Day20 A

Young W. Lim

2017-12-05 Tue

Outline

1. Based on

2. Linked List Data Structure
    - Data Structure
    - Dynamic Memory Allocation
    - Linked Lists

Based on

"C How to Program",
Paul Deitel and Harvey Deitel

Types of Data Structures

- Linked Lists
  - collection of data items linked together
  - insertion and deletion can be performed anywhere in a linked list
- Stacks
  - insertion and deletion can be performed only at the top
- Queues
  - insertion is performed one end of a queue (back, tail)
  - deletion is performed antother end of a queue (front, head)
- Binary Trees
  - efficient search, sorting, eliminaton of duplicate items
  - used for file system directories and compilers

Self-Referential Structures

- contains a pointer member that points to a sturcure of the same type
- can be linked together to form lists, queues, stack and trees
- the pointer member represented as an arrow
  in the figures of these data structures
- A NULL pointer normally indicates the end of a data structure

Dynamic Data Structures

- dynamic data structures grow and shrink at execution time
  - data items in link lists, stacks, queues and binary trees are increasing and decreasing during execution time
- dynamic memory allocation is used
  - `malloc`
  - `calloc`
  - `realloc`
  - `free`

Dynamic Memory Allocation : malloc

- receives the number of bytes to be allocated
- returns a `void *` pointer to the allocated memory
- this `void *` pointer is assigned to a pointer variable of any data type
    - pointer type casting : `(int *)`, `(double *)`
- `int *p`
- `p = malloc( 10 * sizeof(int) );`
- `p = (int *) malloc( 10 * sizeof(int) );`

Dynamic Memory Allocation : calloc, ralloc, free

- the allocated memory
  - not *initialized* : `malloc`
  - zero *initialized* : `calloc` (c for clear)
  - can be *resized* : `realloc` (shrink, grow)
- when an error happens during allocation,
  all these functions return NULL
- `free` deallocates memory
  so that the memory can be reused in the future

Linked Lists (1)

- a linear collection of
  self-referenced structures (called node)
  connected by pointer links
- accessed via a pointer to the first node
- subsequent nodes are accessed via the link pointer member
- the link pointer member of the last node is set to NULL

Linked Lists (2)

- data is stored in a linked list dynamically
    - the length of a list can increase and decrease as necessary
- each node is created as necessary
- a node can contain data of any type
  including other structure object
- normally not stored contiguously in memory
- logically, however, the nodes of a linked list appear to be contiguous