# EECS 10: Computational Methods in Electrical and Computer Engineering 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

- Course Administration
  - Final course evaluation
- Data Structures
  - Structures
    - · Declaration and definition
    - Instantiation and initialization
    - Member access
  - Unions
    - · Declaration and definition
    - Member access
  - Enumerators
    - · Declaration and definition
  - Type definitions

EECS10: Computational Methods in ECE, Lecture 20

(c) 2004 R. Doemer

2

### Course Administration

- Final Course Evaluation
  - Starting this week, until 10th week
  - Nov. 17, 2004, 8am through Dec. 10, 2004, 5pm
  - Online via EEE Evaluation application
- Feedback from students to instructors
  - Completely voluntary
  - Completely anonymous
  - Very valuable
    - · Help to improve this class!
- · Evaluation asks for current grade estimate
  - Suggest to wait after midterm 2 scores are in

EECS10: Computational Methods in ECE, Lecture 20

(c) 2004 R. Doemer

3

### **Data Structures**

- Structures (aka. records)
  - User-defined, composite data type
    - Type is a composition of (different) sub-types
  - Fixed set of members
    - Names and types of members are fixed at structure definition
  - Member access by name
    - Member-access operator: structure\_name.member\_name
- Example:

EECS10: Computational Methods in ECE, Lecture 20

(c) 2004 R. Doemer

4

- Structure Declaration
  - Declaration of a user-defined data type
- Example:

### **Data Structures**

- Structure Declaration
  - Declaration of a user-defined data type
- Structure Definition
  - Definition of structure members and their type
- Example:

EECS10: Computational Methods in ECE, Lecture 20

(c) 2004 R. Doemer

6

- Structure Declaration
  - Declaration of a user-defined data type
- Structure Definition
  - Definition of structure members and their type
- Structure Instantiation
  - Definition of a variable of structure type
- Example:

EECS10: Computational Methods in ECE, Lecture 20

(c) 2004 R. Doemer

7

# **Data Structures**

- Structure Declaration
  - Declaration of a user-defined data type
- Structure Definition
  - Definition of structure members and their type
- Structure Instantiation and Initialization
  - Definition of a variable of structure type
  - Initializer list defines initial values of members
- Example:

EECS10: Computational Methods in ECE, Lecture 20

(c) 2004 R. Doemer

8

4

### **Data Structures** Structure Access - Members are accessed by their name - Member-access operator . Example: struct Student Jane int ID; char Name[40]; 1001 ID char Grade; "Jane Doe" Name **`A'** Grade struct Student Jane = {1001, "Jane Doe", 'A'}; void PrintStudent(struct Student s) ID: 1001 printf("ID: $d\n"$ , s.ID); printf("Name: %s\n", s.Name); Name: Jane Doe printf("Grade: %c\n", s.Grade); Grade: A EECS10: Computational Methods in ECE, Lecture 20 (c) 2004 R. Doemer

### **Data Structures**

- Unions
  - User-defined, composite data type
    - Type is a composition of (different) sub-types
  - Fixed set of mutually exclusive members
    - · Names and types of members are fixed at union definition
  - Member access by name
    - Member-access operator: union\_name.member\_name
  - Only one member may be used at a time!
    - All members share the same location in memory!
- Example:

```
union U { int i; float f;} u1, u2;
u1.i = 42;    /* access to members */
u2.f = 3.1415;
u1.f = u2.f;    /* destroys u1.i! */
```

EECS10: Computational Methods in ECE, Lecture 20

(c) 2004 R. Doemer

10

- Union Declaration
  - Declaration of a user-defined data type
- Example:

```
union HeightOfTriangle; /* declaration */
```

EECS10: Computational Methods in ECE, Lecture 20

(c) 2004 R. Doemer

11

### **Data Structures**

- Union Declaration
  - Declaration of a user-defined data type
- Union Definition
  - Definition of union members and their type
- Example:

EECS10: Computational Methods in ECE, Lecture 20

(c) 2004 R. Doemer

12

6

- Union Declaration
  - Declaration of a user-defined data type
- Union Definition
  - Definition of union members and their type
- Union Instantiation
  - Definition of a variable of union type
- Example:

```
union HeightOfTriangle; /* declaration */
union HeightOfTriangle /* definition */
{ int Height; /* members */
   int LengthOfSideA;
   float AngleBeta;
};
union HeightOfTriangle H;/* instantiation */
```

EECS10: Computational Methods in ECE, Lecture 20

(c) 2004 R. Doemer

13

## **Data Structures**

- Union Declaration
  - Declaration of a user-defined data type
- Union Definition
  - Definition of union members and their type
- Union Instantiation and Initialization
  - Definition of a variable of union type
  - Single initializer defines value of first member
- Example:

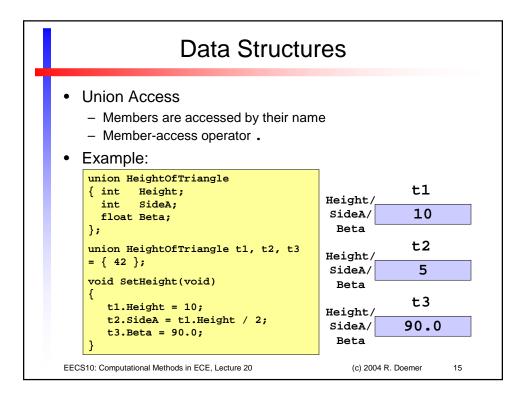
```
union HeightOfTriangle; /* declaration */
union HeightOfTriangle /* definition */
{ int Height; /* members */
   int LengthOfSideA;
   float AngleBeta;
};
union HeightOfTriangle H /* instantiation */
= { 42 }; /* initialization */
```

EECS10: Computational Methods in ECE, Lecture 20

(c) 2004 R. Doemer

14

7



- Enumerators
  - User-defined data type
    - Members are an enumeration of integral constants
  - Fixed set of members
    - · Names and values of members are fixed at enumerator definition
  - Members are constants
    - Member values cannot be changed after definition
- Example:

```
enum E { red, yellow, green };
   enum E LightNS, LightEW;
   LightEW = green;
                                /* assignment */
   if (LightNS == green)
                               /* comparison */
       { LightEW = red; }
EECS10: Computational Methods in ECE, Lecture 20
                                            (c) 2004 R. Doemer
                                                            16
```

# Data Structures • Enumerator Declaration - Declaration of a user-defined data type • Example: enum Weekday; /\* declaration \*/ EECS10: Computational Methods in ECE, Lecture 20 (c) 2004 R. Doemer 17

### **Data Structures**

- Enumerator Declaration
  - Declaration of a user-defined data type
- Enumerator Definition
  - Definition of enumerator members and their value

Example:

EECS10: Computational Methods in ECE, Lecture 20

(c) 2004 R. Doemer

18

9

- Enumerator Declaration
  - Declaration of a user-defined data type
- Enumerator Definition
  - Definition of enumerator members and their value
- Enumerator Instantiation
  - Definition of a variable of enumerator type
- Example:

EECS10: Computational Methods in ECE, Lecture 20

(c) 2004 R. Doemer

19

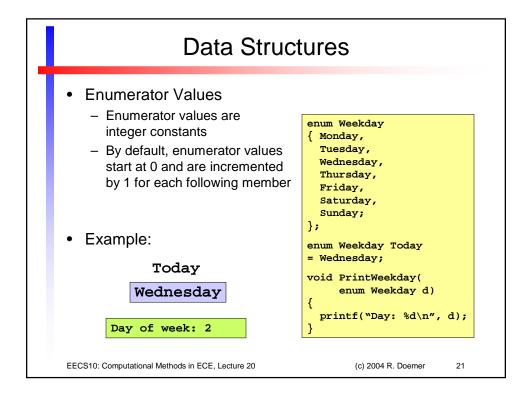
## **Data Structures**

- Enumerator Declaration
  - Declaration of a user-defined data type
- Enumerator Definition
  - Definition of enumerator members and their value
- Enumerator Instantiation and Initialization
  - Definition of a variable of enumerator type
  - Initializer should be one member of the enumerator
- Example:

EECS10: Computational Methods in ECE, Lecture 20

(c) 2004 R. Doemer

20



### **Data Structures Enumerator Values** - Enumerator values are enum Weekday integer constants { Monday = 1, By default, enumerator values Tuesday, Wednesday, start at 0 and are incremented Thursday, by 1 for each following member Friday, Saturday, Specific enumerator values Sunday; may be defined by the user Example: enum Weekday Today = Wednesday; Today void PrintWeekday( Wednesday enum Weekday d) printf("Day: %d\n", d); Day of week: 3 EECS10: Computational Methods in ECE, Lecture 20 (c) 2004 R. Doemer 22

### **Data Structures Enumerator Values** - Enumerator values are enum Weekday integer constants $\{ Monday = 2,$ Tuesday, By default, enumerator values Wednesday, start at 0 and are incremented Thursday, by 1 for each following member Friday, - Specific enumerator values Saturday, Sunday = 1; may be defined by the user Example: enum Weekday Today = Wednesday: Today void PrintWeekday( Wednesday enum Weekday d) printf("Day: %d\n", d); Day of week: 4 EECS10: Computational Methods in ECE, Lecture 20 (c) 2004 R. Doemer

### **Data Structures**

- Type definitions
  - A typedef can be defined as an alias type for another type
  - A typedef definition follows the same rules as a variable definition
  - Type definitions are usually used to abbreviate access to user-defined types

```
Examples:
    typedef long MyInteger;
    typedef enum Weekday Day;
    Day Today;
    typedef struct Student Scholar;
    Scholar Jane, John;
EECS10: Computational Methods in ECE, Lecture 20
                                                    (c) 2004 R. Doemer
                                                                       24
```