# CS 101 Computer Programming and Utilization

Lecture 20

# Switch, structures vs. classes, linked lists

Mar 29, Tuesday 2011

*Prof. R K Joshi*
*Computer Science and Engineering*
*IIT Bombay*
Email: rkj@cse.iitb.ac.in

# Revision

- sorting
  - cannot sort in one single iteration of i:0..n-1
  - sorting numbers in ascending/descending order
  - bubble sort
    - two loops
    - swaps
    - terminating early: if the array gets sorted earlier
    - no. of steps
      - of the order of $n^2$
- searching
  - input: array, element
  - output: position if found
  - if the array is not sorted, in worst case, n-1 steps (comparisons) are needed for an array of n elements
  - binary search on sorted array
    - $\log_2 n$ steps in worst case
    - search at the mid
    - if not found, search in either right or left half, depending upon where the element is likely to be found

# Switch statement

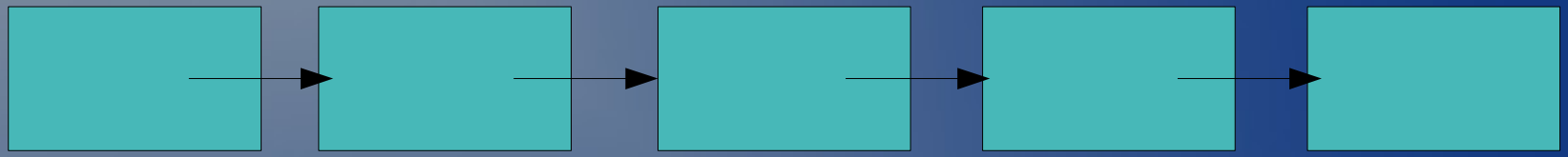
```
cin >> choice;
switch (choice ) {

    case 1: f1(); break;
    case 2: f2(); break;
    case 3: f3(); break;
    default: f4();
}
```

# Structures

```
struct Node {
    int roll;
    string name;
    Node *next;
};
```

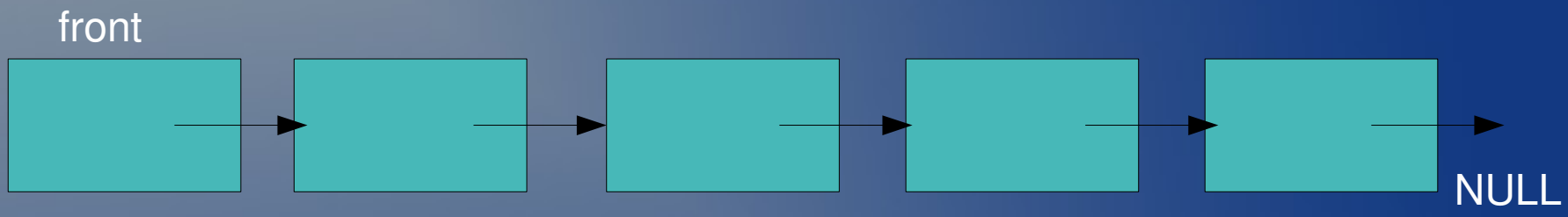
- many values put together (composite)
- can be used to return multiple values
- can also use it for linking structures

# Linked Lists with Structures

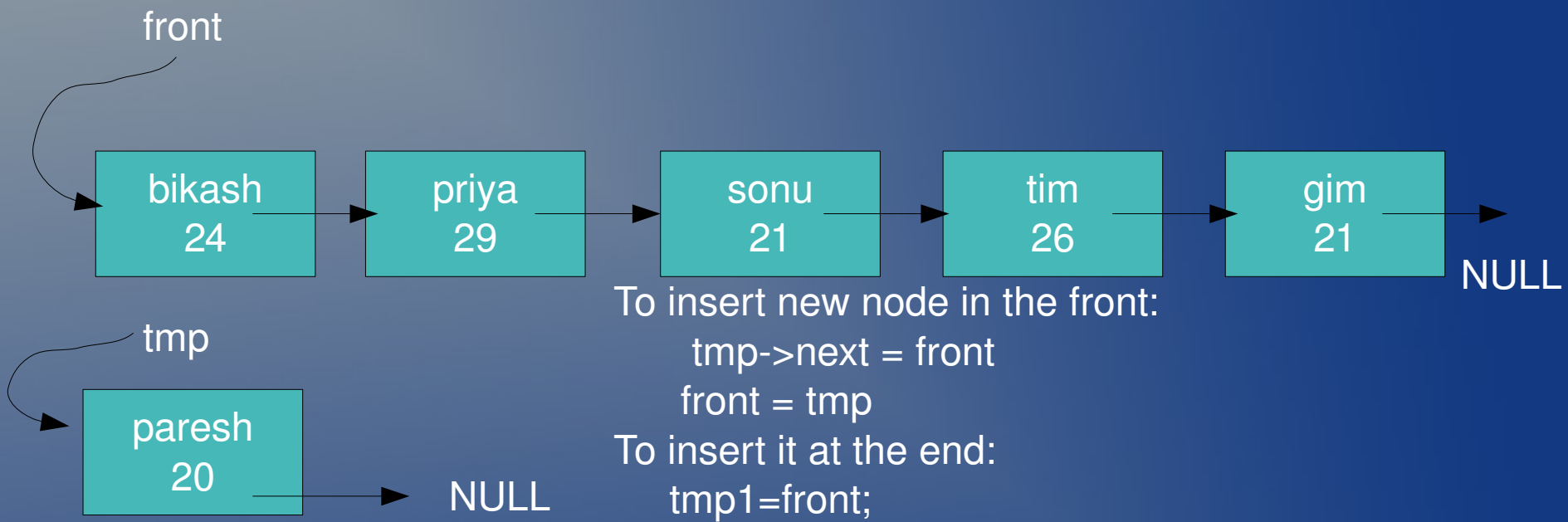


- each node is a structure
- each node has  pointer to the next structure

# Linked Lists with Structures



# Linked Lists with Structures




To insert new node in the front:

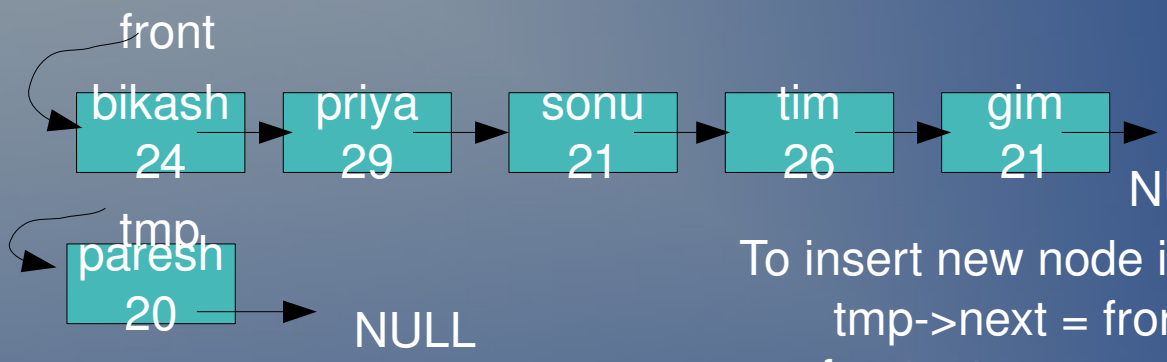
```
tmp->next = front
front = tmp
```

To insert it at the end:

```
tmp1=front;
if (tmp1==NULL) front = tmp
else {
    while (tmp1->next!=NULL) tmp1=tmp1->next
    tmp1->next = tmp
}
```

```
delete (BookRecord *front, int accno);
```

# Linked Lists with Structures




```
struct BookRecord {
  int accno;
  string title;
  int status;
  BookRecord *next;
}
main:
   BookRecord *front = NULL;
   ....
```

To insert new node in the front:

```
      tmp->next = front
     front = tmp
```

To insert it at the end:

```
    tmp1=front;
    if (tmp1==NULL) front = tmp
    else {
        while (tmp1->next!=NULL) tmp1=tmp1->next
        tmp1->next = tmp
    }
```

```
delete (BookRecord *front, int accno);
```

# structures vs. classes

- members in structures are default public
- classes can restrict visibility
- traditionally, structures are used in 'C' language for holding records consisting of many data values
- classes are meant to describe objects having both private state/data and public member functions

# structures vs. classes

- structures + functions : C style programming
  - structures are outside functions
  - functions operate on structures
  - structure go in as parameters to function
- classes with member functions : Object Oriented
  - data elements are private inside a class
  - member functions operate on private data
  - private data is automatically accessible to member functions
  - member function arguments are typically primitive data value, or other objects