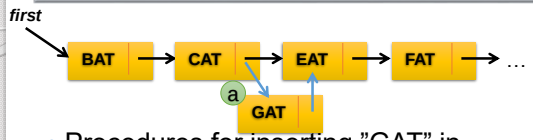
4.2

Representing Chains in C++

4.2

SLL Operation: Insert



- Procedures for inserting ”GAT” in between “CAT” and “EAT” nodes
  - Create a new node “a” and set data field to “GAT”
  - Set the link field of “a” to “EAT” node
  - Set the link field of “CAT” node to “a”

You do not need to move or shift any node!

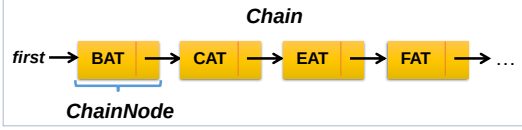
SLL Operation: Delete

- Steps to do when we want to delete the ”EAT” node from the list
  - Locate the node “a” precedes the “EAT” node
  - Set the link field of “a” to node next to “EAT” node
  - Delete the “EAT” node

You do not need to move or shift any node!

Conceptual Design

- Defining a “ChainNode” class
  - Data field
  - Link field
- Designing a “Chain” class
  - Support various operation on ChainNodes



8

4.2.1

ChainNode & Chain Classes

- Composite class

```
class ChainNode
{
friend class Chain;
public:
 // Constructor
 ChainNode(int
value=0, ChainNode*
next=NULL){
   data = value;
   link = next;
   }
private:
   int data;
   ChainNode *link;
};
```

```
class Chain
{
public:
     // Create a chain with two nodes
     void Create2();

     // Insert a node with data=50
     void Insert50(ChainNode *x);

     // Delete a node
     void Delete(ChainNode *x, ChainNode *y);

private:
     ChainNode *first;
};
```

9

ChainNode & Chain Classes

- Nested class

```
class Chain
{
public:
     // Create a chain with two nodes
     void Create2();

     // Insert a node with data=50
     void Insert50(ChainNode *x);

     // Delete a node
     void Delete(ChainNode *x, ChainNode *y);

private:
      class ChainNode{
        public:
          int data;
          ChainNode *link;
     }
     ChainNode *first;
};
```

10

4.2.3

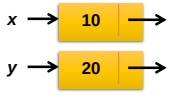
Review Pointer Manipulation

- **Declaration**
  - NodeA *a1=NULL, *a2=NULL;
- **Allocate memory**
  - a1 = new NodeA;
  - a2 = new NodeA[10];
- **Delete memory**
  - delete a1; a1=NULL;
  - delete [] a2; a2=NULL;
- **Dereference**
  - NodeA &a1Ref = (*a1);
- **Access members**
  - a1->memData;
  - a1->memFunc();
  - (*a1).memData;
  - (*a1).memFunc();

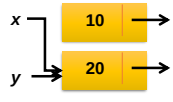
11

Pointer Assignment

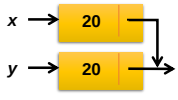
- ChainNode *x, *y;



- x = y;



- *x = *y;

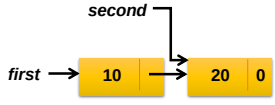


12

4.2.4

Chain Operations

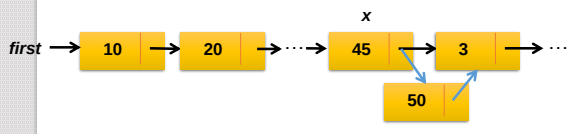
```
void Chain::Create2()
{
   // Create and set the fields of 2nd node
   ChainNode* second = new ChainNode(20,0);

   // Create and set the fields of 1st node
   first = new ChainNode(10,second);
}
```



13

Chain Operations

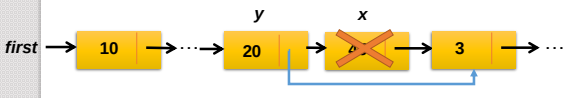
```
void Chain::Insert50(ChainNode *x)
{
   if( x ) // Insert after x
      x→link = new ChainNode(50, x→link);
   else // Insert into empty list
      first = new ChainNode(50);
}
```



14

Chain Operations

```
void Chain::Delete(ChainNode *x, ChainNode *y)
{  // x is the node to be deleted and y is the node
   // preceding x
   if( !x || !y ) throw "cannot delete NULL nodes!";
   if(x==first) first = first->link;
   else y->link = x->link;
   delete x; x=NULL;
}
```



15