УДК 004

ОТОБРАЖЕНИЕ ДИНАМИЧЕСКИХ СТРУКТУР ДАННЫХ В ГРАФИЧЕСКОМ РЕЖИМЕ

О.А. Смирнова

Аннотация. *В статье представлена разработка программы на языке программирования C#, отображающей динамические структуры данных в графическом режиме. Приведены пример и отображения на форме «двунаправленный список символьных строк». Описаны методы для рисования объектов на устройстве отображения.*

Ключевые слова: *языки программирования, C#, двунаправленный (двусвязный) список, базовые классы.*

Введение

C# является мощным и гибким языком программирования, который широко применяется для решения различных задач, и его популярность и востребованность в индустрии программирования продолжают расти. C# отлично подходит для программирования в различных областях и является прямым продолжением успешных компьютерных языков: C и C++. Он наследует синтаксис, ключевые слова и операторы от языка C, а также базируется на объектной модели языка C и улучшает ее. C# активно поддерживается и развивается компьютерными компаниями, что обеспечивает постоянное обновление языка и его инструментария. Стабильность разработки, регулярные обновления и улучшения гарантируют его актуальность и конкурентоспособность в современной индустрии программирования.

C# предоставляет разработчикам инструментарий для создания разнообразных приложений. Он основан на принципах объектно-ориентированного программирования, что способствует созданию чистого, структурированного и модульного кода. Благодаря широкому набору встроенных библиотек и инструментов программисты могут быстро и эффективно разрабатывать приложения для различных платформ, веб-сервисы, мобильные устройства и пр.

Синтаксис данного языка программирования довольно простой и легко читаемый, что делает C# привлекательным для начинающих программистов. Он поддерживает множество функций, которые помогают упростить кодирование и повысить производительность разработчика. C# компилируется в машинный код, что обеспечивает быструю работу программ и оптимальное использование ресурсов компьютера. Кроме того, C# обладает отличной масштабируемостью и поддержкой много-поточности, благодаря чему можно создавать эффективные и отзывчивые приложения.

Двунаправленный список

Двунаправленный (двусвязный) список (рис. 1) – список, состоящий из последовательности элементов, каждый из которых содержит информационную часть (данные) и два указателя на соседние элементы (следующий и предыдущий). В пространстве имен .NETC# System.Collections.Generic содержится класс LinkedList<T>, реализующий двунаправленный (двусвязный) список [2].

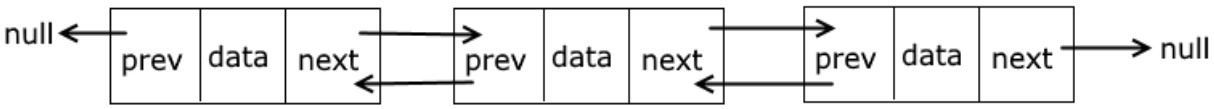

Рис. 1. Пример двусвязного списка

В обычном универсальном списке List<T> каждый элемент – объект типа T, в то время как в LinkedList<T> каждый узел – объект класса LinkedListNode<T>. Класс LinkedListNode<T> имеет следующие свойства:

1. Next – ссылка на следующий элемент типа LinkedListNode<T> в списке. Если следующий элемент отсутствует, то Next имеет значение null.
2. Value – само значение узла, представленное типом T.
3. Previous – ссылка на предыдущий элемент типа LinkedListNode<T> в списке. Если предыдущий элемент отсутствует, то имеет значение null.

Двусвязные (двунаправленные) списки отлично подходят для решения каких-либо геометрических задач. Например, двусвязным списком удобно представлять множество вершин многоугольника: каждый узел списка будет описывать очередную вершину многоугольника, соответственно, можно достаточно легко определить, какая вершина является предыдущей, а какая – следующей [2].

Windows Forms

Windows Forms является одной из самых популярных технологий для создания графических пользовательских интерфейсов на платформе .NET с использованием языка программирования C#. Это мощное средство разработки позволяет создавать интерактивные приложения с удобным и интуитивно понятным интерфейсом.

Одно из главных преимуществ Windows Forms – простота использования. Разработчики могут легко создавать различные элементы управления (кнопки, текстовые поля, списки, таблицы и др.), а затем связывать их с логикой приложения при помощи языка программирования C#. Это делает процесс создания приложений быстрым и эффективным.

Широкий набор функциональных возможностей, присущий Windows Forms, позволяет создавать разнообразные приложения, от простых утилит до сложных корпоративных систем. С его помощью можно легко добавлять анимацию, графику, обработку событий и другие функции, благодаря чему приложения становятся более функциональными и привлекательными для пользователей.

Интеграция с платформой .NET обеспечивает высокую производительность Windows Forms и надежность приложений, а также возможность легкой масштабируемости и поддержки различных устройств и разрешений экранов [1].

Класс Brush

Класс Brush в C# является частью пространства имен System.Drawing, которое предоставляет базовые классы для работы с графикой и рисованием. Brush используется для заливки графических объектов, таких как прямоугольники, эллипсы, линии и др. Brush является абстрактным классом, поэтому нельзя создать экземпляр этого класса напрямую. Вместо этого используются конкретные производные классы, которые предоставляют различные способы заливки объектов. Brush часто применяется вместе с другими классами из пространства имен System.Drawing, такими как Graphics, Pen и Bitmap, для создания графических изображений и пользовательских интерфейсов в приложениях на платформе .NET [1, 3, 4] (табл. 1).

Таблица 1

Описание методов и классов для работы с графикой и рисованием

Методы и кисти	Описание
SolidBrush	Класс в пространстве имен System.Drawing, определяющий кисть одного цвета
HatchBrush	Класс в пространстве имен System.Drawing, используемый для закраски области определенным узором
TextureBrush	Класс в пространстве имен System.Drawing, который позволяет создать кисть, содержащую изображение, и рисовать фигуры, которые затем будут залиты этим изображением
SolidColorBrush	Кисть, закрашивающая область сплошным цветом
RadialGradientBrush	Градиентная кисть, плавно распределяющая заданные цвета от центральной точки к внешним границам. ImageBrush в качестве заполнителя использует не цвет, а изображение
LinearGradientBrush	Кисть, которая позволяет создавать смешанное закрашивание, представляющее собой переход от одного цвета к другому

Класс Graphics

Класс Graphics в C# представляет собой основной инструмент для рисования на графическом устройстве (экране или принтере). С помощью этого класса можно создавать различные графические элементы, включая линии, фигуры, текст и изображения [5].

С помощью объекта Graphics можно выполнять различные операции рисования (табл. 2), такие как рисование линий, прямоугольников, эллипсов и текста. Кроме того, класс Graphics позволяет работать с различными цветами, шрифтами и стилями линий для создания разнообразных графических элементов.

Для использования класса Graphics в C# обычно создают объект этого класса, связанный с конкретным устройством рисования, например с объектом PictureBox или элементом управления Panel. Затем с помощью

методов класса Graphics можно выполнять рисование непосредственно на этом устройстве [1, 5].

Таблица 2

Описание методов для различных операций рисования

Имя метода	Описание
DrawEllipse(Pen, Rectangle)	Рисует эллипс
DrawLine(Pen, Point, Point)	Проводит линию, соединяющую две структуры Point
Clear(Color)	Очищает всю поверхность рисования и выполняет заливку поверхности указанным цветом фона
DrawImage(Image image, int x, int y)	Рисует заданное изображение image, используя его фактический размер в месте с координатами (x, y)
DrawIcon(Icon, Rectangle)	Рисует значок
DrawRectangle(Pen, Rectangle)	Рисует прямоугольник, определяемый структурой Rectangle

Класс Pen

В C# класс Pen представляет собой объект, который используется для определения стиля линии при рисовании на графическом объекте. Объект Pen позволяет задать цвет, ширину и стиль линии для отрисовки на экране [1].

Для использования класса Pen в C# необходимо:

1. Создать объект Pen, указав необходимые параметры (цвет, ширину, стиль и т. д.).

2. Создать объект Graphics, на котором будет происходить рисование.

3. Использовать методы объекта Graphics для отрисовки, передавая созданный объект Pen в качестве параметра для определения стиля линии [1, 5].

Таблица 3

Описание свойств класса для определения стиля линии

Ключевые свойства класса	Что определяют
Color	Цвет линии
Width	Ширину линии в пикселях
DashStyle	Стиль линии (сплошной, пунктирный, пунктирно-точечный и др.)
StartCap и EndCap	Стили концов линии (круглый, квадратный, треугольный и др.)

Класс Font

Класс Font в C# представляет собой объект, который содержит информацию о шрифте текста (название шрифта, размер, стиль (курсив, полужирный) и другие свойства). Этот класс используется для управления отображением текста с помощью различных шрифтов в приложениях, разработанных на платформе .NET [1, 5].

Таблица 4

Описание свойств и методов для шрифта текста

Свойства и методы	Описание
Name	Свойство, которое возвращает или устанавливает название шрифта
Size	Свойство, определяющее размер шрифта
Unit	Единица измерения для размера шрифта (пиксель, дюйм и др.)
Style	Свойство, позволяющее задать стиль шрифта (курсив, полужирный и др.)
SetFont()	Метод для установки шрифта с определенными параметрами
ToString()	Метод, возвращающий строковое представление объекта Font

Общие сведения

Для реализации программы был создан проект FormKurs, содержащий файлы:

1. BidirLinkList.cs (двусвязный список символьных строк).

В данном файле был создан класс BidirLinkList, хранящий первую и последнюю ноды (узловые точки), необходимые для реализации двусвязного списка символьных строк, свойство Count, которое позволяет получить размер ДСС, конструктор BidirLinkList по значению первой ноды, а также конструктор BidirLinkList по массиву (учтено, что если дан пустой список, то выводится ошибка (исключение)).

Применяемые методы:

AppendValue (добавление ноды в конец двусвязного списка по значению это создание ноды с нужным значением и добавление ее в конец двусвязного списка);

AppendNode (добавление ноды сразу в конец двусвязного списка);

InsertNode (добавление ноды на произвольную позицию двусвязного списка);

UpDateFLNode (обновление указателей на первую (First) и последнюю (Last) ноды);

GetNode (поиск и возвращение ноды по индексу, при отсутствии такой ноды – возврат null);

DeleteNode (удаление ноды по ее номеру (ссылки на удаленные ноды не остается));

2. BidirNode.cs (нода двусвязного списка символьных строк).

В данном файле был создан класс BidirNode (строковое значение ноды, ссылки на предыдущую и следующую ноды), который описывает ноду двусвязного списка символьных строк и ее поведение. Учтены следующие два момента:

1) если нода ни на что не указывает, то она является единственной;

2) должна существовать как минимум одна нода.

3. VisualNode.cs (отображаемое на экране представление ноды).

В данном файле создан класс VisualNode – отображаемое на экране представление ноды (с указанием позиции на экране, ссылки на представляемую ноду, а также того, чем рисуются границы, закрашивается фон, цвет и шрифт текста).

4. FormMain.cs (главная форма)

В данном файле происходит создание переменных:

linkedList – экземпляр двусвязного списка, ручное заполнение его нодами;

visualNodes – список (изначально пустой) визуальных представлений нод, то, как будут выглядеть ноды.

Разработан конструктор формы (запускается при запуске программы), автоматическое создание в нем InitializeComponent() – инициализация формы и ее компонентов, а также ручное создание метода InitializeVisualNodes() – заполнение списка визуальных представлений нод и его обновление (создается новый пустой список визуальных представлений нод, берется первая нода из linkedList и создается новая визуальная нода в точке (x, y) с данными, взятыми из FirstNode. Если индекс текущей ноды, которая отрисовывается, находится в границах left и right, то ее BackgroundBrush меняется на Aquamarine, после чего данная нода добавляется в список визуальных нод. Затем происходит сдвиг x на 250 пикселей, чтобы следующая нода отрисовывалась со сдвигом в 250 пикселей, следующая нода берется после выполнения вышеперечисленных действий. Создается метод DisplayLinkedList, который отрисовывает ноды с помощью их визуальных представлений (для каждой ноды в списке вызывается цикл и идет отрисовка ноды, а также стрелок). Метод Redraw обновляет список визуальных представлений нод и вызывает его отрисовку. Событие FormMain_MouseClick – то, что происходит при клике мышью на форму (проверяется, было ли нажатие мышью на какую-либо ноду; если место нажатия находится внутри ноды, происходит обновление границ; после проверки всех условий производится вызов метода отрисовки ReDraw). Событие FormMain_Paint возникает при перерисовке формы. События buttonAddNode_Click и buttonDeleteNode_Click нужны для добавления и удаления нод при нажатии на соответствующие кнопки (проверка существования и местоположения границ и вызов удаления/добавления нод). На рис. 2 представлены файлы проекта FormKurs [5].

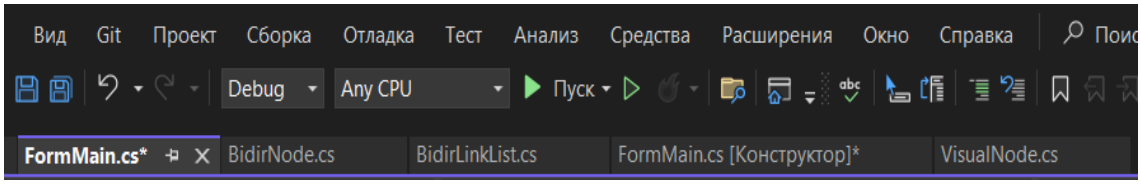

Рис. 2. Файлы проекта FormKurs

На рис. 3 показан результат работы программы – форма, заполненная строками двусвязного списка, также содержащая кнопки «Добавить в конец» и «Удалить выбранное».

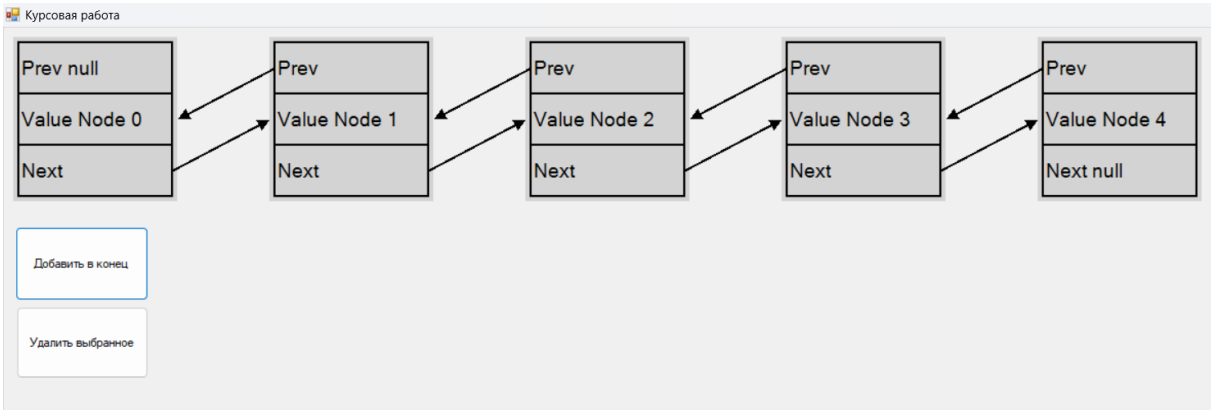

Рис. 3. Форма после выполнения программы

На рис. 4 представлена форма после нажатия мышью на кнопку «Добавить в конец».

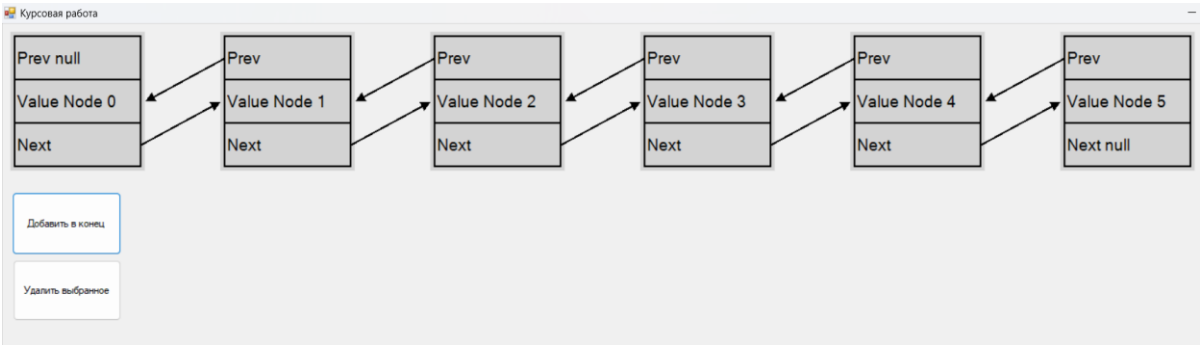

Рис. 4. Форма после нажатия на кнопку «Добавить в конец»

Пример выделения и изменения цвета строк приведен на рис. 5.

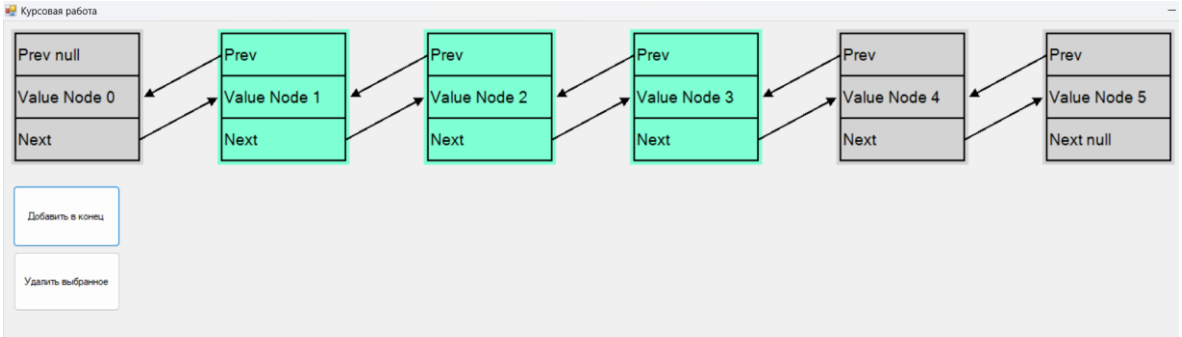

Рис. 5. Форма после выделения нескольких строк

На рис. 6 представлена форма после нажатия мышью на кнопку «Удалить выбранное».

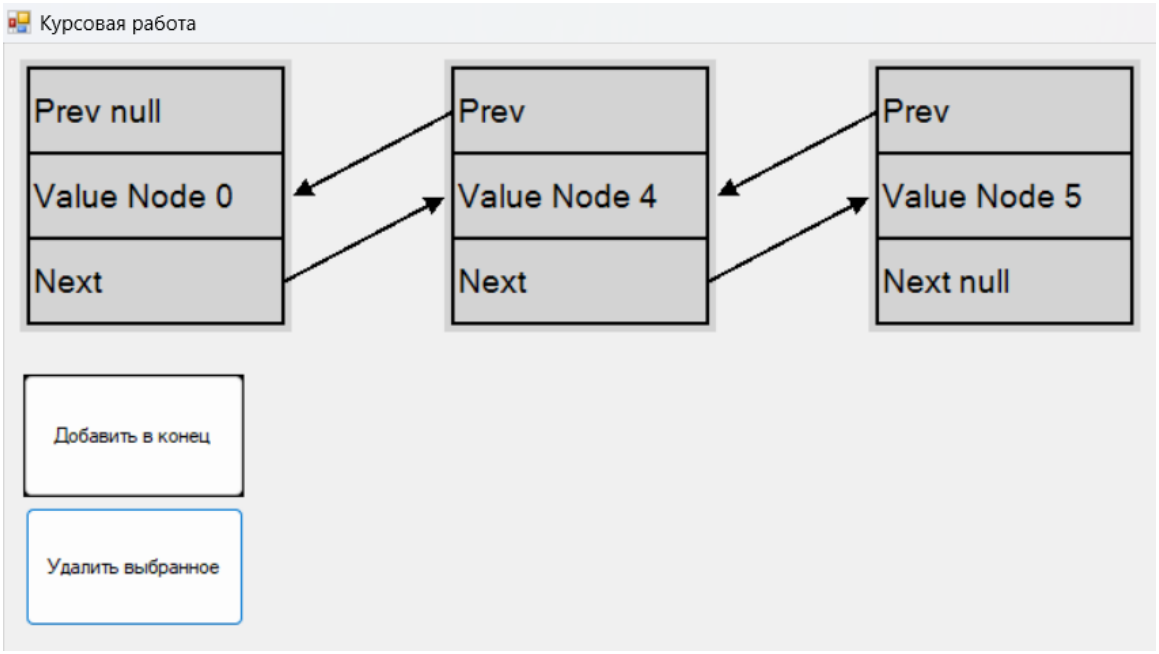

Рис. 6. Форма после нажатия кнопки «Удалить выбранное»

Заключение

Проведен анализ различных классов, необходимых для реализации поставленных задач, а также проанализирован двусвязный (двунаправленный) список.

Библиографический список

1. Двумерная графика на C#, классы Graphics, Pen и Brush. URL: https://c-sharp.pro/общие-замечания-классы-graphics-pen-и-brush/ (дата обращения: 28.03.2024).

2. Двунаправленный (двусвязный) список в C#. URL: https://csharp.webdelphi.ru/dvunapravlennyj-dvusvyaznyj-spisok-v-c-klass-linkedlist/#:~:text=Двунаправленный%20(или%20двусвязный)%20список%20–,LinkedList%3CT%3E%2C%20реализующий%20двунаправленный%20(двусвязный%20список) (дата обращения: 28.03.2024).

3. Использование кисти для заливки фигур – Windows Forms .NET Framework. URL: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/advanced/using-a-brush-to-fill-shapes?view=netframeworkdesktop-4.8 (дата обращения: 28.03.2024).

4. Brushes and Filled Shapes in GDI+ – Windows Forms .NET Framework. URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/advanced/brushes-and-filled-shapes-in-gdi?view=netframeworkdesktop-4.8 (дата обращения: 28.03.2024).

5. Практическое руководство. Создание объектов Graphics для рисования – Windows Forms. NET Framework | Microsoft Learn. URL: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/advanced/how-to-create-graphics-objects-for-drawing?view=netframeworkdesktop-4.8 (дата обращения: 28.03.2024).

DISPLAYING DYNAMIC DATA STRUCTURES IN GRAPHIC MODE

O.A. Smirnova

Abstract. *The article presents the development of a program in the C# programming language that displays dynamic data structures in graphical mode. An example and displays on the form "bidirectional list of character strings" are given. Methods for drawing objects on the display device are described.*

Keywords: *programming languages, C#, bidirectional (doubly linked) list, base classes.*

Об авторе:

СМИРНОВА Ольга Александровна – бакалавр, факультет информационных технологий, ФГБОУ ВО «Тверской государственный технический университет», Тверь. E-mail: olas40660@gmail.com

About the author:

SMIRNOVA Olga Alexandrovna – Bachelor's degree, Faculty of Information Technologies, Tver State Technical University, Tver. E-mail: olas40660@gmail.com