Оглавление

[1. Задание на выполнение курсовой работы 2](#_Toc294650987)

[2. Теоретический материал по используемым динамическим структурам данных и средствам разработки приложений с графическим интерфейсом на С# 2](#_Toc294650988)

[2.1. Список **Ошибка! Закладка не определена.**](#_Toc294650989)

[2.2. Windows Forms. 5](#_Toc294650990)

[3. Описание используемых в программе структур данных 6](#_Toc294650991)

[4.Алгоритм всех функций программы в виде псевдокода 7](#_Toc294650992)

[5. Пример работы программы 9](#_Toc294650993)

[6. Список источников 16](#_Toc294650994)

[Приложение 17](#_Toc294650995)

1. **Задание на выполнение курсовой работы:**

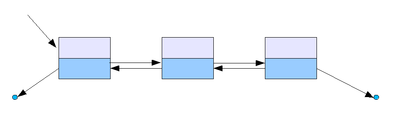
**Вариант 4, уровень 3.**

Создать и отобразить на форме двунаправленный список символьных строк. Выбрать с помощью мыши любую часть этого списка, указав начальную и конечную строку, и выделить ее другим цветом. Создать командную кнопку, вызывающую функцию, удаляющую выбранную часть списка.

1. **Теоретический материал по используемым динамическим структурам данных и средствам разработки приложений с графическим интерфейсом на С#** 
   1. **Список**

Связный список — структура данных, состоящая из узлов, каждый из которых содержит как собственные данные, так и одну или две ссылки («связки») на следующий и/или предыдущий узел списка. Принципиальным преимуществом перед массивом является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера, а порядок обхода списка всегда явно задаётся его внутренними связями.

**Двунаправленный связный список (Двусвязный список)**



Двунаправленный (двусвязный) список – это структура данных, состоящая из последовательности элементов, каждый из которых содержит информационную часть и два указателя на соседние элементы. При этом два соседних элемента должны содержать взаимные ссылки друг на друга.

В таком списке каждый элемент (кроме первого и последнего) связан с предыдущим и следующим за ним элементами. Каждый элемент двунаправленного списка имеет два поля с указателями: одно поле содержит ссылку на следующий элемент, другое поле – ссылку на предыдущий элемент и третье поле – информационное. Наличие ссылок на следующее звено и на предыдущее позволяет двигаться по списку от каждого звена в любом направлении: от звена к концу списка или от звена к началу списка, поэтому такой список называют двунаправленным.

Рассмотрим основные операции, осуществляемыми с двунаправленными списками, такие как:

* создание списка;
* печать (просмотр) списка;
* вставка элемента в список;
* удаление элемента из списка;
* поиск элемента в списке;
* проверка пустоты списка;
* удаление списка.

**Достоинства**

* лёгкость добавления и удаления элементов
* размер ограничен только объёмом памяти компьютера и разрядностью указателей
* динамическое добавление и удаление элементов

**Недостатки**

* сложность определения адреса элемента по его индексу (номеру) в списке
* на поля-указатели (указатели на следующий и предыдущий элемент) расходуется дополнительная память (в массивах, например, указатели не нужны)
* работа со списком медленнее, чем с массивами, так как к любому элементу списка можно обратиться, только пройдя все предшествующие ему элементы
* элементы списка могут быть расположены в памяти разреженно, что окажет негативный эффект на кэширование процессора
* над связными списками гораздо труднее (хотя и в принципе возможно) производить параллельные векторные операции, такие как вычисление суммы
* кэш-промахи при обходе списка

* 1. **Windows Forms.**

**Windows Forms** — название интерфейса программирования приложений(API), отвечающего заграфический интерфейс пользователя и являющегося частью Microsoft.NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обертки для существующего Win 32 API в управляемом коде. Причем управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, С++, так и на VB.Net, J# и др.

С одной стороны Windows Forms рассматривается как замена более старой и сложной библиотеке [MFC](http://ru.wikipedia.org/wiki/Microsoft_Foundation_Classes), изначально написанной на языке [C++](http://ru.wikipedia.org/wiki/C%2B%2B), но с другой стороны, WF не предлагает парадигму, сравнимую с [MVC](http://ru.wikipedia.org/wiki/Model-View-Controller). Для исправления этой ситуации и реализации данного функционала в WF существуют сторонние библиотеки. Одной из наиболее используемых подобных библиотек является [User Interface Process Application Block](http://www.microsoft.com/downloads/details.aspx?familyid=98C6CC9D-88E1-4490-8BD6-78092A0F084E&displaylang=en), выпущенная специальной группой Microsoft, занимающейся примерами и рекомендациями, для бесплатного скачивания. Эта библиотека также содержит исходный код и обучающие примеры для ускорения обучения.

Внутри .NET Framework, Windows Forms реализуется в рамках [пространства имён](http://ru.wikipedia.org/wiki/Пространство_имён_(программирование)) System.Windows.Forms.

Приложения, использующие Windows Forms используют классы System.WinForms. Этот раздел включает такие классы, как Form, который моделирует поведение окон или форм; Menu, который представляет меню; Clipboard, который дает возможность приложениям Windows Forms использовать буфер обмена. Он также содержит многочисленные классы, предоставляющие средства управления, например: Button, TextBox, ListView, MonthCalendar и т.д. Эти классы могут быть включены в приложение либо с использованием только имени класса, либо с использованием полного имени, например: System.WinForms.Button.

Окна получают сообщения WM\_PAINT и большинство перерисовок экрана выполнено в ответ на эти сообщения. В Windows Forms эквивалент сообщения WM\_PAINT - виртуальный метод по имени OnPaint. Производный класс формы может переопределить этот метод в случае надобности выполнять собственную перерисовку в ответ на сообщения WM\_PAINT.

Приложение Windows Forms представляет собой [событийно-ориентированное приложение](http://ru.wikipedia.org/wiki/Событийно-ориентированное_программирование), поддерживаемое Microsoft [.NET Framework](http://ru.wikipedia.org/wiki/.NET_Framework). В отличие от [пакетных программ](http://ru.wikipedia.org/w/index.php?title=Пакетная_обработка&action=edit&redlink=1)большая часть времени тратится на ожидание от пользователя каких-либо действий, как например, ввод текста в [текстовое поле](http://ru.wikipedia.org/w/index.php?title=Текстовое_поле&action=edit&redlink=1) или клика мышкой по [кнопке](http://ru.wikipedia.org/wiki/Кнопка_(техника)).

# Описание используемых в программе структур данных:

Класс Form1 содержит функции Добавление, Рисование, Щелчок мыши, Удаление строк .

1. **Алгоритм всех функций программы в виде псевдокода:**

Начало

Начало класса Form1 наследование Form

Объявление строкового списка sp

число y0

число dy

число iSelBeg

число iSelEnd

Начало функции Form1()

Инициализация компонентов

sp = новый строковый список

iSelBeg = -1

iSelEnd = -1

y0 = 50

dy = 40

Конец функции Form1

Начало функции toolStripTextBox1\_KeyPress(объект sender, событие e)

символ x = e.KeyChar

Если x = 13, то

sp.Добавить(toolStripTextBox1.Text)

toolStripTextBox1.Text = ""

Перерисовать

Все если

Конец функции toolStripTextBox1\_KeyPress

Начало функции Form1\_Paint(объект sender, событие e)

число x = 50

число R = 10

число y = y0

График g = этот.CreateGraphics()

Формат f = новый формат("Arial", 12)

логическая переменная flag = ложь

Для i от 0 до sp.Щетчик с шагом 1

Если i >= iSelBeg и i <= iSelEnd, то

g.Рисовать строку(sp[i], f, Кисть.Red, x, y)

Все если

Иначе

g.Рисовать строку(sp[i], f, Кисть.Black, x, y)

Все иначе

g.Рисовать эллипс(Перо.Blue, x-4\*R, y, 2\*R, 2\*R)

g.Рисовать линию(Перо.Blue, x-3\*R, y+2\*R, x-3\*R, y+4\*R)

g.Рисовать линию(Перо.Blue, x-3\*R-3, y+4\*R-5, x-3\*R, y+4\*R)

g.Рисовать линию(Перо.Blue, x-3\*R+3, y+4\*R-5, x-3\*R, y+4\*R)

g.Рисовать линию(Перо.Blue, x-2\*R, y+R, x, y+R)

g.Рисовать линию(Перо.Blue, x-5, y+R-3, x, y+R)

g.Рисовать линию(Перо.Blue, x-5, y+R+3, x, y+R)

y = y+dy

Все для i

g.Рисовать строку("null", f, Кисть.Black, x-4\*R, y)

toolStripTextBox1.Focus()

Конец функции Form1\_Paint

Начало функции Form1\_MouseDown(объект sender, событие e)

число y = y0

Для i от 0 до sp.Щетчик с шагом 1

Если e.Y >= y и e.Y < y+dy, то

Если iSelBeg < 0, то

iSelBeg = i

Все если

Иначе

iSelEnd = i

Перерисовка

Все иначе

остановить

Все если

y = y+dy

Все для i

Конец функции Form1\_MouseDown

Начало функции toolStripButton1\_Click(объект sender, базовый класс e)

Если iSelEnd == -1 и iSelBeg == -1, то

Показать окно (Нужно сначала указать строки,"+

" а затем удалять!", "Ошибка", Константы.OK, Константы.Error)

Все если

Иначе

Если iSelEnd >= iSelBeg, то

sp.Удалить диапазон(iSelBeg, iSelEnd - iSelBeg + 1)

Перерисовка

Все если

Иначе

Показать окно ("Нужно сначала указать первую строку блока,"+

" а затем последнюю!", "Ошибка", Константы.OK,Константы.Error)

Все иначе

Все иначе

iSelBeg = iSelEnd = -1;

Конец функции toolStripButton1\_Click

Конец класса Form1

Конец

# Пример работы программы:

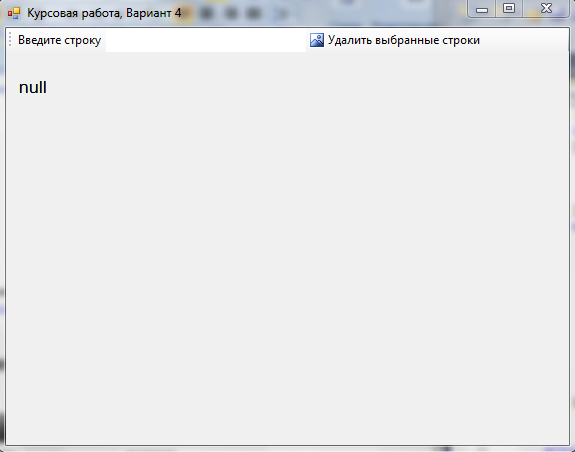
****

Рисунок 1 « Программа в режиме ожидания»

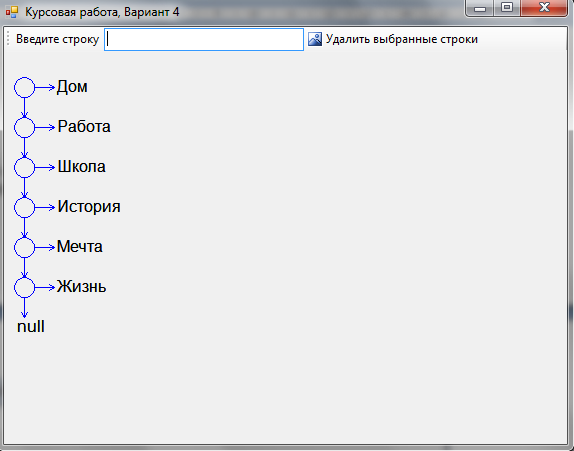
****

Рисунок 2 «Ввод данных и вывод их»

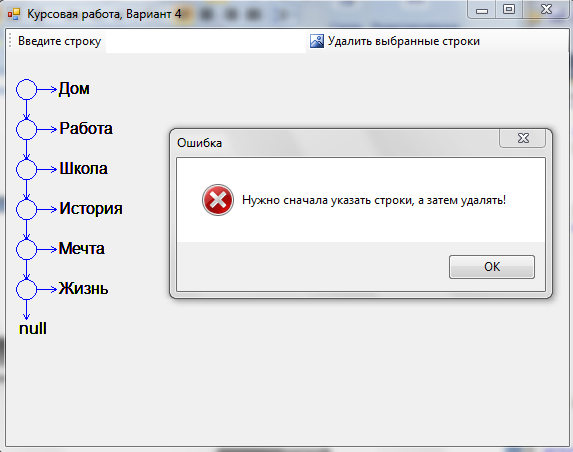
****

Рисунок 3 «Вывод сообщения при отсутствии выбранных строк для удаления»

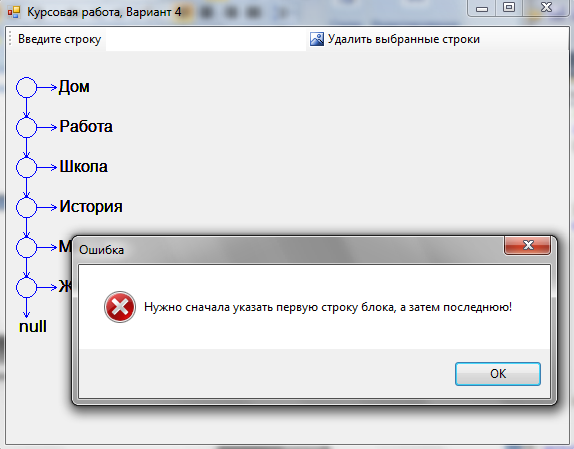


Рисунок 4 «Вывод сообщения при неправильном выделении строк»

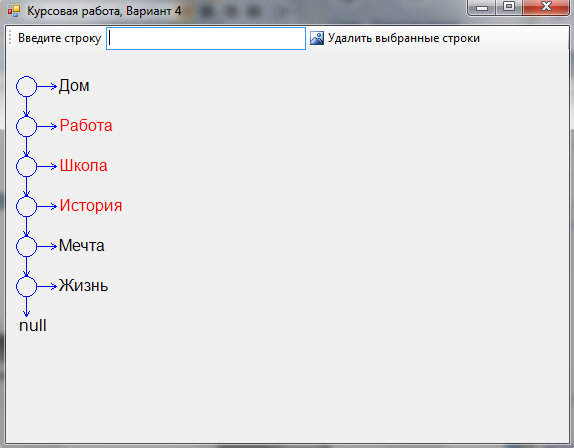


Рисунок 5 «Выделение диапазона строк»

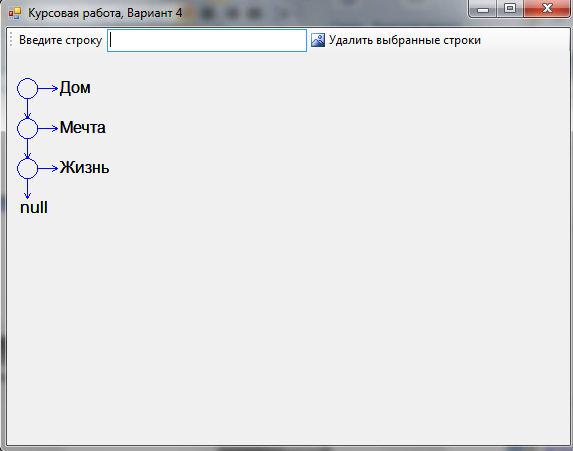


Рисунок 6 «Удаление выделенных строк и вывод полученного списка»

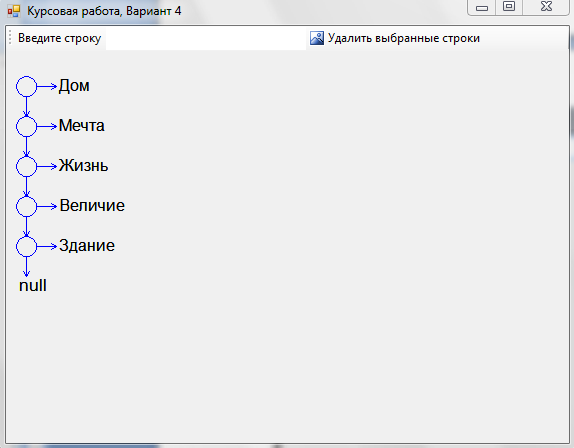


Рисунок 7 «Ввод к оставшимся строкам новые»

# Список источников

<http://www.realcoding.net/article/view/1808> - Windows Forms: Современная модель программирования для создания приложений

<http://ru.wikipedia.org/wiki/Windows_Forms>

<http://www.intuit.ru/department/algorithms/staldata/29/3.html>

<http://ru.wikipedia.org/wiki/%D1%E2%FF%E7%ED%FB%E9_%F1%EF%E8%F1%EE%EA>

# Приложение

**Исходный текст программы**

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;

namespace Курсовая\_Вар\_4

{

public partial class Form1 : Form

{

List<string> sp; // Список строк - используем стандартный класс

int y0; // Координата У первой строки списка

int dy; // Расстояние между строками списка

int iSelBeg; // Индексы начала и конца блока

int iSelEnd;

public Form1()

{

InitializeComponent();

sp = new List<string>();

iSelBeg = -1;

iSelEnd = -1;

y0 = 50;

dy = 40;

}

private void toolStripTextBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

char x = e.KeyChar;

if (x == 13) // ENTER

{

sp.Add(toolStripTextBox1.Text);

toolStripTextBox1.Text = "";

Invalidate();

}

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

int x = 50;

int R = 10;

int y = y0;

Graphics g = this.CreateGraphics();

Font f = new Font("Arial", 12);

bool flag = false;

for (int i = 0; i < sp.Count; i++)

{

// Формируем флаг для закраски блока между выбранными строками

if (i >= iSelBeg && i <= iSelEnd)

{

g.DrawString(sp[i], f, Brushes.Red, x, y);

}

else

{

g.DrawString(sp[i], f, Brushes.Black, x, y);

}

g.DrawEllipse(Pens.Blue, x - 4 \* R, y, 2\*R, 2\*R);

// Вертикальную стрелку

g.DrawLine(Pens.Blue, x - 3 \* R, y + 2 \* R, x - 3 \* R, y + 4 \* R);

g.DrawLine(Pens.Blue, x - 3 \* R - 3, y + 4 \* R - 5, x - 3 \* R, y + 4 \* R);

g.DrawLine(Pens.Blue, x - 3 \* R + 3, y + 4 \* R - 5, x - 3 \* R, y + 4 \* R);

// Горизонтальную стрелку

g.DrawLine(Pens.Blue, x - 2 \* R, y + R, x, y + R);

g.DrawLine(Pens.Blue, x - 5, y + R - 3, x, y + R);

g.DrawLine(Pens.Blue, x - 5, y + R + 3, x, y + R);

y = y + dy; // Определяем положение следующей строки

}

g.DrawString("null", f, Brushes.Black, x-4\*R, y);

toolStripTextBox1.Focus(); // Для ввода следубщей строки

}

private void Form1\_MouseDown(object sender, MouseEventArgs e)

{

// Обрабатываем щелчок мыши - ищем строку !!!

int y = y0;

for (int i = 0; i < sp.Count; i++)

{

if (e.Y >= y && e.Y < y + dy)

{

if (iSelBeg < 0) // Для начальной строки

{

iSelBeg = i;

}

else // Для конечной строки

{

iSelEnd = i;

Invalidate();

}

break;

}

y = y + dy; // Определяем положение следующей строки

}

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

// Удаляем выбранные строки

if (iSelEnd == -1 && iSelBeg == -1)

{

MessageBox.Show("Нужно сначала указать строки,"+

" а затем удалять!", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

if (iSelEnd >= iSelBeg)

{

sp.RemoveRange(iSelBeg, iSelEnd - iSelBeg + 1);

Invalidate();

}

else

MessageBox.Show("Нужно сначала указать первую строку блока," +

" а затем последнюю!", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

iSelBeg = iSelEnd = -1;

}

private void Form1\_Load(object sender, EventArgs e)

{

}

}

}