НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ

CSCB884 Проект по визуално програмиране

Изследване бързодействието на организация „едносвързан списък”, реализирана като наследник на ArrayList

Николай Атанасов Ванков F48897

4/26/2013

Задача

Реализация на линейна структура от данни тип „едносвързан списък“ и сравнение на бързодействието и с вградената generics колекция List<T>

Създаване на два линейни едносвързани списъка, които ще работят с целочислен тип. Първият линеен списък ще бъде реализиран като всеки елемент ще сочи към следващият, а вторият ще използва динамично оразмеряван масив. Ще бъдат сравнявани с .net generics типа List <T>. Тестовете ще се проведат като всяка една колекция се запълни с 1, 200, 500, 5000 и 10000 произволно (random) генерирани числа и се измери времето на добавяне и четене на данните.

Решение на проблема

За да тестваме задачата създаваме потребителски едносвързан линеен списък, в който всеки елемент ще знае за следващия. За целта се реализира класът `LinkedLinearList`, който ще съдържа в себе си *n* на брой елемента от тип `ListNode`. При създаване такъв списък се записва първият елемент и всеки следващ елемент е достъпен чрез свойството **Next** на предходния.

Класът LinkedLinearList притежава:

Променливи:

- `private ListNode currentNode` – текущият елемент в опашката
- `private ListNode headNode` – съдържа първият елемент от списъка

Свойства:

- `public ListNode CurrentNode` – капсулира текущият елемент от списъка в случай, че се наложи ползването му, извън класа
- `public ListNode HeadNode` – капсулира първият елемент от списъка

Методи:

- `public void Add(ListNode passedNode)` – добавя нов елемент
- `public void PrintNodes()` – печата всички елементи на конзолата
- `public void WriteNodes(int numberOfArguments)` – записва елементите в текстови файл
- `public void fillWithRandomData(int numberOfNodes, Random rand)` –запълва със случайно генерирани числа

Макар и линейният списък да е реализиран той още при първите операции ще изостане в сравнение с `List<T>`. Това се дължи на факта, че при всеки добавен елемент от нашия клас, ще се отделя памет и ще се създава нов елемент от тип `ListNode`, докато вграденият в C# тип работи чрез динамично оразмеряван масив. Всеки път, когато лимитът на този масив е запълнен се създава нов, двойно по – голям, и в него се копират старите елементи.

За да може тестовете да бъдат по – обективни създаваме клас, който да имплементира подобна на `List<T>` логика. Ще започва с определен размер и всеки път, когато се достигнат границите ще се създава нов масив, с два пъти повече елементи от предходния.

Тъй като при едната реализация ще губим време при създаване на нов елемент, то при другата ще елиминираме това, но с цената да копираме масив всеки път, щом се създаде нов.

Създава се класа `ArrayLinearList`, който ще притежава следните член-данни:

Свойства:

`public int Capacity` – максималният брой данни, които може да събере масива преди да се наложи да се оразмери

`public int Size` – броят записани полета в масива

Конструктори:

`public ArrayLinearList()` – създава нов лист с размер по подразбиране

`public ArrayLinearList(int size)` – създава нов лист с конкретен размер на масива

Методи:

`private void EnsureCapacity(int minimumCapacity)` - осигурява непрепълването на масива. В случай, че е достигнат максималният размер създава нов масив и прекопира данните в него.

`public void Add(int item)` - добавя нов елемент в списъка

`public void PrintNodes()` – печата данните на конзолата

`public void WriteNodes(int numberOfArguments)` – записва данните в текстов документ

`public void FillWithRandomData(int numberOfNodes, Random rand)` – запълва масива със случайно генерирани данни

Ще изследваме бързодействието на всеки един клас с 1, 200, 500, 5000 и 10000 числа от тип int. Ще създадем масив, който ще съдържа числа, генерирани по случаен принцип и тези числа ще ги запишем във всеки списък за да сме сигурни, че работим с едни и същи данни.

За да проверим времето на записване на данните създаваме класа `TestLists`, който съдържа предефинираните методи `FillWithTestData()` и `FillList()`, който го извиква и измерва времето от извикването до момента, в който отново се върне управлението в метода. За измерване на времето е използван класа `System.Diagnostics.Stopwatch()`, тъй като в нашият случай той ще даде по – точни резултати в сравнение с `DateTime`, особено при работата с малко аргументи.

В класа `CompareLists` се намира входната точка на нашата програма. При извикване тя зарежда масив с random генерирани целочислени числа, зарежда списъци с 1, 200, 500, 5000 и 10000 числа,

като извежда нужното време на конзолата и след това записва данните и проверява отнетото време.

Причината елементите да се запишат на текстов файл вместо да се изведат на екрана е бавната работа на конзолата при печатане на информация.

При стартиране на програмата имаме следните резултати:

	001 елемент	200 елемента	500 елемента	1000 елемента	5000 елемента	10000 елемента
List<T>	00:00:00.000 2945	00:00:00.000 0044	00:00:00.000 0166	00:00:00.000 0251	00:00:00.000 1215	00:00:00.000 3160
Linked Linear List	00:00:00.000 5348	00:00:00.000 1547	00:00:00.000 0235	00:00:00.000 0506	00:00:00.000 4510	00:00:00.000 5061
Array Linear List	00:00:00.000 3099	00:00:00.000 3833	00:00:00.000 0170	00:00:00.000 0312	00:00:00.000 1203	00:00:00.000 2889

Записване на данни в списъците

	001 елемент	200 елемента	500 елемента	1000 елемента	5000 елемента	10000 елемента
List<T>	00:00:00.000 0141	00:00:00.000 2082	00:00:00.000 5170	00:00:00.000 7561	00:00:00.004 9768	00:00:00.004 6871
Linked Linear List	00:00:00.000 2427	00:00:00.000 1369	00:00:00.000 2828	00:00:00.000 8578	00:00:00.002 0572	00:00:00.003 4427
Array Linear List	00:00:00.000 0060	00:00:00.000 1069	00:00:00.000 3428	00:00:00.000 6669	00:00:00.001 6366	00:00:00.003 3333

Записване на данните в текстов файл