Outline

1. The double Data Type
2. Type Conversions
3. The CAST Node
4. Assignment
The **double** Data Type

- Our compiler will recognize two data types.
  - **int** - 4 bytes.
  - **double** - 8 bytes.
- Therefore, we will have four object types.
  - **int**.
  - pointer to **int**.
  - **double**.
  - pointer to **double**.
In certain contexts, an `int` must be converted to a `double`, or vice versa.
- Mixed-mode arithmetic.
- Mixed-mode assignments.

In the syntax tree, this will be represented by a `CAST` node.
For example, suppose \( a \) is an \texttt{int} and \( b \) is a \texttt{double}.
The syntax tree for \( a + b \) would be
The **CAST** Node

The **cast()** Function

```java
TreeNode cast(TreeNode e, int t);
```

Creates a **CAST** node that casts the expression `e` to the type `t`. The expression `e` is attached as the left subtree of the **CAST** node.

- In the `SemanticAction` class, we must add a `cast()` function that creates a **CAST** node.
The mod operator `%` should be applied only to `ints`.

If the C source program applies it to `doubles`, should we

- Cast the `doubles` as `ints` and issue a warning, or
- Print an error message?
The \texttt{cast()} Function

- The \texttt{cast()} function should first see whether the cast is necessary.
- If the type of the expression and the cast type are the same, then \texttt{cast()} should return the expression unchanged.
- Otherwise, it should construct a new \texttt{CAST} tree node with the expression as the left subtree.
Conversions

- Which methods in the `SemanticAction` class should call `cast()`?
- Exactly when should it be called?
- Should it ever be called directly from the CUP file?
Floating-Point Data Types

Robb T. Koether

The double Data Type

Type Conversions

The CAST Node

Assignment

Homework