Memory Allocation in C

Tom Kelliher, CS 320

Jan. 31, 2003

1 Administrivia

Announcements

Assignment

Project 0 handout.

From Last Time

Exercise, structures, pointers.

Outline

1. Introduction.
2. Simple Examples: array, doubly-linked list.

Coming Up

Discussion of Chapter 1.

2 Introduction

1. `malloc()` — memory allocation.

   ```
   void *malloc(size_t size);
   ```

2. `free()` — memory de-allocation.

   ```
   void free(void *ptr)
   ```

Both require `#include <stdlib.h>`.

3 Small Examples

1. A dynamically sized `int` array:

   ```
   void createAndDestroy(int size)
   {
      int *data;
      int i;

      data = (int *) malloc(size * sizeof(int));

      if (data == NULL)
         return;

      for (i = 0; i < size; i++)
         data[i] = 0;

      free((void *) data);
   }
   ```

2. Dynamically allocating and accessing structures:

   ```
   typedef struct qnode {
     struct qnode *next;
     struct qnode *prev;
     int key;
   ```

```
  /* Additional or alternate fields here. */
} qnode;

qnode* head, temp, insert, delete;

/* Don't forget to check malloc's return value! */

head = (qnode*) malloc(sizeof(qnode));
head->next = head->prev = NULL;
head->key = 5;

temp = (qnode*) malloc(sizeof(qnode));
temp->key = 99;
head->next = head->prev = temp;
temp->next = temp->prev = head

temp = (qnode*) malloc(sizeof(qnode));
temp->key = 73;

insert = head;  /* New item inserted AFTER item pointed to by insert. */

temp->next = insert->next;
temp->prev = insert;
insert->next->prev = temp;
insert->next = temp;

delete = head->prev;

/* The following is incorrect, from a memory allocation point-of-view.
 * Why?  How can it be fixed?
 */

free((void*) delete);
delete->prev->next = delete->next;
delete->next->prev = delete->prev;
```