Imperatív programozás
Bevezetés

Kozsik Tamás és mások

A tárgy célja

- Fogalomrendszer
- Terminológia magyarul és angolul
- Tudatos nyelvhasználat
  - Imperatív programozás
  - Procedurális programozás
  - Moduláris programozás
- Részben: programozási készségek
- Linux és parancssori eszközök használata

A tárgy célja, hogy programozási nyelvekkel kapcsolatos fogalmakkal ismertesse meg a hallgatókat (tudás), melyek alapján a hallgatók a programozás során képesek lesznek tudatosan választani a nyelvi eszközök közül (kompetencia). A tárgyalt ismeretkör az imperatív, a procedurális és (kisebb részben) a moduláris programozási paradigmát fedi le, alapot teremtve későbbi, az objektum-orientált és konkurens programozási paradigmákat tárgyaló kurzusoknak. A tárgy utal a vele egy időben tartott *Funkcionális programozás* kurzusra is (és viszont). A tárgy szoros kapcsolatban áll a vele egy időben tartott *Programozás*, illetve (kisebb mértékben) a *Számítógépes rendszerek* kurzusokkal. Jelen tárgynak nem célja, hogy a hallgatókat programozni tanítsa – ez a *Programozás* kurzus feladata –, de természetesen hozzájárul a hallgatók programozási készségeinek fejlődéséhez.

A tárgy figyelembe veszi, hogy az első félévben szerepel a tantervben, így igyekszik nem építeni meglévő ismeretekre. Másrészt nem „bevezető'' kurzus próbál lenni: a meghatározott ismeretkört teljes mélységében (a BSc-záróvizsga szintjén) át kell adja.

A gyakorlatokon Linux operációs rendszert használunk, így a tárgy hozzájárul ahhoz is, hogy a hallgatók felhasználói szinten megismerkedjenek ezzel az operációs rendszerrel, ezen belül is főleg a parancssor használatával. A nyelvi eszközök tudatos használatát azzal is erősíteni kívánja a tárgy, hogy nem integrált fejlesztői környezetben készítjük majd a programokat, hanem közönséges (programozói) szövegszerkesztőkben. Ennek köszönhetően a programírás nem az eszköz által felkínált lehetőségek közüli választás lesz, hanem egy sokkal tudatosabb folyamat.

Használt programozási nyelvek

- C
- Python

Egyszerre két nyelvvel fogunk megismerkedni a félév során: a C-vel és a Pythonnal. Az előbbi az egyik legprimitívebb *magas szintű programozási nyelv*, melyet régóta használnak nagyon sokféle alkalmazási területen. Jelenleg is az egyik legerőteljesebben használt nyelv. Az egyszerűségéből fakadóan kicsit macerás programozni benne, de a gépközelisége miatt a programozók igen hatékony kódot tudnak írni benne. Sokszor arra is használják a C nyelvet, hogy a más nyelveken írt programokat először erre a nyelvre fordítsák, majd ebből generáljanak gépi kódot.

A Python sokkal fiatalabb nyelv, de napjainkban rendkívül népszerű – főleg azok között, akik nem szakképzett programozók, de a saját szakmájuk művelése mellett időnként szükséges programozniuk is. A Python sokkal magasabb szintű, kényelmesebb nyelv, mint a C. Könnyű benne programot írni – de mint majd látni fogjuk, könnyű benne rossz programot írni is.

A két nyelv nagyon sok mindenben különbözik egymástól, ezért jól össze lehet őket hasonlítani, egymással szembe lehet őket állítani. Nagyon hasznos mindkét nyelv ismerete, és ráadásul a későbbi félévekben tanult további programozási nyelvekhez jó alapot szolgáltatnak majd.

1 Paradigmák és nyelvek

Programozási nyelvek

- Ember-gép kommunikáció
- Ember-ember kommunikáció

A programozási nyelv segítségével vesszük rá a számítógépet arra, hogy kiszámoljon nekünk valamit, illetve, hogy azt csinálja, amit parancsolunk neki. A programozási nyelvnek tehát az az egyik célja, hogy minél jobban megértessük magunkat a számítógéppel. Így a leírt program hatékonyan végrehajtható lesz a számítógépen.

Van azonban egy másik, talán még fontosabb cél is: nagy szoftverek fejlesztésénél a programkód az egyik legfontosabb eszköz a fejlesztők közötti kommunikáció során. Tehát a programozási nyelvnek alkalmasnak kell lennie arra, hogy az egyik ember elmondhassa a gondolatait, és azt egy másik ember könnyen megérthesse. Tehát a programozási nyelvnek az emberi gondolkodási sémákat követőnek (is) kell lennie.

Programozási paradigmák

Gondolkodási sémák, szükséges nyelvi eszközök

Például:

- Imperatív programozás
- Funkcionális programozás
- Logikai programozás

- Szekvenciális programozás
- Konkurens programozás
- Párhuzamos programozás
- Elosztott programozás

- Procedurális programozás
- Moduláris programozás
- Objektumelvű programozás

- Aspektuselvű programozás
- Komponenselvű programozás
- Szolgáltatáselvű programozás
- Szerződésalapú programozás

A programozási paradigmák két kérdésre próbálnak válaszolni.

- Milyen gondolkodási sémát kívánunk kifejezni?
- Milyen eszközökre van szükség ehhez?

Sokféle programozási paradigma létezik: ezek a kurzuson az imperatív és a procedurális paradigmával foglalkozunk részletesebben, de a moduláris programozás témakörébe is betekintünk. Ugyanebben a

félévben egy másik tárgyból a funkcionális programozási paradigma kerül bemutatásra. A logikai programozással inkább a mesterszakon lehet találkozni. A következő félévben az objektumelvű programozással foglalkozunk majd. Az alapszakon előkerül még a konkurens programozás is, míg a párhuzamos és elosztott programozás inkább a mesterszakon kerül elő. A negyedik blokkban felsorolt paradigmákkal is találkozhatunk a mesterszakon.

Egy programozási nyelv akár több paradigmát is támogathat. Például a Python nyelv az objektumelvű paradigmát is támogatja, bár az ehhez kötődő nyelvi elemekkel ebben a félévben nem fogunk foglalkozni.

Imperatív programozás

Akkor beszélünk imperatív programokról, amikor explicit mi vezéreljük, hogy a program hogyan változtatja meg az állapotát. A program utasítások sorozataként van megadva, melyeket egymás után végrehajtunk. Az utasítások a számítógép memóriájába írhatnak, onnan olvashatnak. A memória pillanatnyi tartalma határozza meg a program *állapotát*.

Procedurális programozás

A megoldandó feladatot felbonthatjuk az elvégzendő feladatok (algoritmusok) szerint. Ezeket alprogramokként (függvények, eljárások) valósítjuk meg, köztük pl. paraméterátadással, függvény visszatérő értékével kommunikálunk. Ez a procedurális programozás. Ebben az esetben probléma lehet, hogy háttérbe szorulnak az adatszerkezetek. Pl. FORTRAN, Algol60, C, Go nyelvek.

Kezdetben döntően procedurális nyelvek léteztek, hiszen az assembly programok, a FORTRAN, COBOL, Algol60 és társai ilyen elvek mentén épültek fel, bár a Lisp 1957-ben már funkcionális nyelv volt.

Objektumelvű programozás

Amikor a valós világ objektumait próbáljuk modellezni, akkor összegyűjtjük a hasonló tulajdonságúakat, elhanyagoljuk a feladat szempontjából kevéssé fontos különbségeket és absztrakció segítségével egymással egy szűk interfészen kommunikáló osztályokat alkotunk belőlük. Itt az osztályok adatszerkezetén és a rajtuk értelmezett műveleteken van a hangsúly. Ez az objektumelvű (object-oriented) programozás. Pl. Simula67, Smalltalk, Eiffel, Java, C#.

Deklaratív programozás

Más esetekben egyszerűen csak deklarálni akarjuk a program elvárt működését, nem akarjuk explicit meghatározni annak mikéntjét. Ez a deklaratív programozás, amit több kategóriára szoktak bontani.

Funkcionális programozás

A kívánt eredmény egymást hívó függvényekként van definiálva. Ezek a függvények mellékhatás-mentesek, nincsen értékadás, minden memóriaterület egyszer kap csak értéket, és később ez az érték nem változik (referencial transparency). Az ilyen programok helyességét könnyebb belátni. A számításokat, függvényeket könnyen át lehet adni paraméterként ún. magasabb rendű függvényeknek. Ilyen nyelvek pl. Lisp, ML, Haskell, Clean.

Logikai programozás

A rendszer tényeit és következtetési szabályait adjuk meg. Pl. Prolog.

1.1 Alacsony szintű és magas szintű programozás

Gépi kód

Eleinte a számítógépek programozása úgy történt, hogy bedrótozták (de tényleg, a szó szoros értelmében) a programot. Minden újabb program betáplálása újradrótozást igényelt. Ehhez képest már nagy előrelépést jelentett, amikor újradrótozás nélkül lehetett bevinni a gép memóriájába a gépi kódot. A gépi kódot is nehéz programozási *nyelvként* felfogni.

Assembly

```
quickSort:
.LFB1:
        .cfi_startproc
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        subl    $40, %esp
        movl    12(%ebp), %eax
        cmpl    16(%ebp), %eax
        jge     .L6
        movl    16(%ebp), %eax
        movl    %eax, 8(%esp)
        movl    12(%ebp), %eax
        movl    %eax, 4(%esp)
```

Az assembly nyelvek már egyértelműen programozási nyelvnek tekinthetők. Nagyon közel vannak a számítógép utasításaihoz: a gépi kódhoz könnyebben megjegyezhető, szöveges parancsokat határoz meg.

„Magas szintű'' programozási nyelvek

- Fortran
- LISP

- Algol
- COBOL
- BASIC
- **C**

stb.

Modern, kényelmes nyelvek

- **Python**
- **Haskell**
- C++
- Java
- Ada

stb.

1.2 Programozási nyelvek történelme

Ada Lovelace (Analytical Engine, Charles Babbage)

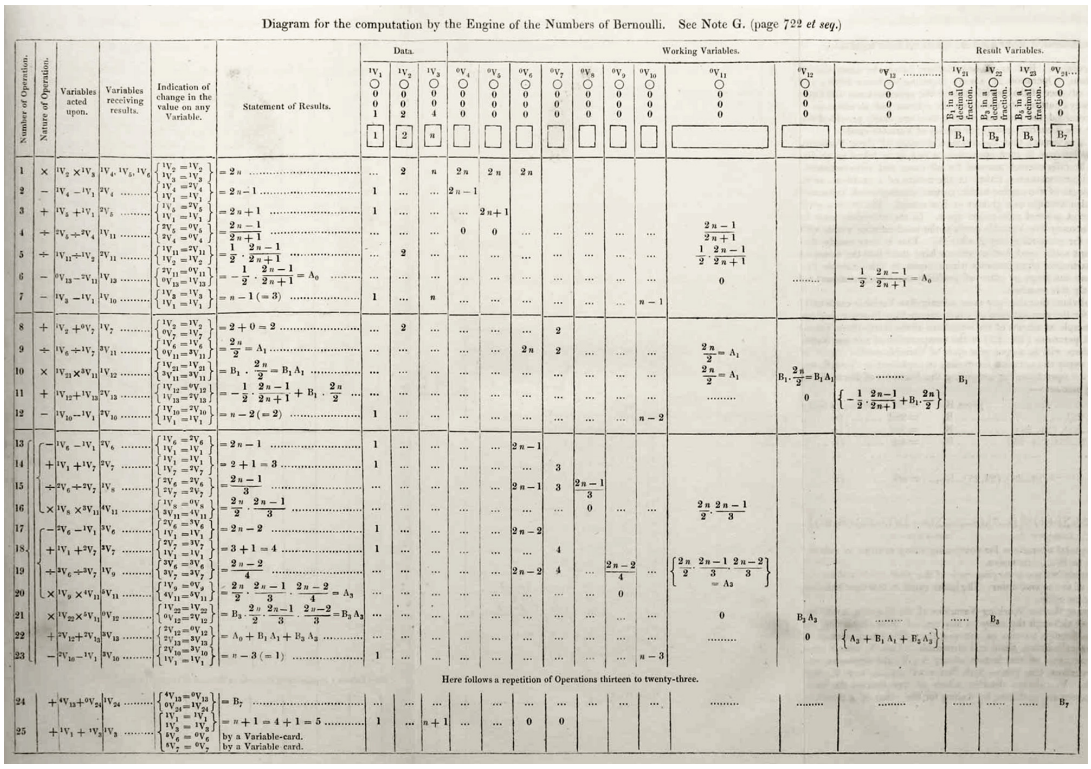


Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

Augusta Ada King, Countess of Lovelace (née Byron, 1815–1852)

A programozás őskora

- Fizikai huzalozás (pl. ENIAC, 1945)
- Gépi kód (Neumann-architektúra, 1945)
- Assembly (1949–)
- Magas szintű programozási nyelvek
  - Plankalkül (Konrad Zuse, 1942–1945)
  - Fortran (John Backus et al., 1954)
  - LISP (John McCarthy, 1958)
  - Algol (1958, 1960, 1968)
  - COBOL (1959)
  - BASIC (Kemény–Kurtz, 1964)

Néhány fontos nyelv

- Simula-67 (Dahl–Nygaard, 1967)
- Pascal (Niklaus Wirth, 1970)
- C (Dennis Ritchie, 1972)
- Ada (1980)
- SQL (Chamberlin–Boyce, 1974)
- C++ (Bjarne Stroustrup, 1985)
- Eiffel (Bertrand Meyer, 1986)

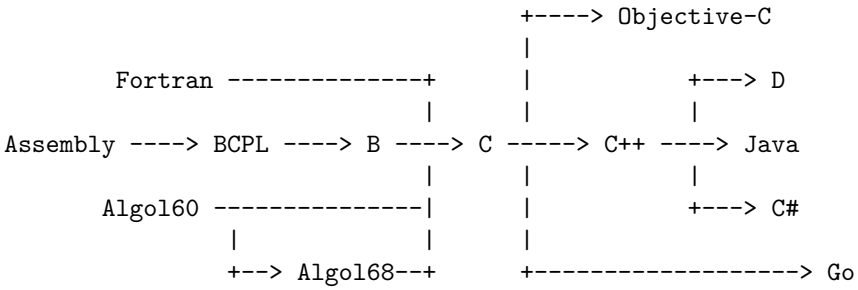
- Erlang (Armstrong–Virding–Williams, 1986)
- Haskell (1990)
- Python (Guido van Rossum, 1990)
- Java (James Gosling, 1995)
- JavaScript (Brendan Eich, 1995)
- PHP (Rasmus Lerdorf, 1995)
- C# (2000)
- Scala (Martin Odersky, 2004)

A Simula-67 volt az első objektum-orientált programozási nyelv. A Pascal is nagyon jelentős, számos másik nyelv származik belőle. Oktatási céllal ma is használják, mert egyszerű és logikus. A többi nyelvet azért soroltam fel, mert ezeket mind tanítjuk az IK-n. Lehet, hogy másokat is, de most ez jutott az eszembe.

Legnépszerűbb nyelvek (2018. szeptember, TIOBE-index)

Sep 2018	Sep 2017	Change	Programming Language	Ratings	Change
1	1		Java	17.436%	+4.75%
2	2		C	15.447%	+8.06%
3	5	⌃	Python	7.653%	+4.67%
4	3	⌄	C++	7.394%	+1.83%
5	8	⌃	Visual Basic .NET	5.308%	+3.33%
6	4	⌄	C#	3.295%	-1.48%
7	6	⌄	PHP	2.775%	+0.57%
8	7	⌄	JavaScript	2.131%	+0.11%
9	-	⌃⌃	SQL	2.062%	+2.06%
10	18	⌃⌃	Objective-C	1.509%	+0.00%
11	12	⌃	Delphi/Object Pascal	1.292%	-0.49%
12	10	⌄	Ruby	1.291%	-0.64%
13	16	⌃	MATLAB	1.276%	-0.35%
14	15	⌃	Assembly language	1.232%	-0.41%
15	13	⌄	Swift	1.223%	-0.54%
16	17	⌃	Go	1.081%	-0.49%
17	9	⌄⌄	Perl	1.073%	-0.88%
18	11	⌄⌄	R	1.016%	-0.80%
19	19		PL/SQL	0.850%	-0.63%
20	14	⌄⌄	Visual Basic	0.682%	-1.07%

A C nyelv kialakulása



A C egy általános célú programozási nyelv, melyet Dennis Ritchie fejlesztett ki Ken Thompson segítségével 1969 és 1973 között a UNIX rendszerekre AT&T Bell Labs-nál. Idővel jóformán minden operációs rendszerre készítettek C fordítóprogramot, és a legnépszerűbb programozási nyelvek egyikévé vált. Rendszerprogramozáshoz és felhasználói programok készítéséhez egyaránt jól használható. Az oktatásban és a számítógép-tudományban is jelentős szerepe van.

A C minden idők legszélesebb körben használt programozási nyelve, és a C fordítók elérhetők a ma elérhető számítógép-architektúrák és operációs rendszerek többségére. (from wikipedia).

A C nyelv fejlődése

- 1969 Ken Thompson kifejleszti a B nyelvet (egy egyszerűsített BCPL)
- 1969 Ken Thompson, Dennis Ritchie és mások elkezdenek dolgozni a UNIX-on
- 1972 Dennis Ritchie kifejleszti a C nyelvet
- 1972-73 UNIX kernel-t újraírják C-ben
- 1977 Johnson Portable C Compiler-e

- 1978 Brian Kernighan és Dennis Ritchie: The C Programming Language könyve
- 1989 ANSI C standard (C90) (32 kulcsszó)
- 1999 ANSI C99 standard (+5 kulcsszó)
- 2011 ANSI C11 standard (+7 kulcsszó)
- 2018 C18 ISO/IEC standard

Mi alapvetően az ANSI C-t, azaz a C90-et fogjuk használni.

A Python nyelv kialakulása és fejlődése

- Inspiráció: ABC, ALGOL 68, APL C, C++, CLU, Dylan, Haskell, Icon, Java, Lisp, Modula-3, Perl, Standard ML
- 1990 Guido van Rossum
- 2000 Python 2.0
- 2008 Python 3.0

Mi a python3-at fogjuk használni, az iparban nagyon sok a python2.7.

Tényleg nagyon fontos, hogy mindig a `python3`-at indítsuk el, mert a sima `python` sok gépteremben a 2.7-es verziót jelenti, és az nagyon más, mint a 3-as. Vannak olyan programok, amik érvényesek az egyik verzióban, de érvénytelenek a másikban, és ami még rosszabb, vannak olyan programok, amelyek mindkét verzióban érvényesek, de mást jelentenek!