Eljárások, függvények

Tartalomjegyzék

Készítette: Gál Tamás

Az alprogramok

A strukturált programozás egyik fontos eszköze az alprogramok használata. **Magas szintű programozási nyelvekben két típusát különböztetjük meg:**
1. Eljárások
2. Függvények

Az alprogramok használata megkönnyíti az összetettebb programok elkészítését. Segítségükkel jól áttekinthető, könnyen javítható vagy módosítható strukturált programszerkezetet alakíthatunk ki.

Az eljárás tulajdonképpen egy névvel ellátott kódrészlet, így tekinthető a programozási nyelv utasítás készletét kibővítő eszköznek is. **Az eljárásoknak nincs visszatérési értékük. Az elkészített eljárásra nevével hivatkozhatunk.**

A függvényeknek viszont visszatérési értékük. Ha nincs mellékhatásuk, akkor a függvényeket a programozási nyelvben használható operátorok körét kibővítő eszközként is felfoghatjuk.

Nem minden magas szintű programozási nyelv teszi lehetővé eljárások készítését. Ilyenkor **void** típusú függvényeket írhatunk, amelynek nincs visszatérési értéke.

Az objektum orientált programozási nyelvekben metódusok határozzák meg az objektumok működését. Szerepük hasonló az eljárásokéhoz és a függvényekéhez.

Kérdések, feladatok

- Mi a különbség az eljárások és függvények között?
- Mit nevezünk alprogramnak?
- Mi az előnye az alprogramok használatának?

Tervezési módszerek

Felülről lefele (top-down) tervezési módszer lényege, lépésenként finomítjuk a programot. A programot előbb nagyobb egységere bontjuk, majd minden lépést finomítunk.

Egy-egy nagyobb programozási feladat kisebb részfeladatokra alprogramokra bontható. Az alprogramok további alprogramokból épülhetnek fel. Felülről lefelé haladva a legbonyolultabb algoritmust is, a használt programozási nyelv utasítás készletével könnyen megvalósítható kis alprogramokra bonthatjuk, amelyek külön-külön elkészíthetők és tesztelhetők. Később pedig újra felhasználhatók.

Alulról felfelé (down-top) tervezési módszer a „téglánkénti építkezés elvén alapul.
Ahogy a legóban külön elkészül egy-egy ház vagy autó, majd ezeket elrendezve egy város, programozásnál a programozási nyelv utasítás készletéből kiindulva a programozó saját alprogramokat készít, amelyekből később még összetettebb alprogramokat hoz létre…, amíg végül a saját alprogramok összeillesztésével elkészül a teljes program.

Az alulról felfelé építkezés esetén sem úszható meg a feladat átfogó vizsgálata. Az elemzés (analízis) és tervezés lépéseinek kihagyásakor az elkészült alprogramok nem fognak megfelelően kapcsolódni, nem állítható össze belőlük a kész program.

Sok programozási nyelvhez elérhetők plusz **függvénykönyvtárak**, amelyek a gyakran előforduló feladatok megoldásához adnak kész elemeket, a a programozási nyelv utasítás készletének és a felhasználható operátorok körének kibővítse révén. Ezek használata meggyorsítja a programozást.
A programozó is készíthet saját függvénykönyvtárakat, amelyekben az általa készített függvényeket gyűjti össze és a későbbi munkáiban is felhasználhatja azokat változatlan, vagy a feladatra igazított formában.

Kérdések, feladatok

- Mutasd be a top-down tervezési módszert!
- Mutasd be a down-top tervezési módszert!
- Mi a különbség down-top és a top-down tervezési módszer között?
- Miért használunk függvénykönyvtárakat.

Eljárások

Gyakran előfordul, hogy a programunkban egy programrészt többször végrehajtunk. Ekkor célszerű az eljárások használata.

Eljárások készítésének előnye, hogy a programban egy többször végrehajtásra kerülő programrészt:
- egyszer kell megírni,
- egy helyen lehet módosítani.

Az eljárások átgondolt használatával jól áttekinthető, könnyen módosítható vagy javítható program készíthető.

Mondatszerű leírás:

```
Eljárás Eljárásnév(formális paraméterek)
  utasítások
Eljárás vége.
```

Formális paraméterek: azok a változók és konstansok, amelyeket az eljáráson belül használhatunk. Nevük eltérhet a hívó programrészben használt névtől.

1. példa:

```
Program Osszead;
var sz1, sz2: integer;

procedure OsszegKi(a, b: integer);
begin
  writeln(a+b);
end;

begin
  sz1 := 1;
  sz2 := 2;
  OsszegKi(sz1,sz2);
end.
```

Mondatszerű leírás

```
Eljárás  OsszegKi(változó a, b:egész)
  KI:  a+b
Eljárás vége

Program Osszead
  Változó sz1, sz2:Egész
  sz1 = 1
  sz2 = 2
  OsszegKi(sz1,sz2)
Program vége
```

2. példa:

```
Program Osszead;
var sz1, sz2: integer;  // GLOBÁLIS változók

procedure OsszegKi(a, b: integer);
var
  Osszeg: integer;      // LOKÁLIS változók
begin
  Osszeg := a+b;
  writeln(Osszeg);
end;

begin
  sz1 := 1;
  sz2 := 2;
  OsszegKi(sz1,sz2);
end.
```

Mondatszerű leírás

```
Eljárás  OsszegKi(változó a, b:egész)
  Változó Osszeg:Egész
  Osszeg = a+b
  KI:  Osszeg
Eljárás vége

Program Osszead
  Változó sz1, sz2:Egész
  sz1 = 1
  sz2 = 2
  OsszegKi(sz1,sz2)
Program vége
```

A globális változók:

- a program bármely pontján elérhetők,
- a program futási ideje alatt végig foglalják a szükséges memóriaterületet,
- összetett programok esetén könnyen előfordulhat, hogy a program nem azt az értéket találja benne, amire számít, mert a program egy másik része felülírta.

A lokális változók:

- csak az adott utasításblokkban, eljáráson vagy függvényen belül érhető el,
- csak addig foglalja a memóriát, amíg adott eljárás vagy függvény kerül végrehajtásra,
- jól kézben tartható a tartalma.

Kérdések, feladatok

- Készítsd el a példaprogramok folyamatábráját!
- Mit nevezünk egy eljárás formális paramétereinek?
- Mi a különbség a lokális és a globális változók között?

Érték és cím szerinti paraméterátadás

Érték szerinti paraméterátadás
A hívó eljárás az átadandó adatokról másolatot készít, a program vermébe ahonnan a hívott eljárás kiveszi az adatokat.. Az érték szerint átadott paraméterek értékét lehet ugyan módosítani, de a módosítás nem marad érvényes a hívott eljáráson kívül.
Cím szerinti paraméterátadás
A hívó eljárás az adat címét adja át a hívott eljárásnak. A hívott eljárás más néven is jelölheti az átvett paramétert, de az átadott és átvett paraméterek típusának ugyanazoknak kell lennie. A paraméter értékének megváltozása az eredeti változó értékének módosítását is eredményezi.

Példa:
```
Program CsereBere;
var sz1, sz2: integer;
procedure CsereE(a, b: integer); // Érték szerinti paraméterátadás
var
  temp: integer;
begin
  temp := a;
  a    := b;
  b    := temp;
end;

procedure CsereC(var a, b: integer); // Cím szerinti paraméterátadás
var
  temp: integer;
begin
  temp := a;
  a    := b;
  b    := temp;
end;

begin
  sz1 := 1;
  sz2 := 2;
  writeln('a:',sz1,' b:',sz2);
  CsereE(sz1,sz2);
  writeln('a:',sz1,' b:',sz2);
  CsereC(sz1,sz2);
  writeln('a:',sz1,' b:',sz2);
end.
```

Mondatszerű leírás
```
Eljárás  CsereE(változó a, b:egész)
  temp := a;
  a    := b;
  b    := temp;
Eljárás vége

Eljárás  CsereC(változó a, b:egész - cím szerint)
  temp := a;
  a    := b;
  b    := temp;
Eljárás vége

Program Osszead
...
Program vége
```

Kérdések, feladatok

- Készítsd el a példaprogram folyamatábráját!
- Mi az érték és a cím szerinti paraméterátadás között?

Függvények

Felépítésük és működésük megegyezik az eljárásokéval. A különbség csupán, hogy v**isszatérési értékkel rendelkeznek.**

Mondatszerű leírás:

```
Függvény Függvénynév(formális paraméterek):függvényérték típusa
  utasítások
  Függvénynév := kifejezés  // vagy return kifejezés
Függvény vége
```

Példa:

```
program Fgv;
const n=20;                      // konstans
var t: array[1..n] of integer;  // Globális tömb

function Veletlen: integer;
var i, van, szam:integer;       // lokális változók
begin
  repeat
    szam:=random(30);
    i:=1;
    while (i<=n) and (T[i] <> szam) do i := i + 1;
  until  i>n;
  Veletlen:= szam;
end;

procedure Feltolt;
var i:integer;
begin
  for i:=1 to n do t[i]:=Veletlen;
end;

procedure Kiir;
var i:integer;
begin
  for i:=1 to n do write(t[i],', ');
end;

begin
  randomize;
  Feltolt;
  Kiir;
end.
```

Összetett példa:

```
program EljarasFgv;
const n=20; {a tömb elemeinek a száma}
var t: array[1..n] of integer;

function Veletlen: integer;
var i, van, szam:integer;
begin
  repeat
    szam:=random(30)+1;
    i:=1;
    while (i<=n) and (T[i] <> szam) do i := i + 1; //Ha nincs még, akkor i=n+1
  until  i>n;                                      //Ha van akkor újra
  Veletlen:= szam;
end;

procedure Feltolt1;
var i:integer;
begin
  for i:=1 to n do t[i]:=Veletlen;
end;

procedure Feltolt;
var i:integer;
begin
  for i:=1 to n do t[i]:=random(30)+1;
end;

procedure Kiir;
var i:integer;
begin
  for i:=1 to n do write(t[i],', ');
end;

procedure Csere(var a, b: integer);  // cím szerinti paraméterátadás
var tmp:integer;
begin
  tmp := a;
  a   := b;
  b   := tmp;
end;

procedure Rendez;
var i, j:integer;
begin
  writeln('Rendezés');
  for i:=n-1 downto 1 do begin
    for j:=1 to i do begin
      if t[j]>t[j+1] then Csere(t[j],t[j+1]);
    end;
  end;
end;

procedure Menu;
var i:integer;
begin
  Feltolt;
  repeat
    writeln();
    writeln('Lehetosegek- 1:Feltölt 2:kiír 3:Rendez 0:Kilép');
    readln(i);
    Case i of
      1: Feltolt1;
      2: Kiir;
      3: Rendez;
    end;
  until i=0;
end;

begin
 randomize;
 Menu;
end.
```

Kérdések, feladatok

- Készítsd el az összetett példaprogram mondatszerű leírását és folyamatábráját!
- A „Veletlen” nevű függvényben melyik programozási tételt használjuk?
- A „Csere” nevű eljárásban milyen paraméterátadást használunk?