Functional Programming

Functions

Prof. Dr. Peter Thiemann

Albert-Ludwigs-Universität Freiburg, Germany

WS 2024/25

Function definition by cases

Example: Absolute value

Find the absolute value of a number

- if x is positive, result is x
- if x is negative, result is $-x$

Function definition by cases

Example: Absolute value

Find the absolute value of a number

- if x is positive, result is x
- if x is negative, result is −x

Definition

```haskell
-- returns the absolute value of x
absolute :: Integer -> Integer
absolute x | x >= 0 = x
absolute x | x < 0 = - x
```

Alternative styles of definition

One equation

```
absolute' x | x >= 0 = x
            | x < 0 = -x
```

Using if-then-else in an expression

```
absolute'' x = if x >= 0 then x else -x
```

Recursion

Standard approach to define functions in functional languages (**no loops!**)

Example: power

Compute `x^n` without using the built-in operator

```
-- compute x to n-th power
power x 0 = 1
power x n | n > 0 = x * power x (n - 1)
```

Example: Counting intersections

Task

- Consider n non-parallel lines in the plane
- How often do these lines intersect (at most)? Call this number $I(n)$.

Example: Counting intersections

Task

- Consider n non-parallel lines in the plane
- How often do these lines intersect (at most)? Call this number $I(n)$.

Base case: $n = 0$ (as simple as possible!)

Example: Counting intersections

Task

- Consider n non-parallel lines in the plane
- How often do these lines intersect (at most)? Call this number $I(n)$.

Base case: $n = 0$ (as simple as possible!)

- Zero lines produce zero intersections: $I(0) = 0$

Example: Counting intersections

Task

- Consider n non-parallel lines in the plane
- How often do these lines intersect (at most)? Call this number $I(n)$.

Base case: $n = 0$ (as simple as possible!)

- Zero lines produce zero intersections: $I(0) = 0$

Inductive case: $n > 0$

Example: Counting intersections

Task

- Consider n non-parallel lines in the plane
- How often do these lines intersect (at most)? Call this number $I(n)$.

Base case: $n = 0$ (as simple as possible!)

- Zero lines produce zero intersections: $I(0) = 0$

Inductive case: $n > 0$

- One line can intersect with the remaining lines at most $n - 1$ times.

Example: Counting intersections

Task

- Consider n non-parallel lines in the plane
- How often do these lines intersect (at most)? Call this number $I(n)$.

Base case: $n = 0$ (as simple as possible!)

- Zero lines produce zero intersections: $I(0) = 0$

Inductive case: $n > 0$

- One line can intersect with the remaining lines at most $n - 1$ times.
- Remove this line. The remaining lines can intersect at most $I(n - 1)$ times

Example: Counting intersections

Task

- Consider n non-parallel lines in the plane
- How often do these lines intersect (at most)? Call this number $I(n)$.

Base case: $n = 0$ (as simple as possible!)

- Zero lines produce zero intersections: $I(0) = 0$

Inductive case: $n > 0$

- One line can intersect with the remaining lines at most $n - 1$ times.
- Remove this line. The remaining lines can intersect at most $I(n - 1)$ times
- Combine the above to $I(n) = I(n - 1) + n - 1$

Definition

Counting intersections

```haskell
-- max number of intersections of n lines
nisect :: Integer -> Integer
nisect 0 = 0
nisect n | n > 0 = nisect (n - 1) + n - 1
```

Questions?

?