Тема 2. СТРУКТУРЫ ДАННЫХ

2.3. Стеки

Стек представляет собой динамическую последовательность элементов с одной точкой доступа, называемой *вершиной* стека. Все элементы последовательности, кроме элемента, расположенного в вершине стека, недоступны. Новый элемент добавляется только в вершину стека, сдвигая остальные элементы последовательности. Исключить можно только элемент из вершины стека, остальные элементы последовательности при этом сдвигаются в сторону вершины стека. Таким образом, стек работает по принципу «последним пришел – первым ушел» и часто называется структурой LIFO (Last In – First Out).

Согласно этому определению зафиксирована вершина стека, а перемещаются его элементы. Очевидно, что прямая реализация такого стека неэффективна, так как время выполнения операций включения и исключения будет пропорционально числу элементов в стеке. Поэтому при реализации стека избегают перемещения его элементов, а делают передвигаемой вершину стека. Для этого используется специальный *указатель вершины стека*, который указывает текущее положение вершины стека.

Операцию включения элемента в стек будем записывать в виде $S \Leftarrow x$ (элемент x поместить в вершину стека S), а операцию исключения из стека – в виде $x \Leftarrow S$ (исключить элемент из вершины стека S и присвоить его значение переменной x). Использование данных обозначений позволяет рассматривать логику работы алгоритма, не обращая внимания на детали реализации стека. В литературе эти операции традиционно называют *PUSH* (включение) и *POP* (исключение). При этом операцию включения элемента x в стек S обычно реализуют в виде процедуры $PUSH(x, S)$, а исключение – в виде функции $POP(S)$, которая удаляет элемент из вершины стека S и возвращает его в качестве своего значения (т. е. операции $x \Leftarrow S$ соответствует присваивание $x \leftarrow POP(S)$).

Одним из наиболее эффективных способов реализации стека является использование последовательного распределения (массива). Массив служит для хранения элементов стека. В качестве указателя вершины стека используется переменная t, являющаяся индексом последнего включенного в стек элемента. Стек состоит из элементов $S_1, S_2, \ldots, S_t$, где S_1 – нижний элемент (дно) стека, а S_t – верхний элемент (вершина) стека. Емкость (глубина) стека определяется размером массива. Пусть для реализации стека выделен массив из m компонентов, тогда всегда должно выполняться условие $t \le m$, в противном случае при включении элемента происходит переполнение, что означает ошибку. Пустой стек соответствует случаю $t = 0$ и, как правило, означает завершение работы алгоритма в целом или некоторого его фрагмента. Реализация операций включения и исключения представлена в алгоритме 2.3.

$S \Leftarrow x$	$x \Leftarrow S$
$t \leftarrow t+1$ **if** $t > m$ **then** // переполнение **else** $S_t \leftarrow x$	**if** $t = 0$ **then** // стек пуст **else** $\begin{cases} x \leftarrow S_t \\ t \leftarrow t-1 \end{cases}$

Алгоритм 2.3. Операции включения и исключения для стека на базе массива

На рис. 2.10 показан пример реализации стека на базе массива (емкость стека равна 8) и процесс изменения его состояния во времени. Пусть стек находится в некотором состоянии *а*, когда он содержит 4 элемента, вершина стека – элемент 9 (рис. 2.10, *а*). После выполнения операций включения $S \Leftarrow 3$ и $S \Leftarrow 8$ стек переходит в состояние *б*, вершиной стека становится элемент 8 (рис. 2.10, *б*). После исключения элемента 8 из вершины стека он переходит из состояния *б* в состояние *в*, вершиной стека становится элемент 3 (рис. 2.10, *в*). Хотя после исключения элемента 8 он по-прежнему присутствует в массиве, в стеке его уже нет, так как вершиной стека является элемент 3.

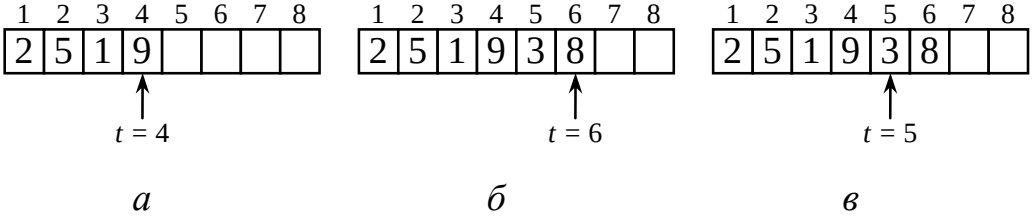

Рис. 2.10. Реализация стека на базе массива:
а – стек содержит 4 элемента, вершина стека – элемент 9;
б – стек после выполнения операций $S \Leftarrow 3$ и $S \Leftarrow 8$;
в – стек после исключения элемента 8 из вершины

Недостатком последовательной реализации стека является фиксированная емкость стека, что может привести к его переполнению. Этот недостаток является следствием противоречия между массивом как статической структурой и стеком как динамической структурой. От указанного недостатка свободна реализация стека с использованием связного распределения (динамических списковых структур).

Стек легко реализуется на базе односвязного списка, узел которого состоит из поля *info*, содержащего элемент последовательности, и поля связи *next* для ссылки на элемент, находящийся в стеке под данным элементом. Поле *next* нижнего элемента стека имеет пустое значение Λ указателя. Функции указателя t вершины стека выполняет указатель *list* связного списка. Стек пуст, если $t = \Lambda$. Очевидно, что операции $S \Leftarrow x$ соответствует процедура *INS_FIRST* включения элемента в начало списка, а операции $x \Leftarrow S$ – процедура *DEL_FIRST* исключения первого элемента списка, модифицированная так, чтобы переменной x присваивалось значение исключаемого элемента. Реализация этих операций представлена в алгоритме 2.4.

$S \Leftarrow x$	$x \Leftarrow S$
$new(l)$ $l.info \leftarrow x$ $l.next \leftarrow t$ $t \leftarrow l$	**if** $t = \Lambda$ **then** // стек пуст **else** $\begin{cases} x \leftarrow t.info \\ l \leftarrow t \\ t \leftarrow t.next \\ dispose(l) \end{cases}$

Алгоритм 2.4. Операции включения и исключения для стека на базе списка

Для многих алгоритмов проверку пустоты стека удобно выделить в отдельную операцию (будем записывать в виде $S = \varnothing$). В этом случае из операции $x \Leftarrow S$ исключается проверка пустоты стека и считается, что она определена только для непустого стека. Операцию, которая устанавливает стек S в начальное состояние (т. е. делает его пустым), будем записывать в виде $S \leftarrow \varnothing$. В ряде случаев множество операций, выполняемых над стеком, дополняется операцией определения значения элемента в вершине стека без его исключения. Иногда может быть полезна операция, которая просто исключает элемент из стека, не возвращая его значение. Эти операции являются простыми модификациями операции исключения и реализуются очевидным образом. Все рассмотренные операции со стеком выполняются за время $O(1)$.