Programozási nyelvek kialakulása, fejlődése. Eljárás és objektumorientált nyelvek jellemzői, alapfogalmai. Grafikus fejlesztői környezetek kialakulása, eseményorientált programozás.

1. Programozási nyelvek kialakulása, fejlődése

A kezdetek:

- 1952: Neumann János tervei elkészül az EDVAC, az első Neumann-elvek alapján épített gép volt
- elsősorban matematikai problémák megoldására lett specializálva
- a programok „gép-függőek” voltak
- a programozás tulajdonképpen nem volt más, mint adott utasítások sorozatának leírása
- a programok megírása sok időt vett igénybe

A 60-as, 70-es évek:

- hosszú, nehéz munkával készített programok gyorsan elavultak
- új, gyorsabb, egyszerűbb programfejlesztési módszerek
- első magas szintű programozási nyelvek
- programozási módszerek, technikák, filozófiák jelentek meg
- általános célú és a specializált programozási nyelvek

A 80-as, 90-es évek:

- objektumorientált programozás

Script-nyelvek:

- PHP (szerver oldali), Javascript (kliens oldali), Perl, VBScript (Visual Basic Script), Actionscript
- Ezen scriptnyelvek némelyike tekinthető hibridnyelvnek is, ugyanis lehetőség van bennük eljárás és objektumorientáltan is programozni. Pl.: PHP, Javascript, Actionscript

A programozási nyelvek 4 generációja:

1. generáció:

- gépi kódban írták
- a processzor közvetlenül végre tudja hajtani
- hátrány a gépfüggőség, nehéz programírás
- előnye, a gyorsaság
- Példa: B8 D4 07 00 00 90 48 75 FC C3

- ezt követte az Assembly, nem bináris
- angol nyelvű szöveges kódot kell írni (pl. ADD)
- kapcsolatban áll a gépi nyelvvel
- Assembler szükséges
- Gépi → Asssembly, illetve Assembly → Gépi
- Példa:

```
.globl main
        .type main,@function
main:
        movl    $2004,%eax
loop1:
        nop
        decl    %eax
        jnz     loop1
        ret
```

2. generáció:

- A programokra megnő az igény, széleskörű alkalmazás
- Magasszintű programnyelvek
- Előny: áttekinthetőség, gyors programírás, kevés hibalehetőség
- Hátrány: szükség van fordítóra, ezáltal lassabb a végrehajtás
- FORTRAN:
  - 1957, IBM
  - Formula Translation, numerikus algoritmusok, matematikai formulák kezelése
  - I/O szempontjából nem volt hatékony
  - A megírt programok 2 részből állnak: főprogram, alprogramok
- ALGOL:
  - 1955, matematikai, műszaki számítások
  - támogatja: dinamikus tárkezelés, rekurzió
  - kutatás, oktatás területén
- COBOL:
  - 1955: CODASYL (Conference on Data Systems Language)
  - A CODASYL által meghatározott elvek (gazdasági életben használható adatfeldolgozó nyelv) alapján készült 1963-ban
  - COBOL: Common Business Oriented Language
  - Napjainkban is használják
- BASIC:
  - Kemény János, Thomas Kurtz
  - Interpreteres
  - 1963: BASIC (Beginners All-Purpose Symbolic Instruction Code)
  - nem tette lehetővé a struktúrált programozást, GOTO

3. generáció:
- a nyelvek egy-egy feladatcsoport megoldására specializálódtak
- PL/1:
  - 1964, IBM
  - adatfeldolgozási és számítási problémákhoz
- PASCAL:
  - Az első nyelv, amely megvalósítja a struktúrált programozást
  - Niklaus Wirth, zürichi Műszaki Főiskola
  - Alapja az ALGOL programnyelv
  - Műszaki és tudományos területeken, oktatásnál használták (használják)
  - A program blokkokból épül fel: begin…end
  - Az utasításokat ; zárja
  - Függvények, eljárások, rekurzió
  - Erősen típusos nyelv
- C:
  - 1972, Dennis Ritchie, UNIX rendszerek fejlesztőeszközének indult
  - alprogramok (csak függvény), rekurzió
  - dinamikus tárkezelés
  - 1989-ben ANSI-C, első szabványos C fordító
  - rendszerprogramozásnál használják elsősorban
  - C++ : OOP-ra épül

4. generáció:
- 4GL rendszerek
- nem nyelvek, hanem magasszintű nyelvre épülő komplex, objektumorientált programfejlesztői környezetek
- első ilyen eszközök a 80-as évek közepén
- Példa: Borland Delphi (Object Pascal), Visual Basic (Basic), Visual C++ (C++), Borland JBuilder (Java)

2. Programnyelvek csoportosítása a kód szerkezete alapján

1. Imperatív:
   - Programmal kell megmondani, hogy a feladatot hogyan kell megoldani
   - Algoritmikus nyelvek
   - A program utasításokból áll
   - Legfontosabb eszköze a változó
   - A programozó gondoskodik a tárkezelésről
   - 2 alfajtája van: eljárásorientált, objektumorientált
2. Deklaratív:
   - A programozó deklarálja a feladatot, de a megoldást nem neki kell
   - Létezik változófogalom, de itt mást jelent
   - A programozó nem nyúlhat a memóriához
   - Utasítások vannak
   - Alfajtái: funkcionális (alapja a függvény, pl. Lisp), logikai (mindent logikai fogalmakkal oldanak meg, pl. Prolog)

Eljárásorientált nyelvek:

- általában fordítóprogramosak
- típusosak: ismerik a típus fogalmát, rendelkeznek beépített típusokkal, létre is lehet hozni típusokat
- 2 legfontosabb típusa a tömb és a rekord
- jellegzetes egysége az alprogram
- nyelvi elemek (lásd később):
  - lexikális egységek
  - szintaktikai egységek
  - utasítások
  - programegységek
  - fordítási egységek
  - program

Objektumorientált nyelvek:

- Simula 67 volt az első, amely ezt az eszközrendszert tartalmazta
- Allan Key 1969-ben írt egy dolgozatot, témája az OOP szemlélet
- Smalltalk: az első, amely megvalósítja az OOP-t. Hátrány: erős hardver kell hozzá, a programozói szakma elutasítja ezt a szemléletet
- A 80-as, 90-es években történt meg a változás, sok nyelvnek kihozzák az OOP-s változatát
- Java: 1995
- OOP nyelvek 2 csoportja: tiszta, hybrid

  - Tiszta: OOP alapján lett tervezve, pl. Smalltalk
  - Hybrid: lehet benne struktúráltan és OO módon is programozni, pl. Java, PHP, Actionscript
- OO nyelvek 3 legfontosabb jellemzője: egységbezárás, polimorfizmus, öröklődés

3. Integrált fejlesztői rendszerek (IDE)

Tartalmaznak:

- Szintaxis orientált szövegszerkesztő a program forráskódjának szerkesztésére, ebben sok opció van a felhasználói felület konfigurálásához (program sorainak számozása, színkiemelések, automatikus formázás, a program bizonyos pontjának felvétele könyvjelzőként, makrók készítése)
- Szintaktikus és szemantikus kódkiemelés, kódkiegészítés, zárójelellenőrzés, TO-DO lista
- Figyelmeztetés a hibáknál, automatikus javítás
- Fordítóprogramot (compiler) vagy értelmezőt (interpreter)
- Programindító mechanizmust
- Debuggert (hibakereső), konfigurálható töréspontok, figyelő ablakok (watch) és nyomkövetési lehetőség (trace)
- Program erőforráshasználata terén is segítséget nyújt: CPU teljesítményelemzés, memóriakezelés, szálkezelés, veremfigyelés
- Grafikus-felület szerkesztési lehetőséget (pl.: vizuális menü-designer, beépített minta felületek)
- Csoportmunka terén is segítséget nyújtanak: projektek megosztása hálózaton, beépített chat, konferenciaszoba, dokumentálásban is segítséget nyújt
- Sablonokat és mintaalkalmazásokat tartalmaz

Szintaxis és szemantika:

- Szintaxis: a programnyelv pontosan meghatározott szabályainak összessége, nyelvtana. Pl.: helyesen írjuk le az utasításokat, nem hivatkozunk deklarálatlan változóra
- Szemantika: a program szintaktikailag helyes, lefordul, de nem úgy működik, ahogy a programozó szeretné. Pl.: 0-val való osztás

4. Eseményvezérelt programozás

- A kommunikációs szekvenciát nem a program, hanem a felhasználó határozza meg
- eseményvezérelt felhasználói felületeket minden pillanatban szinte mindenféle kezelői beavatkozásra fel kell készíteni

- kommunikáció kerete az ablak
- több ablak is lehet egyszerre a képernyőn, de csak 1 aktív
- felhasználói beavatkozások operációs rendszer feldolgozása után jutnak el az aktív ablakhoz, amely reagál erre
- Objektumorientált megközelítésben az ablakokhoz egy osztályt rendelünk
- Események: kattintás, egérmozgatás, egér gombjának lenyomása, stb.
- Példa eseményvezérlésre: weboldalon

5. Programozási eszközök

Karakterkészlet: a program legkisebb eleme a karakter. Minden nyelv definiálja a saját karakterkészletét, amelyekből a forrásprogramot fel lehet építeni.
Karakterek:

- Betűk (a..z, A..Z, _)
- Számjegyek (0..9)
- Egyéb karakterek
    - Műveleti jelek: (+, -, *, /, stb.)
    - Elhatároló jelek (; , aposztróf)
    - Írásjelek
    - Speciális karakterek (~ , @)
    - Szóköz

Van olyan nyelv, amely szerint csak a nagybetű a betű. A nyelvek különbséget tesznek kis és nagybetű egyezés szerint:

- Case sensitive (különbséget tesz kis és nagybetű között)
- Nem case sensitive (nem tesznek különbséget)

Lexikális egységek:

- Többkarakteres szimbólum: a nyelv tulajdonít neki egyfajta jelentést, és más értelmezésben nem használható. Pl.: ++, --, ==
- Szimbolikus név:
    1) Azonosító: olyan karaktersorozat, amely betűvel kezdődik, számmal vagy betűvel folytatódhat. A programozó saját eszközeit nevezi el vele. Szabályos azon.: x, alma, _alma, Nem szabályos azon.: 5alma, al%ma
    2) Kulcsszó, alapszó, foglalt szó: olyan karaktersorozat, amely azonosító felépítésű, a nyelv tulajdonít neki jelentést, a programozó ezt nem változtathatja meg. Általában ezek az utasításnevek.
    3) Standard azonosító: azonosító jellegű, a nyelv tulajdonít neki jelentést, ez a programozó által megváltoztatható. Pl.: beépített függvények (pl. pritln), nevesített konstansok
- Címke: megjelölhetjük vele a végrehajtható utasításokat, hogy hivatkozni tudjunk rá, a program más részéből. A címke lehet azonosító vagy egész szám.

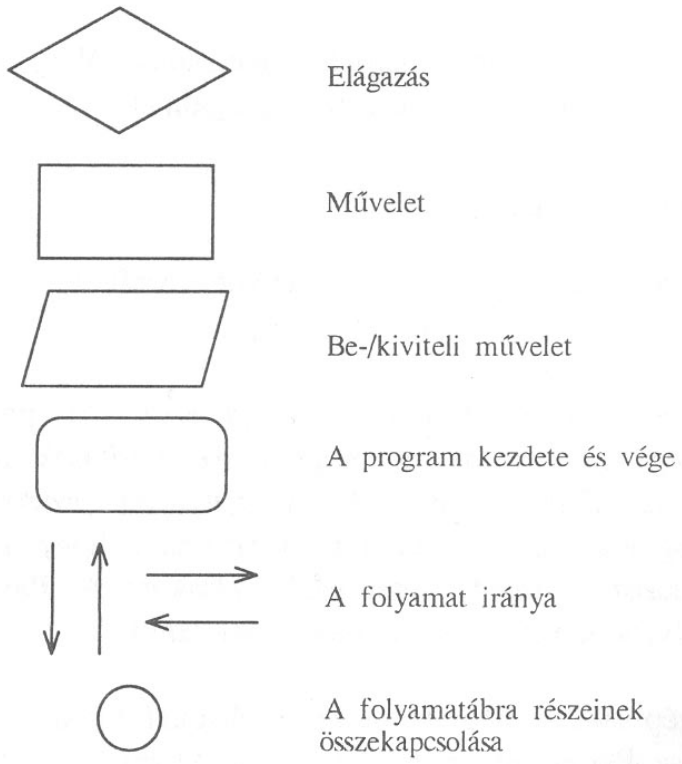
- Megjegyzés: olyan programozási eszköz, amely segítségével nem a fordítóprogramnak, hanem a programozónak szóló üzenetet hagyhatunk a programkódban. A fordítóprogram figyelmen kívül hagyja, ezért tetszőleges karakter lehet benne. Fajtái:
    1) Teljes megjegyzéssor (pl. FORTRAN, a sort 'c'-vel kezdtük)
    2) Egysoros (sorvégi): // után a sor végéig tart
    3) Többsoros megjegyzés: /* …… */
    4) Dokumentációs megjegyzés: /** ….. */ pl. javadoc
- Literál (konstans): olyan programozási eszköz, mellyel fix értékek építhetők be a programba. 2 komponense van: típusa és értéke

6. Az algoritmus megadási módjai

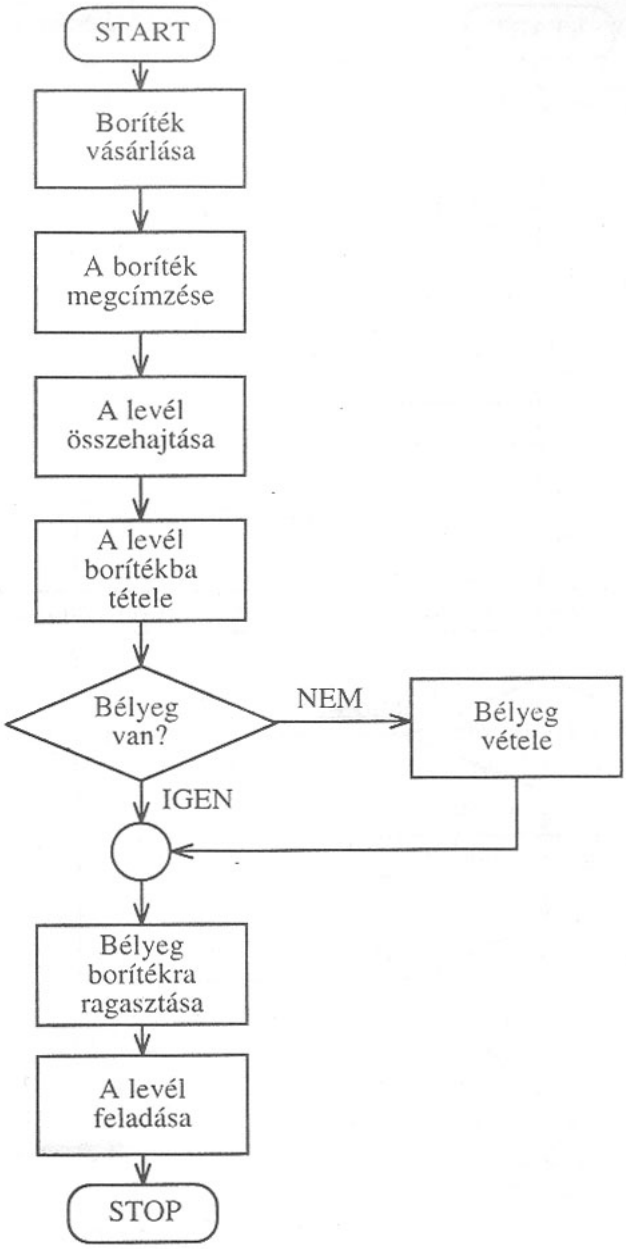
A program írása tervezéssel kezdődik. A tervezés során elkészítjük az algoritmust (egy probléma megoldására kínál megoldási módot).
Az algoritmus megadása 2-féle módon történhet:

- Leírónyelv: szöveges leírás
- Folyamatábra: az algoritmus képi reprezentációja

Folyamatábra elemei:



Egy egyszerű példa feladat:
START
    Boríték vásárlása.

A boríték megcímzése.
A levél összehajtása.
A levél behelyezése a borítékba.
Bélyeg van?
Ha nincs, bélyeg vásárlása.
A bélyeg felragasztása borítékra.
A levél feladása.
STOP

Az algoritmus folyamatábrán: