Programok keretrendszerekkel való ellátása Turbo Pascalban

I. rész

Az elkövetkező lapszámokban megpróbáljuk bemutatni az objektumorientált programozás néhány olyan fortélyát, amely alkalmazható a Turbo Pascal programozási nyelvben, majd rövid programok segítségével bemutatjuk, hogyan illeszthetünk egy programhoz keretrendszert szöveges illetve Windows üzemmódban.

Tervezett cikksorozatunk három problémára tér ki:

— objektumorientált programozás (az objektum tulajdonságai, deklarálása, öröklődés, egységbezárás, többrétűség);

— keretrendszer létrehozása szöveges üzzemmódban Turbo Vision segítségével (menük, státussor, ablakok, *help*-ek létrehozása);

— keretrendszer létrehozása Windows üzemmódban Object Vision segítségével.

Az objektumorientált programozás

Az objektumorientált programozás megpróbálja megközelíteni a természetes gondolkozást, ezért a programozási nyelveket megpróbálták kibővíteni úgy, hogy ezt a tulajdonságot is magába foglalja, vagy megpróbáltak létrehozni olyan nyelveket, melyek eleve erre az elvre épülnek. A Turbo Pascal 5. verziója építi be először az objektum-orientáltságot ebbe a nyelvbe. Az így létrejött programozási nyelv sokkal modulárisabb és strukturáltabb, mint a hagyományos.

Az objektumorientált programozási nyelvet három fő tulajdonság jellemzi:

— egybezárás

— többrétűség

— öröklés.

Az egybezárás

Azt jelenti, hogy az adatokat, adatstruktúrákat és az ezeket kezelő eljárásokat és függvényeket kombináljuk, majd ezeket egységként kezeljük. Ezt az egységet nevezzük **objektumnak.**

A többrétűség

Az objektumok összkapcsolódhatnak és objektumhierarchiát alkothatnak. A többrétűség az jelenti, hogy egy ilyen hierarchián belül egy adott metódus (függvény vagy eljárás) azonosítója akkor is ugyanaz lehet, ha a metódus teste különböző.

Az öröklés

Ha egy objektumból egy újabbat származtatunk, akkor ez örökli az előző objektum egyes adatstruktúráit és metódusait, de ezek átírhatók és/vagy újak hozhatók létre.

Az objektumok deklarálása

Az objektumok deklarálása a *record*hoz hasonlóan a típusdefiníciós részben kezdődik, ezért a továbbiakban összehasonlítjuk e két elemtípust.

Vegyük a következő példát: képzeljünk el egy adatdefiníciót, mely egy vállalat személyzetét tartja nyilván. Tegyük fel, hogy szükségünk van egy személy adataira.

Nézzük meg, hogyan mutat egy **adat** *record*, amely egy személynek az adatait tartalmazza:

```
type
  adat = record
      csnev, sznev   : string [ 30] ;
      kor            : integer;
      beosztas       : string [ 20] ;
  end;
```

Mint ismeretes, egy ilyen tipusú változóra kétféleképpen hivatkozhatunk. Ha a változó mezőire van szükségünk, akkor az eljárás a következő: megjelöljük a változó nevét és ponttal elválasztva az illető mező elnevezését (pl. **valtozo.csnev, valtozo.sznev** stb). Ha viszont az egész változóra szükségünk van, egyszerűen megjelöljük a változó nevét.

Tegyük fel, hogy az alkalmazásunk során szükségünk van arra, hogy tudjuk a személyről, hogy szabadságon van-e vagy sem. Ehhez létre kell hoznunk egy új típust mely tartalmaz egy új mezőt:

```
  szemely = record
      csnev, sznev   : string [ 30] ;
      kor            : integer;
      beosztas       : string [ 20] ;
      szabads        : boolean;
  end;
vagy
  szemely = record
      szem_adat      : adat;
      szabads        : boolean;
  end;
```

Nézzük meg hogyan történik ugyanez objektumok segítségével:

```
type
  adat = object
      csnev, sznev   : string [ 30] ;
      kor            : integer;
      beosztas       : string [ 20] ;
  end;
  szemely = object(adat)
      szabads        : boolean;
  end;
```

Itt az **adat** típus az őstípus, a személy a leszármazottja. Ebből a típusból egy újabb származtatható és így tovább. A származtatott típus mindig örökli az elődei tulajdonságait. Ezek az egymáshoz csatolódó objektumok képezik az objektum hierarchiáját. Egy objektumhierarchián belül minden objektum az őstípusnak nevezett **adat** leszármazottja lesz. Azokat az objektumokat, amelyeket közvetlenül az **adat** típusból származtattuk, közvetlen leszármazottnak nevezzük.

Dávid K. Zoltán

Kolozsvár