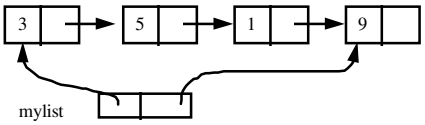
Lecture 12:
Dynamic Data Structures

Examples: linked lists, queues, stacks **(this lecture);**
trees **(Future lectures)**

Features:

- Grow and shrink at execution time
- Defined using pointers and self-reference
- Make programs simpler to design (User just writes simple functions to do data management : `insert(), delete(), search(), ...`)

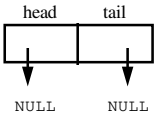
Linked Lists



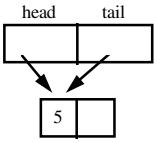

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
       int value;
       struct node *next;
};
struct list
{
       struct node *head;
       struct node *tail;
};
void insert(struct list *l, int value);
void delete(struct list *l);
void main()
{
  struct list mylist = {NULL, NULL};
```

Our Conventions for Lists

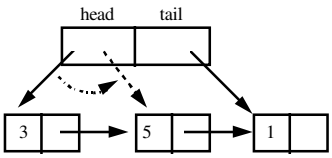
An empty list:



A list with one element:

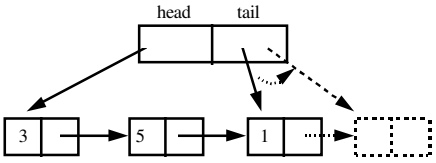


Deleting an element from an existing list:

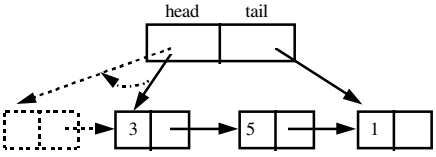


Inserting a new node in the list: 2 options

Option 1: Add at the tail



Option 2: Add at the head



Insert(): Option 1

(same as addto( ) of last lecture)

```
void insert(struct list *l, int value)
{
 struct node *newnode;
 newnode = malloc(sizeof(struct node));
 (*newnode).number = value;
 if ((*l).head == NULL)
    (*l).head = (*l).tail =newnode;
 else
    {
      (*(*l).tail).next = newnode;
      (*l).tail = newnode;
    }
}
void main()
{
  struct list mylist={NULL, NULL};
  insert(&mylist, 3);
  insert(&mylist, 5);
}
```


Insert(): Option 2

```
void insert(struct list *l, int value)
{
 struct node *newnode;
 newnode = malloc(sizeof(struct node));
 (*newnode).number = value;
 if ((*l).head == NULL)
    (*l).head = (*l).tail =newnode;
 else
    {
      (*newnode).next = (*l).head;
      (*l).head = newnode;
    }
}
void main()
{
  struct list mylist={NULL, NULL};
  insert(&mylist, 3);
  insert(&mylist, 5);
}
```

Delete()

(deletes the first node of a list and returns a pointer to this node)

```
struct node *delete(struct list *l)
{
 struct node *delnode;
 delnode = (*l).head;
 if ((*l).head == (*l).tail)
    (*l).head = (*l).tail =NULL;
 else
    (*l).head = (*(*l).head).next;
 return delnode;
}

void main()
{
  struct list mylist={NULL, NULL};
  struct node *mynode;
  insert(&mylist, 3);
  insert(&mylist, 5);
  mynode = delete(&mylist);
  printf("%d", (*mynode).number);
  free(mynode);
}
```


Stacks and Queues

Queue: Linked List with insert() option 1.
Data that went in first, comes out first.
("First In, First Out")

Used to simulate "waiting lines".

Stack: Linked List with insert() option 2.
Data that went in last, comes out first.
("Last In, First Out")

Used to implement recursion.
Used in "interpretive" environments
(postscript, HP calculators)

Postscript

A programming language used to describe images.
(i.e., image is represented *logically* instead of as a collection of *pixels.*

To printer

```
#!
200 110 moveto
100 300 lineto
stroke
100 300 2 0 360 arc
fill
showpage
```

(Printer contains an "interpreter" for postscript. The interpreter uses a stack.)

What does the following C program do?

```
#include <stdio.h>
void main()
{
  int i, j;
  scanf("%d, %d", &i, &j);
  printf("%%");
  printf("!\n");
  printf("%d %d 2 0 360 arc\n", i, j);
  printf("fill\n showpage \n");
}
```