, которая используется для создания параметров для ЄУ РичБокс, позволяет настроить разные аспекты контрола до создания и отображения на форме. CreateParams – внутренняя структура WinForms, которая содержит информацию о создании и стилях окна для ЭУ, используется внутренне во время процесса создания ЭУ	
Данные:		

DataBindings	позволяет связывать свойства ЭУ с источниками данных. С помощью привязки данных можно автоматически обновлять значения ЭУ на основе данных источника, а также обновлять данные в источнике, когда пользователя изменяет значения ЭУ	Синтаксис:
		public ControlBindingsCollection DataBindings { get; }
		DataBindings является только для чтения (read-only) и представляет коллекцию привязок управления для данного элемента управления. В коллекции содержатся объекты типа Binding, каждый из которых представляет одну привязку данных между свойством элемента управления и источником данных.
		`public class Person`
		`{`
		`public string Name { get; set; }`
		`public int Age { get; set; }`
		`}`

		public partial class Form1 : Form
		{
		private BindingSource bindingSource;
		public Form1()
		{
		InitializeComponent();
		// Создаем BindingSource и добавляем некоторые данные
		bindingSource = new BindingSource();
		bindingSource.DataSource = typeof(Person);
		bindingSource.Add(new Person { Name = "John", Age = 30 });
		
		RichTextBox richTextBox1 = new RichTextBox();
		
		// Настраиваем привязку данных к свойству Text элемента RichTextBox
		richTextBox1.DataBindings.Add("Text", bindingSource, "Name");
		
		this.Controls.Add(richTextBox1);
		
		}
		}
		создается элемент управления RichTextBox, а затем привязывается свойство Text этого элемента к свойству "Name" объекта Person, используя BindingSource. Теперь когда объект Person обновляется, текст в RichTextBox автоматически обновляется, отражая значение "Name" объекта Person.
		
		Привязки данных в Windows Forms предоставляют мощный механизм для связывания данных с элементами управления, что упрощает взаимодействие с данными и обновление пользовательского интерфейса на основе изменений в источнике данных.

Tag	позволяет хранить пользовательские данные, связанные с ЭУ. Имеет тип object, что позволяет хранить любой объект или значение в нем.
	Свойство Tag позволяет связывать произвольные данные с элементами управления, что может быть полезно для дополнительной информации или идентификации элементов во время выполнения программы. Однако следует помнить, что это свойство
		Синтаксис: public object Tag { get; set; }
		class Form1 : Form
		{
		public Form1()
		{
		InitializeComponent();
		
		RichTextBox richTextBox1 = new RichTextBox();
		richTextBox1.Text = "Пример использования свойства Tag";
		
		// Сохраняем дополнительные данные (например, ID объекта) в свойстве Tag
		richTextBox1.Tag = 123;
		
		// Получаем дополнительные данные из свойства Tag и выводим их

	не предназначено для замены специальных классов или структур данных для хранения сложной информации. Если требуется хранить более сложные данные, лучше создать свой класс или структуру данных.	int objectId = Convert.ToInt32(richTextBox1.Tag);
		MessageBox.Show("ID объекта: " + objectId);
		
		this.Controls.Add(richTextBox1);
		}
		}

Дизайн:		
(Name)	специальное свойство, доступное для всех ЭУ, предоставляет уникальное имя элемента управления внутри формы. Обычно устанавливается при создании ЭУ и служит для обращения к нему программно	Синтаксис: public string Name { get; set; } RichTextBox richTextBox1 = new RichTextBox(); richTextBox1.Name = "richTextBox1";
GenerateMember	позволяет управлять генерацией членов элементов управления (control members) для формы. Когда добавляется ЭУ на форму в дизайнере, создается соответсвующее поле (член) в коде формы, чтобы можно было обращаться к ЭУ из кода	Свойство GenerateMember позволяет контролировать, будет ли создаваться такое поле для элемента управления при его добавлении на форму или нет. Если GenerateMember установлено в true (по умолчанию), то при добавлении элемента управления на форму будет автоматически создано поле в коде формы, и это поле будет иметь имя, указанное в свойстве Name элемента управления. Если GenerateMember установлено в false, то при добавлении элемента управления на форму поле в коде формы не будет создаваться, и, следовательно, вы не сможете обращаться к элементу управления из кода формы по имени.
Locked	определяет, можно ли редактировать текст внутри ЭУ	• true: Текст в RichTextBox заблокирован и нельзя редактировать (только для чтения). • false: Текст в RichTextBox разблокирован и можно редактировать.
Modifiers	свойство используется в дизайнере форм для устанвоки модификатора доступа ЭУ, позволяет вібрать уровень доступности ЭУ	Свойство не отображается в коде проекта

CausesValidation	определяет, вызывает ли ЭУ событие «проверка» (validation) при потере фокуса.
	Когда пользователь меняет значение ЭУ (например, вводит текст) и затем переходит к другому ЭУ, происходит событие проверки. Во время проверки происходит проверка наличия ошибок ввода и возможно выполнение
		Синтаксис: public bool CausesValidation { get; set; }
		• true (по умолчанию): Элемент управления вызывает событие проверки при потере им фокуса.
		• false: Элемент управления не вызывает событие проверки при потере им фокуса.
		public Form1()
		{
		InitializeComponent();
		
		TextBox textBox1 = new TextBox();
		textBox1.Text = "Текстовое поле";
		
		// Устанавливаем CausesValidation в false
		textBox1.CausesValidation = false;
		
		this.Controls.Add(textBox1);
		}
		
		protected override void OnValidating

	пользовательского кода для обработки ошибки.	`(System.ComponentModel.CancelEventArgs e)`
		`{`
		`base.OnValidating(e);`
		
		`// Код проверки ввода, который вызывается при потере фокуса элементом управления, если CausesValidation установлено в true`
		`}`

Anchor	определяет, как ЭУ будет «прикреплен» (привязан) к краям его контейнера при изменении размеров контейнера. Можно автоматически подстраивать размеры и положение ЭУ относительно родительского контейнера, чтобы ЭУ сохранял свою позицию и размер при изменении размеров формы или другого контейнера	Свойство Anchor представляет собой перечисление AnchorStyles, которое содержит следующие значения:
		• None: не привязан к каким-либо краям контейнера и не изменяет свои размеры при изменении размеров контейнера.
		• Top: прикрепляется к верхнему краю контейнера и сохраняет расстояние от верхнего края при изменении высоты контейнера.
		• Bottom: прикрепляется к нижнему краю контейнера и сохраняет расстояние от нижнего края при изменении высоты контейнера.
		• Left: прикрепляется к левому краю контейнера и сохраняет расстояние от левого края при изменении ширины контейнера.
		• Right: прикрепляется к правому краю контейнера и сохраняет расстояние от правого края при изменении ширины контейнера.
		Свойство Anchor можно установить путем объединения значений AnchorStyles с помощью операции "или" (|).
		`RichTextBox richTextBox1 = new RichTextBox();`
		`richTextBox1.Text = "Текст в RichTextBox";`
		`richTextBox1.Anchor = AnchorStyles.Top | AnchorStyles.Left;`

Dock	определяет, как ЭУ будет заполнять доступное пространство в родительском контейнере (масштабироваться при изменении контейнера)	Значение свойства Dock - это перечисление DockStyle, которое содержит следующие значения: • None (по умолчанию): RichTextBox не заполняет доступное пространство контейнера и сохраняет свой размер. • Top: располагается в верхней части родительского контейнера и растягивается по ширине контейнера. • Bottom: располагается в нижней части родительского контейнера и растягивается по ширине контейнера. • Left: располагается в левой части родительского контейнера и растягивается по высоте контейнера. • Right: располагается в правой части родительского контейнера и растягивается по высоте контейнера. • **Fill: заполняет всё доступное пространство родительского контейнера по ширине и высоте.** Значение Dock можно установить, присваивая одно из значений перечисления DockStyle:
Location	определяет позицию (координаты) верхнего левого угла ЭУ в родительском контейнере.	Синтаксис: public Point Location { get; set; } Point - это структура в C#, представляющая пару координат X и Y в двумерном пространстве. // Устанавливаем позицию верхнего левого угла

		RichTextBox в контейнере (форме) richTextBox1.Location = new Point(100, 50);

Margin	устанавливает отступы (поля) вокруг текста внутри контрола.
	Свойство Margin является объектом типа Padding, который представляет внутренние отступы контрола для каждой из четырех сторон: верхней (Top), нижней (Bottom), левой (Left) и правой (Right). Величина отступов задается в пикселях.
		Синтаксис: public Padding Margin { get; set; }
		
		// Устанавливаем внутренние отступы: 10 пикселей сверху, 20 пикселей снизу, 5 пикселей слева и 15 пикселей справа
		richTextBox1.Margin = new Padding(5, 10, 15, 20);

DefaultSize	статическое свойство, представляет размер ЭУ по умолчанию для экземпляров РичТексБокс, созданных с использованием конструктора без параметров.	Посмотреть, установить нельзя
		•
МЕТОДЫ		

Find	поиск текста внутри РичБокса, если текст найден, его можно выделить и выполнить другие действия	public int Find(string textToFind, int start, int end, RichTextBoxFinds options);
		• textToFind – что ищем
		• start – начальная позиция поиска (индекс символа, с которого нужно начать поиск)
		• end – конечная позиция поиска
		• options – параметры поиска, определяющие дополнительные условия поиска (представляет собой перечисление RichTextBoxFinds, можно комбинировать их значения с помощью побитовых операторов)
		Возвращает:
		индекс первого символа найденого текста или -1, если текст не найден
		
		`private void Form1_Load(object sender, EventArgs e)`
		`{`
		`RichTextBox richTextBox1 = new RichTextBox();`
		`richTextBox1.Text = "Это пример текста для поиска в RichTextBox.";`
		`this.Controls.Add(richTextBox1);`
		
		`// Ищем слово "пример" в тексте`
		`int index = richTextBox1.Find("пример", 0, richTextBox1.Text.Length, RichTextBoxFinds.None);`
		
		`if (index != -1)`
		`{`
		`richTextBox1.Select(index, "пример".Length); // Выделяем найденный текст`
		`}`
		`else`
		`{`
		`MessageBox.Show("Текст не найден.");`

		} }