részmegoldásokból épül fel. A *c* tömböt a fenti képlet alapján töltöm fel – lentről felfelé, soronként – hiszen leírja miként építhető fel az optimális megoldás, lépésről-lépésre, az optimális részmegoldásokból. Végül a *c[1][1]*-ben lesz az optimális út hossza. Ahhoz, hogy meglegyen maga az út is, annyi szükséges még, hogy végigmenjek a *c* tömbön Greedy módra.

A Dinamikus programozásnak van egy rekurzív változata is, amikor a fenti képletet rekurzívan használom ugyan de – a részeredmények eltárolása által – ügyelek arra, hogy ne oldjak meg többször azonos részfeladatokat.

A következő részben arról olvashatsz, hogy milyen érdekfeszítő következtetésekhez vezethetnek el a technikák ezen párhuzamos bemutatása.

37				
30	25			
25	16	15		
7	15	11	4	
2	5	8	3	1

Kátai Zoltán

A programozási nyelvek osztályozása

Programozási nyelveket több szempont szerint is osztályozhatunk, különféle metszeteket készíthetünk, különböző nyelvosztályokat állíthatunk fel, de az egyes jellemzők közé éles határ nem húzható. Hibrid nyelvekről akkor beszélünk, ha az adott nyelv egy osztályozási szempont szerint több osztályba tartozik. Napjaink programozási nyelveinek többsége hibrid.

Amatőr és professzionális nyelvek

Az *amatőr programozási nyelvekre* az interaktivitás, a sok nyelvi elem, a gyors nyelvi fejlődés jellemző. Ezekben a nyelvekben a programok szerkezete egyszerű, és ezek speciális gépi tulajdonságokra épülnek rá.

A *professzionális nyelvekre* a modularitás, a magas fokú stabilitás és a kevés nyelvi elem a jellemző. Ezen nyelvek igen hatékonyak, sok lehetőséggel bírnak és gépfüggetlen kódot generálnak, vagy a kód átvihető más architektúrájú gépekre is.

Emberközeliség

A *gépi nyelvek* használatával minden hardver lehetőség kihasználható, azonban a memóriacímeket, a memória-kiosztást és a programkódot önerőből kell megvalósítani (a memóriában lévő utasításkódokat közvetlenül a programozó adja meg). A megírt programok gépközeliek, közvetlenül a processzor utasításkészletére épülnek.

Az *alacsony szintű nyelvek* géporientált nyelvek ugyan, de megjelennek a szimbolikus utasítások, azonosítók és címkenevek, megjelenik a feltételes vezérlésátadás fogalma, az eljárások és a visszatérések. Az adatokat deklarálni, definiálni lehet és a tárhely is ennek függvényében foglalódik le. Megjelennek a makrók és a direktívák. A közvetlen kódok helyett rövid, könnyen megjegyezhető szavakat alkalmazunk (mnemonikok) a könnyebb megjegyzés, a jobb átláthatóság kedvéért. Jobban áttekinthetők a címzési módok, a programokba megjegyzéseket szúrhatunk be, külön fordítható egységekkel dolgozhatunk.

A *magas szintű nyelvek* már feladatorientáltak. Megjelenik a típus és a változó fogalma, kifejezések kiértékelésével komoly számításokat lehet elvégezni egyszerűen, megjelennek a ciklusok, elágazások. A nyelvek eljárásokat, függvényeket tudnak használni és komoly paraméterátadó mechanizmusokkal vannak felruházva.

A *metanyelvekre* azért van szükségünk, hogy segítségükkel más nyelveket tudjunk leírni, ezáltal kizárható a különböző deklarációkban meghúzódó többértékűség.

Típusok használata

A *nem típusos nyelvek* esetében ha létezik is a változó fogalma, ez nincs semmiféle típushoz kötve.

A *típusos nyelveknél* megjelenik a típus fogalma, amely meghatározza, hogy a változó milyen értékeket vehet fel, mekkora memóriatartományra van szüksége, milyen műveletek végezhetők el vele stb.

A *szigorúan típusos nyelvek* esetében szigorú szabályok írják elő a típusok közötti átalakításokat, konverziókat.

Alapelvek szerint

A *procedurális nyelvek* egy adott probléma megoldásának algoritmusát írják le.

Az *imperatív nyelvek* osztályába a Neumann architektúrához szorosan kötődő algoritmikus nyelvek tartoznak, amelyeknél fő programozási egység az utasítás, és ezek egymásutánisága vezérli a processzort. A tár bizonyos területén lévő értékeket módosíthatjuk, így változókról beszélhetünk. A programozó mondja meg, hogy mit és hogyan kell csinálni.

A *deklaratív nyelvek* osztályába azok a matematikai logikára, vagy függvényhasználatra épülő nem algoritmikus nyelvek tartoznak, amelyeknél a programozó csak a megoldandó feladatot írja le, a megoldást magát a rendszer végzi el. Ezeknél a nyelveknél nem létezik utasításfogalom, a tárhely értékeit nem lehet módosítani, nem léteznek adatok, vagy ezeknek teljesen más a szerepük.

Az *applikatív nyelvek* függvények változókra történő alkalmazásaival operálnak. Nincs mellékhatás.

A *funkcionális nyelvek* magas szintű függvények használatára és operátor definíciókra épülnek. Az operátorok függvényeket manipulálnak, mintha azok egyszerű adatok lennének.

A *definíciós nyelvek* olyan applikatív nyelvek, amelyeknél a megfeleltetések (értékadások) definíciókként vannak értelmezve.

Az *egyszeres megfeleltetésű nyelvek* esetén egy változó a láthatósági területén csak egyszer fordulhat elő a bal oldalon (egyszer vehet fel értéket).

Az *adatfolyam (dataflow) nyelvek* az adatfolyam architektúrák programozási nyelvei.

A *logikai nyelvek* predikátumokra és relációkra épülnek. Tényekből szabályok segítségével következtetéseket tudnak levonni általában rezolúció-kalkulust használva.

A *megkötésorientált nyelvek* a megoldandó feladatot megkötések sorozataként fogalmazzák meg és oldják meg.

Az *objektumorientált nyelvek* esetén a megoldandó feladatot osztályok definiálásával fogalmazzuk meg és oldjuk meg. Az osztályok zárt egységnek tekintik az adatokat és az őket kezelő eljárásokat. Az osztályokból objektumokat példányosítunk, amelyek egymással kommunikálnak.

A *konkurens nyelvek* segítségével a párhuzamos, konkurens, osztott, többszálas programokat tudjuk megfogalmazni.

A *negyedik generációs nyelvek* (4GL) nagyon magas szintű nyelvek, melyek segítségével a megoldandó feladat természetes nyelven, vagy diagrammok használatával fogalmazható meg. A fordítóprogram választja ki a megfelelő adatszerkezeteket vagy algoritmusokat.

A *lekérdező nyelvek* az adatbázis-programozás fő kommunikációs eszközei, interfészei.

A *specifikáló (leíró) nyelvek* a szoftver vagy hardver tervezésének formális leírását szolgálják.

Az *assembly nyelvek* a gépi kód szimbolikus jelölésére szolgálnak egy adott számítógép architektúrán.

A *köztes nyelveket* a fordítóprogramok használják mint ábrázolás rendszert. Lehetnek szöveges vagy bináris formátumúak.

A *metanyelvek* más nyelvek deklarálására szolgálnak.

Az *egyéb, vagy más alapelvekre épülő nyelvek* (nemkonvencionális nyelvek) képezik az utolsó nagy nyelvosztályt. Ilyenek a különböző párhuzamos, adatfolyam, szisztolikus működést leíró nyelvek, vagy minden olyan nyelv amely egy bizonyos speciális probléma megoldására volt tervezve.

Generációk szerint

Az elektronikus számítógépek nagy generációi tulajdonképpen meghatározták a programozási nyelvek generációit is. Ilyen értelemben beszélhetünk *első (1GL), második (2GL), harmadik (3GL), negyedik (4GL)* és *ötödik (5GL) generációs programozási nyelvekről.*

Az első generációs nyelveket (1946-1955) a teljes mértékű processzorfüggőség jellemezte. Az utasítások bitsorozatok voltak, amelyeket a gép előlapján lévő kapcsolókkal lehetett megadni.

A második generációs nyelvek (1955-1963) tulajdonképpen az assembly színtű nyelveket foglalják magukban, ekkor jelenik meg a mnemonik fogalma, ekkor jelennek meg a fordítóprogramok (*compiler*) és a szerkesztők (*linker*).

A harmadik generációs nyelvek (1963-1973) már magasszintű programozási nyelvek.

A negyedik generációs nyelvek (1973-) napjaink programozási eszközei. Bonyolult lekérdező nyelvek, programkód generátorok, interaktív fejlesztői környezetek, melyek már túl vannak a magasszintű nyelvek osztályán.

Az ötödik generációs nyelvek (1981-) pedig valójában két fogalmat takarnak, egy gép közelebbi, de magasszintű nyelvet, amely tulajdonképpen a számítógép operációs rendszerét jelenti, és egy természetes nyelvet, amely során az ember és gép közötti kommunikáció („programírás") megvalósul.

Számítási modellek szerint

Azon absztrakt modelleket követve, amelyeknek alapján az algoritmusokat végre kell hajtani, a feladatot meg kell oldani, a következő nagy paradigmákat különböztethetjük meg:

- egyszerű operációs paradigma
- a Neumann-féle paradigma
- az automata feldolgozás paradigmája
- az adatbázis-kezelés paradigmája
- a funkcionális paradigma
- a logikai paradigma
- a párhuzamos paradigma
- az objektumorientált paradigma
- a vizuális paradigma
- az ötödik generációs paradigma

Kovács Lehel