A számítástechnika kémiai alkalmazása

Szerkeszti: BAKOS MIKLÓS

A számítástechnika kémiai, művelettani és kémiai technológiai alkalmazásáról még nincs kézikönyv vagy tankönyv. Valószínűleg nem is lesz, legalábbis olyan, amilyenre a mai generációnak lenne szüksége. Ez a generáció ui. a számítástechnika elterjedése előtti korszakban tanulta a szakmáját, most azt kellene megmutatni neki egy ilyen könyvben, hogy a számítástechnika olyan szolgáltatás, amelyet a szakmában intelligens feladatok megoldására igénybe lehet venni. Egy utánunk következő generáció tagjai számára a számítógép épp oly természetes adottság lesz, mint a mai számára a büretta, a logarléc és a tömegspektrográf. Ez az utánunk következő generáció a kémiát, művelettant és a kémiai technológiát számítástechnikai adottságaival együtt fogja tanulni, nem fog számára a probléma felvetődni, hogy miképpen kell a kémiai tudományokba kívülről bevinni a számítástechnikát. Ezért nem is lesz szüksége a számítástechnika kémiai, művelettani és kémiai technológiai alkalmazásáról szóló külön könyvre.

A mai vegyész- és vegyészmérnök-generáció számára a számítástechnika „szakmaidegen". Nehezen vesz tudomást róla és ímmel-ámmal kezdi tanulgatni. A haladást gátló szociológiai vonatkozások elemzése nem feladatunk, annyi azonban igen, hogy megkíséreljük a számítástechnikát szakmai közelbe hozni és megmutatni, hogy a mai generáció tagjainak nem kell e l ő b b számítástechnikai szakemberré átalakulnia, hogy majd u t ó b b a számítástechnikát eredeti szakmájában felhasználhassa.

A cikksorozat első három cikke a számítógépről és a programozásról szól. Nem a programozás oktatása a célunk e cikkekkel, hiszen kitűnő programozási tankönyvek jelentek meg magyarul is. De az a vegyész kolléga, aki nem akarja a programozás egyébként szórakoztató mesterségét elsajátítani, mégsem fecsérli az időt azzal, ha ezeket a cikkeket elolvassa, mert megismerkedik a számítástechnika és a programozás tolvajnyelvével, azzal a tolvajnyelvvel, amely menthetetlenül és visszavonhatatlanul elterjed a tudományban és technikában: a kémiában, a művelettanban és kémiai technológiában is.

Ezt a tolvajnyelvet fel kell használnunk a következő 10—12 cikkben, amelyekben épp azt akarjuk kifejteni és példákon mutatni, hogy a számítástechnika hogyan használható a kémiában, művelettanban és kémiai technológiában: mi minden van máris előkészítve arra, hogy a mai generáció a számítástechnikai szolgáltatásokat igénybe vegye.

Benedek Pál

Az elektronikus számítógépekről

BENEDEK PÁL*—
MEZEI MIHÁLY*

Az elektronikus számítógépek nagyszerűségéről, nélkülözhetetlenségéről, elterjedéséről manapság számos kijelentést lehet hallani. Ezek már közhelyszámba mennek, de ennek ellenére a legnagyobb részük igaz. Az ipar és a technika nagyiramú fejlődése, a tervezés mind szélesebb körben való elterjedése olyan problémákat vet fel, melyek sok szempont figyelembevételén alapuló gyors döntést igényelnek.

Ha például egy sok terméket előállító vegyi gyárban valamelyik üzemegység hirtelen leáll, akkor az összes ezzel kapcsolatban álló üzem termelését módosítani kell. Természetesen ilyen módosítást sokféleképpen lehet végrehajtani, mondjuk úgy, hogy az egész vegyi gyárat leállítják, vagy pedig leállítják azokat az üzemeket, amelyek a leállt üzem termékeit használják fel. Elképzelhető természetesen, az is, hogy az érintett üzemeket valamilyen más lehetséges üzemmódra állítják át, és így folytatják a termelést. Már az is időigényes feladat, hogy megállapítsuk, az új állapothoz tervezett üzemmódok egyáltalán megvalósíthatók-e (vagyis összhangban van-e az egyik üzemrész termelése a másik anyagszükségletével stb.). A probléma sokszorosára bővül, ha meg akarjuk keresni azt az új üzemmód-kombinációt, ami optimálisnak tekinthető, azaz a legtöbb nyereséget hozza a vegyigyárnak az adott szituációban.

Hasonlóképpen a legkülönbözőbb területekről mutathatunk olyan problémákat, amelyek elektronikus számítógépek nélkül nem oldhatók meg. A Holdra simán leszálló űrhajó rakétamotorjának irányítása mindig az adott helytől és sebességtől függ, az ebből adódó korrekciókat viszont nagyon gyorsan kell végrehajtani (még mielőtt az űrhajó a Holdnak ütközne).

* Magyar Vegyipari Egyesülés Mérnöki Irodája, Budapest

A kibernetika kiinduló pontja egy hasonló típusú probléma volt a második világháború alatt: Hogyan lehet lelőni egy ellenséges repülőgépet. Nyilvánvaló, hogy a gép pályáját rövid idő alatt kell nagyon pontosan meghatározni, nehogy kijusson a légvédelmi ágyuk hatósugarából. Az már akkoriban kiderült, hogy a feladatot „jól célzó tüzérek" nem képesek megoldani.

A mai társadalomban az elektronikus számítógépek alkalmazása egyes területeken a szó szoros értelmében élet-halál kérdésévé válik. A társadalmi lét más területein a helyzet nem ilyen drámai, de úgy tűnik, hogy az elektronikus számítógépek alkalmazása bizonyos mértékben meggyorsíthatja a fejlődést a kémiai tudományokban is és a vegyiparban is. A vegyész ugyanis nagyon sokféle eszközt használ mindennapi munkájában a bürettától kezdve mondjuk a tömegspektrométerig és az elektronikus számítógép is belépett és egyre nagyobb mértékben lép ezeknek az eszközöknek a körébe. Akkor, amikor bürettát vagy tömegspektrométert használunk napi gyakorlatunkban, akkor többé-kevésbé ismerjük a bürettának vagy a tömegspektrométernek a működését legalábbis lényeges vonásaiban. Ha tehát az elektronikus számítógépet is olyan eszköznek tekintjük, mint az előbbieket, vagyis olyannak, amelyet a vegyész napi munkájában felhasználhat, akkor nyilván ennek az eszköznek a működését is nagy vonásaiban ismernünk kell.

Az elektronikus számítógépek felépítése

Az elektronikus számítógépeknek két, egymástól alapvetően különböző típusa van: léteznek *analóg* és *digitális* gépek. A különbség onnan adódik, hogy az analóg gépekben a számokat fizikai mennyiségek (áramerősség, szögelfordulás stb.) képviselik, míg a digitális gépekben a számokat jegyenként a kettes számrendszerben ábrázolják.

Jelen dolgozatban a digitális gépekről, felépítésükről szeretnénk ismertetést adni a felhasználó szempontjából. Ezen azt értjük, hogy noha a gép szerkezeti elemei természetesen mindig egyben műszaki elemek is, erre legfeljebb az elnevezésekben utalunk. Ha pl. mágnesszalagról beszélünk, akkor nem egy speciális magnó-szalagra gondolunk (noha fizikailag erről van szó), hanem egy olyan egységre, amelyik a gépen belül van, és képes számokat tárolni és reprodukálni adott sebességgel.

A digitális gépek legfontosabb tulajdonsága a következő:

Azt mindenki tudja, hogy az elektronikus számítógépek nagy sebességgel végeznek aritmetikai műveleteket. Ez azonban önmagában nem elég. Hiába végeznénk el egy szorzást mondjuk egy ezredmásodperc alatt, ha az eredményt le kellene írnunk, vagy be kellene táplálnunk a gépbe egy újabb művelethez, mint a mechanikus vagy elektromechanikus asztali számológépek esetében, mert akkor nem használnánk ki a műveleti sebességet.

Az elektronikus számítógépek hatékonyságát a nagy műveleti sebesség mellett a *programozhatóságuk* adja, vagyis az a tulajdonságuk, hogy nemcsak számokat képesek beolvasni (és természetesen tárolni), hanem olyan utasítássorozatot is, amely megmondja, hogy a beolvasott számokkal milyen műveleteket végezzen el, az illető műveletek eredményeit milyen további műveletekhez használja fel, s. i. t. Így érhetjük el azt, hogy a számítási idő legnagyobb részét most már a műveletek végrehajtása töltse ki.

Nézzük meg milyen szerkezeti elemekből épül fel egy digitális számítógép.

Valamennyi digitális számítógép általános jellemvonása, hogy az egyes műveleteket egy meghatározott helyen, az illető műveletnek megfelelő műveleti egységben végzi. (Esetleg ilyen egység a gépben több is van, de ez nem változtat a lényegen.) Ebből az következik, hogy a műveleteket a gép időben egymás után hajtja végre, tehát kell egy olyan egységnek lennie, ahol a részeredményeket (és természetesen az előre beolvasott, vagy még ki nem adott végeredményeket) tárolni tudja. Hasonló egységre van szükség a feladat utasítássorozatának, a programnak a tárolásához. Ezt a két feladatot a gép egy közös egységgel, a *memóriával* oldja meg.

Végül ahhoz, hogy a gép adatokat kaphasson és adhasson, kell egy beviteli, más néven *input* és egy kiviteli, másnéven *output egység*.

A memóriával szemben alapvető követelmény egyrészt az, hogy az *elérési idő*, vagyis az az idő, ami egy szám tárolásához, vagy valamilyen művelet számára való előkészítéshez kell, olyan nagyságrendű legyen, mint a művelet ideje, (mert ellenkező esetben, mint már rámutattunk, a nagy műveleti sebességet nem tudnánk kihasználni), másrészt pedig az, hogy minél több adatot lehessen benne tárolni. A két követelmény azonban ellentmond egymásnak. Ezt a két ellentmondó követelményt úgy próbálják kielégíteni, hogy különböző típusú memóriaegységeket építenek be a gépbe, különböző tárolókapacitással és különböző elérési idővel.

A leggyorsabban elérhető memóriaegység, a *központi memória*, másnéven ferrit, (ui. fizikailag ferritgyűrűkből épül fel, melyek két különböző irányú mágnesezettségű állapota 0-t vagy 1-et jelent). Ez viszonylag kis kapacitású (néhány ezer, nagyobb gépeknél néhány tízezer szám), de az elérési idő a műveleti időknél általában kisebb.

A központi memória különböző, nagyobb elérési idejű, de nagyságrendekkel nagyobb kapacitású memóriaegységekkel áll kapcsolatban. Ezek a *háttérmemóriák*. Megfelelő adatmozgató utasításokkal elérhető az, hogy csak azok az adatok (esetleg a programnak is csak az a része) legyen a ferritben, amelyekre a számolás aktuális szakaszában szükség van, a többiek a háttérmemóriákban legyenek. A háttérmemória a gyakorlatban mágnesdob, mágneslemez (diszk), mágnesszalag vagy mágneskártya stb.

A számítógép és a felhasználó közötti információcsere

Mindazt, amit a gépekkel közölni akarunk, betűk és számok segítségével kell megfogalmaznunk. A gép számára azonban egy írott vagy nyomtatott betű felismerése is bonyolultabb feladat, mint azt

az első pillanatban gondolná az ember, ezért ezeket kódolnunk kell egy, a gép által felismerhető formába. A kódolás többféleképpen oldható meg, leggyakrabban papírszalag vagy kártya lyukasztásával. Ennek az a lényege, hogy minden betűnek és számjegynek egyetlen lyukkombináció felel meg, és a különböző lyukkombinációk egymásután alkalmazása a nekik megfelelő betűk vagy számjegyek egymásután való leírásával egyenértékű. (Lyukkombináción azt értjük, hogy egy sorban hol van lyuk és hol nincs.) Attól függően, hogy egy számjegy, ill. betű (közös nevükön *karakter*) kódolására maximálisan hány lyukat használhatunk, beszélünk különböző csatornás kódokról. A Telex kód pl. 5 csatornás, a számítógépeknél gyakran használt Flexowriter kód pedig 8 csatornás. Gyakorlatilag tehát arról van szó, hogy azt a szöveget vagy számadatrendszert, amelyet a géppel közölni akarunk, olyan írógépen írjuk le, amely az írással egyidejűleg elvégzi a kódolást és lyukszalagot ad ki. Ezt a lyukszalagot nagy sebességgel képes a számítógép beviteli egysége letapogatni.

A kivitel történhet úgy is, hogy a gép, megfelelő átalakító berendezés segítségével, mindjárt a kívánt karaktereket írja ki a kódok helyett, például írógépre vagy sornyomtatóra, (azaz egy olyan berendezésre, amely egyszerre egy sort nyomtat, tehát az írógépnél sokkal gyorsabb), de történhet ugyanolyan formában is, mint a bevitel. Ebben az esetben a kiviteli egységekről kijött anyag, az output, egy további gépi számolás vagy számolások inputja lehet, vagy pedig külön berendezés segítségével kell azt olvasható formába átalakítani.

Az itt elmondotak tehát azt jelentik, hogy a bevitel kódolási folyamattal jár együtt, a kivitel pedig dekódolással. A kódolás és dekódolás megoldható az input/output egységgel is (például a géppel közvetlen kapcsolatban álló írógép esetén), de megoldható a számítógéptől függetlenül is (pl. szalaglyukasztó és olvasó írógép segítségével).

A számítógép-„generációk"

A kereskedelemben, vagyis mindenki számára hozzáférhetően a számítógépek kb. 15 esztendővel ezelőtt jelentek meg és ez alatt a 15 év alatt a számítógépek három egymást követő nemzedékével ismerkedhettünk meg. Az első generációba tartozó számítógépeket az jellemezte, hogy elektroncsövekből építették fel őket, a második generáció gépeit félvezetőkből építették, a harmadik generációba tartozókat pedig integrált áramkörökből. Az egyik generációról a következőre való átváltás azzal járt, hogy csökkent a számítógép térfogata, súlya és elektromos teljesítményfelvétele, míg egyidejűleg megnövekedett az üzembiztonság, a műveleti sebesség és a gép központi memóriája, és a számítógépek egyre nagyobb méretű feladatok megoldására váltak alkalmasakká. A harmadik generációs számítógépek körében további új lehetőségeket is megvalósítottak. Nem jelentett még elvi újdonságot az, hogy egy számítógépet több terminállal* láttak el, amelyek esetleg a számítógéptől néhányszáz kilométer távolságban helyezkedtek el, de az igen, hogy kidolgozták az időosztásos számítógépeket, amelyek egyidejűleg több feladat elvégzésén képesek dolgozni és végül megvalósították több számítógép összekapcsolásának különböző módozatait. A számítógépek negyedik generációját mikrointegrált áramkörökből fogják felépíteni. A mikrointegrált áramkörökből épített számítógépek térfogata várhatóan már nagyon kicsi lesz és alighanem a mai legnagyobb teljesítményű számítógépeket mikrointegrált áramkörökből épített gép esetén össze lehet csomagolni egy akkora térfogatban, mint amilyet most mondjuk egy 40× ×45×20 cm méretű asztali számítógép képvisel. Téves lenne azonban erről arra a következtetésre jutni, hogy az ilyen típusú számítógépek a vegyész íróasztalán helyet kapnak. Nemcsak arról van szó, hogy ezek a gépek valószínűleg nagyon drágák lesznek. A központi egység még nem minden.

A központi egységhez perifériális berendezések tartoznak: adattárolók, valamint beolvasó és kiíró szerkezetek. Ez utóbbiakon keresztül valósul meg az ember—gép kapcsolat. A perifériális berendezések méretcsökkentése korántsem megoldott, habár ezeknél is meg lehet különböztetni három generációt.

* Terminálnak nevezik a számítógép olyan adatbeviteli és -kiviteli végegységét, amely a géptől távol van elhelyezve és amellyel az összeköttetést megfelelő telex-, ill. telefonvonal valósítja meg.

A számok és utasítások ábrázolása a számítógépben

A következőkben a számok és utasítások ábrázolásáról lesz szó. Azt már említettük, hogy ezt a két feladatot a gép ugyanabban az egységben, a memóriában oldja meg. Ehhez még azt is hozzá kell tenni, hogy azonos módon. A memória cellákból áll, amelyekben egy szám, vagy egy utasítás tárolható. Minden cellában bizonyos számú elektronikus vagy mágneses elem van, melyek mindegyike különböző állapotban lehet. Az egyes elemek aktuális állapota mondja meg azt, hogy milyen szám, vagy milyen utasítás van a cellában.

Maradjunk egy kicsit a számoknál. Említettük, hogy a digitális gép kettes számrendszerben számol, azaz a számok kettes számrendszerben felírt számjegyeit ábrázolja. Ha a tízes számrendszerhez ragaszkodnánk, akkor vagy olyan elektronikus elemeket kellene alkalmazni, melyek kettő helyett tíz különböző állapotban lehetnek, vagy pedig négy kétállapotú elemet kellene felhasználni egy számjegy ábrázolásához. Az első esetben bonyolultabb elemekből kellene felépíteni a gépet, a második esetben feleslegesen sok elemre volna szükség, mert a négy kétállapotú elem 16 különböző számjegyet is tudna ábrázolni, tehát nem lennének kihasználva.

A számábrázolásnak két alapvető típusa van: *fixpontos* és a *lebegőpontos*. *Fixpontos* számábrázolás esetén az egyes elemekhez állapotuktól függően a 0, ill. 1 számjegyeket rendeljük hozzá, és a tizedespontot előre meghatározott helyen, (pl. az utolsó számjegy után) tételezzük fel.

Lebegőpontos számábrázolás esetén a számot normált alakjában ábrázoljuk, tehát a számot a mantissza és az exponens együttesen határozza meg. Ez a két szám azonban nem kerül két külön-

böző cellába, hanem a cella elemeinek egy részét (pl. az első tíz elemet) külön fixpontos számnak tekintjük, ez ábrázolja a mantisszát.

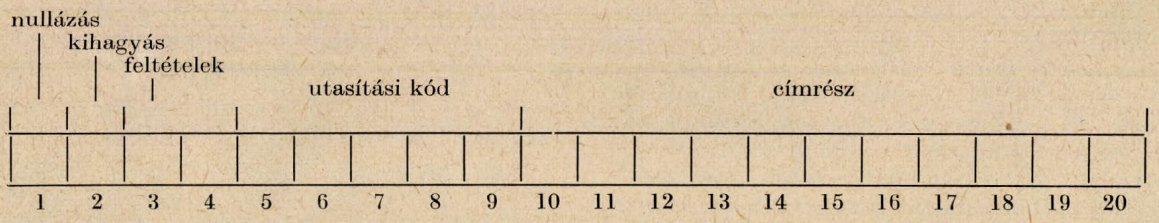
Látható, hogy ily módon ugyanannyi elem segítségével lényegesen nagyobb számokat is tudunk ábrázolni, mint fixpontosan.

Egy kétállású elem állapotának ismerete (feltéve, hogy a két állapot bekövetkezése egyenlő valószínűségű) az információelmélet szerint egy bites információ. Ennek alapján egy cella elemeit bitnek is szokás nevezni és az általuk megvalósított állapotot bit-kombinációnak.

Láttuk, hogy a bit-kombinációkkal hogyan lehet számokat ábrázolni. Amennyiben egy bit-kombináció utasítást reprezentál, abban az esetben minden pozíciónak megvan a maga speciális jelentése, és a végrehajtandó műveletet ezek a speciális információk határozzák meg. Belátható azonban, hogy ily módon csak igen egyszerű utasításokat tudunk ábrázolni.

Legyenek pl. gépünkben 20 bites cellák, és legyen a központi memóriában $1024 = 2^{10}$ cella. Ekkor egy cella annyi információt tud csak adni, amennyit 20 Bar-kochba típusú kérdésre adott válaszból nyerhetünk (tehát 20 bitet), ami viszont nem sok, mert csupán annak eldöntésére, hogy a memória melyik cellájából (a memória cellái meg vannak számozva) vegye a gép a művelet kiindulásaként szolgáló számot, 10 kérdésre van szükség a gépünknél.

Tételezzük fel, hogy összeadást úgy tud a gépünk végezni, hogy egy konkrét, speciális cella értékéhez, a szummátorhoz hozzáadja egy kiválasztott cella értékét. Az erre vonatkozó utasításnak tartalmaznia kell a kiválasztott cella sorszámát (ez az utasítás címrésze), és az összeadás műveletének kódszámát. Természetesen külön utasítással lehet fixpontos, ill. lebegőpontos számokat összeadni, ami azt jelenti, hogy kétféle utasításnak kell lennie az összeadásra. A cellaszám, mint már láttuk 10 bitet foglal el, mert a 2^{10} cella közül kell választani. Amennyiben 32-nél kevesebb műveletet ismer a gép, a művelet kódszáma 6 bitet foglal el. A fennmaradó két bitnek még lehet valami speciális jelentése, pl. ha az 1. bit egyes, akkor a művelet végrehajtása előtt a gépünk a szummátort nullázza, ha pedig a 2. bitje 1, akkor nem a soron következő, hanem az azután következő cellából veszi a következő utasítást, a 3. és 4. bitek 0 vagy 1 volta szerint a gép az utasítást csak bizonyos feltételek teljesülése esetén hajtja végre (például csak akkor, ha a szummátorban pozitív szám áll):



Egy ilyen utasítás tehát csak egyetlen matematikai művelet elvégzését tudja vezérelni (esetleg bizonyos feltételektől függően).

Meg kell jegyeznünk azt, hogy a gép nem „tudja", hogy egy adott cellájában szám van-e, vagy utasítás. A program a feladat végrehajtását egy adott cellánál kezdi, feltételezi, hogy ott utasítás van, azt végrehajtja, majd a következő sorszámú cellából veszi a következő végrehajtandó utasítást. Annak elkerülésére, hogy a gép ne tekintsen minden cellát utasításnak (ami programozási hiba következtében könnyen előfordulhat és beláthatatlan eredményekhez vezet), lehetőség van olyan utasítás beiktatására, hogy a következő végrehajtandó utasítást ne a soronkövetkező, hanem egy másik cellából vegye.

Definiáljuk feltételezett gépünkön a következő műveleteket:

a) Adja hozzá fixpontosan a szummátorhoz a kiválasztott cella értékét
(kódszám: 20)

b) Legyen a kiválasztot cella értéke a szummátor értéke
(kódszám: 21)

Ekkor pl. ha a 15. és a 16. cellában levő számokat össze akarjuk adni, és az eredményt a 20. cellába kívánjuk helyezni, akkor ezt elérhetjük gépünkkel, amennyiben a 17—19 cellában a következő utasítások vannak:

15. cella:
|0 0 0 0 0 0 0|1 0 0 0 0 0 0|1 1 0 0 0 0|
4144 fixpontosan ábrázolva

16. cella:
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1|
11 fixpontosan ábrázolva

17. cella:
|1|0|0 0|0 1 0 0|0 0 0 0 0 0 1 1 1 1|
A törölt szummátorhoz hozzáadjuk a 15. cella tartalmát (4144-et).

18. cella:
0|0|0 0|0 1 0 1 0 0|0 0 0 0 0 1 0 0 0 0|
A szummátorhoz hozzáadjuk a 16. cella tartalmát (11-et).

19. cella:
|0 1|0 0|0 1 0 1 0 1|0 0 0 0 0 1 0 1 0 0|
Lehozzuk a szummátorban levő összeget (4155-öt) a 20. cellába, és gondoskodunk arról, hogy a 20. cella tartalmát ne tekintse most a gép utasításnak.

20. cella:
|0 0 0 0 0 0 0|1 0 0 0 0 0 0|1 1 1 0 1 1|
$4144 + 11 = 4155$ fixpontosan ábrázolva.

A bitek közé írt választóvonalak alapvetően képzeletbeliek, vagyis például a 18. cellában levő

utasítás egyben 20496-ot is jelenthet fixpontosan ábrázolva.

Ez az „uniformizálás" a számok és utasítások között mint látjuk, bizonyos hátrányokkal jár, de döntő előnye (a gép szerkezeti egyszerűsödése mellett), hogy az utasításokkal műveleteket végezhetünk, aminek az eredménye egy újabb utasítás lehet, másszóval a program, azaz a bevitt utasítássorozat programozottan módosítható. Ennek a lehetőségnek az előnyeit felesleges tovább hangsúlyoznunk. Példánkban látjuk, hogy ha a 19. cella 2. bitjét elfelejtettük volna egyesre állítani, akkor a gép az összeadás után 4155 számot mint utasítást kezdené végrehajtani, venné azt a cellaszámot, ami a 4155 fixpontos alakjában a 11—20 bitek jelentenek, azaz 155-öt, és azt az utasítást, melynek kódszámát éppen 4155 5—10. bitjei adják, azaz 1-et, s.i.t. és ezt hajtaná végre, ami teljesen ellenőrizhetetlen eredményekhez vezetne.

Ha azonban valahol egy adatsorozatot tárolunk, amelynek minden elemével ugyanazt a műveletet kívánjuk elvégezni, akkor ezt megoldhatjuk úgy, hogy amikor a sorozat első elemével végrehajtotta a gép az utasítást, adjon hozzá 1-et az utasítást tartalmazó cellához. Belátható, hogy ez pont azt eredményezi, hogy az utasítás címrésze 1-el nőni fog, ami azt jelenti, hogy ezután az adatsorozat következő elemével fogja gépünk ugyanazt a műveletet elvégezni.

Az a három generáció, amely a hardware és a gép — ember vonatkozásában megfigyelhető volt, megfigyelhető a software (azaz a számítógéphez írt programok összessége) fejlődésében is. Az első számítógépek esetében a programot gépi kódban kellett megírni. A programok írása gépi kódban nem túlságosan nehéz feladat, de nem lehet állítani azt sem, hogy nagyon kellemes. Ezért amikor a számítógépek második generációja megjelent (és ez hozta létre tulajdonképpen a „demográfiai robbanást" a számítógépek történetében), kidolgozták az ún. programozási nyelveket.

A számítógép gondolkodóképességéről

Az elektronikus számítógépet „nagysebességű trotli"-nak nevezték egyszer a matematikusok. Most már látjuk, hogy ebben sok igazság van. A gép meglehetősen „vakon" végzi a dolgát, teljesen „öntudatlanul". Mint láttuk, nem tud megkülönböztetni egy utasítást egy számtól. Nem tudja megállapítani azt sem, hogy egy adott helyre beolvasott-e már egy számot, vagy sem, mert minden cellában minden elem valamilyen állapotban van, és ez lehet éppen a kívánt állapot is. Végül pedig az utasítások egyszerűsége a feladat szinte óvodás lebontását kívánja az embertől, mert a gép, csak a legegyszerűbb fogalmakkal képes dolgozni, és csak a legegyszerűbb utasításokat képes végrehajtani.

Ha a gép maga meg is érdemli a „trotli" jelzőt, ad egy olyan lehetőséget, amivel a trotliságon túl lehet lépni: Amikor összetett aritmetikai kifejezés kiszámítását lebontjuk elemi műveletekre, hamar észrevesszük, hogy a munka meglehetősen gépies lesz, vagyis igen egyszerű szabályokat kell ismételten alkalmazni. (Kiszámítjuk a kifejezésben szereplő összeadandókat, félretesszük és a végén összeadjuk. Az összeadandók kiszámításakor először kiszámítjuk a tényezőket, félretesszük és a végén összeszorozzuk. Amennyiben valamelyik tag törtkifejezés, külön számoljuk a számlálót és a nevezőt, majd félretesszük, és ezután osztunk s.i.t.). Ez viszont azt jelenti, hogy ha ezeket a lebontási szabályokat meg tudjuk fogalmazni a gép által érthető elemi utasításokkal, akkor írhatunk egy olyan programot, mely elkészít egy utasítássorozatot, mely egy adott összetett aritmetikai kifejezés értékét számítja ki, anélkül, hogy az elemi műveletekre való lebontást a programozónak kellene elvégeznie. Ez az utasítássorozat lényegében azokból az utasításokból fog állni, melyeket mi magunk is írtunk volna az adott aritmetikai kifejezés kiszámítására. vagyis ez az utasítássorozatot gyártó program írta meg a nekünk szükséges programot a végső alakjában. Mivel az utasítások ugyanúgy tárolódnak mint a számok, belátható, hogy ez elvileg lehetséges, mert az utasítássorozat elkészítése visszavezethető megfelelő számok közötti matematikai műveletekre.

Fordító programok

Ha viszont sikerül ilyen programot írni, akkor azt állandóan a gépben tárolva lényegében elértük azt, hogy most már adhatunk összetettebb utasításokat a gépnek. Természetesen ez a programkészítő program is csak bizonyos utasításokat és fogalmakat képes megérteni, azaz lebontani elemi utasításokra. Egy ilyen megengedett utasítás és fogalomrendszer lényegében egy egyszerű nyelv, amely ismer néhány fogalmat, műveletet, utasítást, és ezek bizonyos kombinációját. Azt a programot, mely az ilyen „nyelven" írt utasítássorozatot, azaz programot lebontja az adott gép elemi utasításaira *fordítóprogramnak* nevezzük.

Az utóbbi években több ilyen nyelvet dolgoztak ki. Ezek egymástól egyrészt a felhasználás különbözősége miatt a fogalomrendszerükben különböznek, másrészt a fogalomrendszerük bonyolultságában. Az nyilvánvaló, hogy minél bonyolultabb fogalmakkal tud egy nyelv dolgozni, a vele való programozói munka annál könnyebb. Ugyanakkor a fordítóprogram hatékonysága csökken a nyelv bonyolultságával, mert az mindig az általános esetre van felkészülve, és egy konkrét program lebontásakor nem tudja kihasználni a különböző, az adott speciális helyzetből adódó egyszerűsítési lehetőségeket. A programozási nyelveket más szempontból két csoportba lehet sorolni, nevezetesen eljárás-orientált nyelvek és probléma-orientált nyelvek csoportjára. Az előbbi csoportba sorolható nyelveket használjuk műszaki-tudományos számítások elvégzésére, az utóbbi nyelvek közé tartozókat pedig az adatfeldolgozó programok írására lehet kiválóan használni.

A programozási nyelveknek nemcsak az az előnyük, hogy összetettebb fogalmakkal képesek dolgozni mint a gép saját utasításrendszere, másnéven gépikódja, hanem az is, hogy univerzálisak, azaz minden gépre többé-kevésbé (kisebb módosítások-

tól eltekintve) érvényesek. A gépikódok meglehetősen különböznek egymástól, mert azok erősen igazodnak a gépek műszaki felépítéséhez, így ez az univerzalitás határozottan előnyös. Természetesen minden géphez meg kell írni az illető nyelv fordítóprogramját.

A következő cikkben az ALGOL nyelv fogalomrendszerét szeretnénk ismertetni. Nem célunk, hogy az olvasót megtanítsuk programozni ALGOL nyelven, csupán a nyelvet szeretnénk bemutatni az olvasónak, mint a fenti gondolatoknak egy igen elterjedt, jól használható realizációját.

Program, programcsomag, programrendszer

A második generációba tartozó számítógépek elterjedésével kapcsolatban hallatlan mennyiségben születtek meg különböző programok. Kezdetben, mindenki saját maga számára írt programokat. Később azonban kiderült, hogy ez nem célravezető dolog és sokkal helyesebb, hogyha bizonyos feladat megoldására már egyszer született egy program, akkor ezt a már megírt programot minden további nélkül felhasználjuk, ha ugyanazt a feladatot kell ismét megoldani. Így jöttek létre a programkönyvtárak nagy számítógépgyártó cégek, illetőleg számítástechnikai központok körül. Csakhamar kiderült, hogy ugyanannak a feladatnak a megoldására több programot is készítettek. Mondjuk a feladat differenciálegyenlet-rendszer megoldása. Ennek a a matematikai feladatnak numerikus megoldására ismeretes a Runge-Kutta módszer, meg a „prediktor-korrektor" módszer és még néhány más módszer is. Amikor egy reakciókinetikus a saját kinetikai vizsgálataival kapcsolatban differenciálegyenlet-rendszert akar megoldani és erre fel akar használni valamiféle programot, amelyet már mások megírtak, akkor döntenie kell, hogy a rendelkezésre álló programok közül melyiket válassza. Ez a döntés nem túlságosan egyszerű, mert rendszerint az adott feladat természetétől és paramétereitől függ, hogy ezt, azt, vagy a harmadik módszert célszerű felhasználni. Ezért tehát praktikus ezeket a programokat valahogyan összecsomagolni és programcsomagként kezelni. A programcsomagot az jellemzi, hogy azonos adatbeviteli és adatkiírási szerkezete van és a programcsomag felhasználójától függ, hogy a kidolgozott számításmenetek közül melyiket használja, vagy ha többet is akar használni, akkor ezeket milyen sorrendben vegye igénybe.

Van a programcsomagnak egy másik típusa is, amelyet az jellemez, hogy nem azonos feladatok megoldására szolgáló programok vannak benne összegyűjtve, hanem éppen eltérő feladatok megoldására szolgáló programok és azért vannak ezek „összecsomagolva", mert a felhasználó szempontjából ezek a különböző programok gyakran, s egymás után fordulnak elő. (Példánkban mondjuk, a kémikus számára nagyon hasznos lehet egy programcsomag, amely képes vegyületek fizikai kémiai tulajdonságainak becslésére.)

A harmadik generációba tartozó software legmagasabbrendű terméke az ún. programrendszer. A programrendszer programoknak vagy programcsomagoknak olyan gyűjteménye, amely lehetővé teszi, hogy az egyik programmal kiszámított adatok egy, a számítási sorrendben következő program bemenő adatai legyenek. Az adatoknak az egyik programból a másikba való átvitele egy programrendszeren belül azután a körülményektől függően lehet teljesen automatikus vagy lehet olyan is, amely emberi döntést és beavatkozást követel, s ennek fejében interaktív munkát tesz lehetővé. Az interaktív munkának feltétele a megfelelő ember-gép kapcsolat.

A programozási munka

A software emberi tevékenység eredményeképpen jön létre és talán érdemes az ember-ember kapcsolatot is megfigyelni, azt ti., hogy ez a kapcsolat hogyan alakul a feladatok természetének megváltozásával. Ha a feladatot annak bonyolultsága szempontjából három kategóriába osztjuk (és ezek a kategóriák nagyjából követik a software generációkat), azt mondhatjuk, hogy egyedi számítások, összetett számítások és rendszerek számítása a a három kategória, amely történetileg kialakult és amelyek ma egymás mellett léteznek. Az egyedi számítások elvégzésénél egyetlen programozó készíti a programot és ez a programozó lehet ha úgy tetszik vegyész, vagy más szakember, aki saját maga számára valamilyen munkájával kapcsolatban egyedi számítást akar elvégezni.

Az összetett számításoknál a helyzet némiképpen más. Ilyenkor ugyanis négyfajta ember együttműködésére van szükség. Az első a fizikai, kémiai, közgazdasági vagy más jelenség *modelljét* elkészítő specialista; a második a matematikus, aki ezt a modellt mint matematikai problémát nézi és kezeli, megadja azt a számításmenetet, amely a modellből megfogalmazott matematikai probléma megoldására vezet; ezt követi a programozó munkája, aki a matematikus által kidolgozott gondolatmenetet (algoritmust) a programozástechnika szabályai szerint programmá fogalmazza meg; végül bekapcsolódik a negyedik a kódoló, aki a programot kellő részletességgel megírja. Meg kell említeni szinte zárójelben a programozó szerepét. Az algoritmust lehet így és úgy programozni, és nagymértékben ettől függ, hogy mennyi időt vesz igénybe az adott számítás végrehajtása.

Egy programkönyvtárból vett programot, amelyet gyakran használunk, programozó kollégáink az utóbbi négy év alatt többször átalakítottak — az eredeti algoritmust változatlanul hagyva — és programozástechnikai javításokkal a számítási sebességét az eredeti negyvenszeresére fokozták. E példa megmutatja, hogy jogos a programozást önálló munkának tekinteni. A felsorolt különböző szakmájú és felkészültségű embereknek ilyen együttesét teamnak lehet nevezni és ha ezt a kifejezést elfogadjuk, akkor azt mondhatjuk, hogy egy összetett számítás programjának elkészítése egy team feladata. Így tehát, míg az egyedi számítások esetén egyetlen szakember sokoldalú felkészültségén és mesterségbeli tudásán múlik a használható program létrejötte, addig az összetett számítások esetén az egyének mesterségbeli tudásán túlmenően

a kooperáció megvalósításának minőségétől függ a programnak a létrejötte.

Említettük, hogy az utóbbi időben nagy programrendszereket is kidolgoznak. Ezeknek a nagy programrendszereknek a készítésében ismét újfajta technika alakult ki, az ember-ember kapcsolatot illetően. Az ilyenfajta munka a rendszer analízisével kezdődik el, amelynek során azt kell felderíteni, hogy ennek a rendszernek milyen elemei vannak, valamint azt, hogy ezeket az elemeket milyen hálózat kapcsolja össze egymással. A rendszeranalízis a rendszermérnök (system's engineer) feladatkörébe tartozik és ugyancsak az ő feladatköre az is, hogy az egész rendszernek az analízise alapján megalkossa az általános modelljét, hogy aztán ebben az általános modellben az egyes elemek kidolgozását megfelelő teamekre bízza. A hierarchikus szervezés tipikus példájáról van itt szó. Az összetett számítások esetében egyetlen teamen belüli kooperáció jellemezte az ember-ember kapcsolatot, a nagy rendszerek számításával kapcsolatos programrendszerek elkészítésében a kooperációnál magasabbrendű tudományos munkaszervezés jellemzi vagy hozza létre azokat az ember-ember kapcsolatokat, amelyek végül is egy működő programrendszer kidolgozására vezetnek.

РЕЗЮМЕ

В сообщении сперва дано краткое описание конструкционного устройства дигитальных ЭВМ, затем подробно обсуждены вопросы связанные с программами управляющими работой ЭВМ. Представлены, далее, элементарные составляющие части программ, и показано, какими способами можно выполнить решение некоторых сложных проблем при помощи этих составляющих частей.

SUMMARY

A short introduction is given on the structural construction of the digital electronic computing mashines. The problems in connection with the programs controlling the operation of the computers are described in detailes. For one side, the elementar blocks of the programs and for the other, the methods which are necessary to solve a complex problem with the aid of these blocks are shown.