Московский авиационный институт   
(государственный технический университет)   
  
Факультет прикладной математики   
  
Кафедра вычислительной математики и программирования

Курсовой проект по курсам

“Языки и методы программирования” и “Алгоритмы и структуры данных”

8 факультет, 2 курс, весенний семестр 2020/21 учебного года

Студент: Стрыгин Д.Д.

Группа: М8О-106Б-19, №20

Преподаватель: Дубинин А.В.

Содержание:

* Введение
* Задание
* Связные списки
* Итераторы
* Описание программы
* Заключение
* Список источников

Введение

Составить и отладить программу на языке Си для обработки линейного списка заданной организации с отображением списка на динамические структуры (группы 1, 2, 3, 8) или на массив (только с индексным доступом, без применения ссылок и указателей, для групп 4, 5, 6, 7). Навигацию по списку следует реализовать с применением итераторов. Предусмотреть выполнение одного нестандартного и четырёх стандартных действий:

1. Печать списка.

2. Вставка нового элемента в список.

3. Удаление элемента из списка.

4. Подсчёт длины списка.

Задание

Тип списка: линейный однонаправленный.

Задание: удалить элементы списка со значениями, находящимися в заданном диапазоне.

Связные списки

Связанные списки являются второй по частоте использования структурой данных после массивов. Они являются достаточно простой реализацией динамических структур данных, использующие указатели (pointers) для реализации. Понимание работы указателей является необходимым условием для того, чтобы понять связанные списки. Кроме того — требуется понимание динамического выделения памяти и знать, что такое структуры и как ими пользоваться.

Связные списки имеют несколько основных преимуществ:

* элементы могут быть добавлены или удалены из середины списка
* нет необходимости объявления размера при инициализации

Но имеют и недостатки:

* связанные списки не имеют возможности случайного доступа к элементам — т.е. нет возможности получить элемент внутри списка, без того, чтобы пройтись по всем элементам до него
* для работы списков требуется динамическое выделение памяти и указатели, что усложняет код и может привести к утечкам памяти
* связанные списки требуют больше ресурсов операционной системы, т.к. их элементы выделяются динамически и каждый элемент должен хранить дополнительный указатель

**Применение**

Линейные списки находят широкое применение в приложениях, где непредсказуемы требования на размер памяти, необходимой для хранения данных; большое число сложных операций над данными, особенно включений и исключений. На базе линейных списков могут строиться стеки, очереди и деки. Представление очереди с помощью линейного списка позволяет достаточно просто обеспечить любые желаемые дисциплины обслуживания очереди. Особенно это удобно, когда число элементов в очереди трудно предсказуемо.

**Виды списков**

*Линейный однонаправленный список*

Это структура данных, состоящая из элементов одного типа, связанных между собой последовательно посредством указателей. Каждый элемент списка имеет указатель на следующий элемент. Последний элемент списка указывает на NULL. Элемент, на который нет указателя, является первым (головным) элементом списка. Здесь ссылка в каждом узле указывает на следующий узел в списке. В односвязном списке можно передвигаться только в сторону конца списка. Узнать адрес предыдущего элемента, опираясь на содержимое текущего узла, невозможно.

В информатике линейный список обычно определяется как абстрактный тип данных (АТД), формализующий понятие упорядоченной коллекции данных. На практике линейные списки обычно реализуются при помощи массивов и связных списков. Иногда термин «список» неформально используется также как синоним понятия «связный список». К примеру, АТД не типизированного изменяемого списка может быть определён как набор из конструктора и основных операций:

* Операция, проверяющая список на пустоту.
* Три операции добавления объекта в список (в начало, конец или внутрь после любого (n-го) элемента списка);
* Операция, вычисляющая первый (головной) элемент списка;
* Операция доступа к списку, состоящему из всех элементов исходного списка, кроме первого.

Характеристики:

* Длина списка. Количество элементов в списке.
* Списки могут быть типизированными или не типизированными. Если список типизирован, то тип его элементов задан, и все его элементы должны иметь типы, совместимые с заданным типом элементов списка. Чаще списки типизированы.
* Список может быть сортированным или несортированным.
* В зависимости от реализации может быть возможен произвольный доступ к элементам списка.

*Двусвязный список (двунаправленный связный список)*

Здесь ссылки в каждом узле указывают на предыдущий и на последующий узел в списке. Как и односвязный список, двусвязный допускает только последовательный доступ к элементам, но при этом даёт возможность перемещения в обе стороны. В этом списке проще производить удаление и перестановку элементов, так как легко доступны адреса тех элементов списка, указатели которых направлены на изменяемый элемент.

*Кольцевой связный список*

Разновидностью связных списков является кольцевой (циклический, замкнутый) список. Он тоже может быть односвязным или двусвязным. Последний элемент кольцевого списка содержит указатель на первый, а первый (в случае двусвязного списка) — на последний.

Как правило, такая структура реализуется на базе линейного списка. С каждым кольцевым списком дополнительно хранится указатель на первый элемент. В этом списке ссылки на NULL не встречается.

Также существуют циклические списки с выделенным головным элементом, облегчающие полный проход через список.

Итераторы

Итератор — интерфейс, предоставляющий доступ к элементам коллекции (массива или контейнера) и навигацию по ним. В различных системах итераторы могут иметь разные общепринятые названия. В терминах систем управления базами данных итераторы называются курсорами. В простейшем случае итератором в низкоуровневых языках является указатель.

Использование итераторов в обобщённом программировании позволяет реализовать универсальные алгоритмы работы с контейнерами.

Главное предназначение итераторов заключается в предоставлении возможности пользователю обращаться к любому элементу контейнера при сокрытии внутренней структуры контейнера от пользователя. Это позволяет контейнеру хранить элементы любым способом при допустимости работы пользователя с ним как с простой последовательностью или списком. Проектирование класса итератора обычно тесно связано с соответствующим классом контейнера. Обычно контейнер предоставляет методы создания итераторов.

Итератор похож на указатель своими основными операциями: он указывает на отдельный элемент коллекции объектов (предоставляет доступ к элементу) и содержит функции для перехода к другому элементу списка (следующему или предыдущему). Контейнер, который реализует поддержку итераторов, должен предоставлять первый элемент списка, а также возможность проверить, перебраны ли все элементы контейнера (является ли итератор конечным). В зависимости от используемого языка и цели, итераторы могут поддерживать дополнительные операции или определять различные варианты поведения.

Иногда счётчик цикла называют «итератором цикла». Тем не менее, счётчик цикла обеспечивает только перебор элементов, но не доступ к элементу. Отличия от индексации.

**Отличия от индексов**

В процедурных языках программирования широко используется индексация, основанная на счётчике цикла, для перебора всех элементов последовательности (например, массива). Хотя индексация может использоваться совместно с некоторыми объектно-ориентированными контейнерами, использование итераторов даёт свои преимущества:

* Индексация не подходит для некоторых структур данных, в частности, для структур данных с медленным произвольным доступом или вообще без поддержки такового (например, список или дерево).
* Итераторы предоставляют возможность последовательного перебора любых структур данных, поэтому делают код более читаемым, удобным для повторного использования и менее чувствительным к изменениям структур данных.
* Итераторы могут предоставлять дополнительные возможности при навигации по элементам. Например, проверку отсутствия пропусков элементов или защиту от повторного перебора одного и того же элемента.
* Некоторые контейнеры могут предоставлять возможность модифицировать свои объекты без влияния на сам итератор. Например, после того как итератор уже «прошёл» первый элемент, можно вставить дополнительные элементы в начало контейнера без каких-либо нежелательных последствий. При использовании индексации это проблематично из-за смены номеров индексов.

Возможность модификации контейнера во время итерации его элементов стала необходимой в современном объектно-ориентированном программировании, где взаимосвязи между объектами и последствия выполнения операций могут быть не слишком очевидными. Использование итератора избавляет от этих видов проблем.

Описание программы

Программа общается с пользователем при помощи строковых команд и выполняет следующие функции со сложностью:

1. – добавление элемента **b** на место **a**. Сложность – O(1).

2 – удаляет элемент на месте **a**. Сложность – O(1).

3 – удаляет элементы в заданном числовом диапазоне. Сложность – O(n).

Заключение

В курсовом проекте были рассмотрены связные списки их виды и области, где они применяются. В ходе подготовки проекта были изучены методы работы со списками при помощи итераторов. В процессе выполнения проекта была написана программа, в которой реализованы основные функции работы со списками, а также функция, которая удаляет элементы в заданном диапозоне.

Список источников:

* <https://ru.wikipedia.org/wiki/Итератор> Итератор
* <https://rtfm.co.ua/c-svyazannye-spiski/#Удаление_определённого_элемента> Связный список
* <https://ru.wikipedia.org/wiki/Связный_список> Связный список (виды)
* <https://studopedia.su/9_72555_primenenie-lineynih-spiskov.html> Связный список
* методичка