Double Linked List Programming needed for Proj 2.

Singly Linked List Programming: DS0 Ch. 5

OOP style compared to classical C style... C style OK for this course.

CSI 310: Lecture 12
Back to Proj 2 data structures...
2. struct node for the linked list of char * data.

1. C-strings for text lines.

Projo. 2 Dynamic objects:

The dynamic memory used shrinks as well as grows as needed, dynamically.

Partially filled array.

dynamically allocated. DSO CH. 4 covers the dynamically allocated
such data structures can be virtually unlimited in size if the objects are

Dynamic objects that contain some pointer type that hold addresses of structure type

A linked data structure consists of some structure type objects (variables)
multitudes of Java Library and Language Classes are dynamic.

PT: All of the objects (variables, instances) of the interesting and full featured

(matte and free in C

coded operation (new and delete) executed when the program runs.

But a dynamic variable is created or destroyed only by an explicitly
function is called and are destroyed (storage released) when it returns.

3. CSI201 Local extern (C/C++ automatic) variables are created when a
input data can determine if and how many dynamic variables are created.

2. Dynamic variables are created during process execution, unpredictable.

I. Dynamic variables are not declared. They are not named by identifiers.

You need pointers to access them! In Java you need "references".

Dynamically Allocated Variables

University at Albany Computer Science Dept.
p1 = new int[4];

new

To allocate an array of say 4 ints, do the same except use the array form for

p1 = new int;

Then make program execute:

e.g.: int *p1;

& p1

variable to hold its address.

To allocate ONE variable of any type, say int, make sure you have a pointer

Doing dynamic allocation: use new.
It consists of nodes, a Head pointer, and (usually) a Tail pointer. Nodes are very fast (in fact, (small) constant time). Known locatons are one-dimensional sequences, so entity insertion and deletion near entries at

Linked List: Concrete linked data structure good for implementing
thead=TAIL=new node(23.6);
node * TAIL;
node * HEAD;

...;

node * TAIL;
val-TAIL data;
private:
    ...

{ }
data = val * p
node p = NULL;
node (node (const val-TYPE val = val-TLP),
typedep double val-TLP;
public:
    }
};

class node

<table>
<thead>
<tr>
<th>Classical C Style</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OOP/Modern C++ Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building a 1-Item Linked List</td>
</tr>
</tbody>
</table>

University at Albany Computer Science Dept.
With our classical style code, (7) and (5) are executed before and (6) afterwards.

(6) Copy "ditto" into automatic variable HEAD.

(7) Copy the address (pointer) to the new node) into automatic variable TAIL.

(8) Copy 0 (NULL) into the TAIL field.

(9) Copy (float) 23.6 into the data field of the new node variable.

(10) Run the dynamic memory allocation "new" of the C++ support library.

For both styles, the computer does the same things with the same results.
{if (TAIL!=NULL) TAIL=HEAD;

    // HEAD before copying
    if (YOU == you lose your
        HEAD = tp;
    tp->TP = HEAD;
    tp->DATA = 14.7;
    tp = new node;
    node * tp;
    ...
    }

    node * TTP;
    val = tp;
    private:
    ...

    { data = TTP;  
        TTP = p;
    }

    node * p = NULL;
    node const val-TP = val-TP;
    typedef double val-TP;

    public:
    class node
    }

  }
The old value of HEAD was the ADDRESS of the node "housing"

23.6; it was NOT 23.6.

The old value before step 4:

1. If TAIL == NULL, the original list was empty, so TAIL = HEAD gives TAIL.
2. This operation OVERWRITES the previous value in HEAD, so it must be done AFTER step 3.
3. Copy the address of the new node (the one containing 14.7) into automatic variable HEAD.
4. This operation OVERWRITES the previous value in HEAD, so it must be done AFTER step 3.
5. Copy pointer value from HEAD into the T1nk held of the new node.
6. Copy (float) 14.7 into the data held of the new node variable.
7. Run the dynamic memory allocator "new" of the C++ support library.

For both styles, the computer does the same things with the same results.

University at Albany Computer Science Dept.
meaning, plus practice solving problems!
understanding of variables, data and (C/C++) language.
Have patience: Speed and elegance come from precise steps.
If it fails for boundary cases, try to fix it and repeat.
Then, check if for any boundary case(s). (E.G: Empty list and I-item list)
and draft code if not.
Check that it works in the general case. Re-design idea in the general case.
Advice: Try to design and draft code for an algorithm.
otherwise. It's good!

This last C++ statement works when the list was empty and does nothing.
public:
    node *TAIL;
    node *HEAD;
private:
};

class edictorcore {  // Note class edictorcore definition uses node:
    public:
        struct node {  // Note USE completely public struct class/struct node:
            char *data;  // for NULL
            struct node *prev;
            struct node *next;
        };
    #include <cstddeef>  // for NULL
    #define edictorcore-struct
    #ifndef edictorcore-struct
    #endif
    #define edictorcore  // #endif
    #include <edictorcore.h>

    #define edictorcore-struct
    #ifndef edictorcore-struct
    #endif
    #define edictorcore  // #endif
    #include <edictorcore.h>

    #define edictorcore-struct
    #ifndef edictorcore-struct
    #endif
    #define edictorcore  // #endif
    #include <edictorcore.h>

    // simplicify:

    Alternative: Do this if you are having trouble, haven’t started (really?) or love
    Main/Save “Linked List Toolkit” is optional.

    University at Albany Computer Science Dept. 13
{ /* etc */
editorcore();

HEAD=TAIL=NULL;
*/
*/
{ 
// Stuff Omitted...
node *cutTAIL;
node *cutHEAD;

3. Cursor position is AT THE END of the buffer.
   // or LAST LINE. //
2. Cursor position is BEFORE the 2nd, 3rd, etc,
   (insert new lines at the beginning in all cases) //
1. Cursor position is THE FIRST BUFFER LINE
node *cursor; // YOUR JOB: How(?) to distinguish
private:

inserted at the beginning of the main list. //

post: the line referred to by pch is
allocated c-string. //
pre: pch holds the address of a dynamically

vold insertFRONT (char * pch )
private:

University at Albany Computer Science Dept.
```c
void editorcore::insertfront(char * ph) {
    node * tp = new node;
    tp->prefix = NULL; // HEAD is a class editorcore DATA MEMBER.
    tp->next = prex; // NULL is a standard macro.
    if (TAIL==NULL) { // if (TAIL==NULL)
        HEAD = tp; // HEAD is a class editorcore DATA MEMBER.
        TAIL = tp; // TAIL is a standard macro.
    }
    prex = tp;
    node *temp = pch;
    data = temp->node-pointer;
    if (getline(editorcore::interpret(ph), temp)) {
        // include <stdlib.h>
        // include "editorcore.h"
        // include editorcore.hxx
        // University at Albany Computer Science Dept.
    }
    return;
}
```
ie., executes return, or fails out.

`Returns`  

Automatically deallocated `when` executed.

(i.e., called, invoked, etc.)

Automatically allocated `when` executed; `interpreter::interpreter` is activated

tp is a local storage external variable.

NOT the variable named tp.

POINTS TO.

delete tp; deletes the OBJECT/STORAGE/VARIABLE the value of tp.

Main and Save error warn "Don't code delete tp;"
```c
if (HEAD == TAIL)
    TAIL = t;
if (TAIL) TAIL->TINK = t;
if (TAIL) TAIL->TINK = t;
t->TINK = NULL;
t->data = 33.3;
node * t = new node;
...

void set_tk(node * p)
{
    node * tink;
    double data;
    struct node
    
    typedef double val_tpf;

public:
    class node
    
    OOP Modern C++ Style

Adding an item to the end: Requires modifying last node.

University at Albany Computer Science Dept.

Classical C Style

```
TAIL before dereferencing it, to set the last ink!  

// woe is you if you lose your
...if (ptr == NULL) ...

...can be shortened to...

...inside conditionals: all non-NUL values convert to true there. So,

2. Rule of C/C++: The NULL pointer value is converted to boolean false.

When isn't there a last node?

MUST be conditional on whether there is a last node.

NUL link with the address of the new node, the code for modifying it.

Since the last node of the original list must be modified (to replace the

Remarks:
node *t = new node(33.3);
if (HEAD) HEAD = TAIL;
if (!HEAD) HEAD = TAIL;
does nothing here..
TAIL = t;
{
    HEAD = t;
    assert (HEAD);
}
else
{
    TAIL->LINK = t;
    assert (TAIL);
}
/* list is *not* empty */

Alternatively (nicer) code:
pointer to (addr of) a char

another cut\0

THIS LINE'S BEEN CUT\0

FIRST MAIN LIST TEXT LINE\0

2ND LINE\0

THE THIRD\0

NULL

NULL

NULL

NULL

My editor core

HEAD

TALL

CUT HEAD

CUT TAIL

You design

cursor and

any others

pointer to (addr of) a node
Given this doubly linked list of 3 C-string addresses,

between TWO and THl3,¿?

how can you insert the C-string containing Dogs

Th3

Two

One

NULL

NULL

NULL

HEAD

TAIL
Given the original data structure, how can you change it to:

1. Dogs
2. NULL
3. NULL
4. One
5. TWO
6. THREE
Copy char array's address into it.

Allocate a dnode, saving its address.

Copy C-string into it.

Allocate char array (for C-string) saving its address.

Determine char array length needed is 5, for "dogs".
Given the original data structure, how can you change it to this one?
Remember, we want to insert Dogs between the two circled nodes. We NEED the addresses of the 2 circled nodes so we can modify their values.
struct dnode {
  dnode* pfore; // the first dnode.
  dnode* pback; // the last dnode.
  char* data; // data==addr of a C-string.

  // data!=NULL.

  // data==addr of a C-string.
  // pfore==addr of the previous dnode if any.
  // pback==addr of the next dnode if any.
};

The field (data member) names will be omitted in our data structure diagrams.
Alternatively, if we know the address of the right node, we can program:

```
if tpright -> pfore;
```

Therefore, if tpright contains the address of the circled node:

```
tpright = tpright -> pback;
```

The following code calculates the address of the node:

If tpright contains the address of the left circled node:

```
tpright = tpright;  
```
But, how can we obtain ANY one of these addresses?

\texttt{node * tpleft; node * tpright; (C++ declarations for these variables)}

(2) Use the value in a "cursor" variable, or search the linked list forward or backward.

(1) Starting from value of \texttt{head} or \texttt{tail}, traverse or search the linked list forward or backward.
Relevant data str. is circled:

How can we connect it? Now a new dnode containing the addr. of the dogs' string is accessible via `tpn` AND 2 variables hold the addresses of dnodes we will put it between.

"NULL" head tail．
to connect (which is the new dnode) into the linked list:

1. Set 2 link fields
   \[ tpleft->pfore = tpright; \]
   \[ tpright->pback = tpleft; \]

2. Set 1 link field in each of
   \[ tpleft->pright = tpright; \]
   \[ tpright->pback = tpleft; \]

Only 2 more steps remain
The result of step 1, tpn−>fore=tpn−notin

I

TPRIGHT

NUL

TPRIGHT

TPRIGHT

TPRIGHT

The result of step 1,
The result of step 2,}\n\nIs \n\n\nVolla
\n\nTwo
\n\nTWO
\n\nTLP
\n\nTPLEFT
\n\nTPRIGHT
\n\nNULL
\n\nDogs
\n\nTHE
\n\n145
\n\n37
\n\nUniversity at Albany Computer Science Dept.
The temporary pointer variables are not needed any more. (out of the picture)
Possible cursor feature:

A bigger picture after the changes:
array alloc, and C-stuff), man scripts, etc. Given here, in chapter 5, chapter 4 (on dynamic
explanations, analyses, etc. Give here, in chapter 5, chapter 4 (on dynamic
Your job is to solve these programming problems, using the declarations,°

2. Removing a node; and moving it into the cat-list?

OTHER pointers must also be set.

Hint: When a node is inserted, its own 2 pointers must be set, AND 2

6. Inserting a node into the EMPTY list?

5. Inserting a node at the BEGINNING?

4. Inserting a node after the LAST?

implemented.

3. Moving the cursor up or down. EASY, depends on how the cursor is

puzzle for you.

2. Making the cursor work to indicate the end of list? Hmm...that's a

1. Implementing a cursor in front of a line? EASY! use a node pointer.

What about