

Komputeralgebra rendszerek

Függvények

Czirbusz Sándor

`czirbusz@gmail.com`

Komputeralgebra Tanszék
ELTE Informatika Kar

2009-2010 tavasz

Index I

1 Definiálási módok

- Formulák
- Függvénydefiniálás Maple–ban
- Függvénydefiniálás Sage–ben
- Utólagos függvénydefiniálás
- Lépcsős függvények a Maple–ban
 - Mint case-utasítás
 - A piecewise procedúra
- Lépcsős függvények a Sage–ben
 - Anonim függvények

2 A MAPLE procedúrái

- Definiálás
 - Rekurzió
 - Műveletek függvényekkel

3 A Sage procedúrái

Formulával

Nem igazi függvény!

$$T := T0 * \exp(-c * t)$$

Értékkadás:

Maple–ban **Maple–ban**

- **subs** vagy az **eval** segítségével
- **Eval** és **value** segítségével

Sage–ben

- `T.subs(t = ...)`

A nyíl operátor és a procedúrák

Nyíl operátor

`fn := parameter(ek)-> kif`

- Többváltozós is és 0 változós is lehet
- paraméteres függvény megadására nem alkalmas

Procedúrával

`fn := proc(parameterek) ... end proc`

A legrugalmasabb forma. Nem kötelező az explicit **return**. ekkor az utolsó végrehajtott értékkel tér vissza.

„Röptében”

Az **f(x) := kifejezés** forma az újabb Maple-verziókban működik, a rendszer megkérdezi, valóban függvényt akarunk-e definiálni.

A lambda operátor és a procedúrák

A lambda operátor

$fv = \mathbf{lambda}$ paraméterek: kif

Procedúrával

def fn(paraméterek): ...

Kötelező a **return**, a struktúra Python-módra az indentálással azonos.

„Röptében”

f(x) = kifejezés

Utólagos függvénydefiniálás

Ha meggondoltuk magunkat Maple-ban

- `fname := unapply(formula, name_seq)`
- `fname := convert(CompSeq(params=[var_seq],[formula], procedure)`
- **Tolnert, FromInert**

Ugyanaz Sage-ban

- ha a formulának van **.function()** belső függvénye, akkor ennek segítségével
- ha nincs, de van **.polynom()** belső függvénye, akkor azon keresztül
- egyszerűbb esetekben működik a **lambda** paraméterek, kifejezés segítségével

Mint case-utasítás

Csak a Java előtti Maple-verziókban!

```
function_name:=parameter(s)->
  if cond1 then
    val1
  elif cond2 then
    val2
    .....
  elif condN then
    valN
  else
    default
  endif
```

A piecewise procedúra

```
function_name :=parameter(s)->piecewise(  
    cond1, val1,  
    cond2, val2,  
    .....  
    condN, valN,  
    default  
)
```

Előnye: "ráengedhető" a Maple matematika apparátusa

piecewise

A konstrukció eltérő a Maple-étől

- Szintaxis: **piecewise(paramlist[, var=None])**
- A „paramlist” elemei (l, f) alakú párok, ahol „ l ” egy intervallum, „ f ” pedig egy függvény
- Az intervallumoknak összefüggőeknek kell lenniük, a közös ponton a függvény-érték azonos ☹
- A függvény lehet előre definiált függvény, polinom, lambda operátor
- Az opcionális változóval jelezzük, mi a függvény-változó
- A helyes definíciót nem ellenőrzi a Sage ☹

Anonim függvények

- Mindkét rendszer lehetővé teszi, hogy rövid függvényeknek ne adjunk nevet
- Maple-ban ez a nyíloperátor
- Sage-ban a lambda
- Többnyire funkcionális konstrukciókban, egyszeri műveletekre, iterátorokban.

Definiálás I

```
proc( parameter_seq )[::ret_type]  
    [local name_seq;]  
    [global name_seq;]  
    [options name_seq;]  
    [description string;]  
    [uses name_seq;]  
    statements  
end proc
```

Paraméterek típusjegyzetése a :: -vel

Visszaadott érték ha nincs explicit **return** utasítás: az utoljára számított. A **proc** operandusai

Definiálás II

- op 1 A paramétersorozat
- op 2 A lokális változók
- op 3 Az opciók
- op 4 A `remember` táblázat
- op 5 A leíró rész
- op 6 A globális változók
- op 7 Lexikális tábla (a nemdefiniált változók kapcsolata környező proc-okkal)
- op 8 A visszaadott érték típusa, ha definiált

Definiálás III

Tesztelgetés

- `kernelopts(profile=true)` - a procedúrahívások figyelése
- `exprofile`, `excallgraph`
- A remember listázása: `interface(verboseproc=3)`, `print(proc)`
- `op(4,eval(proc))`
- Törlés: **forget**(proc,[n]). Mellékhatás: a 'proc/nevű procedúrák táblái is törlődnek.
- Alternatív törlés: **subsop**

Rekurzió

A "**remember**" táblázat.

- Az **option** remember hatására építi föl
- Lekérdezhető - a procedúra struktúra 4. eleme:
op(4, eval(procname));
- Felülírható: **procname(arg): = value;**
- Törölhető: **forget(procname, index)**

Műveletek függvényekkel

- Elemi műveletek: összeadás, szorzás, stb.
- Kompozíció
Általában $f@g$, önmagával ($f@@n$), inverz: $f@@(-1)$

A Sage procedúrái

- Python örökség: a struktúrát csak az indentálás jelzi.
- A **def fn(paraméterek):** után a bekezdés 4 karakterrel belejbb kerül.
- A procedúratörzs addig tart, míg ezt az indentálást megszüntetjük (vagyis visszakerülünk a „def” első betűje oszlopába).
- Bárhol elhelyezett **return** megszakítja a procedúrát
- Visszatérést okoz a **yield** is; ekkor iterátort „gyártunk”

Példák

Lásd a munkalapokon