Лабораторная работа №7

Тема: Структуры данных – связный список, стек, очередь.

Задание:

Создать приложение, реализующее основную функциональность списков, стеков, очередей на основе связных списков и сравнение производительности с теми же функциональными элементами, основанными на динамических массивах, в соответствии с индивидуальным заданием.

Помимо явно поддерживаемых языками программирования сложных типов данных существует набор стандартных решений, основанных на структурах. Одной из таких составных структур данных является список.

Из повседневной жизни мы знаем, что список — это динамический последовательный набор однотипных элементов. Изначально, списки появились именно для хранения динамических наборов (динамические массивы были придуманы позднее). Поскольку это динамическая структура данных, ее использование тесно связано с динамической памятью и указателями.

Какие различия между списками и динамическими массивами, и каковы их преимущества и недостатки?

Массивы записаны в памяти последовательно. Элементы списка могут располагаться в произвольном порядке, с «дырами» свободной динамической памяти между ними.

В массиве можно легко обратиться к любому элементу по индексу. Список — структура строго последовательного доступа.

В массиве отсутствуют ссылки элементов друг на друга (за ненадобностью), в списке они есть и занимают хотя и небольшое, но заметное количество памяти (накладные расходы). (на самом деле, накладные расходы на элементы списков малого размера еще больше — помимо ссылок, каждый элемент списка — самостоятельная область динамической памяти, соответственно, добавляются накладные расходы менеджера памяти).

При добавлении или удалении элемента массив надо переформировать целиком. В списке надо изменить всего лишь пару указателей.

Соответственно, получатся, что массивы рационально применять для сравнительно небольших, не часто изменяющихся наборов данных малого размера. Списки — для больших элементов, большого набора и подверженного частым изменениям. Кроме того, списки могут применяться при работе с данными строго последовательного доступа. В случае очень больших наборов выступает еще одна особенность — может не быть достаточно большого непрерывного блока памяти для хранения массива.

Каким образом можно сравнить эффективность списка и массива? Самый простой путь — сравнить время выполнения тех или иных операций.

Для получения времени можно воспользоваться функцией GetTickCount из модуля Windows. Она возвращает количество миллисекунд с момента загрузки системы, и удобна для измерения небольших непрерывных интервалов времени. Для сверхмалых и очень больших интервалов она неприменима — точность составляет примерно 16 мс, а через 49.7 суток счетчик обнуляется.

Использовать ее можно следующим образом:

private void button1\_Click(object sender, EventArgs e)

{

int StartTime, ResultTime;

StartTime = Environment.TickCount;

DoSomethingSlow();

ResultTime = Environment.TickCount - StartTime;

MessageBox.Show(String.Format("Время выполнения {0:f2} сек", (float)ResultTime/1000.0));

}

Таким образом, вам необходимо сравнить время выполнения соответствующих варианту операций с использованием элементов данных размером 16 байт и 100 Кбайт:

unsafe struct SmallData

{

public fixed byte a[16];

}

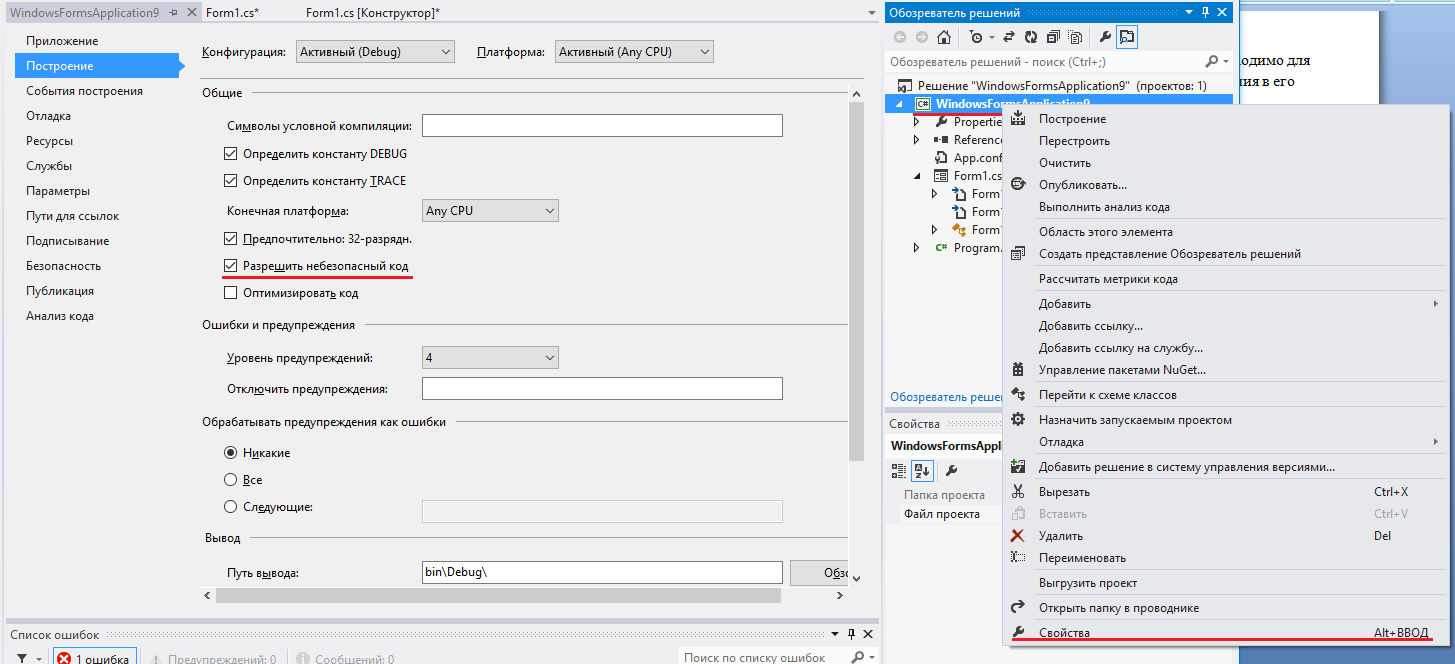
unsafe struct LargeData

{

public fixed byte a[102400];

}

Обратите внимание, что типы объявлены сос пецификатором unsafe. Это необходимо для реализации «настоящих» статических массивов. Для работы данного приложения в его свойствах необходимо разрешить небезопасный код.



Для реализации структур на основе массивов и структур на основе связных списков мы воспользуемся стандартными шаблонами, объявленными в пространстве имен System.Collections.Generic.

List<SmallData> SmallList = new List<SmallData>();

LinkedList<LargeData> LargeLinkedList = new LinkedList<LargeData>();

Тиы структур стек и очередь определяются их использованием. Все операции должны быть повторены в цикле 2500 раз, каждый раз с измерением времени исполнения.

Вариант 1.

1. Очередь на основе двусвязного списка. Операции помещения и удаления элемента.
2. Стек на основе двусвязного списка. Операции протолкнуть и вытолкнуть.
3. Двусвязный список. Операции поиска элемента, добавления элемента в конец.

Вариант 2.

1. Очередь на основе односвязного списка. Операции помещения и удаления элемента.
2. Стек на основе двусвязного списка. Операции протолкнуть и вытолкнуть.
3. Двусвязный список. Операции удаления элемента, добавления элемента в начало.

Вариант 3.

1. Очередь на основе двусвязного списка. Операции помещения и удаления элемента.
2. Стек на основе односвязного списка. Операции протолкнуть и вытолкнуть.
3. Односвязный список. Операции поиска элемента, добавления элемента в конец.

Вариант 4.

1. Очередь на основе односвязного списка. Операции помещения и удаления элемента.
2. Стек на основе двусвязного списка. Операции протолкнуть и вытолкнуть.
3. Односвязный список. Операции удаления элемента, добавления элемента в начало.