Wyrażenia typów bazowych

Do typów bazowych należą:

- `unit` — typ z jedną wartością (jak w C)
- `bool` — wartości logiczne
- `int` — liczby całkowite
- `float` — liczby zmiennopozycyjne
- `char` — znaki
- `string` — napisy

i notacja dla stałych w poszczególnych typach:

- `() : unit`
- `true, false : bool`
- `17, 0xfa12, 0o723, -0b111011 : int`
- `2.3, -0.5e-3 : float`
- `'a', '\n', '\001' : char`
- `"napis" : string`

Podstawowe operacje:

- na wartościach logicznych: `not`, `&`, `&&`, `or`, $||$
- na liczbach całkowitych:
  - arytmetyczne: `~-`, `+`, `-`, `*`, `/`, `mod`, `succ`, `pred`, `abs`
  - bitowe: `land`, `lor`, `lxor`, `lnot`, `lsl`, `lsr`, `asr`
  - stałe: `max_int`, `min_int`
- na liczbach zmiennopozycyjnych:
  - arytmetyka: `+.`, `-.`, `*.`, `/.`, `**`
  - funkcje matematyczne: `exp`, `acos`, `asin`, `atan`, `atan2`, `cos`, `cosh`, `log`, `log10`, `sin`, `sinh`, `sqrt`, `tan`, `tanh`
  - zaokrąglenia: `ceil`, `floor`
  - konwersje z liczbami całkowitymi: `float`, `truncate`
- na napisach:
  - konkatenacja: `^`
  - konwersje: `string_of_bool`, `string_of_int`, `string_of_float`, `int_of_string`, `float_of_string`

- uniwersalne:
  - porównania strukturalne: `=`, `<>`, `<`, `<=`, `>`, `>=`
  - `min`, `max`
  - identyczność: `==`, `!=`
  - wyrażenie warunkowe; oto składnia:

```
if <wyrażenie logiczne> then
  <wyrażenie1>
else
  <wyrażenie2>
```

    Zarówno `<wyrażenie1>` jak i `<wyrażenie2>` muszą dawać wartość tego samego typu. Przykład:

```
if 3 < 4 then 3.1 else 3.9
```

Typy złożone

Do typów złożonych należą pary i krotki, rekordy, listy i inne algebraiczne typy danych (predefiniowane lub zdefiniowane przez użytkownika).

Na początku ograniczymy się do krotek i rekordów.

Krotki

Krotka jest n–tką elementów, z których każdy może być dowolnego typu. Krotki zapisujemy jako wartości oddzielone przecinkiem. Czasami może być potrzebne ujęcie ich w nawiasy, na ogół jest to zbędne.

Oto proste przykłady:

```
# 3, ();;
- : int * unit = 3, ()

# (10 / 2, 10 mod 2 = 0, "parzyste");;
- : int * bool * string = 5, true,
    "parzyste"

# 10 / 2, (10 mod 2 = 0, "parzyste");;
- : int * (bool * string)
```

```
      = 5, (true, "parzyste")

# 10 / 2, (10 mod 2 = 0, "parzyste"), cos;;
- : int * (bool * string) * (float -> float)
      = 5, (true, "parzyste"), <fun>

# (3 + 4, "identyczne") = (9 - 2, "identyczne");;
- : bool = true

# ((13 / 3, 3 > 5), 7 * 7) =
                      ((2 * 2, not true), 50);;
- : bool = false

# (3 + 4, "identyczne") == (9 - 2, "identyczne");;
- : bool = false
```

Pary są szczególnym przykładem n–tek. Dla par (i tylko dla par!) standardowo zdefiniowane są operacje rzutowania na pierwszą (`fst`) i drugą współrzędną `snd`):

```
# 3, ();;
- : int * unit = 3, ()

# fst (3, ());;
- : int = 3

# snd (10 mod 2 = 0, "parzyste");;
- : string = "parzyste"
```

Rekordy

Rekordy, podobnie jak w Pascalu, są wartościami złożonymi z pól o ustalonych nazwach, typach (jak zwykle dowolnych) i wartościach. W odróżnieniu od Pascala, w Camlu nie ma konstrukcji odpowiadającej rekordowi z wariantami (analogiczną rolę pełnią algebraiczne typy danych, o których pomówimy w przyszłości). Również odmiennie niż w Pascalu, pola rekordu nie mogą być zmieniane.

W Camlu, podobnie jak w Pascalu, typy rekordów trzeba zadeklarować przed pierwszym użyciem. Uwaga : nazwy etykiet w definicjach różnych typów muszą być różne!

Ty rekordowy deklaruje się tak:

```
type typ\_rekordowy =
  { <nazwa pola 1> : <typ 1>; ... ;
    <nazwa pola n> : <typ n> }
```

Wartość rekordową zapisujemy również w nawiasach klamrowych, po nazwie pola podając znak równości i wartość dla danego pola.

Robi się to tak:

```
{ <nazwa pola 1> = <wyrażenie 1>; ... ;
   <nazwa pola n> = <wyrażenie n> }
```

A to jest przykład:

```
# type pora = { dzien : string; godzina : int} ;;
type pora = { dzien: string; godzina: int }

# type spotkanie =
    { data : int * int * int;
      pora : pora };;
type spotkanie =
    { data: int * int * int;
      pora: pora }

# type jakistam =
  { tak : int;
    nie : int;
    abojawiem : string };;
type jakistam =
  { tak: int;
```

```
    nie: int;
    abojawiem: string }

# { nie = 0; tak = 1; abojawiem = "?" } ;;
- : jakistam = {tak=1; nie=0; abojawiem="?"}

# { nie = 0; tak = 1; abojawiem = "?" }
  =
  { abojawiem = "?"; tak = 1; nie = 0} ;;
- : bool = true

# { nie = 0; tak = 1; abojawiem = "?" }
  !=
  { abojawiem = "?"; tak = 1; nie = 0} ;;
- : bool = true

# { nie = 0; tak = 1; abojawiem = "?" }
  =
  { abojawiem = ""; tak = 7; nie = 3} ;;
- : bool = false
```

Dostęp do pól rekordów odbywa się przez kropkę, składnia jak w Pascalu czy C:

```
# { nie = 0; tak = 1; abojawiem = "?" }.nie;;
- : int = 0

# { data = (7, 10, 94);
    pora =
      { dzien = "piatek";
        godzina = 15
      }
  } . pora . dzien ;;
- : string = "piatek"
```