Задание. Реализовать линейный список и основные операции, используя связное распределение (с помощью массивов).

Линейные списки. Связное распределение

При связанном распределении для хранения множества не требуется выделения непрерывной области памяти (в отличие от последовательного распределения).

При таком распределении осуществляется последовательный доступ (sequential access) к элементам списка.

Реализация линейных списков посредством связных списков позволяет эффективно организовать операции вставки и удаления, пожертвовав для этого удобства дополнительной памятью для хранения указателей.

При связном распределении элементы множества $X = \{x_1, \dots, x_n\}$ могут располагаться в произвольных областях памяти. Чтобы сохранить информацию о порядке следования элементов, необходимо каждому элементу x_i поставить в соответствие указатель (ссылку) на следующий элемент x_{i+1}.

Значение указателя можно трактовать как адрес области памяти, где находится соответствующий элемент. Такое представление множества называется **связанным (связным) списком.**

Каждый элемент списка, называемый узлом, состоит из двух полей: в поле info размещается сам элемент, а в поле связи next – указатель на следующий за ним элемент. Доступ к узлу возможен только в том случае, если на него указывает хотя бы один указатель.

Реализация связных списков с помощью (посредством) массивов

Рассмотрим реализацию списка в виде двух массивов A и B.

Сами элементы списка будут храниться в массиве A. Пусть i – индекс элемента в списке, тогда $A[i]$ – сам элемент; $B[i]$ – индекс, по которому в массиве A располагается следующий за $A[i]$ элемент списка. Если k – индекс последнего элемента в списке, то $B[k]$=0.

Кроме того, существуют две переменные: ns – индекс начала занятых компонентов в массиве A и nf – индекс начала свободных компонентов массива A.

Тогда список $2, 12, 17$ может быть реализован следующим образом: $ns = 1$, $nf = 4$.

i	1	2	3	4	5	6
$A[i]$	2	17	12	-	-	-
$B[i]$	3	0	2	5	6	0

Пример. Представление с помощью массивов двух связных списков $X = \{a, b, c\}$ и $Y = \{d, e, g\}$

			Y		5 Free			X 8		
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
$info$		**c**	**d**	**b**		**g**		**a**		**e**
$next$	**9**	**0**	**10**	**2**	**1**	**0**	**0**	**4**	**7**	**6**

В массиве **info** хранятся элементы множества, а в массиве next – указатели, т.е. индексы позиций в массивах, где расположены последующие элементы.

Поскольку при таком представлении нет встроенных механизмов выделения и освобождения памяти, они должны быть предусмотрены при реализации.

Чтобы вести учет использования позиций массивов, все свободные позиции объединены в отдельный связный список свободных позиций с указателем free, который представляет собой индекс первой свободной позиции в списке.

Данный список может обеспечить обслуживание нескольких связанных списков, имеющих одинаковый тип элементов.

Процедура выделения памяти $new(p)$ для обеспечения включения нового элемента в список сначала должна проверить наличие свободных позиций (если $free = 0$, то свободных позиций нет).

Затем, если имеются свободные позиции ($free \neq 0$), определить индекс первой из них (установив значение указателя $p = free$) и исключить эту позицию из списка свободных позиций (присвоив указателю $free$ значение $next(free)$).

Например, пусть требуется включить элемент f в список Y после элемента e, хранящегося в $info(10)$.

Тогда процедура $new(p)$ установит указатель $p = 5$, и, установив указатель $free = next(free) = 1$, исключит позицию 5 из списка свободных позиций.

Операция включения запишет элемент f в $info(5)$ и установит $next(5) = next(10) = 6$ и $next(10) = 5$.

	Free 1		Y 3					X 8		
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
$info$		**c**	**d**	**b**	**f**	**g**		**a**		**e**
$next$	**9**	**0**	**10**	**2**	**6**	**0**	**0**	**4**	**7**	**5**

Процедура освобождения памяти $dispose(p)$ позицию p, которая стала свободной в результате исключения элемента из списка, включает в начало списка свободных позиций, записав значение указателя $free$ в позицию p массива $next$ и установив новое значение указателя $free = p$.

В качестве примера рассмотрим исключение элемента e из списка $Y = \{d, e, f, g\}$.

Исключаемый элемент **e** хранится в $info(10)$, а предшествующий ему элемент d – в $info(3)$. Операция исключения устанавливает значение $next(3) = next(10) = 5$ и передает процедуре $dispose(p)$ значение $p = 10$ для освобождения памяти, которая включает эту позицию в начало списка свободных позиций, установив значения $next(10) = free = 1$ и $free = 10$. Хотя исключенный элемент $\boldsymbol{e}$ по-прежнему присутствует в $info(10)$, в списке Y его уже нет, т.к. позиция 10 находится в списке свободных позиций.

			Y 3					X 8		Free 10
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
$info$		**c**	**d**	**b**	**f**	**g**		**a**		**e**
$next$	**9**	**0**	**5**	**2**	**6**	**0**	**0**	**4**	**7**	**1**

Т.о., список свободных позиций имеет одну точку доступа: новый элемент всегда добавляется в начало, исключается всегда первый элемент, т.е. список ведет себя как стек.

Вместо нескольких массивов, когда каждому полю узла связанного списка соответствует свой массив, можно использовать один массив, элементами которого являются объекты комбинированного типа (записи). Каждая запись представляет собой узел списка и состоит из тех же полей.

Т.о., операция включения и исключения выполняются за некоторое фиксированное время, не зависящее от размеров множеств.

Так же легко за фиксированное время выполняются операции конкатенации (сцепления) и разбиения списков. Для сцепления двух списков достаточно установить значение поля next последнего элемента первого списка (при этом должен быть обеспечен доступ к последнему элементу, например, с помощью специального указателя), равным значению указателя на первый элемент второго.

Разбиение на два списка можно выполнить, если обеспечен доступ к узлу, непосредственно предшествующему месту разбиения.

Основным достоинством связанного распределения является их удобство для реализации динамических множеств, поскольку большинство операций, ориентированных на модификации множеств, выполняются за время, не зависящее от их размеров.

Существенным недостатком связанного распределения является невозможность прямого доступа к элементу множества (за исключением первого), т.е. возможен только последовательный доступ.

Например, если необходимо получить доступ к элементу x_i, следует последовательно, начиная с первого элемента, перемещаться по $i-1$ узлам списка, пока не будет достигнут узел с элементом x_i. Это требует времени в среднем $O(n)$.

Другой недостаток связан с тем, что необходима дополнительная память для хранения указателей.