Announcements

- HW6 due Fri at 11:55pm
- No new homework this week
  - Prepare for Midterm 2 next week
  - Review in recitation next Tue

Multiway Branches

(C)
switch (e) {
    case const1: stmt1;
    ...
    case constn: stmtn;
    [default: stmt;]
}

What are the semantics?

Multiway Branches

Java or C#?
switch (e) {
    case const1: stmt1;
    ...
    case constn: stmtn;
    [default: stmt;]
}

Java - allows fall through
but cases only
at top level

C# - no fall through
requires a "break"/"goto" at the end of each case.

Multiway Branches

Java or C#?
switch (e) {
    case const1: stmt1;
    ...
    case constn: stmtn;
    [default: stmt;]
}

Design choices and C's choice?
Multiway Branches

Discuss and come up with an example of where you might prefer to use a "switch" versus an "if" and/or vice versa

- switch if have a chain cases
- what are the possible values for an expression
- "nominal type"
- chimericness

Multiway Branches: SML

case e of
  l => ...
  2 => ...
  3 => ...
  _ => ...

Guarded Commands

if cond1 then ...
  t1
else cond2 then ...
  t2
else ...

for (let i = low; i <= high; i = i + step) { … } (C)

FOR i := low TO high BY step DO END

foreach (var in list) { … } (C#)

- What is different? Give an example or explain what you can do in C but not the others (pros and cons?)

Guarded Commands

IA-64 assembly languages

if cond1 then ...
  t1
else if cond2 then ...
  t2
else ...

for (let i = low; i <= high; i = i + step) { … } (C)

Loops

for (expr1; expr2; expr3) { … } (C)

FOR i := low TO high BY step DO END

foreach (var in list) { … } (C#)
About your classmates

- Burned old furniture in a bonfire at aunt's house
- Takes random hikes in the middle of the night
- Previously an English major
- 2nd in campus-wide air guitar contest (and won an electric guitar)
- Plays the drums
- DJed at Church nightclub

Subtyping

One-Slide Summary

- If T is a subtype of U then any expression of type T can be used in a context that expects a U; this is called subsumption.
- A conversion is a function that converts between types.
- Widening conversions convert between types respecting subsumption, while narrowing is counter to subsumption.

Is MYSTERY broken?

- We can't initialize a variable of subrange type. Why?

```plaintext
VAR x : [1 TO 10];
x := 1;
```

Subset Interpretation of Subtyping

- Types as a set of values

\[ T \subseteq U \]

"T is a subtype of U"
Subtyping

- We can view types as sets of values
- Subtyping is a relation between types induced by the subset relation between value sets

Intuition:
- If T is a subtype of U, then any expression with type T also has type U
- If T is a subtype of U, then any expression of type T can be used in a context that expects a U
- Subtyping is reflexive and transitive

Subsumption Rule

\[
\text{If } T <: U \text{ and } e : T, \text{ then } e : U
\]

How is subtyping useful?

```java
void f(U a) {
    ... 
    // some piece of code used on multiple types
}

b: T T <: U
f(b) ✓

c: U
f(c) ✓
```

Example of inclusion/subtyping polymorphism

```java
procedure plus (x: integer); integer = begin return x + 1 end
end

var x: [0..10];
plus(x); : integer

var y: integer
plus(x, y); : integer
```

Subtyping

- We can view types as sets of values
- Subtyping is a relation between types induced by the subset relation between value sets

Intuition:
- If T is a subtype of U, then any expression with type T also has type U
How is subtyping useful?

T t;
U u;
\( u := t \);

1. \( T = U \) (type equality)
2. \( T \leq U \) (subsumption)

\( U \leq T \) \( u := (U) t \)

Type checking subranges

- First, figure out which assignments we should allow.
- Second, discuss how we can use subtyping to check them.

```
VAR x : [1 TO 10];
VAR y : INTEGER;
x := 2;
y := 2;
x := 200;
y := 200;
x := y;
y := x;
```

Subtyping may break type safety

- With "bad" definitions of the subtyping relation, we may break type safety.
- How can we make \( 11(8) \) (apply 8 to the 11) type check by just adding the subsumption rule?

Plan for Safe Subtyping

- Define rules for base types based on subsets
  - \( \text{SHORT} \leq \text{INTEGER} \quad [a \ TO \ b] \leq \text{INTEGER} \)
- Define rules for type constructors that respect safety

Subtyping for Subranges

- \( [a \ TO \ b] \leq [c \ TO \ d] \) if

Subtyping for Subranges

- \( [a \ TO \ b] \leq [c \ TO \ d] \) if
Subtyping for Pairs

• \( T \times T' \preceq U \times U' \) if

Subtyping for Pairs

• \( T \times T' \preceq U \times U' \) if

Subtyping for Records

• \( \{ l_1 : T_{-1}, \ldots, l_n : T_n \} \preceq \{ l_1 : T_{-1}', \ldots, \\
  l_m : T_{m}' \} \) if

For Next Time

• No new reading
• Online discussion forum
  - \( \geq 1 \) substantive question, comment, or answer each week
• Work on HW6