

CSE 3302
Programming Languages


Semantics (cont.)

 Chengkai Li
 Fall 2007



Lecture 3 - Syntax, Fall 2007 CSE3302 Programming Languages, UT-Arlington
 ©Chengkai Li, 2007


Overloading and Resolution



- Overloading: one name refers to multiple entities in scope.
- Overload Resolution: select one entity.
- Name itself isn't sufficient in resolution: need extra information (often data types)

Lecture 3 - Syntax, Fall 2007 CSE3302 Programming Languages, UT-Arlington
 ©Chengkai Li, 2007


Function/Method Overloading



- C: no overloading
- C++/Java/Ada: resolution by number and types of parameters.
 - Perfect if exact match exists;
 - No perfect match: different conversion rules
 - Ada: automatic conversions not allowed.
 - Java: conversions allowed in certain directions.
 - C++: automatic conversions more flexible.
 - e.g.,
 - `int sum(int a, int b) {...}`
 - `double sum(double a, double b) {...}`
 - `double sum(double a, int b) {...}`
- `sum(1); sum(1, 2); sum(1.0, 2.0); sum(1, 2.0);`
- Ada even uses return type for resolution

Lecture 3 - Syntax, Fall 2007 CSE3302 Programming Languages, UT-Arlington
 ©Chengkai Li, 2007


Environment: Names to Locations



- Location: one specific attribute of names
- Environment: binding names to locations
 - Conceptually part of symbol table.
 - But usually considered separately.
- Static vs. dynamic
 - FORTRAN: completely static
 - LISP: completely dynamic
 - Algol-descendants (C, C++, Ada, Java) : combination
 - global variables: static
 - local variables: dynamic

Lecture 3 - Syntax, Fall 2007 CSE3302 Programming Languages, UT-Arlington
 ©Chengkai Li, 2007

Environment: as Bindings



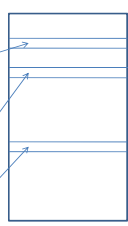
⊗

Ⓜ

integer, 1, global, address=0x1F560A


integer, 42, local to q, address=0x1D0980

character, 'a', global, address=0x1BC4F3



Lecture 3 - Syntax, Fall 2007 CSE3302 Programming Languages, UT-Arlington
 ©Chengkai Li, 2007

Environment: another view

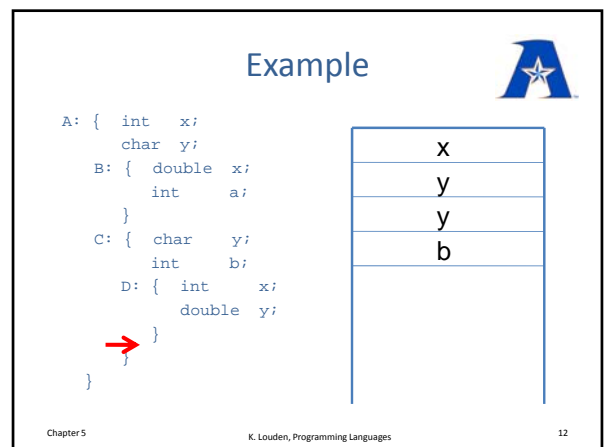
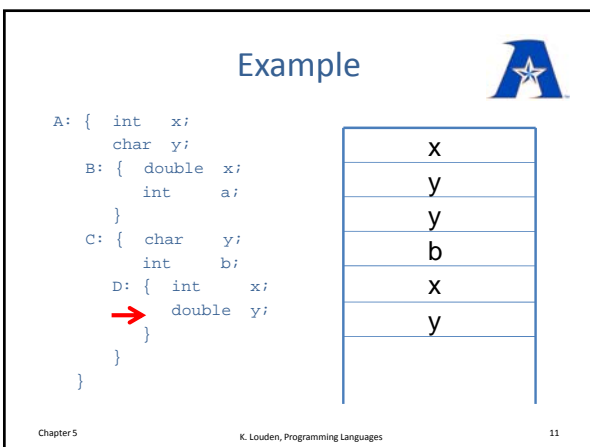
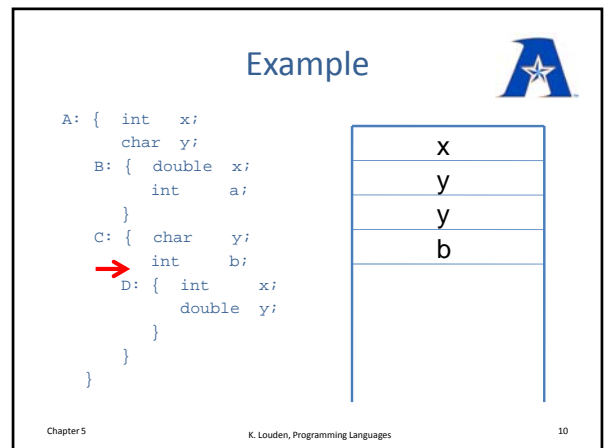
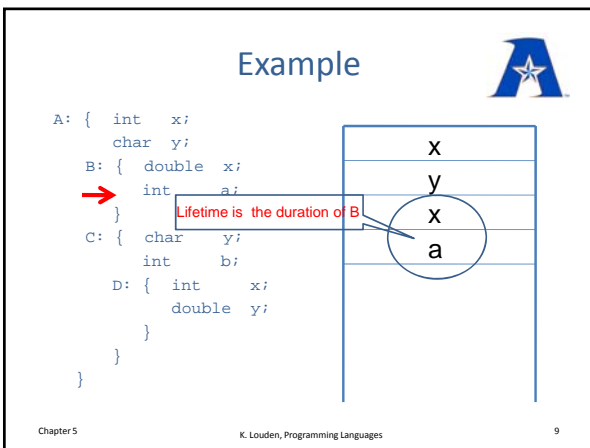
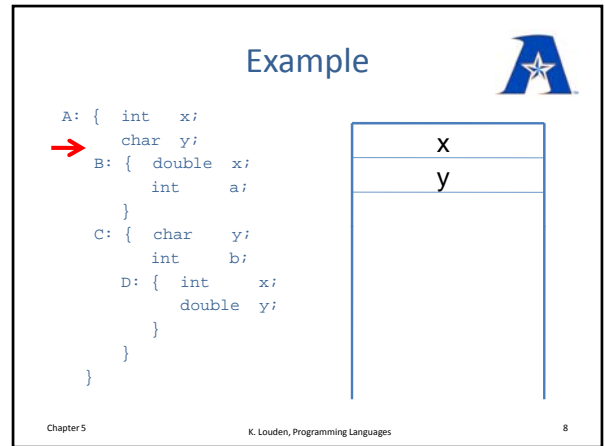
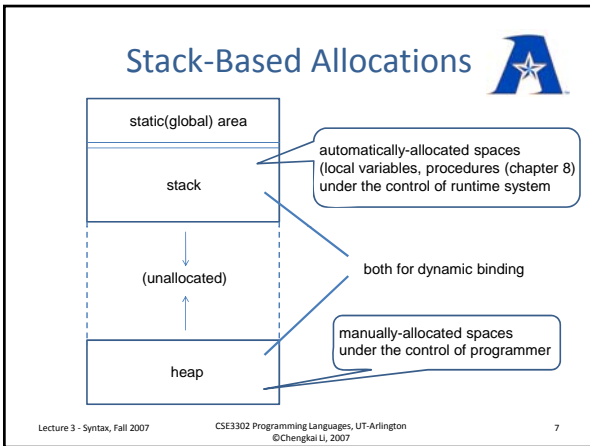



x

y

y

Lecture 3 - Syntax, Fall 2007 CSE3302 Programming Languages, UT-Arlington
 ©Chengkai Li, 2007






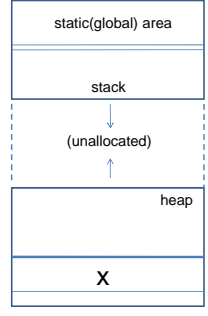
- What are the most difficult bugs that you have to deal with?

Lecture 3 - Syntax, Fall 2007 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, 2007 13

Heap-Based Allocation




- **C**
`int *x;
 x=(int *)malloc(sizeof(int));
 free(x);`
- **C++**
`int *x;
 x= new int;
 delete x;`
- **Java**
`Integer x = new Integer(2);
 //no delete
 //need garbage collection)`



Lecture 3 - Syntax, Fall 2007 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, 2007 14


Scope vs. Lifetime



- Lifetime beyond scope:
 - alive in scope hole
 - alive outside scope
- Scope beyond lifetime (unsafe)

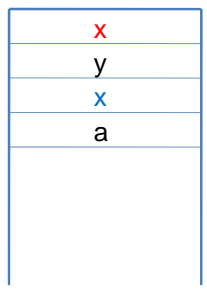
Lecture 3 - Syntax, Fall 2007 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, 2007 15

Example: Alive in scope hole




```

A: { int x;
    char y;
    B: { double x;
        int a;
    }
    C: { char y;
        int b;
        D: { int x;
            double y;
        }
    }
    }
```



Chapter 5 K. Louden, Programming Languages 16

Example: Alive outside scope



```

int func(void) {
    static int counter = 0;
    counter += 1;
    return counter;
}


main()
{
    int i;
    int x;
    for (i=0; i<10; i++) { x=func(); }
    printf("%d\n", x);
}

```

counter

Chapter 5 K. Louden, Programming Languages 17

Example: Scope beyond lifetime



Dangling pointer:

```

int *x, *y, *z;

x=(int *) malloc(sizeof(int));
y=x;
free(x);

z=(int *) malloc(sizeof(int));

printf("%d\n", *y);

```

Chapter 5 K. Louden, Programming Languages 18

Box-and-Circle Diagram for Variables

Name — **Other Attributes** — **Value**
Location

I-value **r-value**

x **y**

assignment **x = y**

Lecture 3 - Syntax, Fall 2007 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, 2007 19

Box-and-Circle Diagram for Variables

x **y**

Java:
 Student x = new Student("Amy");
 Student y = new Student("John");
 x.setAge(19);
 x = y;
 y.setAge(21);

assignment by sharing

Lecture 3 - Syntax, Fall 2007 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, 2007 20

Box-and-Circle Diagram for Variables

x **y**

assignment by cloning **x = y**

Lecture 3 - Syntax, Fall 2007 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, 2007 21

Problems with Pointers

- Alias that causes side-effects.


```
Student x = new Student("Amy");
Student y = new Student("John");
x.setAge(19);
x = y;
y.setAge(21);
```
- Dangling References:


```
int *x;
{ int y;
  y = 2;
  x=&y;
}
printf("%d\n", *x);
```

Lecture 3 - Syntax, Fall 2007 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, 2007 22