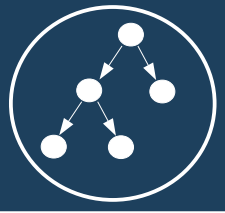


Data Structures and Algorithms

Структуры данных. Список на основе массива

Список

Список — это абстрактный тип данных, представляющий собой упорядоченный набор значений, в котором некоторое значение может встречаться более одного раза. Список динамическая структура данных.

Поддерживаемые операции:

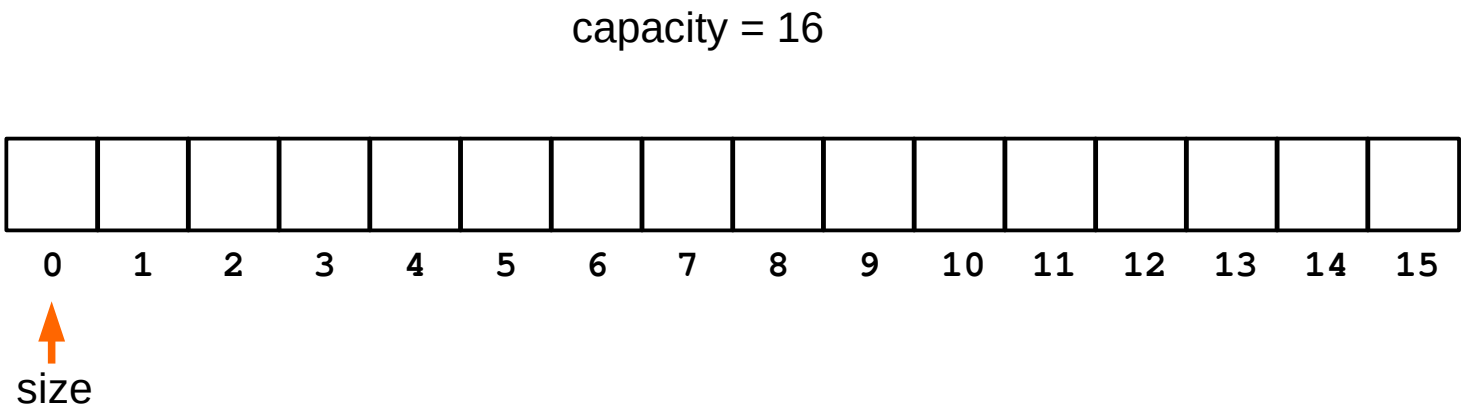
- Добавление элемента в список
- Удаление элемента из списка
- Получение элемента по индексу
- Замена элемента по индексу
- Получение размера списка

Список на основе массива

В качестве основы списка можно использовать массивы переменной длинны. В таком случае получение и замена элемента массива по индексу реализуется особенно просто. Сложности возникнут только со вставкой и удалением элемента из списка. Особое внимание следует уделить вопросам увеличения и уменьшения размера списка на основе массива.

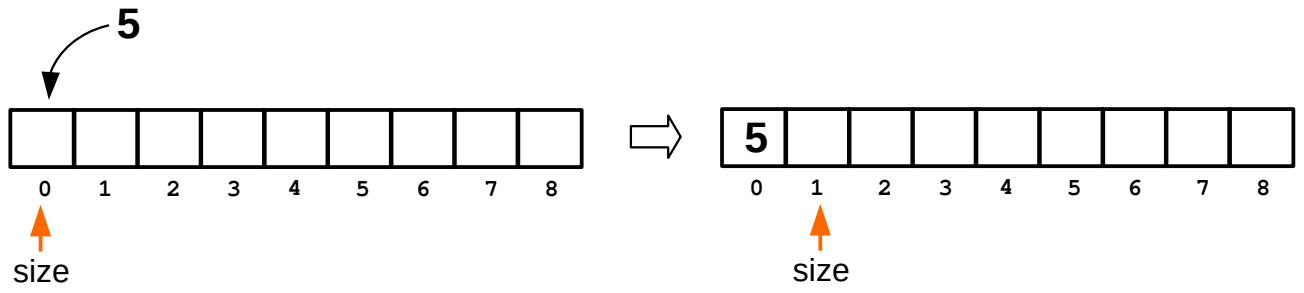
Реализации списка на основе массива

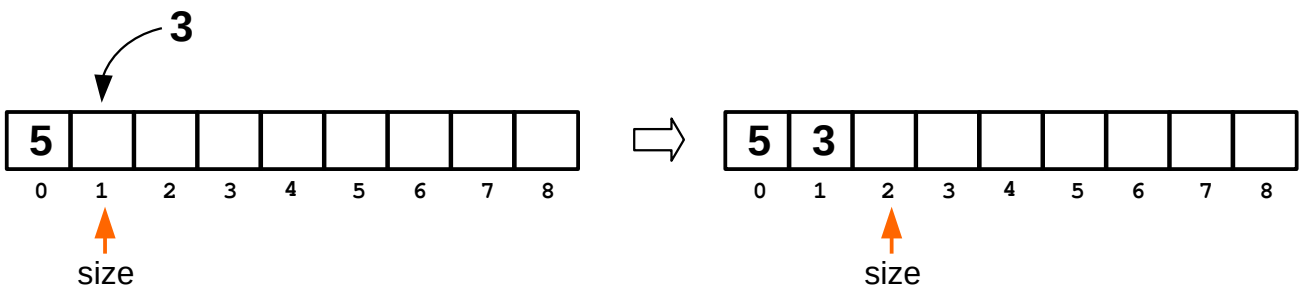
В качестве основы списка берем массив нужного типа данных. Его размер будем называть capacity(емкость). Также введем дополнительную переменную size(размер), она будет указателем на место для добавления элемента и к тому же используется для получения количества добавленных элементов. При создании списка устанавливается в начало массива.



Добавление значения в конец списка

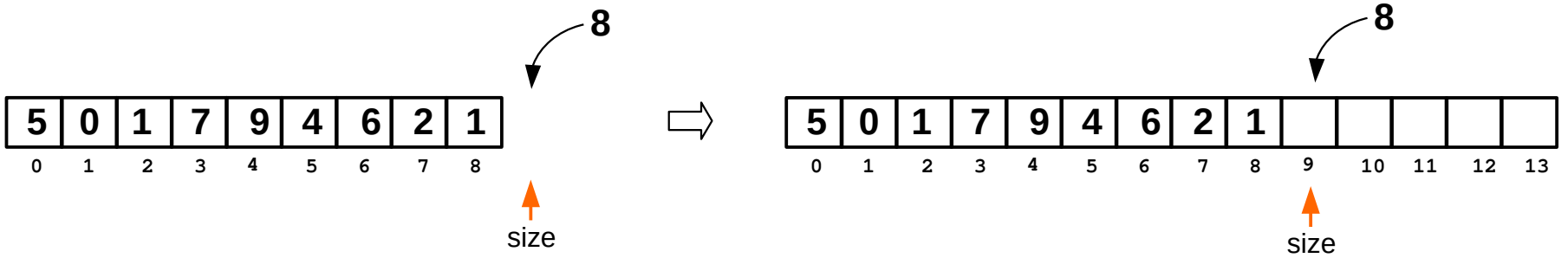
Если size меньше чем capacity, то добавляем элемент на индекс size и увеличиваем size на единицу.

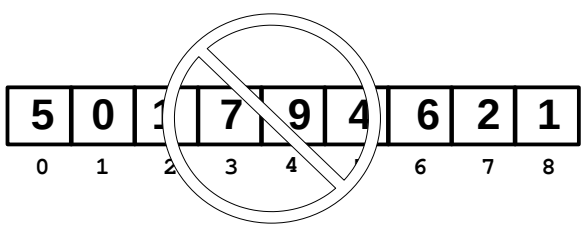




Добавление значения в конец списка

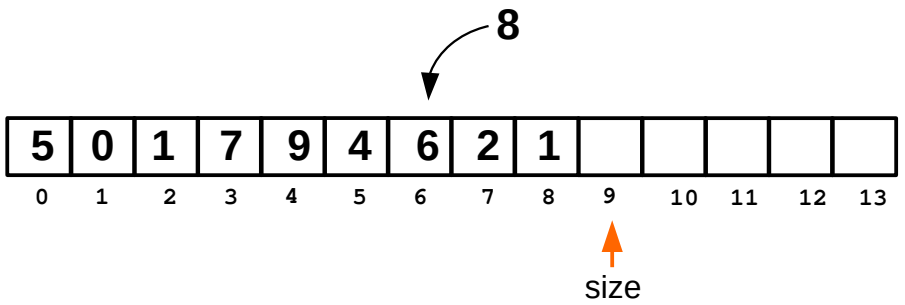
Если size равно capacity, то создаем новый массив размером (capacity * 3)/2 + 1. Копируем данные из базового массива в новый. Указываем, что теперь для хранения используется новый массив. Добавляем элемент на индекс size и увеличиваем size на единицу.



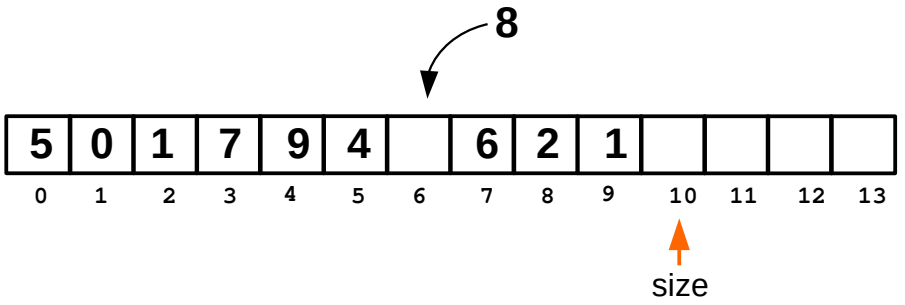


Освобождаем память занимаемую старым массивом.

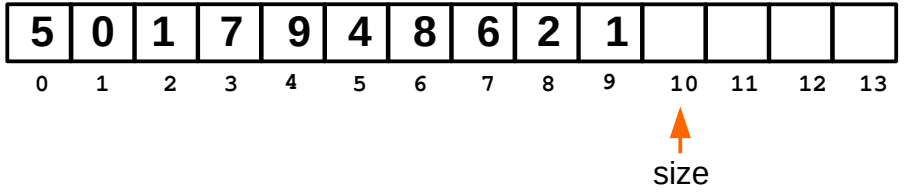
Вставка значения по индексу



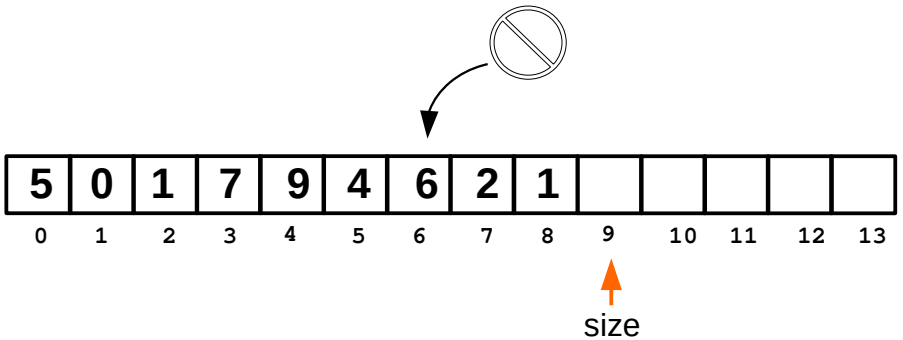
Проверяем достаточно ли места для вставки. Если нет, запускаем процесс увеличения размера. Сдвигаем правую часть массива (от индекса на который вставляем элемент до size) на одну позицию вправо (желательно вызвать быструю функцию копирования массива блоками).



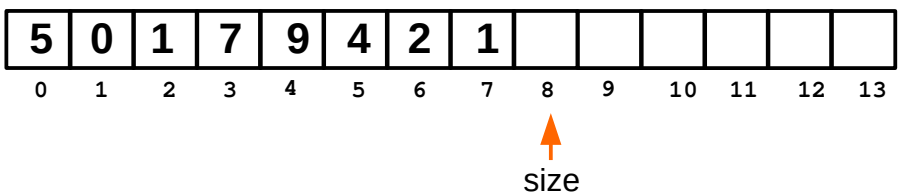
Ставим элемент на нужный индекс. Увеличиваем size на единицу.



Удаление элемента по индексу

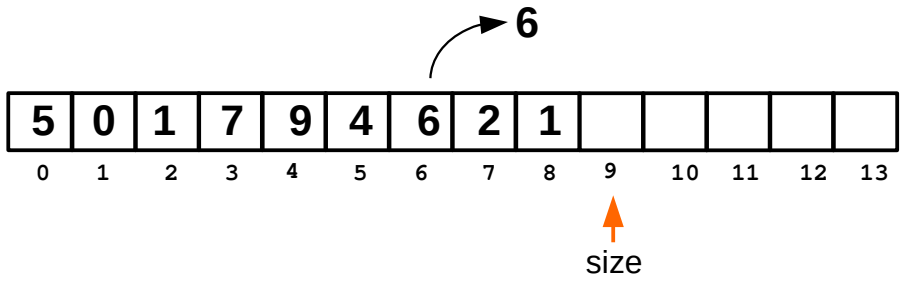


Сдвигаем правую часть массива (от индекса удаляемого элемента до size) на одну позицию влево (желательно вызвать быструю функцию копирования массива блоками).

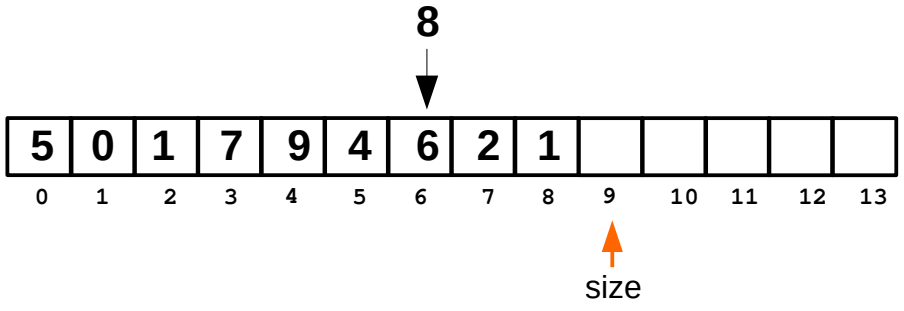


Уменьшаем значение size на единицу.

Получение и замена элемента по индексу



При получении значения по индексу сначала проверяют корректность индекса. После чего возвращаем значение по индексу.

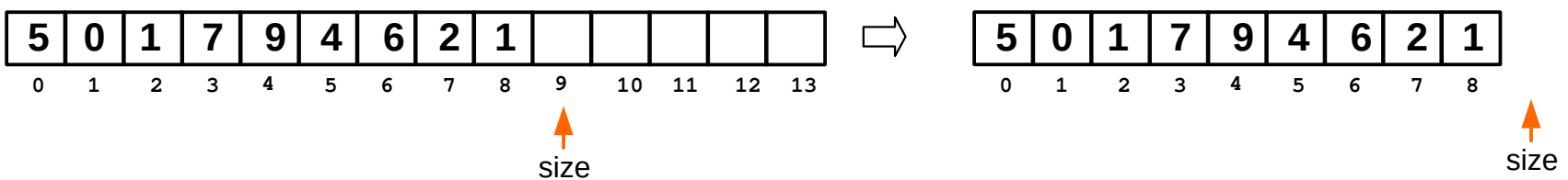


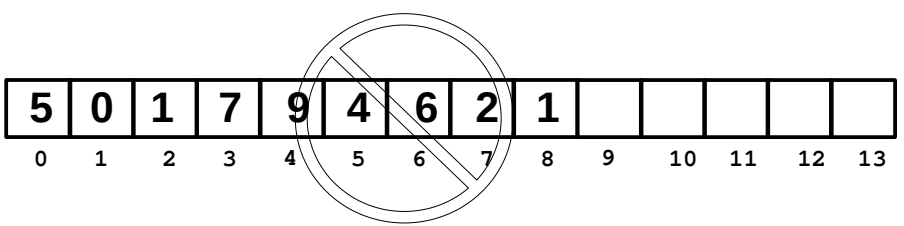
При замене значения по индексу сначала проверяют корректность индекса. После чего заменяем значение по индексу.

Уменьшение размера списка

В большинстве случаев список только увеличивает свою емкость. Автоматического уменьшения емкости не предусматривают. Для уменьшения емкости используют функцию, вызов которой осуществляется по желанию разработчика. В этой функции обычно устанавливают capacity равное size.

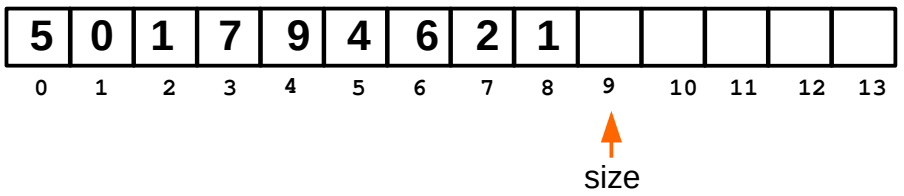
Создают новый массив размером size. Копируют данные из основного массива в новый. Указываем что новый массив теперь используется вместо основного.





Освобождаем память занимаемую старым массивом.

Получение размера списка



Для получения размера списка достаточно вернуть значение size.

Оценка сложности операций

Операция	Сложность операции в худшем случае
Вставка элемента	O(n)
Удаление элемента	O(n)
Получение по индексу	O(1)
Изменение по индексу	O(1)
Получение размера	O(1)

Python

Реализация на Python

Отсутствие массивов в Python

Так как в Python отсутствует поддержка массивов, то реализация списка становится бессмысленной.

Реализация на Java

Описание структуры списка

```
class ArrayBasedList {
     private int[] dataArray;
     private int size;
     private int capacity;
     private final int DEFAULT_CAPACITY = 10;

     public ArrayBasedList() {
          dataArray = new int[DEFAULT_CAPACITY];
          capacity = dataArray.length;
          size = 0;
     }
```

Методы добавления

```
public void add(int value) {
      if (size >= capacity) {
            boolean resizeResult = upResize();
            if (!resizeResult) {
                  throw new RuntimeException("Cannot add element");
            }
      }
      dataArray[size] = value;
      size += 1;
}

public void addByIndex(int value, int index) {
      if (index < 0 || index >= size) {
            throw new IndexOutOfBoundsException();
      }
      if (size >= capacity) {
            boolean resizeResult = upResize();
            if (!resizeResult) {
                  throw new RuntimeException("Cannot add element");
            }
      }
      System.arraycopy(dataArray, index, dataArray, index + 1, size - index);
      dataArray[index] = value;
      size += 1;
}
```

Метод удаления

```
public void deleteByIndex(int index) {
     if (index < 0 || index >= size) {
          throw new IndexOutOfBoundsException();
     }
     System.arraycopy(dataArray, index + 1, dataArray, index, size - index);
     size -= 1;
}
```

Методы для увеличения, уменьшения емкости и очистки

```
public boolean upResize() {
      if (capacity >= Integer.MAX_VALUE - 1) {
            return false;
      }
      long newCapacityL = (capacity * 3L) / 2L + 1L;
      int newCapacity = (newCapacityL < Integer.MAX_VALUE - 1) ? (int) newCapacityL : Integer.MAX_VALUE - 1;
      dataArray = Arrays.copyOf(dataArray, newCapacity);
      capacity = newCapacity;
      return true;
}

public void trimToSize() {
      dataArray = Arrays.copyOf(dataArray, size);
      capacity = dataArray.length;
}
public void clear() {
      dataArray = new int[0];
      capacity = dataArray.length;
      size = 0;
}
```

Метод для получения размера

```
public int size() {
     return size;
}
```

Реализация на Fortran

Описание списка

```
type Array_Based_List
    integer,allocatable::data_array(:)
    integer::l_size, capacity

    contains
        procedure,pass::init
        procedure,pass::add
        procedure,pass::up_resize
        procedure,pass::add_by_index
        procedure,pass::delete_by_index
        procedure,pass::get_size
        procedure,pass::get_by_index
        procedure,pass::set_by_index
        procedure,pass::trim_to_size
        procedure,pass::clear
        procedure,pass::show_list
end type Array_Based_List
```

Методы инициализации и увеличения размера

```
subroutine init(this)
    class(Array_Based_List)::this
    if(.not. allocated(this%data_array)) then
        allocate(this%data_array(1))
        this%capacity = 1
        this%l_size = 1
    end if
end subroutine init

subroutine up_resize(this)
    class(Array_Based_List)::this
    integer,allocatable::temp_array(:)
    integer::new_capacity
    new_capacity = (this%capacity * 3)/2 + 1
    allocate(temp_array,source = this%data_array)
    deallocate(this%data_array)
    allocate(this%data_array(new_capacity))
    this%data_array(1:this%l_size-1) = temp_array
    this%capacity = new_capacity
    deallocate(temp_array)
end subroutine up_resize
```

Методы добавления

```
subroutine add(this, data_value)
    class(Array_Based_List)::this
    integer, intent(in)::data_value
    if (this%l_size > this%capacity) then
        call this%up_resize()
    end if
    this%data_array(this%l_size) = data_value
    this%l_size = this%l_size + 1
end subroutine add

subroutine add_by_index(this, data_value, l_index, op_result)
    class(Array_Based_List)::this
    integer,intent(in)::data_value, l_index
    logical, intent(inout)::op_result
    if(l_index < 1 .or. l_index >= this%l_size) then
        op_result = .false.
        return
    end if

    if (this%l_size + 1 > this%capacity) then
        call this%up_resize()
    end if
    this%data_array(l_index+1:this%l_size+1) = this%data_array(l_index:this%l_size)
    this%data_array(l_index) = data_value
    this%l_size = this%l_size + 1
    op_result = .true.
end subroutine add_by_index
```

Метод удаления

```
subroutine delete_by_index(this, l_index, op_result)
    class(Array_Based_List)::this
    integer,intent(in)::l_index
    logical, intent(inout)::op_result
    if(l_index < 1 .or. l_index >= this%l_size) then
        op_result = .false.
        return
    end if
    this%data_array(l_index:this%l_size-2) = this%data_array(l_index+1:this%l_size-1)
    this%l_size = this%l_size - 1
    op_result = .true.
end subroutine delete_by_index
```

Методы получения и установки по индексу

```
integer function get_by_index(this, l_index, op_result)
    class(Array_Based_List)::this
    integer,intent(in)::l_index
    logical, intent(inout)::op_result
    if(l_index < 1 .or. l_index >= this%l_size) then
        op_result = .false.
        return
    end if
    get_by_index = this%data_array(l_index)
    op_result = .true.
end function get_by_index

subroutine set_by_index (this, l_index, data_value, op_result)
    class(Array_Based_List)::this
    integer,intent(in)::l_index, data_value
    logical, intent(inout)::op_result
    if(l_index < 1 .or. l_index >= this%l_size) then
        op_result = .false.
        return
    end if
    this%data_array(l_index) = data_value
    op_result = .true.
end subroutine set_by_index
```

Методы для получения размера

```
integer function get_size(this)
    class(Array_Based_List)::this
    get_size = this%l_size - 1
end function get_size
```

Методы для уменьшения размера и очистки

```
subroutine trim_to_size(this)
    class(Array_Based_List)::this
    integer,allocatable::temp_array(:)
    allocate(temp_array,source = this%data_array(:this%l_size-1))
    this%data_array = temp_array
    this%capacity = size(this%data_array, dim = 1)
    deallocate(temp_array)

end subroutine trim_to_size

subroutine clear(this)
    class(Array_Based_List)::this
    if(allocated(this%data_array)) then
        deallocate (this%data_array)
        this%capacity = 0
        this%l_size = 0
    end if
end subroutine clear
```

Список литературы

1) Роберт Седжвик, Кевин Уэйн «Алгоритмы на java 4-е издание» Пер. с англ. - М. : ООО "И.Д. Вильямс", 2013. ISBN 978-5-8459-1781-2.