Anonymous Functions Anonymous Functions First Class •0000

Objectives

Introduction to Higher Order Functions

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign DEPARTMENT OF COMPUTER SCIENCE

- ► Explain the concept of first class citizen.
- ▶ Use sectioning and lambda to define anonymous functions.
- ► Change the behavior and interface of a function by using another function.





First Class Anonymous Functions

First Class

First Class Functions Defining Functions the Usual Way

An entity is said to be first class when it can be:

▶ **Assigned** to a variable, **passed** as a parameter, or **returned** as a result

Examples:

- ► APL: scalars, vectors, arrays
- ► C: scalars, pointers, structures
- ► C++: like C, but with objects
- ► HASKELL, LISP, OCAML: scalars, lists, tuples, functions

The Kind of Data a Program Manipulates Changes the Expressive Ability of a Program.

Some HASKELL Functions

```
1 \operatorname{sqr} a = a * a
2 hypotsq a b = sqr a + sqr b
```

Sample Run

```
sqr :: Integer -> Integer
2 sqr :: Num a => a -> a
3 hypotsq :: Num a => a -> a -> a
4 Prelude> sqr 10
5 100
6 Prelude > hypotsq 3 4
7 25
```





 First Class
 Anonymous Functions
 First Class
 Anonymous Functions

 ○○○◆○
 ○○○○◆○
 ○○○○◆○
 ○○○○○

Example: Compose

1 inc x = x + 1

Example

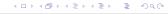
Example: Twice

- ▶ One handy function allows us to do something twice.
- ► You will see this function again!

Twice

First Class

```
htwice f x = f (f x)
Here is a sample run ...
Prelude> :t twice
twice :: (t -> t) -> t -> t
Prelude> twice inc 5
7
Prelude> twice twice inc 4
```



Anonymous Functions

Creating Functions: Lambda Form

► Functions do not have to have names.

```
_{1}\x -> x + 1
```

First Class

- ► The parts:
 - ► Backslash (a.k.a. *lambda*)
 - Parameter list
 - Arrow
 - ► Body of function

```
1prelude> (\x -> x + 1) 41
242
```

Creating Functions: Partial Application

Standard Form vs. Anonymous Form

```
1 inc :: (Num t) => t -> t
2 inc a = a + 1
3 inc = \a -> a + 1
4
5 plus :: (Num t) => t -> t -> t
6 plus a b = a + b
7 plus = \a -> \b -> a + b
```

▶ What do you think we would get if we called plus 1?

4日 → 4周 → 4 差 → 4 差 → 1 差 9 9 0 ○

Anonymous Functions

Anonymous Functions First Class Anonymous Functions First Class

Creating Functions: Partial Application

Standard Form vs. Anonymous Form

```
inc :: (Num t) => t -> t
inc a = a + 1
inc = \a -> a + 1

fplus :: (Num t) => t -> t -> t

fplus a b = a + b

fplus = \a -> \b -> a + b
```

▶ What do you think we would get if we called plus 1?

```
inc = plus 1
```



Anonymous Functions

First Class

Curry and Uncurry

▶ Suppose you have a function tplus that takes a pair of integers and adds them.

```
1tplus :: (Integer,Integer) -> Integer
2tplus (a,b) = a + b
```

- ▶ But you really wish it took its arguments one at a time.
- ► There's a function curry :: (a,b) -> c -> a -> b -> c that will convert it for you! See if you can write it.

4 D > 4 B > 4 E > 4 E > E 9 Q C

η -equivalence

An Equivalence

$$f \equiv \x \rightarrow f x$$

► Proof, assuming f is a function...

$$f z \equiv (\langle x - \rangle f x) z$$

These are Equivalent

```
1 plus a b = (+) a b
2 plus a = (+) a
3 plus = (+)
```

1 inc x = x + 1

So are These

