CSCA48 Tutorial 5 - ADTs, Lists, Memory Allocation

Tabeeb Yeamin, github.com/tabeebyeamin

February 13, 2020

Agenda

- Abstract Data Types Review
- Lists and Linked Lists
- Allocating Memory on Demand
- Exercise: Building a Dynamic Array

Abstract data types

- A set of *objects* together with a set of *operations*
- Provides a general description of how a collection is structured
- Describes the operations that will be supported
- Data structures are implementations of ADTs

Collections and Lists - the List ADT

- List is a sequential collection
- Order is not relevant (can have ordered or unordered lists)
- Supports: Insertion, Deletion, Search (and possibly Update)

The Linked List

```
struct Node {
    int data;
    struct Node* next;
};
```

Allocating memory on demand

```
calloc( # of items , size of each item in bytes)
```

- Finds available place in memory that has the request capacity, reserves it, wipes it out and returns a pointer to the reserved chunk.

```
Review_Node *new_review =
(Review_Node *)calloc(1, sizeof(Review_Node));
```

- calloc returns pointer without any attached data type so you have to cast it.
- always release (**free**) all the memory you allocated, or you could have memory leaks.

But What About malloc()?

```
Review_Node *new_review =
(Review_Node *) malloc (1*sizeof(Review_Node));
```

- There's no difference in the size of the memory block allocated.
- calloc() zero-initializes the buffer, while malloc() leaves the memory uninitialized.
- calloc() = malloc() + memset(ptr, 0, size)
- malloc() is actually faster than calloc since it doesn't initialize
- use calloc() if you want to initialize allocated memory to 0 by default

Discussion of Calloc vs Malloc:
https://stackoverflow.com/questions/1538420/difference-between-malloc-and-calloc

Exercise: Building a Dynamic Array

- Find the starter code on Piazza @312
- Complete the function that initializes the array to a given capacity - think about calloc()
- Complete the function that reads review information from console (using scanf() and fgets() - if you need help check the Unit 3 notes)
- Complete a function that inserts a review into the array, and if the array is full, resizes the array to double the size. Resizing means requesting a new array of twice the size, and copying data over, then deleting the old array