

# EECS 22: Advanced C Programming

## Lecture 20

Rainer Dömer

doemer@uci.edu

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

## Lecture 20: Overview

- Dynamic Data Structures
  - Double-linked List
  - Example: List of student records (continued)
    - `Student.h`
    - `Student.c`
    - `StudentList.h`
    - `StudentList.c`
    - `Makefile`

## Dynamic Data Structures

- Double-Linked List (with header)
    - Example: List of Student Records
1. Empty list

Length	0
First	●
Last	●

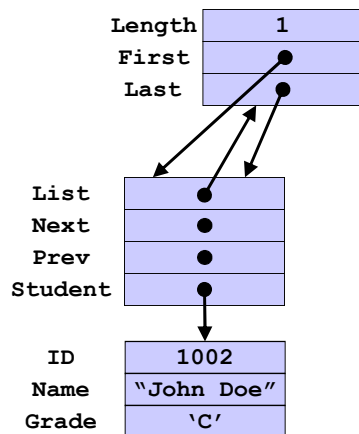
EECS22: Advanced C Programming, Lecture 20

(c) 2016 R. Doemer

3

## Dynamic Data Structures

- Double-Linked List (with header)
    - Example: List of Student Records
1. Empty list
  2. Add a student



EECS22: Advanced C Programming, Lecture 20

(c) 2016 R. Doemer

4

## Dynamic Data Structures

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

1. Empty list
2. Add a student
3. Append a student

EECS22: Advanced C Programming, Lecture 20
(c) 2016 R. Doemer
5

## Dynamic Data Structures

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

1. Empty list
2. Add a student
3. Append a student
4. Prepend a student

EECS22: Advanced C Programming, Lecture 20
(c) 2016 R. Doemer
6

## Dynamic Data Structures

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

ID	1001	1002	1003	...	1042
Name	"Jane Doe"	"John Doe"	"Jim Doe"	...	"Z End"
Grade	'A'	'C'	'B'	...	'A'

1. Empty list
2. Add a student
3. Append a student
4. Prepend a student
5. ...

EECS22: Advanced C Programming, Lecture 20
(c) 2016 R. Doemer
7

## Dynamic Data Structures

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

```

typedef struct Student
    STUDENT;
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;
};

struct Student
{
    int ID;
    char Name[SLEN+1];
    char Grade;
};
                
```

EECS22: Advanced C Programming, Lecture 20
(c) 2016 R. Doemer
8

## Dynamic Data Structures

- Example `Student.h`

```

/* Student.h: header file for student records */
#ifndef STUDENT_H
#define STUDENT_H

#define SLEN 40

struct Student
{
    int ID;
    char Name[SLEN+1];
    char Grade;
};
typedef struct Student STUDENT;

/* allocate a new student record */
STUDENT *NewStudent(int ID, char *Name, char Grade);

/* delete a student record */
void DeleteStudent(STUDENT *s);

/* print a student record */
void PrintStudent(STUDENT *s);

#endif /* STUDENT_H */

```

## 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);

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

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

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

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

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

#endif /* STUDENT_LIST_H */

/* EOF */

```

EECS22: Advanced C Programming, Lecture 20

(c) 2016 R. Doemer

11

## Dynamic Data Structures

- Example `StudentList.c` (part 1/9)

```

/* StudentList.c: maintaining lists of student records */
#include "StudentList.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <assert.h>

/* allocate a new student entry */
static SENTRY *NewStudentEntry(STUDENT *s)
{
    SENTRY *e;
    e = malloc(sizeof(SENTRY));
    if (! e)
        { perror("Out of memory! Aborting...");
          exit(10);
        } /* fi */
    e->List = NULL;
    e->Next = NULL;
    e->Prev = NULL;
    e->Student = s;
    return e;
} /* end of NewStudentEntry */

...

```

EECS22: Advanced C Programming, Lecture 20

(c) 2016 R. Doemer

12

## Dynamic Data Structures

- Example `StudentList.c` (part 2/9)

```

.../* delete a student entry */
static STUDENT *DeleteStudentEntry(SENTRY *e)
{
    STUDENT *s;
    assert(e);
    s = e->Student;
    free(e);
    return s;
} /* end of DeleteStudentEntry */

/* allocate a new student list */
SLIST *NewStudentList(void)
{
    SLIST *l;
    l = malloc(sizeof(SLIST));
    if (!l)
        { perror("Out of memory! Aborting...");
          exit(10);
        } /* fi */
    l->Length = 0;
    l->First = NULL;
    l->Last = NULL;
    return l;
} /* end of NewStudentList */
...

```

EECS22: Advanced C Programming, Lecture 20

(c) 2016 R. Doemer

13

## Dynamic Data Structures

- Example `StudentList.c` (part 3/9)

```

...
/* delete a student list (and all entries) */
void DeleteStudentList(SLIST *l)
{
    SENTRY *e, *n;
    STUDENT *s;

    assert(l);
    e = l->First;
    while(e)
        { n = e->Next;
          s = DeleteStudentEntry(e);
          DeleteStudent(s);
          e = n;
        }
    free(l);
} /* end of DeleteStudentList */
...

```

EECS22: Advanced C Programming, Lecture 20

(c) 2016 R. Doemer

14

## Dynamic Data Structures

- Example `StudentList.c` (part 4/9)

```

.../* append a student at end of list */
void AppendStudent(SLIST *l, STUDENT *s)
{
    SENTRY *e = NULL;
    assert(l);
    assert(s);
    e = NewStudentEntry(s);
    if (l->Last)
    {
        e->List = l;
        e->Next = NULL;
        e->Prev = l->Last;
        l->Last->Next = e;
        l->Last = e;
    }
    else
    {
        e->List = l;
        e->Next = NULL;
        e->Prev = NULL;
        l->First = e;
        l->Last = e;
    }
    l->Length++;
} /* end of AppendStudent */

```

EECS ...

## Dynamic Data Structures

- Example `StudentList.c` (part 5/9)

```

.../* prepend a student at beginning of list */
void PrependStudent(SLIST *l, STUDENT *s)
{
    SENTRY *e = NULL;
    assert(l);
    assert(s);
    e = NewStudentEntry(s);
    if (l->First)
    {
        e->List = l;
        e->Next = l->First;
        e->Prev = NULL;
        l->First->Prev = e;
        l->First = e;
    }
    else
    {
        e->List = l;
        e->Next = NULL;
        e->Prev = NULL;
        l->First = e;
        l->Last = e;
    }
    l->Length++;
} /* end of PrependStudent */

```

EECS ...



## Dynamic Data Structures

- Example `StudentList.c` (part 6/9)

```

.../* remove the first student from the list */
STUDENT *RemoveFirstStudent(SLIST *l)
{
    SENTRY *e = NULL;
    assert(l);
    if (l->First)
    {
        e = l->First;
        l->First = e->Next;
        if (l->First)
            { l->First->Prev = NULL;
            }
        else
            { assert(l->Last == e);
            l->Last = NULL;
            }
        l->Length--;
        return DeleteStudentEntry(e);
    }
    else
    { return(NULL);
    }
} /* end of RemoveFirstStudent */
...

```

EECS22: Advanced C Programming, Lecture 20

(c) 2016 R. Doemer

17

## Dynamic Data Structures

- Example `StudentList.c` (part 7/9)

```

.../* remove the last student from the list */
STUDENT *RemoveLastStudent(SLIST *l)
{
    SENTRY *e = NULL;
    assert(l);
    if (l->Last)
    {
        e = l->Last;
        l->Last = e->Prev;
        if (l->Last)
            { l->Last->Next = NULL;
            }
        else
            { assert(l->First == e);
            l->First = NULL;
            }
        l->Length--;
        return DeleteStudentEntry(e);
    }
    else
    { return(NULL);
    }
} /* end of RemoveLastStudent */
...

```

EECS22: Advanced C Programming, Lecture 20

(c) 2016 R. Doemer

18

## Dynamic Data Structures

- Example `StudentList.c` (part 8/9)

```

...

/* print a student list */
void PrintStudentList(SLIST *l)
{
    SENTRY *e;
    assert(l);
    e = l->First;
    while(e)
    { PrintStudent(e->Student);
      e = e->Next;
    }
} /* end of PrintStudentList */

...

```

## Dynamic Data Structures

- Example `StudentList.c` (part 9/9)

```

#ifdef MAIN
int main(void)
{
    STUDENT *s = NULL;
    SLIST *l = NULL;
    l = NewStudentList();
    s = NewStudent(1001, "Jane Doe", 'A');
    AppendStudent(l, s);
    s = NewStudent(1002, "John Doe", 'C');
    AppendStudent(l, s);
    s = NewStudent(1000, "New Kid", 'F');
    PrependStudent(l, s);
    PrintStudentList(l);
    s = RemoveFirstStudent(l);
    AppendStudent(l, s);
    s = RemoveLastStudent(l);
    PrependStudent(l, s);
    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: 1000
Student Name: New Kid
Student Grade: F
Student ID: 1001
Student Name: Jane Doe
Student Grade: A
Student ID: 1002
Student Name: John Doe
Student Grade: C
==5908== HEAP SUMMARY:
==5908==   in use at exit: 0 bytes in 0 blocks
==5908== total heap usage: 9 allocs, 9 frees, 328 bytes allocated
==5908== All heap blocks were freed -- no leaks are possible
==5908== ERROR SUMMARY: 0 errors from 0 contexts
%
```