Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Чувашский государственный университет имени И.Н. Ульянова»

ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ

Назарова Ольга Васильевна

Понятие динамических и статических структур данных

Используемые в программировании данные можно разделить на две большие группы: данные статической структуры и данные динамической

структуры.

Данные статической структуры – это данные, взаиморасположение и взаимосвязи элементов которых всегда остаются постоянными. Количество элементов –величина постоянная.

Данные динамической структуры –это данные, внутреннее строение которых формируется по какому

-либо закону, но количество элементов, их взаиморасположение и взаимосвязи могут динамически изменяться во время выполнения программы согласно закону формирования.

Динамическая структура построена как цепочка взаимосвязанных элементов

одного типа, обычно называемых узлами, каждый из которых включает:

– **информационные поля (поле)** – где содержатся необходимые данные. Информационное поле само может являться интегрированной структурой – массивом, другой динамической структурой и т.п.;

– **адресные поля (поле)** – содержат один или несколько указателей, связывающий данный элемент с другими элементами структуры.

Количество информационных и адресных полей определяется классом структуры.

К динамическим структурам относят:

- однонаправленные (односвязные) списки;
- двунаправленные (двусвязные) списки;
- кольцевые (циклические) списки;
- стеки;
- деки;
- очереди;
- бинарные деревья и др.

Они отличаются способом связи отдельных элементов и/или допустимыми операциями.

Однонаправленный список – линейная структура, где каждый элемент (кроме последнего) имеет указатель на следующий элемент. У последнего элемента указатель равен NULL.

Двунаправленный список – линейная структура, где каждый элемент (кроме первого и последнего) имеет указатели на следующий элемент и на предыдущий элемент. У первого элемента указатель на предыдущий элемент равен NULL, а у последнего – указатель на следующий равен NULL.

Кольцевые списки – циклическая структура. Они могут быть однонаправленными и двунаправленными. Особенностью является то, что ссылка последнего элемента указывает на первый элемент, а в двунаправленном списке у первого элемента дополнительно присутствует ссылка на последний элемент списка.

В **однонаправленных**, **двунаправленных** и **циклических списках** добавление, включение и исключение элементов производится в любом месте.

Очередь – линейный список, в котором все включения элементов производятся на одном конце списка, а все исключения (и обычно всякий доступ) – на другом по принципу FIFO (First In – First Out, «первым пришёл – первым вышел»).

Стек – линейная структура данных, в которой добавление и удаление элементов осуществляется в одной стороны списка (с конца) по принципу LIFO (Last In – First Out, «последним пришёл – первым вышел»).

Дек –линейная структура данных, в которой можно добавлять и удалять элементы с обоих концов списка.

Бинарные деревья – древовидная структура данных, в которой каждый элемент (родительский узел) имеет ссылки на два других элемента (дочерних), называемых левым и правым наследниками (потомками).

Средством доступа к динамическим структурам и их элементам является указатель (адрес) на место их текущего расположения в памяти. Как правило, это указатель на первый элемент структуры.

Достоинства связного представления данных:
– размер структуры ограничивается только доступным объемом машинной памяти;
– при изменении логической последовательности элементов структуры требуется не перемещение данных в памяти, а только коррекция указателей;
– большая гибкость структуры, возможность обеспечить ее изменчивость.

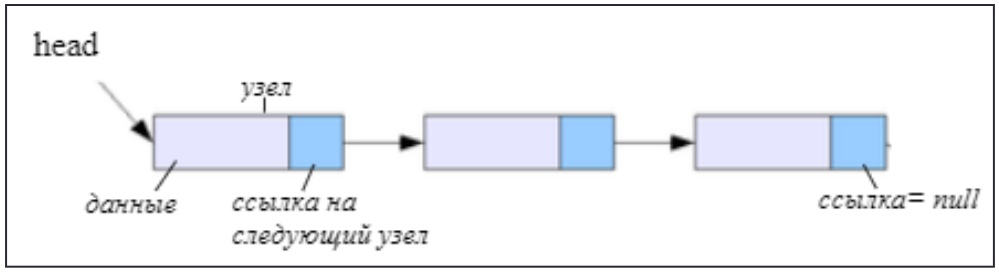
Основные недостатки:
– на поля связок расходуется дополнительная память;
– доступ к элементам связной структуры может быть менее эффективным по времени.

В языке Java существует достаточное количество стандартных классов, реализующих динамическое взаимодействие элементов. Это, прежде всего, классы коллекций. Среди них списки, например ArrayList, LinkedList, очереди ArrayDeque и много других.

Формирование и вывод односвязного списка

Линейный однонаправленный список (его еще называют односвязным списком) – это простейшая динамическая структура данных.

Ее схема представлена на рис. 1.



Существует два основных способа построения односвязного списка.

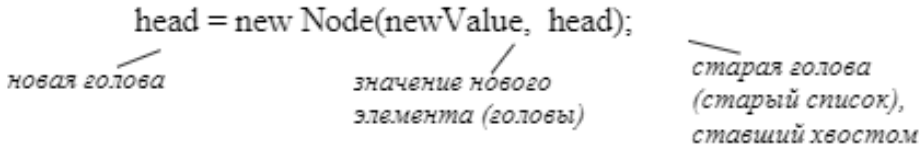
Первый способ – изначально создается «голова», а новые элементы помещаются в конец списка, образу

я хвост.

Второй способ – изначально создается «хвост» и «голова» меняет свое положение при наращивании списка. В нижеприведенном примере показаны оба варианта.

Изменение линейного односвязного списка

Добавление элемента в начало списка (с головы). Данная задача решается в одно действие: создается новый узел, который становится новой «головой» списка, а старая ссылка на «голову» становится ссылкой на следующий элемент.



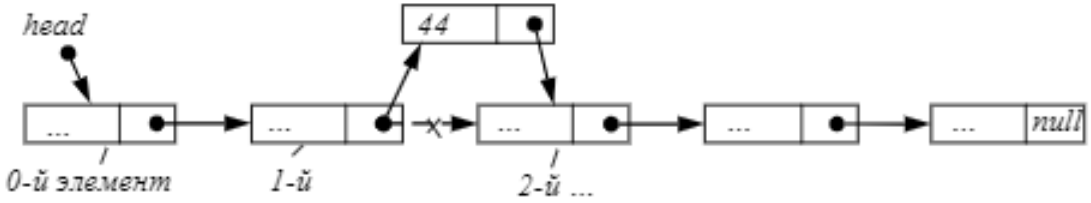
Добавление элемента в конец списка. Для того чтобы добраться до последнего элемента, необходимо пройтись по всему списку с «головы», пока не будет получен элемент, у которого ссылка next на следующий элемент равна null. Далее добавляется новый элемент.

Ниже представлен фрагмент программы:

```
// создается новый элемент со значением 123 – будущий хвост
    Node newtail=new Node(123, null);

// для перемещения по списку используется вспомогательная переменная ref,
// которой в качестве начальной ссылки передается указатель на «голову»
    ref = head;
    while (ref.next != null) {  // пока не последний элемент
      ref = ref.next;
    }
// указателю последнего элемента присваиваем новый «хвост» (элемент)
   ref.next=newtail;
```

Добавление элемента в список в указанное место (вставка элемента). Рассмотрим пример вставки второго элемента со значением 44. Схематично вставка элемента показана на рис. 2. Изначально полю next нового элемента присваивается ссылка на узел, который был ранее под номером 2, а потом полю next первого элемента присваивается ссылка на новый узел (именно в такой последовательности, иначе список будет разорван).



Фрагмент кода вставки элемента в список.

```
// создается новый элемент со значением 44 – для вставки
 Node newNode=new Node(44, null);
  ref = head;   // используем временную переменную
  int k=1;      // счетчик элементов
// поиск нужного положения узла для вставки
  while (ref.next!= null && (k<2 )) {
    ref = ref.next;
    k++;
  }
// переброска ссылок при вставке элемента
  newNode.next=ref.next.next;
  ref.next=newNode;
```

Удаление элемента с начала списка (с головы).

Эта задача такая же простая, как и добавление элемента с головы, так как указатель находится на начальном элементе. Новой головой становится ее ссылка на следующий элемент.

head=head.next;

Удаление последнего элемента списка (с хвоста).

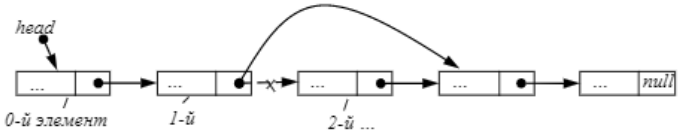
Для его удаления необходимо переместиться в конец списка на предпоследний элемент. Ниже приведен фрагмент кода.

```
// создаем вспомогательную переменную
   ref = head;
// перемещаемся на предпоследний элемент
   while (ref.next.next != null) {
     ref = ref.next;
   }
// полю next предпоследнего элемента присваиваем null
  ref.next=null;
```

Удаление элемента с заданным номером из списка.

Для этого ссылку элемента, предшествующего удаляемому, необходимо перебросить на элемент, расположенный после удаляемого.

Рассмотрим пример удаления второго элемента. Схематично удаление элемента показано на рис. 3, где полю next первого элемента задается указатель на третий элемент.



Ниже приведен фрагмент кода удаления второго элемента.

```
ref = head; // создаем вспомогательную переменную
k=1;
 // поиск положения узла, предшествующего удаляемому
while (ref.next!= null && (k<2 )) {
  ref = ref.next;
  k++;
}
 // переброска ссылки для исключения ненужного элемента из списка
 ref.next=ref.next.next;
```