Programozás III

PROGRAMOZÁSI ALAPOK

1. Meg akarja-e csinálni a tárgyat ebben a félévben?

igen	64
nem	0
ahogy alakul	0

2. Akar-e hétről-hétre tanulni?

igen	57
nem	3
inkább bulizom, és csak ha marad időm	1
nincs biztos válasz:	1

3. Segítene-e, ha a félév közben is lenne két kötelezően beadandó hf?

igen	56
nem	7
nincs válasz:	1

87.5% igen

4. Mennyire van tisztában az objektumorientált paradigmával?

a) egyáltalán nem	1
b) ismerem, de nem értem	19
c) ismerem és jól értem	36
d) a könyökömön jön ki	3
b-c között	4
b és d (??)	1

MIÉRT TANULUNK TÖBB NYELVET?

A nyelv szerepe:

– gondolatközlés

– gondolatformálás

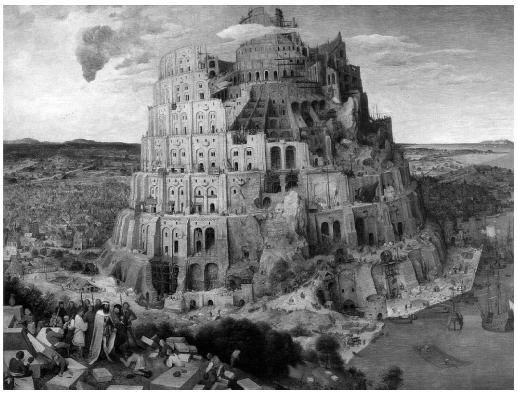
A programozási nyelv szerepe:

– gondolatközlés

– gondolatformálás

NYELVEK

Hány nyelvet beszélnek a világon?



Brueghel: Bábel tornya

kb.:

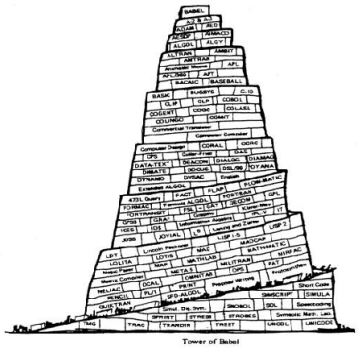
3 – 10 ezer között

ebből írott nyelv kb.:

2-300

PROGRAMOZÁSI NYELVEK

Kb. hány programozási nyelv lehet?



Tower of Babel

Sammet, J.E.: Programming Languages, 1969
(Bild nach Pieter Brueghel, 1563)

Becslések szerint több, mint 4000.

Ebből jóval kevesebb a szabványosított.

PROGRAMOZÁSI NYELVEK

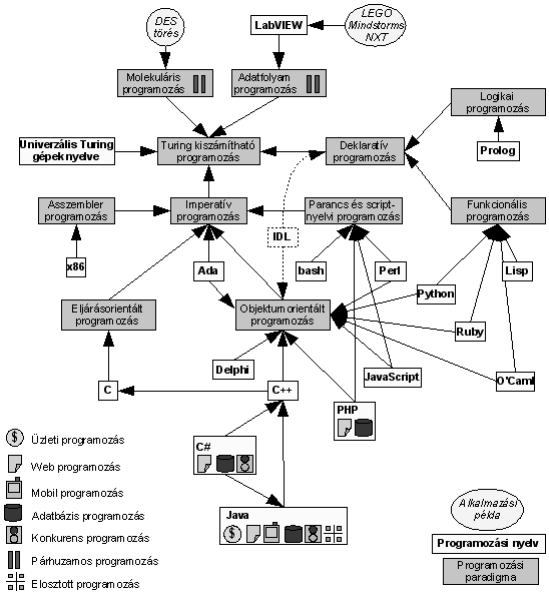
A programozási nyelvek fejlődése:

http://www.levenez.com/lang/

Néhány példa – ld. nehany_nyelv_bemutatasa.zip

Forrás:
korábbi ELTE-s anyag, már leszedték

PROGRAMOZÁSI NYELVEK



PROGRAMOZÁSI NYELVEK

Nyelvek osztályozása (1):

- Paradigmák szerint:
  - blokkszerkezetű, procedurális
  - objektum alapú, osztály alapú, objektumorientált
  - osztott, párhuzamos
  - funkcionális
  - logikai
  - adatbázis

paradigma: Egy tudományterület valamely általánosan elfogadott, zárt nézetrendszere.

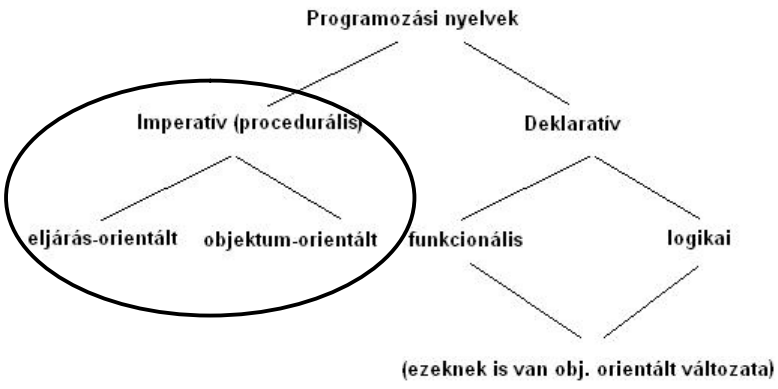
PROGRAMOZÁSI NYELVEK

Nyelvek osztályozása (2):

- A nyelv célja szerint:
  - rendszerprogramozási
  - üzleti
  - tudományos
  - adat alapú
  - lista-kezelő
  - vektor-kezelő
  - string-kezelő
  - parancsfeldolgozó

(Peter Wegner) – Nyékyné jegyzete alapján

PROGRAMOZÁSI NYELVEK



PROGRAMOZÁSI NYELVEK

Imperatív nyelvek:

„Hogyan” oldjuk meg a feladatot – (Turing gép modell)

Változó: egy memóriarekesz kap nevet és értéket.

A program állapota: a változó nevek/értékek és az éppen végrehajtott utasítás.

Program: állapotátmenetek sorozata.

Imperatív program: állapotok és az őket módosító utasítások.

Eljárás: állapotátmenetek bizonyos értelemben jellemző sorozata.

Procedurális programozás: imperatív programozási eljárások együttese.

PROGRAMOZÁSI NYELVEK

Deklaratív nyelvek:

Nem az a kérdés, hogy „hogyan”, hanem, hogy „mi” a megoldandó feladat. $\Rightarrow$ A feladat pontos megfogalmazásán van a hangsúly.

Változó: egy érték kap nevet (mint a matematikai változó fogalma).

A megoldás megkeresése a futtató környezet dolga.

ELMÉLETI ALAPOK

Miért és hogyan tudja megérteni a számítógép a programozási nyelveket?

Kell egy eszköz (algoritmus) a programnyelv értelmezéséhez.

Kell egy eszköz (algoritmus) a program végrehajtásához.

ELMÉLETI ALAPOK

– **Chomsky**: formális nyelvek

Ezen az elméleti alapon működnek a fordító programok

Generatív nyelvtan: Leírja, hogyan lehet előállítani a nyelv lehetséges jelsorozatait. $\Rightarrow$ programgenerálás

Analitikus nyelvtan: Ez alapján eldönthető, hogy a bemenő jelsorozat a nyelvtannal leírt nyelvnek megfelel vagy sem. $\Rightarrow$ szintaktikus elemzés

– **Turing**: automaták

Az automaták ismerik fel a különböző formális nyelveket. $\Rightarrow$ programok végrehajthatósága

ELMÉLETI ALAPOK

Church-Turing tézis:
Tapasztalataink alapján úgy gondoljuk, hogy azokat a feladatokat lehet digitális számítógéppel, tehát valamilyen programozási nyelven megírt algoritmussal megoldani, amelyeket Turing-géppel is meg lehet oldani, és megfordítva: amely feladatokat nem tudunk Turing géppel megoldani, azokat semmilyen algoritmussal (digitális számítógépre valamilyen nyelven megírt programmal) sem tudjuk megoldani.

Irodalom:

Bach Iván: Formális nyelvek,Typotex,2001.

Demetrovics János, J. Denev, R. Pavlov: A számítástudomány matematikai alapjai,Tankönyvkiadó,Budapest,1989.

stb.

PROGRAMOZÁSI ALAPOK

Cél:

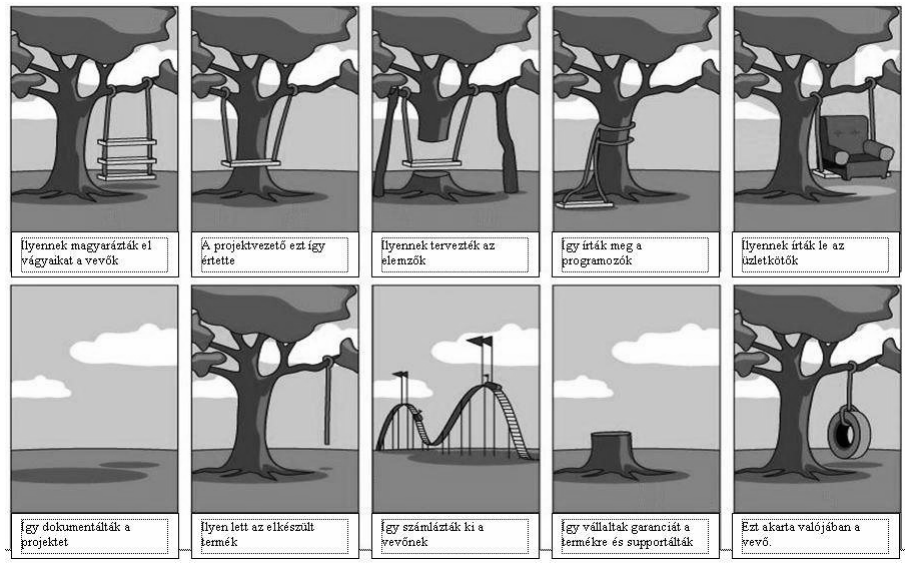
Jó minőségű programtermék

Eszköz:

Programozási nyelv(ek)

Segítség:

Programozási módszertan



Innentől önálló feldolgozás

(ill. inkább a korábban tanultak önálló átismétlése)

PROGRAMOZÁSI ALAPOK

Szoftverek minőségi szempontjai:

- helyesség
- megbízhatóság
- robusztusság (robosztus), biztonságosság
- karbantarthatóság
- újrafelhasználhatóság
- kompatibilitás
- hordozhatóság
- barátságosság
- hatékonyság
- tesztelhetőség
- stb.

PROGRAMOZÁSI ALAPOK – SZOFTVERMINŐSÉG

Helyesség:
A program helyes, ha pontosan megoldja a feladatot és megfelel a kívánt specifikációnak. Alapvető a pontos és minél teljesebb specifikáció.

Megbízhatóság:
Egy program megbízható, ha helyes, és abnormális (a specifikációban nem leírt helyzetekben) sem történik katasztrófa, hanem valamilyen „ésszerű" működést produkál.

Robusztus
egy nyelv, ha megakadályozza vagy futás közben kiszűri a programozási hibákat, **biztonságos**, ha megakadályozza, hogy rosszindulatú programok kerüljenek a rendszerünkbe

PROGRAMOZÁSI ALAPOK – SZOFTVERMINŐSÉG

Karbantarthatóság:
A karbantarthatóság annak mérőszáma, hogy milyen könnyű a programterméket a specifikáció változtatásához adaptálni. Egyes felmérések szerint a szoftver költségek 70 %-át szoftver karbantartásra fordítják!
A karbantarthatóság növelése szempontjából a két legfontosabb alapelv:

- a tervezési egyszerűség
- a decentralizáció (minél önállóbb modulok létrehozása)

Újrafelhasználhatóság:
Az újrafelhasználhatóság a szoftvertermékek azon képessége, hogy egészben vagy részben újra felhasználhatóak új alkalmazásokban.

PROGRAMOZÁSI ALAPOK – SZOFTVERMINŐSÉG

Kompatibilitás:
A kompatibilitás azt mutatja meg, hogy milyen könnyű a szoftver- termékeket egymással kombinálni.

Hordozhatóság:
A program hordozhatósága azt mutatja, mennyire könnyű a programot más géphez, konfigurációhoz, vagy operációs rendszerhez – általában más környezethez – átalakítani.

Barátságosság:
A program emberközelisége, barátságossága a felhasználó számára rendkívül fontos: ez megköveteli, hogy az input logikus és egyszerű, az eredmények formája áttekinthető legyen.

PROGRAMOZÁSI ALAPOK – SZOFTVERMINŐSÉG

A program **hatékonyság**a a futási idővel és a felhasznált memória méretével arányos.
Minél gyorsabb, illetve minél kevesebb memóriát használ, annál hatékonyabb.

Tesztelhetőség:
A tesztelhetőség, áttekinthetőség a program karbantartói, fejlesztői számára fontos.

stb…

PROGRAMOZÁSI ALAPOK – SZOFTVERTERVEZÉS

Tervezési szempontok:

- Jól definiált szintaktikai és szemantikai leírás
- Megbízhatóság
  - kivételkezelés, helyesség-ellenőrzés
- Karbantarthatóság, kiterjeszthetőség
  - új adattípusok definiálása
  - operátorok túlterhelése
- Gyors fordíthatóság, hatékony tárgykód
- Ortogonalitás (a nyelv elemei külön-külön is elérhetőek legyenek és kombinálásukkor ne legyen közöttük kölcsön-hatás)
- Hordozhatóság, általánosság, stb.

PROGRAMOZÁSI ALAPOK – NYELVEK

Mi a programozási nyelv?

- Egy jelölés …
- Eszköz
  - a számítógép vezérlésére
  - programozók közötti kommunikációra
  - algoritmusok leírására
  - magas szintű tervezésre
  - a feladat bonyolultságának kezelésére

PROGRAMOZÁSI ALAPOK – NYELVEK

A Neumann-elvű nyelvek tulajdonságai:

- **utasítás-orientált nyelvek:**
  az utasítások egymás utáni vagy ismételt végrehajtásával kapjuk meg az eredményt.
- A memória rekeszeinek absztrakciójaként bevezetik a **változó** fogalmát.
- **értékadás:**
  A kiszámított értékeket tárolni kell. Az "eltárolás" művelete az értékadás.
- Az utasítások ismételten végrehajthatók.

PROGRAMOZÁSI ALAPOK – NYELVEK

Általános nyelvi fogalmak:

• A nyelv **szintax**isa azoknak a szabályoknak az összessége, amelyek az adott nyelven írható összes lehetséges, formailag helyes programot (jelsorozatot) definiálják.
(Reguláris kifejezések, BNF forma)

• Az adott nyelv programjainak jelentését leíró szabályok összessége a nyelv **szemantiká**ja.

PROGRAMOZÁSI ALAPOK – NYELVEK

Általános nyelvi fogalmak:

• **gépi kód**: olyan bitsorozat, amely a számítógép processzora számára közvetlen utasításként értelmezhető. Pl.: B8 D4 07 00 00 90 48 75 FC C3

• **assembly** nyelv: egy alacsony szintű programozási nyelv, amelyben közvetlen utasításokat adhatunk a processzornak, így nagyon kicsi és nagyon gyors programokat lehet írni. Végrehajtható utasításai általában egy gépi kódú utasításnak felelnek meg. A gépi kódra való fordítást az assembler végzi.

• **magas szintű nyelv**: utasításkészlete független az adott számítógép architektúra utasításkészletétől, végrehajtásuk előtt egy fordítóprogramnak kell őket assembly kódra, illetve gépi kódra fordítani.

PROGRAMOZÁSI ALAPOK – NYELVEK

A programok végrehajtása:

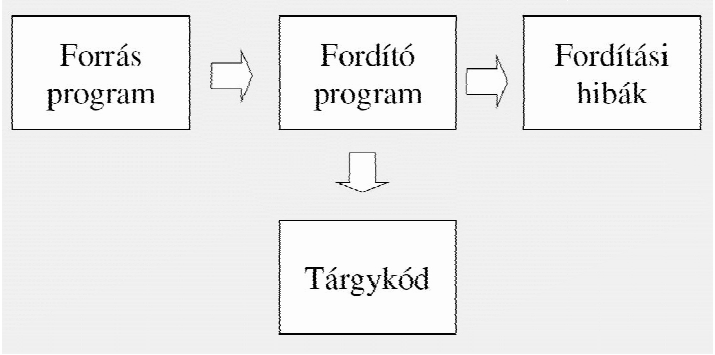
• Az *interpreter* egy utasítás lefordítása után azonnal végrehajtja azt.

• A *fordítóprogram* átalakítja a programot egy vele ekvivalens formára, ez lehet a számítógép által közvetlenül végrehajtható forma, vagy lehet egy másik programozási nyelv.

Fordítás:
A lefordítandó program a *forrásprogram*. A fordítás eredményeként kapott program a *tárgyprogram*.
Az az idő, amíg az utasítások fordítása folyik, a fordítási idő.
Az az idő, amíg az utasítások végrehajtása folyik, a végrehajtási idő.

PROGRAMOZÁSI ALAPOK – NYELVEK

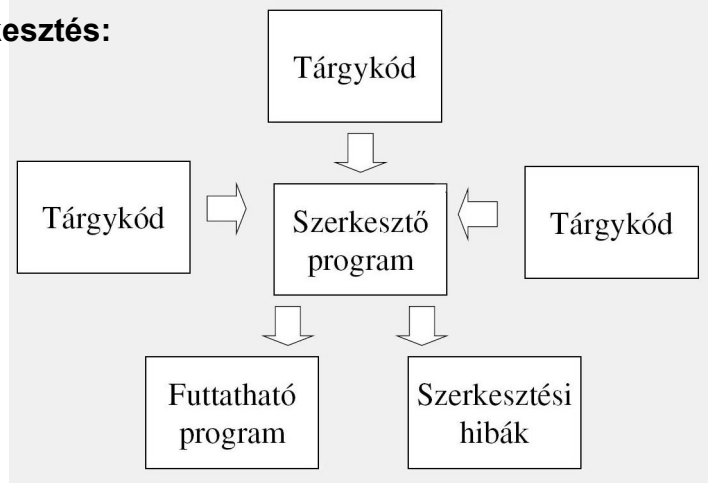
Fordítás:



Fordítási egység vagy modul: A nyelvnek az az egysége, amely a fordítóprogram egyszeri lefuttatásával, a program többi részétől elkülönülten lefordítható.

PROGRAMOZÁSI ALAPOK –NYELVEK

Szerkesztés:



Több modulból álló programok esetében a fordítás és a végrehajtás között: össze kell szerkeszteni a programot.

PROGRAMOZÁSI ALAPOK – KITÉRŐ

Project szemléletű programozás:

A forrásprogramot nem egyetlen nagy egybefüggő egységként kezeljük, és fizikailag sem egy nagy fájlban tároljuk, hanem a logikailag különálló részeket külön fájlban és külön egységként raktározzuk.
Így, ha elég általánosan írjuk meg a modulokat, akkor akár több programban is felhasználhatjuk őket.

PROGRAMOZÁSI ALAPOK – NYELVEK

Programegység, fordítási egység:

• A programegység egy részfeladatot megoldó, tehát funkcionálisan összefüggő programrész.

• A fordítási egység programegységek halmaza.

PROGRAMOZÁSI ALAPOK – NYELVEK

A programegységek részei:

• A **specifikációs rész** írja le az egységnek a más egységből elérhető „kapcsolódási pontjait".

• A **törzs** tartalmazza az egység funkcióját megvalósító programot.

Törzs:

– deklarációs rész (ha van)

– utasítássorozat

PROGRAMOZÁSI ALAPOK – NYELVEK

Programírással kapcsolatos néhány alapfogalom:

- **Hatókör**
  A programszövegnek azt a szakaszát, amelyen belül az adott deklaráció érvényben van, a deklaráció ***hatókör***ének nevezzük.

- **Blokkstruktúra**
  A programegységek egymásba ágyazásával előálló hierarchikus struktúra. Blokkstruktúrában egy programegység deklarációinak hatásköre kiterjed az összes *tartalmazott* programegységre is.

PROGRAMOZÁSI ALAPOK – NYELVEK

Programírással kapcsolatos néhány alapfogalom:

- **Globális - lokális**:
  – Egy programegységben a tartalmazó programegységek által deklarált azonosítókat (a tartalmazott programegység szemszögéből) *globális* azonosítóknak,
  – a programegységben deklarált azonosítókat magában a programegységben *lokális* azonosítóknak nevezzük.

PROGRAMOZÁSI ALAPOK – DEKLARÁCIÓS PÉLDA

```
class Pelda1{

   static int x = 1;

   public static void main(String[] args){
       int y = 2;
       x = x+y;
       kiir();
   }

   static int osszeg(int x, int y){
       int z = 3;
       return x+y+z;
   }

   static void kiir(){
       System.out.println("egy: " + osszeg(x,y));
       System.out.println("kettő: " + osszeg(4,5));
       System.out.println("három: " + (osszeg(4,5) - z));
   }
}
```

Mit ír ki?

PROGRAMOZÁSI ALAPOK – NYELVEK

Programírással kapcsolatos néhány alapfogalom:

- **Memóriakezelés**
  - A változóhoz a memóriaterület hozzárendelése (*allokálás*a)
    - automatikus, a definíció kiértékelésekor
    - a programozó rendelkezik róla
  - a változó által lefoglalt terület felszabadítása
    - automatikus,
    - a programozó hatáskörébe tartozik.
- **Élettartam**

  A változókhoz a szükséges memóriaterület lefoglalása (allokálása) és annak felszabadítása közötti időt a változó *élettartam*ának nevezzük.

PROGRAMOZÁSI ALAPOK – NYELVEK

Programírással kapcsolatos néhány alapfogalom:

- **statikus változó:**
  – élettartama a program egész működése idejére kiterjed
  – mindig az ún. statikus (main) memória-területen helyezkedik el
  – a statikus terület egyszer, a program betöltésekor kerül lefoglalásra
- **dinamikus változó**:
  – a program explicit módon foglal le területet, a dinamikus (heap) memória-területen, amire a címével, ún. pointerrel lehet hivatkozni.
  – egyes nyelvek tartalmaznak utasítást a dinamikus területek felszabadítására is.

PROGRAMOZÁSI ALAPOK – MÓDSZERTAN

Programozási módszertan:

- **moduláris tervezés**: a programot részfeladatokra bontjuk, az egyes részeket külön programban dolgozzuk ki, ezeket a független modulokat lefordítjuk és a végső programban összeépítjük őket.
- **strukturált tervezés**: szintén részfeladatokra bontjuk a programot, de a strukturált programozás fontos alapelve a **lépésenkénti finomítás elve**.
- **objektumorientált tervezés**: a program egészét egyedi jellemzőkkel rendelkező, önmagukban is részben vagy teljesen működőképes, zárt objektumok halmazából építi fel. Egy egységbe foglalja az adatokat és a rajtuk végzett műveleteket.

PROGRAMOZÁSI ALAPOK – MÓDSZERTAN

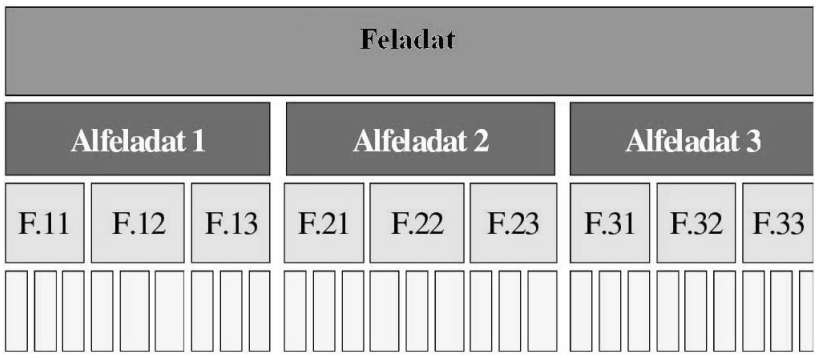
Moduláris dekompozíció:

• A moduláris dekompozíció a feladat több egyszerűbb részfeladatra bontását jelenti, amely részfeladatok megoldása már egymástól függetlenül elvégezhető. Ennek segítségével csökkentjük a feladat bonyolultságát.

• Általában a módszert ismételten alkalmazzuk, azaz az alrendszereket magukat is dekomponáljuk. Ezzel lehetővé tesszük azt is, hogy a feladat megoldásán egyszerre több ember is dolgozzon.
A módszer egy fával ábrázolható, ahol a fa csomópontjai az egyes dekompozíciós lépéseknek felelnek meg.

PROGRAMOZÁSI ALAPOK – MÓDSZERTAN

Moduláris dekompozíció:



PROGRAMOZÁSI ALAPOK – MÓDSZERTAN

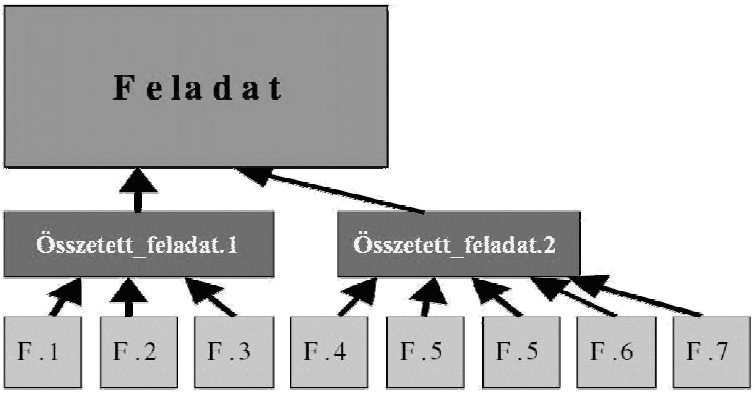
Moduláris kompozíció:

- Olyan szoftverelemek létrehozását támogatja, amelyek szabadon kombinálhatók egymással.
- Programjainkat minél inkább már meglévő program-egységekből szeretnénk felépíteni.

Célunk: a programkészítés során minél több, már létező, szabványos elemet használjunk fel.

PROGRAMOZÁSI ALAPOK – MÓDSZERTAN

Moduláris kompozíció:



PROGRAMOZÁSI ALAPOK – MÓDSZERTAN

A modularitás néhány alapelve :

- ***A modulokat nyelvi egységek támogassák***
  – A modulok illeszkedjenek a használt programozási nyelv szintaktikai egységeihez.
- ***Kevés kapcsolat legyen***
  – Minden modul minél kevesebb másik modullal kommunikáljon!
- ***Gyenge legyen a kapcsolat***
  – A modulok olyan kevés információt cseréljenek, amennyi csak lehetséges!
- ***Információ elrejtés***
  – Egy modullal kapcsolatos információk mindegyike legyen rejtett, kivéve azok, amelyeket explicit módon nyilvánosnak deklaráltunk.

PROGRAMOZÁSI ALAPOK – MÓDSZERTAN

Strukturált programozás:

(E. W. Dijkstra): elkészítendő programunk egy olyan absztrakt gép (*A)*, amelynek egyetlen utasítása van: *„oldd meg a feladatot”*.

Mivel nincs ilyen gépünk $\Rightarrow$ definiálunk egy egyszerűbb B absztrakt gépet (meghatározzuk az utasításkészletét), és ezzel az utasításkészlettel elkészítjük az előző (A) gépet „szimuláló” programot, azaz a feladat megoldását.

PROGRAMOZÁSI ALAPOK – MÓDSZERTAN

Strukturált programozás (folyt.):

Ha nincs ilyen B gépünk sem, az eljárást meg kell ismételni, azaz egy még egyszerűbb C absztrakt géppel és egy programmal most a B gépet kell szimulálni. Az eljárást tovább folytathatjuk.
Akkor vagyunk készen, ha olyan utasításkészlethez jutunk, amelyik már létezik egy konkrét gépen, illetve egy olyan programozási nyelvben, amelynek fordítóprogramja rendelkezésünkre áll.

PROGRAMOZÁSI ALAPOK – MÓDSZERTAN

Strukturált programozás:

Magyarul: Föntről lefelé fokozatosan közelítjük a megoldást. (Top down)

A legkisebb modul az eljárás, amely adatokon dolgozik.
A zártság érdekében megpróbáljuk a globális adatok számát minimalizálni. A lokális adatok elvesznek az eljárásból való kilépéskor $\Rightarrow$ az ilyen zártság nem minden esetben megoldás.

(Wirth): Program = adat + algoritmus

(Böhm, Jacopini, Mills):
Bármely algoritmus leírható a szekvencia, szelekció, iteráció véges sokszori alkalmazásával.

PROGRAMOZÁSI ALAPOK – MÓDSZERTAN

Objektum-orientált paradigma:

Alapja: a természetes emberi gondolkozás

Az objektumorientált modellezés során elvonatkoztatunk, megkülönböztetünk, osztályozunk, általánosítunk, leszűkítünk, kapcsolatokat építünk, stb. $\Rightarrow$ absztrahálunk.

Olyan programozási módszer, amely lehetővé teszi különböző bonyolult változók (objektumok) létrehozását és kezelését.

Objektum-orientált program: Egymással kapcsolatot tartó, együttműködő objektumok összessége, ahol minden objektumnak megvan a jól meghatározott feladata.