Occam 2.

Készítette: Szabó Éva

Függvények

- Célja: hogy egy bonyolult számítást elkülönítsenek, illetve újrafelhasználhatóvá tegyenek.
- Alapelv, hogy nem okozhatnak mellékhatást.
- Nem használhatnak kimenő paramétereket , ezért minden formális paraméterüket konstansként kell deklarálni.
- Nem módosíthatnak globális változókat
- A visszatérési értékeiket kötelesek vagyunk felhasználni.

Függvények deklarálása

TÍPUS1, TÍPUS2, ... , TÍPUSn ***FUNCTION*** *név(formális_paraméterek)*

lokális változódeklarációk:

VALOF

valamilyen processz (elemi/összetett)

RESULT *retval1, retval2, ... ,retvaln*

:

Ahol:

- *TÍPUS1, TÍPUS2, ... , TÍPUSn* a visszatérési értékek típusai, legalább egynek kell lennie, tömb esetén meg kell adni a méretét, lehetnek különbözőek is.
- A formális paramétereket ugyanúgy kell megadni, mint az eljárásoknál, de mindegyiknek konstansnak kell lennie (*VAL* kulcsszóval). Mivel konstans csatorna nem létezik, ezért csatorna nem lehet függvény paramétere.
- a *RESULT* paraméterei lesznek a konkrét visszatérési értékek, ezek csak (lokális) változók lehetnek, konstansok, literálok nem. Annyinak kell lennie, ahány visszatérési típust megadtunk és olyan sorrendben, ahogy azokat megadtuk. Pontosan 1 db *RESULT* utasításnak kell szerepelnie a *VALOF* blokkjában.

helye, szerepe a nyelvben

működése

fajtái

vományos - több teljesülő őr esetén *elvileg* véletlenül vá

ALT - több teljesülő őr esetén az elsőt választja

is: (a szögletes zárójelben szereplő részek opcionálisak,

```
--------------------------------------------------------

-- 1. Két darab felhasználótól bekért számot állíts
nagyság

-- szerinti sorrendbe, és írd ki őket! Az
összehasonlításhoz

-- használj külön függvényt!

--------------------------------------------------------

#USE "course.lib"

PROC nl(CHAN OF BYTE out)

  SEQ

    out.string("*n",1,out)

    flush(out)

:
```

```
INT, INT FUNCTION sort(VAL INT a, b)
  INT x,y:
  VALOF
    IF
      a>b
        x,y:=b,a
      TRUE
        x,y:=a,b
    RESULT x,y
:
```

```
PROC main (CHAN OF BYTE in, out, err)
  INT egy,ketto, kisebb,nagyobb:
  SEQ
    out.string("Kerek ket szamot:", 20, out) -- a szöveg nincs 20 hosszú, ezért lesz előtte pár szóköz
    nl(out)
    out!'*t' -- outra írunk és nem flusholjuk, ezért nem fog egyből megjelenni
    in.int(egy, 5, in, out)
    out!'*t'
    in.int(ketto, 5, in, out)
    out!'*n'
    kisebb, nagyobb:=sort(egy, ketto) -- függvényhívás
    out.string("A nagyobb: *t", 15, out)
    out.int(nagyobb, 5, out)
    nl(out)
:
```

ALT

- Az ALT az Occam egy speciális szerkezete, más nyelvekben nem találunk ennek megfelelő szerkezetet.
- Egy folyamat kiválasztását őrök segítségével végzi, melyek lehetnek tiltottak vagy engedélyezettek.
- Az ALT minden egyes csatornát figyel és a hozzá tartozó őr engedélyezetté válik, ha az általa figyelt csatornára egy folyamat készen áll adat küldésére.
- Az őr, melynek csatornájára nem érkezik adat, tiltott.

- Egy őr általánosan a követezőképpen néz ki:

  <feltétel> **&** *<csatorna>* **?** *<változó>*

- A *<feltétel>* elhagyható, ha nem írjuk ki, akkor a feltétel *TRUE*-nak felel meg.
- Az *ALT* csak azokat az őröket veszi figyelembe, mely előtt a feltétel *TRUE*.
- Ha egy őr engedélyezetté válik, akkor a benne megadott változó felveszi a csatornáról érkező adat értékét és „elindítja” a hozzá tartozó folyamatot.

ALT deklarálása

ALT

[feltétel1 ***&*** *]csatorna1* ***?*** *változó1*

folyamat1

[feltétel2 ***&*** *]csatorna2* ***?*** *változó2*

folyamat2

...

[feltételn ***&*** *]csatornan* ***?*** *változón*

folyamatn

példa

```
ALT
  c1 ? x
    y := y+x
  c2 ? x
    z := z+x
```

- Az x változó értéke attól függ, hogy c1-re vagy c2-re érkezik előbb adat. Mivel a program írásakor nem tudhatjuk, hogy melyik csatornáról fog adat érkezni, ezért az ALT-ot tartalmazó programok nemdeterminisztikusak.
- A holtpont veszélye sincs kizárva, mert azt sem tudhatjuk előre, hogy a felsorolt csatornákra egyáltalán fog-e érkezni adat.

Egy olyan feladat, amelyhez hasonló lehet akár a ZH-n is

```
PROC P1(CHAN BYTE in, out)
  BYTE c:
  SEQ
    in ? C
    WHILE c <> '*n'
      SEQ
        IF
          (c >= '0') AND (c <= '9')
            out ! C
          TRUE
            SKIP
        in ? C
    out ! '*n'
:
```

```
PROC P2(CHAN BYTE in, out)
  BYTE c:
  SEQ
    in ? C
    WHILE c <> '*n'
      SEQ
        IF
          c = '9'
            out ! '0'
          TRUE
            out ! (c + 1)
        in ? C
        out ! ' '
:

PROC main(CHAN BYTE keyboard, screen, error)
  CHAN BYTE csatorna:
  PAR
    P1(keyboard, csatorna)
    P2(csatorna, error)
:
```