Universidad Nacional Autónoma de México
Facultad de Ciencias
Lenguajes de Programación

•••

Karla Ramírez Pulido

Tipos

¿Cuántos tipos conocen?

- int
- float
- double
- long
- short
- byte
- bit

- number
- char
- string
- boolean
- object
- list

¿Algún otro tipo?

Clasificación de lenguajes basada en tipos

1. Lenguajes con tipificado explícito o también conocidos como Lenguajes fuertemente tipificados *(strong typing languages)* Haskell, Java, Pascal, ...

```
int  foo (int x) {

    int x + int x;

}
```

2. Lenguajes con tipificado implícito. Scheme, Racket, LISP, ...

```
(define foo

    (lambda (x)

        (+ x x)))
```

¿Qué es un “tipo”?

Es cualquier propiedad de un programa que se puede establecer sin ejecutar el mismo. Mucha gente se refiere a ellos como la abstracción de un conjunto de valores

Son parte del lenguaje:

- Tipos primitivos

Puede extender el lenguaje:

- Tipos Abstractos de Datos

Sintaxis y Semántica:

1. INT
2. int
3. Int
4. Integer
5. INTEGER

Semántica: es un tipo de dato de 32 bits con signo para almacenar valores numéricos. Cuyo valor mínimo es -231 y el valor máximo 231-1.

Ventajas de tener lenguajes con tipos explícitos

Los tipos ayudan a autodocumentar los programas. El tener tipos explícitamente proveen una descripción aproximada acerca del comportamiento de un programa.

Los compiladores/intérpretes pueden explotar los tipos para hacer los programas más rápidos en desempeño y en manejo de espacio (optimizaciones) por ejemplo en la liberación de espacio por parte de los recolectores de basura.

El sistema de tipos ayuda a prevenir algunos errores asociados a los tipos, tanto en tiempo de compilación como de ejecución (es decir, tanto estática como dinámicamente).

El Sistema Verificador de Tipos

Checador de tipos

OJO: en español no existe la palabra checador, sino verificador.

El sistema de tipos ayuda a detectar errores asociados a los tipos (en la etapa de depuración) de un programa.

El sistema de tipos ayuda a “cachar” errores de tipo antes de ejecutar un programa.

El diseño de un sistema de tipos debe de contemplar:

- Adquirir más información para así poder obtener mejor y más precisas conclusiones acerca del comportamiento de un programa sin tener que ejecutarlo (i.e. estáticamente)
- Adquirir más información es difícil por lo que:
  - Un lenguaje podría tener restricciones de uso para sus primitivas (u otras operaciones).
  - Se debe de considerar que esto se verá reflejado en el tiempo en que se haga la verificación de tipos.
  - Aún y cuando no todas las partes de un programa se ejecuten en una ejecución, todas deben de estar verificadas.

¿Qué regresa cada una de las siguientes líneas?

1. {fun {x} }

   error: ¿?

1. {+ 3 {fun {x} x} }

   error: num-n: not a number

3. {with {f {fun {x} {+ x 1}}}
   {+ 3 {f 5}} }

   = {+ 3 {fun{x=5} {+ 5 1}} }

   = {+ 3 6}

   = 9

¿Qué regresa el siguiente código?

```
{with {f  {fun {x}
              {fun {y} {+ x y}}}}
   {+ 3  {f   5}} }
```

```
= {+  3 {{fun {x=5}
              {fun {y} {+ 5 y}}}} }
```

```
= {+  3   {fun {y} {+ 5 y}} }
= {+ 3  fun}
```

error: ¿de quién?

1. de la suma {+ 3 fun}
2. de la fun {y} ... i.e. del 2do.Parámetro

¿Qué regresa el siguiente código?

```
{+ 3
   {if0  {read-number}
         5
         {fun {x} x}} }
```

1. Escenario A:
   a. cuando read-number es 0

      if0 {0} ⇒ 5

1. Escenario B:
   b. cuando read-number es 1

      if0 {1} ⇒ {fun {x} x}

¿Qué regresa el siguiente código?

```
{+ 3
    {if0  {read-number}
          5
          {fun {x} x}} }
```

Escenario A:

a. cuando read-number es 0

   if0 {0} ⇒ 5

   ⇒ {+ 3 5} = 8

¿Qué regresa el siguiente código?

```
{+ 3
   {if0  {read-number}
         5
         {fun {x} x}} }
```

Escenario B:

a. cuando read-number es 1

   if0 {1} ⇒ {fun {x} x}

⇒ {+ 3 {fun{x} x} }

error: suma...

Gracias

¿Dudas?