FREE eBook

LEARNING linked-list

Free unaffiliated eBook created from
Stack Overflow contributors.

#linked-list

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: linked-list

It is an unofficial and free linked-list ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official linked-list.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with linked-list

Remarks

This section provides an overview of what linked-list is, and why a developer might want to use it.

It should also mention any large subjects within linked-list, and link out to the related topics. Since the Documentation for linked-list is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

Detailed instructions on getting linked-list set up or installed.

Design using Sentry Node

When designing a linked list, you can avoid all the special-cases (empty list, first node, last node, etc) by using a sentry node. Let's see how that is done:

```
struct Node
{
    Node* next;
    Node* prev;
    T data;
};

// helper function to link 2 nodes
void Link(Node* n1, Node* n2)
{
    n1->next = n2;
    n2->prev = n1;
}

// this inserts new data before 'here'
Node* Insert(Node* here, const T& data)
{
    Node* item = new Node{0,0,data};  // create new item. use T's copy-constructor
    Link(here->prev, item);           // link in new node. item comes before here,
    Link(item, here);                 // so in-between `here->prev´ and `here´
    size += 1;                        // update size
    return item;
}

// erase one item
Node* Erase(Node* here)
{
    Node* nxt = here->next;           // save next item for return value
    Link(here->prev, here->next);     // unlink item. no special cases needed when using
sentry
    delete here;                      // delete item. this will call T's destructor
```

```
    size -= 1;                          // update size
    return nxt;
}
```

This looks like it would fail for en empty list for example, but with a sentry node the list is never truly empty, it always contain the sentry node, that link to itself if there is no data-nodes. The sentry node also double as the one past last marker.

```
Node* sentry;
void Init()
{
    sentry = (Node*)your_preferred_allocator();
    Link(sentry, sentry);
    size = 0;
}
```

A more comprehensive tutorial can be found at https://pastebin.com/DXunz58Q

Read Getting started with linked-list online: https://riptutorial.com/linked-list/topic/9811/getting-started-with-linked-list

Credits

S. No	Chapters	Contributors
1	Getting started with linked-list	Community, sp2danny