CSE 326
Lecture 3

Spring Quarter 1995

Example Pseudocode

Ok

- Enqueue(Q,x)

  p := newnode;
  p.data := x;
  p.next:= nil;
  rear.next := p;
  rear := p

Better

- Enqueue(x)

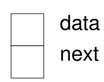
  local pointer p;

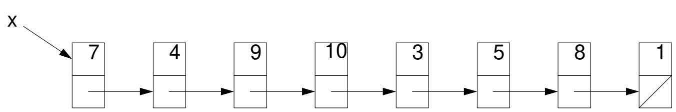
  p := newnode;
  p.data := x;
  p.next:= nil;
  rear.next := p;
  rear := p

- variables that are not local and not passed in are part of the scope of the class where the function or procedure is defined.

Recursion Example

- Finding the maximum in a list.





- max(x);

  local integer m;

  if x = nil then return(-infinty)
  else { m := max(x.next);
  if m > x.data then return(m)
  else return(x.data) }

Guides in Recursion

- Recursive algorithms on well designed data structures are often short, elegant, and efficient.
- In some cases recursive algorithms can be converted to interative algorithms improving efficiency.
- Guides:

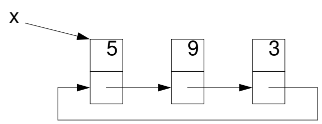
  1. There must be some base case or cases.
  2. Progress toward the base cases must be made at each recursive call.
  3. In designing a recursive algorithm alway assume that the recursive calls work.
  4. Never duplicate a recursive call on the same data. (This is different than number 2 above.)

Causes of Lack of Progress

- Inadequate basis

```
bad(n);
  if n = 0 then return(0)
    else return(bad(n/3 + 1) + n)
```

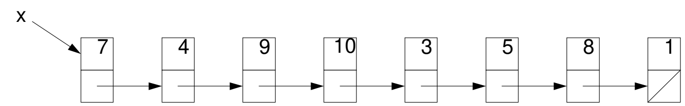
- Data has a cycle, so basis is never reached.



Iterative Max

- max(x);

```
if x = nil return(-infinity)
  else {
    m := x.data;
    while not (x.next = nil) do {
      x := x.next;
      if x.data > m then m := x.data };
    return(m) }
```



More on Unix and C++

- Dan Fasulo on keyboards