https://github.com/IchPOch/Algoritm3Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский химико-технологический университет имени Д.И. Менделеева»

Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

Выполнил студент группы КС-36 (Епринцев Михаил Алексеевич)

Ссылка на репозиторий: (https://github.com/IchPOch/Algoritm3)

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Лобанов Алексей Владимирович

Крашенинников Роман Сергеевич

Дата сдачи: (17.03.23)

Оглавление

[Описание задачи. 2](#_Toc63548272)

[Описание метода/модели. 2](#_Toc63548273)

[Выполнение задачи. 2](#_Toc63548274)

[Заключение. 2](#_Toc63548275)

# Описание задачи.

В рамках лабораторной работы требуется реализовать двусвязный список. Также нужно:

* Использовать шаблонный подход, обеспечивая работу контейнера с произвольными данными.
* Реализовывать свой итератор предоставляющий стандартный для языка механизм работы с ним(для С++ это операции ++ и операция !=, для python это )
* Обеспечивать работу стандартных библиотек и конструкции for each если она есть в языке, если их нет, то реализовать собственную функцию использующую итератор.
* Проверку на пустоту и подсчет количества элементов.
* Операцию сортировки с использованием стандартной библиотеки.

Список должен реализовывать операции:

* добавления элемента после произвольного элемента
* удаление произвольного элемента из списка

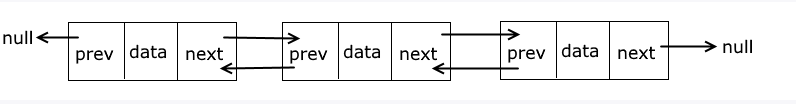
Для демонстрации работы структуры необходимо создать набор тестов(под тестом понимается функция, которая создаёт структуру, проводит операцию или операции над структурой и удаляет структуру):

* заполнение контейнера 1000 целыми числами в диапазоне от -1000 до 1000 и подсчет их суммы, среднего, минимального и максимального.
* Провести проверку работы операций вставки и изъятия элементов на коллекции из 10 строковых элементов.
* заполнение контейнера 100 структур содержащих фамилию, имя, отчество и дату рождения(от 01.01.1980 до 01.01.2020) значения каждого поля генерируются случайно из набора заранее заданных. После заполнение необходимо найти всех людей младше 20 лет и старше 30 и создать новые структуры содержащие результат фильтрации, проверить выполнение на правильность подсчётом кол-ва элементов не подходящих под условие в новых структурах.
* Заполнить структуру 1000 элементов и отсортировать ее, проверить правильность использую структуру из стандартной библиотеки и сравнив результат.
* Перемешать все элементы отсортированного списка в случайном порядке.

# Описание метода/модели.

Двусвязный список — связная структура данных в информатике, состоящая из набора последовательно связанных записей, называемых узлами. Каждый узел содержит два поля, называемых ссылками, которые указывают на предыдущий и последующий элементы в последовательности узлов. Ссылка на предыдущий элемент корневого узла и ссылка на последующий элемент последнего узла указывают на некого рода прерыватель, обычно сторожевой узел или null, для облегчения обхода списка. Если в списке только один сторожевой узел, тогда список циклически связан через него. Двусвязный список можно представить, как два связных списка, которые образованы из одних и тех же данных, но расположенных в противоположном порядке.

Две ссылки позволяют обходить список в обоих направлениях. Добавление и удаление узла в двусвязном списке требует изменения большего количества ссылок, чем аналогичные операции в связном списке. Однако данные операции проще и потенциально более эффективны (для некорневых узлов) - при обходе не нужно следить за предыдущим узлом или повторно обходить список в поиске предыдущего узла, плюс его ссылка может быть изменена.



# Выполнение задачи.

Реализовывал на языке C#,

Класс DoublyNode - содержит поля для хранения значения

Класс DoublyLinkedList - реализует сам двусвязный список.

Далее рассмотрим сам класс списка:

1. Add() - добавляет в конец списка новый элемент. Для этого создается узел, в который сразу через конструктор передается значение. Далее если в списке нет элементов, то head и tail ссылаются на один и тот же элемент, в ином случае во временную переменную node передается tail, tail становится равным новому узлу, предыдущий элемент tail’а начинает ссылаться на node и указатель на следующий элемент от node становится равным tail.

public void Add(T data)

{

DoublyNode<T> node = new DoublyNode<T>(data);

if (head == null)

head = node;

else

{

tail.Next = node;

node.Previous = tail;

}

tail = node;

count++;

}

1. AddFirst() - добавляет элемент в начало списка. То же самое что и в конец только в начало

public void AddFirst(T data)

{

DoublyNode<T> node = new DoublyNode<T>(data);

DoublyNode<T> temp = head;

node.Next = temp;

head = node;

if (count == 0)

tail = head;

else

temp.Previous = node;

count++;

}

1. RemoveAtt() - удаляет узел под определенным индексом. в цикле перебираем список и находим нужный нам узел. Удаляем его, перебивая указатели у соседних элементов друг на друга. Также мы должны учесть являлся ли удаленный элемент началом и/или хвостом списка. В таком случае мы задаем указателю на предыдущий/следующий элемент nullptr.

public bool RemoveAtt(int index)

{

var current = this.head;

for (int i = 0; i < index; i++)

{

current = current.Next;

}

if (current != null)

{

// если узел не последний

if (current.Next != null)

{

current.Next.Previous = current.Previous;

}

else

{

// если последний, переустанавливаем tail

tail = current.Previous;

}

// если узел не первый

if (current.Previous != null)

{

current.Previous.Next = current.Next;

}

else

{

// если первый, переустанавливаем head

head = current.Next;

}

count--;

return true;

}

return false;

}

1. Size() - возвращает размер списка.

public int Size()

{

return this.count;

}

1. IsEmpty() - возвращает true, если список пустой и false в ином случае.

public bool IsEmpty { get { return count == 0; } }

1. Sort() - сортирует список. Через методы

public void Sort()

{

quickSort(head, tail);

}

1. quicksort()- где идёт проверка на окончание сортировки и

private void quickSort(DoublyNode<T> lo, DoublyNode<T> hi)

{

if (lo != null && hi != null && lo != hi && lo != hi.Next)

{

DoublyNode<T> p = partition(lo, hi);

quickSort(lo, p.Previous);

quickSort(p.Next, hi);

}

}

1. partition() – где происходят сами замены

private DoublyNode<T> partition(DoublyNode<T> lo, DoublyNode<T> hi)

{

DoublyNode<T> pivot = hi;

DoublyNode<T> i = lo, j = lo;

while (j != hi)

{

if (j.Data.CompareTo(pivot.Data) < 0)

{

swap(j, i);

i = i.Next;

}

j = j.Next;

}

swap(i, pivot);

return i;

}

1. swap() – меняет местами элементы списка

private void swap(DoublyNode<T> a, DoublyNode<T> b)

{

T temp = a.Data;

a.Data = b.Data;

b.Data = temp;

}

1. Shuff() - перетасовывает список. Сначала записываем последний узел в переменную iNode. Запускаем цикл от size - 1 до 0. В цикле получаем случайное число j от 0 до i, в цикле находим узел под этим индексом и записываем его в переменную jNode и обмениваемся значениями между ними.

public void Shuff()

{

Random rnd = new Random();

var iNode = this.tail;

for(int i = this.count - 1; i>=0;i--)

{

int k = rnd.Next(0,i);

var jNode = this.head;

for(int j = 0; j < k; j++)

{

jNode = jNode.Next;

}

swap(iNode, jNode);

iNode = iNode.Previous;

}

}

1. BackEnumerator() – скролит все эллементы списка, сделано для foreach()

public IEnumerable<T> BackEnumerator()

{

DoublyNode<T> current = head;

while (current != null)

{

yield return current.Data;

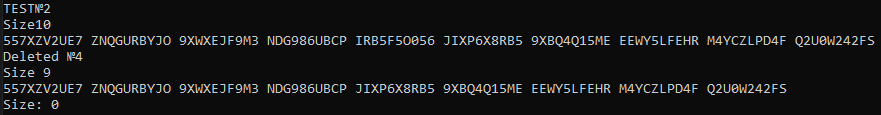
current = current.Next;

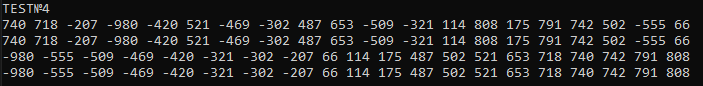
}

}

# Заключение.

В ходе данной лабораторной работы, я провел ряд тестов, в которых проверил работоспособность двусвязного списка. В первом тесте я заполнил структуру 1000 случайных целых чисел в диапазоне от -1000 до 1000. Подсчитал их сумму, среднее арифметическое значение, а также нашел минимальный и максимальный элемент. Вывод программы: 

Во втором тесте я проверил работу операций вставки и изъятия элементов на коллекции из 10 строковых элементов. Использовал функции Add(), RemoveAt(). Вывод программы:

В четвертом тесте я заполнил два списка (мой и из стандартной библиотеки) случайными значениями от -1000 до 1000. Списки до сортировки имеют одинаковые элементы на одних и тех же позициях. Вывод программы:

В пятом тесте я перемешал все элементы отсортированного списка в случайном порядке. Вывод программы:

Подведем итог, в ходе лабораторной работы я изучил и протестировал двусвязный список. Данная структура является одной из базовых. На ее основе могут быть построены другие структуры такие как стек, очередь, дека.