**4.3. Комбинированные структуры данных: массивы и списки списков**

Более сложным случаем является использование массива списков или списка списков. Здесь каждый элемент массива или основного списка является началом вспомогательного списка, причем эти вспомогательные списки могут содержать разное число элементов (но, конечно, одного типа).

|  |
| --- |
| **nil** |
| адрес |

|  |
| --- |
| next |
| адрес |

|  |
| --- |
| next |
| **nil** |

|  |  |  |  |
| --- | --- | --- | --- |
| адрес  вспом.  списка | адрес  вспом.  списка | . . . . . | адрес  вспом.  списка |

|  |
| --- |
| Эл. 1 |
| next |

|  |
| --- |
| Эл. 1 |
| next |

|  |
| --- |
| Эл. 1 |
| next |

|  |
| --- |
| Эл. 1 |
| **nil** |

|  |
| --- |
| Эл. 1 |
| next |

|  |
| --- |
| Эл. 2 |
| **nil** |

|  |
| --- |
| Эл. 2 |
| next |

|  |
| --- |
| Эл. 2 |
| **nil** |

|  |
| --- |
| Эл. 2 |
| next |

|  |
| --- |
| Эл. 3 |
| **nil** |

|  |
| --- |
| Эл. 3 |
| **nil** |

В обоих случаях в первую очередь вводится тип данных, описывающий структуру элементов вспомогательного списка:

#### **Type** pSubList = ^ TSubList;

TSubList = **record**

<описание информационных полей>;

next : pSubList;

**end**;

После этого описывается либо основной массив указателей, либо структура элементов основного списка:

#### **Type** TMainArray = **array** [ 1 . . N ] **of** pSubList;

pMainList = ^ TMainList;

TMainList = **record**

NextMain : pMainList;

FirstSub : pSubList;

**end**;

Обработка таких структур включает больше операций, поскольку практически любая типовая операция (поиск, просмотр, добавление, удаление) может выполняться как с основным массивом или списком, так и с любым вспомогательным списком. Например, полный поиск или полный проход реализуется двойным циклом: внешний цикл проходит по элементам основного списка, а внутренний обрабатывает отдельно каждый вспомогательный список. Для этого необходимы две ссылочные переменные: pCurrentMain для прохода по основному списку и pCurrentSub – для прохода по вспомогательному списку.

pCurrentMain := “адрес первого элемента основного списка”;

**while** pCurrentMain <> **nil do**

**begin**

pCurrentSub := pCurrentMain^.FirstSub;

**while** pCurrentSub <> **nil do** pCurrentSub := pCurrentSub^.next;

**end**;

pCurrentMain := pCurrentMain^.NextMain;

Добавление и удаление элементов во вспомогательных списках выполняется обычным образом. Небольшие особенности имеет удаление элемента из основного списка, поскольку в этом случае как правило надо удалить и весь вспомогательный список. Поэтому прежде всего организуется проход по вспомогательному списку с удалением каждого элемента, а потом уже производится удаление элемента основного списка.

Иногда удобно в элементах основного списка хранить не только адрес первого элемента вспомогательного списка, но и адрес последнего элемента. Естественно, что при необходимости как основной, так и вспомогательные списки могут быть двунаправленными.

Кроме того, поскольку стеки и очереди можно рассматривать как частные случаи списков, легко можно реализовать такие структуры как массив или список стеков, массив или список очередей и.т.д.