Lecture 01 - Data Structures

- We will examine some basic data structures:
    - The Array
    - The List
    - The Stack
    - The Queue
    - The Record

Arrays

- An array is a data structure consisting of a fixed number of data items of the same type.
- Any array element is directly accessible via an index value.

```
x = array[27], initial = letters[7]
```

- Arrays can have more than on index, multidimensional arrays.
    - An array of real numbers is an array with 400 elements:

```
heights: array[1...20, 1...20]
```

- Initializing an array takes n operations for an array of n elements.

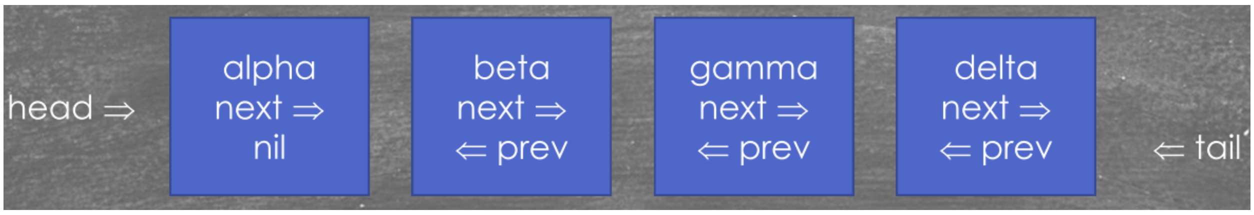
Lists

- A list is a collection of items arranged in some order.
- Unlike an array, elements of a list cannot be directly accessed via an index.
- List items (nodes) are records containing data and a pointer to the next node in the list.

```
type node = record {
    contents: stuff;
    next: ^node;
    prev: ^node
}
```

- Special pointers head (and tail for doubly linked lists) are maintained to point to the first (and last) elements of the list:

```
head: ^node
tail: ^node
```



- Insert an item onto a list start:

```
item: ^node
procedure list_add_start(item)
    item^.next = head
    head = item
```

- Insert an item onto a list end:

```
procedure list_add_end(item)
    tail^.next = item
    item^.prev = tail
    item^.next = nil
    tail = item
```

- Insert an item into a list after a specific node:

```
item: ^node
procedure list_add_mid(item, match)
    ptr: ^node
    ptr = head
    while ptr^.contents != match & ptr^.next != nill do
        ptr = ptr.next
    item^.prev = ptr
    item^.next = ptr^.next
    ptr^.next = item
    ptr = item^.next

    if ptr = nil then
        tail = item
    else
        ptr^.prev = item
```

Stacks

- A stack is a data structure which holds multiple elements of a single type.
- Elements can be removed from a stack only in the reverse order to that in which they were inserted (**LIFO** --> **Last In, First Out**).
- A stack can be implemented with an array and an integer counter to indicate the current number of elements in the stack.

```
stack: array[1...50]
ctr: integer
ctr: 0
```

- To put an element on the stack:

```
procedure push(elem)
    ctr = ctr + 1
    stack[ctr] = elem
```

- To remove an element from the stack:

```
procedure pop(elem)
    if ctr = 0 then
        elem = nil
    else
        elem = stack[ctr]
        ctr = ctr - 1
    fi
```

Queues

- A queue is a data structure which holds multiple elements of a single type.
- Elements can be removed from a queue only in the order in which they were inserted (**FIFO** --> **First In, First Out**).
- A queue can be implemented with an array and two integer counter to indicate the current start and next insertion positions.

```
queue: array[1...50]
start: integer
next: integer
start = 1
next = 1
```

- To put an element in the queue:

```
procedure enqueue(elem)
    queue[next] = elem
    next = next + 1
    if next > 50 then next = 1
```

- To take an element out of the queue:

```
procedure dequeue(elem)
    if start = next then
        elem = nil
    else
        elem = queue[start]
    fi
    start = start + 1
    if start > 50 then start = 1
```

Records (Structures)

- A record is a data structure consisting of a fixed number of items.
- Unlike an array, the elements in a record may be of differing types and are named.

```
type person = record {
    name: string
    age: integer
    height: real
    female: Boolean
    children: array[1:10] of string
}
```

- An array may appear as a field in a record.
- Records may appear as elements of an array.
  - e.g. `staff: array[1...50] of person`
- Records are typically addressed by a pointer.
  - Declare a boss to be a pointer to records of type person.

```
type boss = ^person
```

- Fields of a record are accessible via the field name.

```
staff[5].age
boss^.name
```