EECS 22: Advanced C Programming

Lecture 21

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 21: Overview

- Course Administration
  - Midterm exam
  - Midterm course evaluation
- Dynamic Data Structures
  - Double-linked list: Element insertion, deletion
  - Example: Student records
    - **`StudentList.h, StudentList.c`**
      - New **`InsertStudentBefore`**
      - New **`InsertStudentAfter`**
  - Graphical data structure display in **`ddd`**

Course Administration

- Midterm Exam Feedback
  - Results are finally in, scores posted
  - Part 1, multi-choice questions
    - Check-all-that-apply questions show more mistakes
    - Some of the fill-in-the-code questions appear to be harder
  - Part 2, programming on paper went well
    - Static vs. auto, local vs. global variable issues
    - Header files shall contain include guards
    - Header file inclusion must match Makefile dependencies
  - Overall, very positive results!

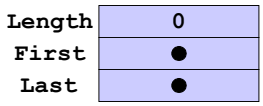
Course Administration

- Midterm Course Evaluation Feedback
  - Participation higher than usual
    - 94 out of 186 students (50.54%)
    - ➢ Thank you!
  - Overall result
    - Extremely positive, very encouraging!
      - *"Clear slides", "well-structured", "helpful examples", …*
  - Specific suggestions for improvement
    - Split labs to 3 times a week, create a discussion section
    - TA's office hour
    - Thursday due date, Friday labs *"makes no sense"*
    - More actual coding and debugging in class.
    - Better communication with the TA's
  - Grades match my expectation for the class! ;-)
    - 63 A, 23 A-, 6 B+, 1 C+, 1 Not Applicable

Dynamic Data Structures

- Double-Linked List (with header)
  - Example: Sorted List of Records

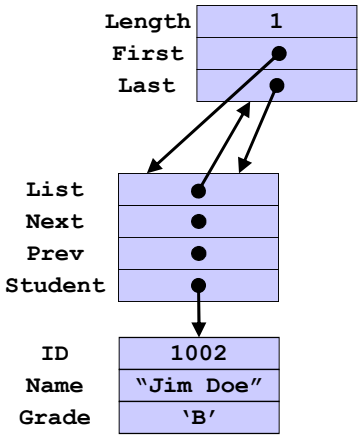
1. Empty list



Dynamic Data Structures

- Double-Linked List (with header)
  - Example: Sorted List of Records

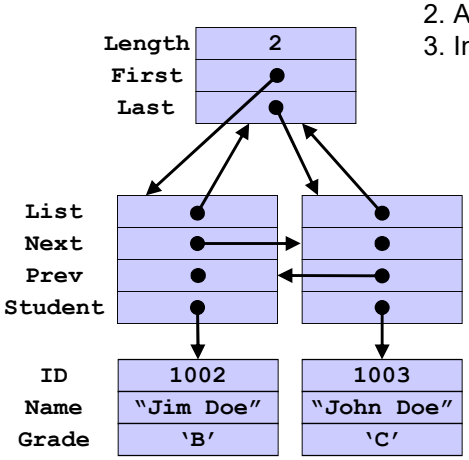
1. Empty list
2. Append a record



Dynamic Data Structures

- Double-Linked List (with header)
  - Example: Sorted List of Records

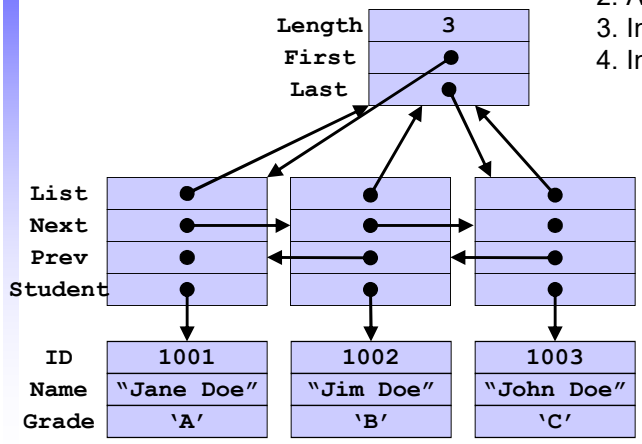
1. Empty list
2. Append a record
3. Insert after a record



Dynamic Data Structures

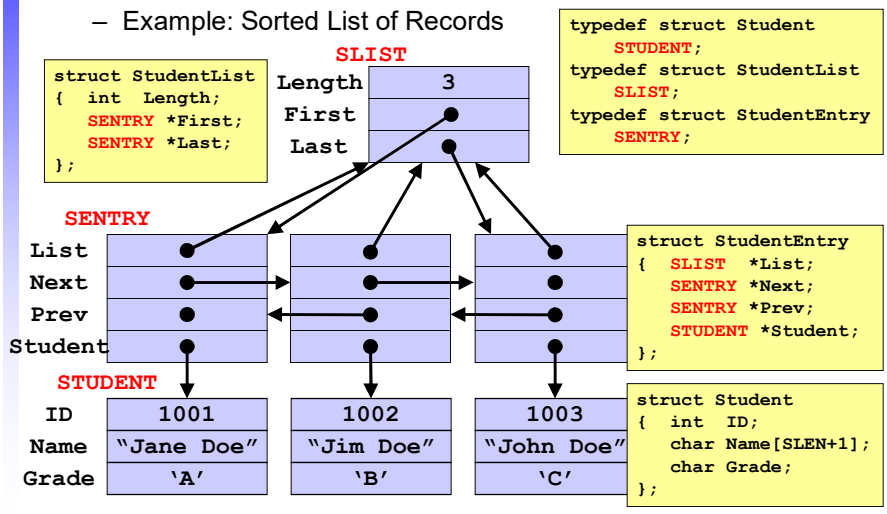
- Double-Linked List (with header)
  - Example: Sorted List of Records

1. Empty list
2. Append a record
3. Insert after a record
4. Insert before a record



Dynamic Data Structures

- Double-Linked List (with header)
  - Example: Sorted List of Records



Dynamic Data Structures

- Example **`StudentList.h`** (part 1/2)

```
/* StudentList.h: header file for lists of student records      */

#ifndef STUDENT_LIST_H
#define STUDENT_LIST_H

#include "Student.h"

typedef struct StudentList SLIST;
typedef struct StudentEntry SENTRY;

struct StudentList
{  int  Length;
   SENTRY *First;
   SENTRY *Last;
};

struct StudentEntry
{  SLIST  *List;
   SENTRY *Next;
   SENTRY *Prev;
   STUDENT *Student;
};

...
```

Dynamic Data Structures

- Example **`StudentList.h`** (part 2/2)

```
...
/* allocate a new student list */
SLIST *NewStudentList(void);

/* delete a student list (and all entries) */
void DeleteStudentList(SLIST *l);

/* prepend/append a student at beginning/end of list */
void PrependStudent(SLIST *l, STUDENT *s);
void AppendStudent(SLIST *l, STUDENT *s);

/* insert a student before/after an existing one */
void InsertStudentBefore(SENTRY *e, STUDENT *s);
void InsertStudentAfter(SENTRY *e, STUDENT *s);

/* remove the first/last student from the list */
STUDENT *RemoveFirstStudent(SLIST *l);
STUDENT *RemoveLastStudent(SLIST *l);

/* print a student list */
void PrintStudentList(SLIST *l);

#endif /* STUDENT_LIST_H */

/* EOF */
```


Dynamic Data Structures

- Example **`StudentList.c`** (part 1/4)

```
/* StudentList.c: maintaining lists of student records  */
/* author: Rainer Doemer                                */
/* modifications:                                       */
/* 11/10/11 RD added InsertStudentBefore, InsertStudentAfter */
/* 11/08/11 RD version for double-linked lists          */
/* 11/03/11 RD initial version                          */

#include "StudentList.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <assert.h>

/* unmodified functions omitted here for brevity */

...
```

Dynamic Data Structures

- Example **`StudentList.c`** (part 2/4)

```
...

/* insert a student before an existing one */
void InsertStudentBefore(SENTRY *e, STUDENT *s)
{
    SENTRY *New;
    New = NewStudentEntry(s);
    New->List = e->List;
    New->Next = e;
    New->Prev = e->Prev;
    e->Prev = New;
    if (New->Prev)
      { New->Prev->Next = New;
      }
    else
      { assert(New->List->First == e);
        New->List->First = New;
      }
    New->List->Length++;
} /* end of InsertStudentBefore */

...
```


Dynamic Data Structures

- Example **`StudentList.c`** (part 3/4)

```
...

/* insert a student after an existing one */
void InsertStudentAfter(SENTRY *e, STUDENT *s)
{
    SENTRY *New;
    New = NewStudentEntry(s);
    New->List = e->List;
    New->Next = e->Next;
    New->Prev = e;
    e->Next = New;
    if (New->Next)
      { New->Next->Prev = New;
      }
    else
      { assert(New->List->Last == e);
        New->List->Last = New;
      }
    New->List->Length++;
} /* end of InsertStudentAfter */

...
```

Dynamic Data Structures

- Example **`StudentList.c`** (part 4/4)

```
#ifdef MAIN
int main(void)
STUDENT *s = NULL;
    SLIST *l = NULL;
    SENTRY *e = NULL;
    l = NewStudentList();
    s = NewStudent(1002, "Jim Doe", 'B');
    AppendStudent(l, s);
    e = l->First;
    assert(e->Student == s);
    s = NewStudent(1003, "John Doe", 'C');
    InsertStudentAfter(e, s);
    s = NewStudent(1001, "Jane Doe", 'A');
    InsertStudentBefore(e, s);

    PrintStudentList(l);

    DeleteStudentList(l);
    l = NULL;
    return 0;
} /* end of main */
#endif /* MAIN */
/* EOF */
```


Dynamic Data Structures

- Example Session

```
% vi StudentList.c
% vi Makefile
% make
gcc -Wall -ansi -g -c Student.c -o Student.o
gcc -DMAIN -Wall -ansi -g StudentList.c Student.o -o StudentList
% valgrind ./StudentList
==5908== Memcheck, a memory error detector
Student ID:    1001
Student Name:  Jane Doe
Student Grade: A
Student ID:    1002
Student Name:  Jim Doe
Student Grade: B
Student ID:    1003
Student Name:  John Doe
Student Grade: C
==5908== HEAP SUMMARY:
==5908==     in use at exit: 0 bytes in 0 blocks
==5908==  total heap usage: 7 allocs, 7 frees, 264 bytes allocated
==5908== All heap blocks were freed -- no leaks are possible
==5908== ERROR SUMMARY: 0 errors from 0 contexts
%
```

Graphical Data Structure Display

- Data Display Debugger **`ddd`**
  - Graphical frontend for GDB
    - Requires *X* forwarding client (e.g. *Xming* in addition to *Putty*)
  - Displays separate windows
    - Menu bar and command buttons
    - ➢ Graphical display area for *pointers* and *data structures*!
    - Source code (and assembly code) browser
    - Command line interface
  - ➢ Can display data structures built by pointers as graphs
    - Very useful tool to visualize and debug dynamic data structures
  - Example: **`StudentList.c`**

```
% vi StudentList.c
% make StudentList
gcc -Wall -ansi -g -c Student.c -o Student.o
gcc -DMAIN -Wall -ansi -g StudentList.c Student.o -o StudentList
% ddd ./StudentList
```


Graphical Data Structure Display