

Stack ADT

CS2263 – Systems Software Development



1

Learning Outcomes

At the conclusion of this lecture students should be able to:

- itemize the subset of the list ADT functionality required for a stack ADT
- Implement that functionality as a linked-list to create a stack ADT

2

References

- Lu, Yung-Hsiang. 2015. Intermediate C Programming. CRC Press. New York. (Chapter 18)
- Remember – the textbook code is available from github:
 - <https://github.com/yunghsianglu/IntermediateCProgramming>

3

What is a List ADT?

A collection of items (int, char*, Point2D ...)

- Public interface
- Private implementation

4

List ADT: Public Interface

Declared in the .h file, but in general functions can be

- Creation
- Read
 - Search
 - Report
- Update
 - Add an object at some location
 - Remove an object from some location
- Deletion
 - Including all objects

5

List ADT: Private Implementation

- Includes all functions specified in .h, but can also include "private" functions.
- Review of
 - Data structure
 - Creation
 - Searching
 - Adding links
 - Removing links
 - Deletion

6

List ADT: Data Structure

```
typedef struct charlist {
    pCharLink head;
    pCharLink tail;
    pCharLink iterator; /* not implemented */
    int nLinks;
}charList, *pCharList;
```

7

List ADT: Creation

- Need to declare at least a head pointer to the first link

```
/*
 * Create an empty List
 */
pCharList mallocCharList(){
    pCharList pCLThis = (pCharList)malloc(sizeof(charList));
    if(pCLThis == (pCharList)NULL )return pCLThis;
    pCLThis->nLinks = 0;
    pCLThis->head = (pCharLink)NULL;
    pCLThis->tail = (pCharLink)NULL;
    pCLThis->iterator = (pCharLink)NULL;

    return pCLThis;
}
```

8

List ADT: Reading

- “Chaining” through the list

```
/*
 * print the payloads of the list
 */
void fprintfCharList(FILE* pFin, pCharList pCLThis){
    pCharLink working;
    working = pCLThis->head;
    while(working != (pCharLink)NULL){
        /* print payload here */
        working = working->next;
    }
    return;
}
```

9

List ADT: Updating

- Need to find the place to add to/remove from
 - Application specific (at least for now)
- Three cases for addition/removal
 - First position (need to involve head pointer)
 - Last position (the next pointer maybe NULL)
 - In between (the general case)

```
/*
 * Delete an object from a list
 */
int delFromCharList(pCharList pCLThis, int iPos){
    pCharLink working;
    pCharLink temp;
    /* range checks */

    /* Case I: it's the head */
    /* Cases II and III: chain to the position */
    /* Case III: It's the last link in the list */
    /* Case II: It's in between */
    return 0;
}
```

10

List ADT: Deleting

- “Chaining” through the list (just like reading)
- Removing links requires care

```
/*
 * Free List, including all links/nodes
 */
void freeCharList(pCharList pCLThis)
{
    /* free all links */
    pCharLink working;
    working = pCLThis->head;
    while(pCLThis->head != (pCharLink)NULL ){
        working = pCLThis->head->next;
        freeCharLink(pCLThis->head);
        pCLThis->head = working;
    };
    free(pCLThis);
    return;
}
```

11

Stack ADT

Subsets of List ADT

12

Stack ADT: Public Interface

Declared in the .h file, but in general functions can be

- Creation
- Read
 - ~~Search~~
 - Report
- Update
 - ~~Add an object at some location~~
 - Add an object (PUSH)
 - ~~Remove an object from some location~~
 - Remove an object (POP)
- Deletion
 - Including all objects

13

Stack ADT: Private Implementation

Differences from a list ADT

- Data structure
- Creation
- Adding links
- Removing links
- Deletion

14

Stack ADT: Data Structure

```
typedef struct charstack {
    pCharLink head;
}charStack, *pCharStack;
```

15

Stack ADT: Creation

- Need to declare at least a head pointer to the first link

```
/*
 * Create an empty List
 */
pCharList mallocCharList(){
    pCharList pCLThis = (pCharList)malloc(sizeof(charList));
    if(pCLThis == (pCharList)NULL )return pCLThis;
    pCLThis->head = (pCharLink)NULL;

    return pCLThis;
}
```

16

Stack ADT: Updating

- We will add to/remove from the head (easiest)
- Three cases for addition/removal
 - First position (need to involve head pointer)
 - ~~Last position (the next pointer maybe NULL)~~
 - ~~In between (the general case)~~

```

/*
 * Delete an object from a list
 */
int popFromCharList(pCharList pCLThis){
    /* really just get the value from position 1,
     * then remove the link at position 1.
     */
    pCharLink temp;
    if(pCLThis->head == (pCLThis)NULL) return EOF;
    return getPayloadCharLink(pCLThis->head);
    temp = pCLThis->head->next;
    freeCharLink(pCLThis->head);
    pCLThis->head = temp;
    return 0;
}

```