Rekurzió egyszeruen és érdekesen

V. rész

Rekurzív eljárások

Az alapveto különbség a függvények és az eljárások között, hogy a függvény kiszámít valamit, és ezt visszatéríti, mint eredményt, az eljárás pedig elvégez valamit. A rekurzív eljárásoknak is fo jellegzetessége, hogy meghívják önmagukat. Természetesen itt is érvényes az a megkötés, hogy a rekurzív hívásnak feltételhez kötöttnek kell lennie, hogy biztosítva legyen a rekurzív hívások láncolatából való kiszabadulás. Ebbol adódóan a rekurzív eljárásokban is általában van egy ún. kulcs IF, amelynek egyik ága rekurzív (itt kerül sor a rekurzív hívásra) a másik pedig nem.

Ha, úgy fogjuk fel a rekurzív eljárást, mint amely ezen kulcs IF köré épül, akkor a következo vázat kapjuk (Pascal változat):

```
Procedure rek_elj (<paraméter_lista>);
<lokális deklarálások>
  BEGIN

      If <feltétel> then                          a zóna
            Begin                                 b zóna

                  Rek_elj (<paraméter_lista>);    B zóna

            End
            Else                                  X zóna
                                                  A zóna

END;
```

Bár a fenti sablon a rekurzív eljárásoknak egy nagyon leegyszerusített vázlata, mégis sokat segíthet a tanulmányozásukban.

Amint megfigyelhetted a kulcs IF rögzítése 5 területet határolt el a rekurzív eljárásban:

a – a kulcs IF elotti terület
A – a kulcs IF utáni terület
b – a kulcs IF rekurzív ágán a rekurzív hívás elotti terület
B – a kulcs IF rekurzív ágán a rekurzív hívás utáni terület
X – a kulcs IF nem rekurzív ága

A rekurzív eljárás bármely utasítása csakis a fenti területek valamelyikére kerülhet.

Ha gondolatban lefuttatjuk a rek_elj eljárást akkor könnyen nyomon követhetjük, hogy az egyes területek utasításai mikor, hányszor és milyen sorrendben hajtódnak végre. Mindez abban segíthet neked, hogy világosan átlásd milyen hatása van annak, ha egy utasítást egy bizonyos területre írsz. Tegyük fel, hogy futtatása során, a rek_elj eljárás – a rekurzió következményeként –, *n*-szer fog meghívódni, ennyiszer lesz átjárva.

A következo ábra bemutatja, hogy a különbözo zónák, hányszor, milyen sorrendben, valamint mely átjárások során kerülnek végrehajtásra.

$$a_1 f_1 b_1 a_2 f_2 b_2 \ldots a_{n-1} f_{n-1} b_{n-1} a_n f_n X_n A_n B_{n-1} A_{n-1} \ldots B_2 A_2 B_1 A_1$$

I I I H

f – feltétel

I – azt jelzi, hogy a feltétel értéke logikai IGAZ

H – azt jelzi, hogy a feltétel értéke logikai HAMIS

– az indexek jelzik, hogy a rekurzív eljárás hányadik átjárásáról van szó

– például B2 jelentése: sor kerül – a második átjárásban – a B zóna utasításainak végrehajtására.

A fenti ábra fontos következtetésekhez vezethet el. Milyen következménye lesz, ha egy utasítás egy bizonyos területre kerül?

a terület: a *hívások sorrendjében* hajtódik végre, *annyiszor ahány átjárás*ra kerül sor.

A terület: *a hívások fordított* sorrendjében hajtódik végre, *annyiszor ahány átjárás*ra kerül sor.

b terület: a hívások sorrendjében hajtódik végre, *de egyszer kevesebbszer* mint a *a terület* hiszen az utolsó átjárásban nem kerül sor a végrehajtására.

B terület: a hívások *fordított* sorrendjében hajtódik végre, *de egyszer kevesebbszer* mint az *A terület*, hiszen az utolsó átjárásban nem kerül sor a végrehajtására.

X terület: csak az utolsó átjárásban hajtódik végre.

A következo részben egy konkrét feladaton fogom bemutatni, miként használható ez a megközelítés rekurzív eljárások írására!

1. Írj rekurzív eljárást, amely karaktereket olvas be vakon, addig amíg '*' karaktert ütünk, a képernyore pedig a következoképpen írja ki oket:

Például, ha a bemenet: ABCD*

Kimenet:

a) *DCBA
b) DCBA
c) ABCDDCBA
d) ABCD**DCBA
e) ABCD*DCBA

Íme a megoldás:

Pascal

```
Procedure eljA;
Var c:char; {mindenik átjárásnak meg lesz a saját c-je}
begin
      c:=readkey; {beolvasás vakon az a zónában}
      if c<>'*' then eljA;
      Write(c); {kiírás az A zónában }
end;
```

C/C++

```
void eljA()
{
      char c; // mindenik átjárásnak meg lesz a saját c-je
      c=getch(); // beolvasás vakon az a zónában
      if (c!='*') eljA;
      putchar(c); // kiírás az A zónában
}
```

Pascal

```
Procedure eljB;
Var c:char; {mindenik átjárásnak meg lesz a saját c-je}
begin
      c:=readkey; {beolvasás vakon az a zónában}
      if c<>'*' then
                  begin
                    eljB;
                   Write(c); {kiírás a B zónában }
                   end;
end;
```

C/C++

```
void eljB()
{
      char c; // mindenik átjárásnak meg lesz a saját c-je
      c=getch(); // beolvasás vakon az a zónában
      if (c!='*')
            {
             eljB;
             putchar(c); // kiírás a B zónában
            }
}
```

Pascal

```
Procedure eljC;
Var c:char; {mindenik átjárásnak meg lesz a saját c-je}
begin
      c:=readkey; {beolvasás vakon az a zónában}
      if c<>'*' then
                  begin
                     Write(c);  {kiírás a b zónában }
                     eljC;
                     Write(c);  {kiírás a B zónában }
                   end;
end;
```

C/C++

```
void eljC()
{
      char c; // mindenik átjárásnak meg lesz a saját c-je
      c=getch(); // beolvasás vakon az a zónában
      if (c!='*')
            {
             putchar(c); // kiírás a b zónában
             eljC;
             putchar(c); // kiírás a B zóná-ban
            }
}
```

Pascal

```
Procedure eljD;
Var c:char; {mindenik átjárásnak meg lesz a saját c-je}
begin
      c:=readkey; {beolvasás vakon az a zónában}
      Write(c); {kiírás az a zónában }
      if c<>'*' then eljD;
      Write(c); {kiírás az A zónában }
end;
```

C/C++

```
void eljD()
{
      char c; // mindenik átjárásnak meg lesz a saját c-je
      c=getch(); // beolvasás vakon az a zónában
      putchar(c); // kiírás az a zónában
      if (c!='*') eljD;
      putchar(c); // kiírás az A zónában
}
```

Pascal

```
Procedure eljE;
Var c:char; {mindenik átjárásnak meg lesz a saját c-je}
begin
      c:=readkey; {beolvasás vakon az a zónában}
      if c<>'*' then
                  begin
                     Write(c);  {kiírás a b zónában }
                     eljE;
                     Write(c);  {kiírás a B zónában }
                   end
                else Write(c);  {kiírás az X zónában }
end;
```

C/C++

```
void eljE()
{
      char c; // mindenik átjárásnak meg lesz a saját c-je
      c=getch(); // beolvasás vakon az a zónában
      if (c!='*')
            {
             putchar(c); // kiírás a b zónában
             eljE;
             putchar(c); // kiírás a B zónában
            }
      else putchar(c); // kiírás az X zónában
}
```

Ennek az öt cikkbol álló sorozatnak a mottójában Comeniustól idéztem, de nem o volt a valaha élt legnagyobb tanító, hanem minden kétséget kizáróan, Jézus Krisztus. Tanítói sikerének a titka többek között abban állt, hogy egyszeruen tudott elmondani mély igazságokat. Ezt gyakran úgy tette meg, hogy muvészien alkalmazott mindennapi életbol vett szemléltetéseket, valamint rávezeto, illetve véleménykikéro kérdéseket. Ezzel a módszerrel nekünk, tanároknak is sikerülhet olyan mély fogalmakat is, mint például a rekurzió egyszeruen és élvezetesen tanítani. Egyetértetek ezzel diákok?

Kátai Zoltán