

# Enhancing ADTs

CS2263 – Systems Software Development

*Tonight Is A Wonderful Time To Fall In Love, April Wine (Stand Back, 1974) <https://www.youtube.com/watch?v=8HUKtmAULJI>*

1

## Learning Outcomes

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

- Describe what an ADT would look like if the payload details were completely divorced from the ADT.
- Discuss the utility (pros and cons) of an ADT structured like this
- Write the function prototypes to implement this idea for a Stack ADT

2

## References

- Lu, Yung-Hsiang. 2015. Intermediate C Programming. CRC Press. New York. (Chapter 9.2)
- See L25Resources

3

## How Generic are Our ADTs?

### So far

- Modularize code
  - Thematically (e.g. `utils.h`)
  - Encapsulate data and functionality (e.g. `Strings.h`)
  - Abstract both representation and functionality (e.g. ADTs)
- What if we could implement generic ADTs?
  - Data type independence
  - C++ templates
  - Java Generic Types (e.g. `list<T>`)

4

## A Generic Node, List

```

12  typedef struct adtnode {
13      void* payload;
14      struct adtnode* next;
15  } ADTNode, *pADTNode;
16
17  /* The main event: The list */
18  typedef struct adtlist {
19      pADTNode head;
20      pADTNode tail;
21      int nLinks;
22  }ADTList, *pADTList;

```

5

## Functionality

- Create
- Read
  - get
  - print
  - find
- Update
  - insert
  - remove
- Delete

List-only actions:

- Create
- What about:
- Read
    - get: should return a *copy*
    - print: how/what to *print*
    - find: determining *equality*?
  - Update
    - insert: need to make a *copy*
    - remove: need to *free* the payload
  - Delete
    - remove: need to *free* the payload

*Need for data type-specific helper functions*

- Where have we seen this before?

qsort()

6

```

pADTList mallocADTList();
void freeADTList(pADTList pThis, void(*freeObj)(const void*));
int addToADTList(pADTList pThis, void* pv, int iPos, void(*dupObject)(const void* pObj));
int delFromADTList(pADTList pThis, int iPos, void(*freeObj)(const void* pObj));
int findInADTList(pADTList pThis, void* pv, int(*compareObj)(const void* pObj1, const void* pObj2));
void* getFromADTList(pADTList pThis, int iPos, void(*dupObj)(const void* pObj));
int fprintfCharList(FILE* pFIn, pADTList pThis, int(*fprintfObj)(const void* pObj));

```

## Generic Functionality

7

## Sample Function

```

/*
 * Free List, including all links/nodes
 */
void freeADTList(pADTList pThis, void(*freeObj)(const void*)) {
    if(pThis == (pADTList)NULL) return; //guard
    /* free all links */
    pADTNode working = pThis->head;
    while(pThis->head != (pADTNode)NULL){
        working = pThis->head->next;
        (*freeObj)(pThis->head);
        pThis->head = working;
    };
    free(pThis);
    return;
}

```

8