TOM programmeerimiskeel

Elinor Toodo

Tutvustus (1)

- Objekt-orienteeritud programmeerimiskeel
- Aluseks Objective C ja sarnaneb ka Java'le
- Kompilaator- Tesla (1998) ja tomc
- Arendati 90ndatel - 2002 (enam ei arendata) „Programmers without deadlines“ poolt
- Mitmene pärimine, korteežid, laiendused
- TOM – lihtsalt nimi sest

  “Names don't need to have a meaning. :)”

Tutvustus (2)

- Programmile saab juurde panna laiendusi (*categories*) selle töö ajal
- Laiendused:
  - Saavad välja kutsuda/lisada teisi klasse/meetodeid/muutujaid.
  - Ei vaja uuesti kompileerimist
- Käsu lõpus ; ja plokid piiratud { }

Koodist programmiks

- Koodifail laiendiga .t

  hello.t

- Lihtsaim viis teha GNU makefile sisuga:

```
UNIT=           hello
TOM_SRC=        hello

TOM_MAKEFILES_DIR=  /usr/lib/tom/makefiles
include $(TOM_MAKEFILES_DIR)/GNUmakefile.bin
```

- Programmi käivitamine tavalisel viisil

  ./hello

Andmetüübid

Täisarvulised:

- Byte - 8-bitine märgita (‘a’)
- Char - 16-bitine märgita
- Int - 32-bitine märgiga (123456, 0377, 0xff)
- Long - 64-bitine märgiga (0L, 1l)

Ujukomaarvud:

- Float - tavatäpsusega (1.0, 1e23, 4.2e1)
- Double - topelt-täpsusega (1d)

Näiteid muudest tüüpidest

- Tühi tüüp (*void)*
  - Näitab tüübi puudumist
  - Kasutatakse meetodi tagastuse tüübina
  - Tal ei ole vaikeväärtust
  - Selle tüübiga ei saa mingeid operatsioone teha
- Viit (*pointer*)
  - Viita saab ainult edasi anda
  - Kasutatakse massiivides, teises keeles kirjutatud koodi integreerimiseks või silumiseks
  - Null pointer’ile viidata ei saa

Operaatorid

- ++ , --
- - (miinus), ~ (inversioon), !
- * , / , %
- + , - (lahutamine)
- <<, >> (aritmeetiline nihe)
- >>> (loogiline nihe paremale)
- & (bitikaupa AND)
- | (bitikaupa OR)
- ^ (bitikaupa välistav-OR)
- <, <=, >=, >
- ==, !=
- && (tingimuslik AND)
- || (tingimuslik OR)
- -> (implikatsioon)
- ?: (if-then-else)
- = (assignment)

Muutujad

Deklareerimine:

- Koos tüübiga

```
int a;
int a = 1, b = 2;
```

- Väärtustamata omandab muutuja tema tüübi vaikeväärtuse (näiteks 0 või FALSE)
- Nimi väikeste tähtedega, võib sisaldada – ja _

Väärtustamine:

```
a = 1;
```

Nähtavused, täpsustused (1)

- Mutable – Genereeritud set-meetodiga

```
mutable int starting_value;
void
 set_starting_value  int _new_value
{
 starting_value = _new_value;
}
```

- Private – Klassisisestele meetoditele kättesaadav
- Protected – Klassi ja alamklasside meetoditele kättesaadav

Nähtavused, täpsustused (2)

- Public – Automaatsed juurdepääsu meetodid

```
public int starting_value;
int
  starting_value
{
  = starting_value;
}
```

- Static – Eksisteerib ainult defineerivas klassis
- Obsolete – Muutuja, mis ei ole jääv
- Local – Väärtus salvestatakse lõimesiseselt
- Redeclare – Muutuja tüübi muutmiseks

Meetodid

- Nimi algab väikese tähega, võib sisaldada suuri tähti
- Meetodid toetavad mitme väärtusega tagastust
- Meetodi nimi võib koosneda mitmest osast

```
int
multiply int a
        by int b
{
  return a * b;
}
```

- Meetodi väljakutsel on sõnumi vastuvõtjaks self (mingi klassiobjekt)

```
int result = 1;
result = [self multiply result by 2];
```

Meetodid (2)

- Tagastus – kõik parameetrite tüübid plus tühi tüüp *void.*
- Tagastuseks kaks võimalust:
  - *return* võtmesõna

```
int
  returnOne
{
  int result = 1;
  [self deallocateResources];
  return result;
}
```

  - Tagastav omistamine (*return assignment*)

```
int
  returnOne
{
  = 1;
  [self deallocateResources];
}
```

- Kui tagastust ei väärtustata, tagastatakse tüübi vaikeväärtus

Korteežid (*tuples*)

- Korteež - väärtuste grupp. Sulgudes ja väärtused eraldatud komadega. Neid saab ka üksteise sisse paigutada.
- Muutujaid ei saa deklareerida korteež-tüüpi
- Väärtuste tüübid ei pea olema ühesugused.

  näiteks (1, 3.14) on korteež tüübiga (int, float)

- Korteeži osi saab väärtustada ainult korraga.
- Kasutatakse näiteks meetodites või mitmelises väärtustamises
- Lihtne väärtuste vahetamine: (a, b) = (b, a);

Massiivid

- Deklareerimine

  Süntaksi näidete napuse tõttu ei ole teada kuidas deklareeerimine käis

- Pikkuse küsimine

  int n = [argv length];

- Elementide juurdepääs (0 <= int index < [argv length])

  String s = [argv at index];

Tsüklid

- *While* tsükkel

```
int counter;
while (counter < 10)
 {
    counter = counter + 1;
 }
```

- *Do while* tsükkel

```
int counter;
do
 {
    counter = counter + 1;
 } while (counter < 10);
```

- *For* tsükkel

```
int counter;
for (counter = 0; counter < 10; counter++)
 {
 }
```

Tsüklid pärinevad C keelest

Tingimus-struktuurid

- ?: operaator - *if true ? then : else*

```
int a = 0;
int b = 1;
int c = a > b ? a : b;
```

- *If else* konstruktsioon

```
int n = 1
if (n == 0)
  [[[stdio out] print („n=“, n)] nl];
else
  [[[stdio out] print „not null"] nl];
```

- Kui else osa puudub, saab vaikeväärtuse

```
int b = 3;
int a = if (b < 0) b;                (ehk a = 0)
```

Klassid

- Nimi võib sisaldada suuri tähti
- Klassid toetavad lihtsat mitmest pärilust (*multiple inheritance*)

päriluse näide:

```
implementation class A: B, counter
end;

implementation instance A
end;
```

```
implementation class Counter
Int
  main Array argv
{
  [[[stdio out] print "Hello, world!"] nl];
}
end;

implementation instance Counter
{
  int current_value;
}

int
  nextValue
{
  current_value += 1;
  = current_value;
}

end;
```

Allikad

Dokumentatsioon:

http://gerbil.org/tom/

- http://gerbil.org/tom/doc/tome/

Natuke üldist infot:

http://en.wikipedia.org/wiki/TOM_%28object-oriented_programming_language%29