Упражнение по работе со стеком: включение, исключение элемента

Необходимо:

1. Разобраться в представленных частях программы.
2. Вставить требуемые операторы в указанные места программы.
3. Проверить правильность ее работы.

Программа состоит из шести модулей, одного заголовочного файла и файла данных

l_main.c — главная программа;
l_input.c — программа ввода информации из файла;
l_print.c — программа печати стека;
l_push.c — программа добавления элемента в стек;
l_pop.c — программа удаления элемента из стека;
l_rprint.c — программа печати стека (рекурсивная);
l_head.h — заголовочный файл;
key.txt — файл данных.

Задание разбито на три подзадачи:

1. Вставка необходимых операторов в программу **l_push.c** (каталог 1)
2. Вставка необходимых операторов в программу **l_pop.c** (каталог 2)
3. Вставка необходимых операторов в программу **l_rprint.c** (каталог 3)

Каждый каталог содержит **makefile**, тексты программ (*.c) и объектные файлы (*.o). **makefile** необходим для получения исполняемого файла и проверки работы программы.

Результат работы необходимо собрать в каталоге 4, т.е. каталог должен содержать все шесть исходных модулей, заголовочный файл и файл данных.

Теория

ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ

Отметим, что как число объектов в структурах данных, так и связи между этими объектами могут изменяться со временем.

Например, строяться новые населенные пункты или прокладываются новые дороги; в списки добавляются или из списков исключаются некоторые члены. Иногда (и достаточно часто) такие изменения происходят во время работы программы. Такие "нестационарные" структуры данных выделены в отдельный класс.

О п р е д е л е н и е: Динамическими структурами данных будем называть структуры, которые могут создаваться, меняться и уничтожаться в процессе выполнения программы.

Очевидно, такие структуры наиболее часто встречаются при работе программ в реальном времени.

Из всего многообразия возможных структур данных, возникающих в основном из-за различной сложности связей между объектами, мы рассмотрим более или менее подробно лишь два наиболее часто встречающихся в программировании вида: линейные списки (и их частные случаи — стек и очередь) и деревья (как характерный пример, нелинейных структур).

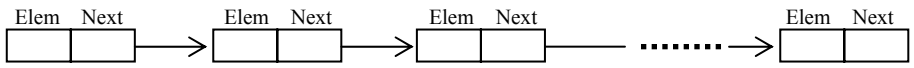
ЛИНЕЙНЫЕ СПИСКИ

Определения и основные операции

О п р е д е л е н и е: Линейный список — множество, состоящее из n элементов (n >= 0), со следующими свойствами:

1) существует первый элемент (при n > 0);
2) для любых не крайних элементов существует предыдущий и последующий;
3) существует последний элемент (при n > 0).

Иными словами, объекты линейного списка связаны лишь с двумя соседними. Данная структура поэтому хорошо представляется спрямленной последовательностью связей (отсюда и ее название):



Примеры групп объектов, подпадающих под данное определение, многочислены и многообразны, в частности: стопка любых предметов (например, подносов), книги, тетради (как список листов) и т.д.

Подробнее рассмотрим очередь к врачу. Очевидно, что здесь имеем множество пациентов, причем множество может быть и пустым. Также ясно, что есть первый и последний элемент, и для любых не крайних существует предыдущий и последующий.

Операции, которые чаще всего производят над линейными списками, являются:

1) просмотр списка и поиск элемента;
2) включение нового элемента;
3) исключение элемента;
4) объединение двух списков в один;
5) копирование списка и другие операции.

В нашем примере очереди к врачу эти операции означают:

1) Кто-то хочет узнать, как далеко он находится от начала и опрашивает людей (с начала или от себя к началу). Фактически просматривая очередь.
2) Пришел еще один больной (и встал в конец очереди или, если "по номеркам", то в середину!).
3) Кто-то отчаялся ждать и ушел домой (или пошел к врачу).
4) Было два врача и две очереди, но один врач ушел ...
5) Оставшийся после окончания приема врача народ скопировал свой порядок в список, чтобы в следующий раз восстановить порядок или медицинская сестра всех переписала перед началом приема врача в порядке очереди.

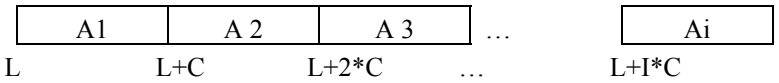
Возратимся к программированию. Отметим, что, по-видимому невозможно найти представление (структурированных) данных, при которых все возможные операции выполнялись бы одинаково эффективно. Однако, к счастью, в редких случаях требуется выполнение всех операций, чаще необходимо обеспечить эффективность работы лишь нескольких операций. В зависимости от этого выбирают как способ представления структуры данных (в частности, линейного списка), так и конкретный подвид структур (например, для линейных списков — стеки, очереди, деки и т.п.).

Способы представления списков

Существует два основных способа представления списков: последовательный и связный. При последовательном — элементы списка располагаются в памяти ЭВМ последовательно друг за другом; при связном — информация группируется в пары (например, записи), где один компонент — очередной элемент списка, а второй — ссылка (адрес) на пару со следующим (или предыдущим) элементом. Т.е. расположение в памяти ЭВМ элементов линейного списка может быть представленно следующим образом:

а) при последовательном представлении (таблично и рисунком):

Номер лемента	Адрес в памяти машины	Информация
1	L	A1
2	L+C	A2
3	L+2*C	A3
4	…	…
5	L+I*C	A+I*C
6	…	…



A1, A2, A3, ... — элементы списка; L — адрес первого эелемента в памяти машины; C — объем памяти, отводимый для размещения одного элемента.

б) при связном представлении (таблично и рисунком):

Номер лемента	Адрес в памяти машины	Информация
1	L1	A1, L2
2	L2	A2, L3
3	L3	A3, L4
4	…	…
5	Li	Ai, L_{i+1}
6	…	…

где L1, L2, L3, ... — адреса памяти в ЭВМ соответствующих пар, в которых находится информация об элементе и адресе следующего элемента. При этом адреса L1, L2, L3, ... могут быть не упорядочены.

В подробно рассматриваемом нами примере — очереди к врачу, последовательное представление означает, что люди в очереди сидят на линейно расположенных стульях в правильном порядке (причем подряд!). При связном представлении этого нет, т.е. народ сидит как попало, но у каждого имеется ссылка на предыдущуего (иначе анархия, приводящая обычно к скандалу); эта ссылка в данном случае существует в памяти человека в виде образа впереди стоящего пациента.

Отметим, что в поликлиниках, если очередь длинная, то стихийно (но далеко не случайно!) реализуется

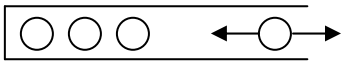
связное представление, а, например, в магазине — обычно последовательное. Связано это скорее всего с тем, что перемещаться стоя легко, а сидя сложно. Также и в программировании при некоторых условиях оказывается предпочтительнее один вид представления, а при других — другой.

Частные случаи линейных списков: стек, очередь, дек

В программировании наиболее часто используются несколько частных случаев линейных списков, когда дополнительно к свойствам определения на структуры данных накладываются новые ограничения.

Определение: Стеком будем называть линейный список, в котором все включения и исключения элементов разрешено производить только на одном конце (считается, что включение и исключения самые частые (основные) операции!).

Примеры — магазин автомата с патронами, железнодорожный состав вагонов в тупике и т.д.

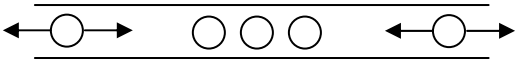


Определение: Очередь — линейный список, в котором все включения производятся на одном конце, а исключения — на другом.

Примером является любая очередь, если не разрешено новые элементы вставлять и исключать из середины.

Определение: Дек — линейный список, где исключения и включения производятся с обоих концов.

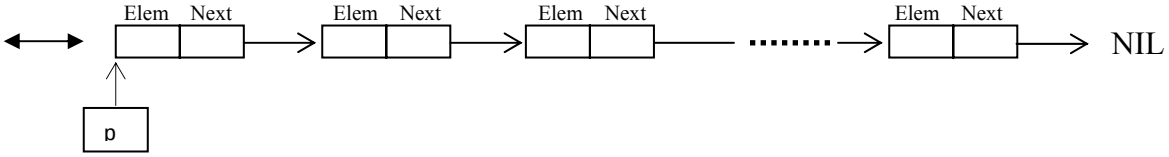
Пример простой и не искусственный привести трудно, некоторую аналогию имеет вход и выход из вагона электрички.



Отметим, что над стеком, очередью и деком могут производиться разнообразные операции, включая перечисленные выше для линейных списков. Однако, из данных определений ясно, что данные частные случаи линейных списков ориентированны на ситуации, когда в основном производятся операции включения и исключения элементов. Специфичность этих операций (место выполнения) делает более удобным связное представление, хотя последовательное представление не исключено.

Мы рекомендуем использовать следующий порядок связи элементов.

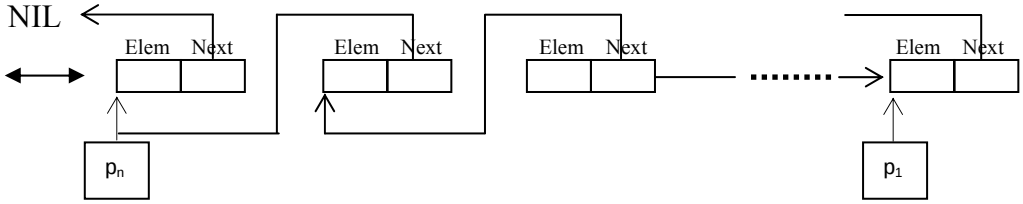
Для стека:



где p — указатель, в которой хранится адрес верхнего элемента стека. Включение и исключение элементов производится со стороны верхнего элемента.

Для очереди

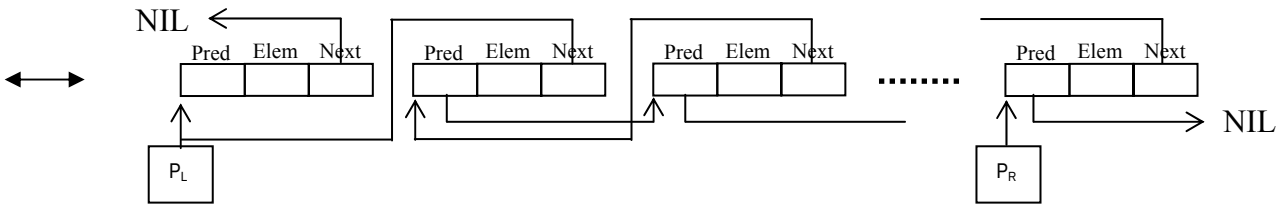
Отметим обратное направление связи элементов в очереди по сравнению со стеком, что определяется изменением направления "стока" элементов.:



где в переменных ссылочного типа p1 и pn будем хранить адреса соответственно первого и последнего элемента очереди.

Включение производится со стороны последнего элемента, а исключение со стороны первого.

Для дека (для любознательных):



где P_L и P_r — ссылки на левый и правый конец дека, и т.к. исключение и включение элементов возможно с обоих концов, то приходится некоторым образом совместить стек и очередь, создавая так называемый 'двухсвязный' линейный список.

Теперь о том, где и как используются рассмотренные выше частные случаи линейных списков. Стеки и очереди применяются в тех случаях, когда происходит выборка объектов из некоторого их множества для дальнейшей их обработки. Например, из огромных звездных каталогов обычно выбирается некоторая группа звезд по каким-то признакам (например, один спектральный класс, расположение в заданной площадке неба и т.п.) для дальнейшей их статистической или другой обработки. При этом часто неважно, что использовать стек или очередь (хотя и принято строить стек).

Иная ситуация, когда программа работает в режиме реального времени. Часто здесь возникает необходимость образовывать из объектов линейный список, причем так, что прибывший первым элемент уходил бы (обрабатывался) тоже первым. Этот принцип реализуется в очереди (в стеке правило — первым пришел и последним ушел).

Т.о. при работе с данными большого объема, а также в трансляторах, текстовых редакторах и во многих других программных средствах используются стеки.

Для функционирующей в реальном времени системной программы, управляющей процессом обработки программ пользователей ЭВМ (и называемой операционной системой), создаются очереди: очередь заданий для выполнения их центральным процессором ЭВМ, выходные очереди данных и т.п.

Дек используется значительно реже, чем очередь и стеки.