2004 A.M. Turing Award Winners

Vinton G. Cerf
Robert E. Kahn

Citation
For pioneering work on internetworking, including the design and implementation of the Internet's basic communications protocols, TCP/IP, and for inspired leadership in networking.

Press Release
New York Times Article

Biographical Information:
Vinton G. Cerf, MCI
Robert E. Kahn, CNRI

Linked List - II

Default Memberwise Assignment

• Assignment operator ( = )
  - Can assign one object to another of same type
  - Default: memberwise assignment
    • Each right member assigned individually to left member
• Passing, returning objects
  - Default: pass-by-value
    • Copy of object passed, returned
    • Compiler-generated copy constructor
  - You must provide your own copy constructor when there are member pointer variables
    • And destructor too
Consider the following two ShoppingList objects:

- What would the result of `slist2 = slist1;` be if the default copy behavior took place?

- `slist2` becomes a “shallow copy” of `slist1`.
- `slist2` doesn’t get a copy of `slist1` data.
- All references to `slist2` data are lost!

Here is what `slist2` would look like as a “deep copy” of `slist1`:

- Note that `slist2` has its own set of data.
ShoppingList::ShoppingList(const ShoppingList &slist) {
    Node *cur1,
    *cur2;
}

Clone Example (2 of 8)
this->deleteAll();
this->size = slist.size;

Clone Example (3 of 8)
this->front = new Node(*(slist.front).item);
Clone Example (4 of 8)

```cpp
cur1 = this->front;
cur2 = slist.front;
```

Clone Example (5 of 8)

```cpp
while (cur2->link) {
cur2 = cur2->link;
cur1->link = new Node(*cur2);
cur1 = cur1->link;
}
```

Clone Example (6 of 8)

```cpp
while (cur2->link) {
cur2 = cur2->link;
cur1->link = new Node(*cur2);
cur1 = cur1->link;
}
```
Clone Example (7 of 8)

```c
cur1 = cur2 = 0;
```

![Diagram of a linked list with three nodes A, B, and C, showing the front and size]

Clone Example (8 of 8)

```
}
```

![Diagram of a linked list with three nodes A, B, and C, showing the front and size]

Doubly Linked List

- Each node has 2 pointers, one to the node before and one to the node after
- No longer limited to left-to-right pointer movement
- Typically, a rear pointer is employed to simplify reverse traversals

![Diagram of a doubly linked list with nodes A, B, and C, showing the front and rear]
Circular Linked List

- “Last” node’s link points at the “front” node
- Concept of “front” optional

![Circular Linked List Diagram]

---

---