A 7a. ábrán egy párhuzamos áramlásba helyezett golyó körül kialakuló áramlási vonalak láthatók (az áramlási tér egy sík metszetében). A 7b. ábrán egy párhuzamos áramlás áramvonalai láthatók, ezt az áramlást a **v**=const. vektoregyenlet írja le, mivel az áramlási tér minden pontjában a sebesség állandó. A 7c. ábrán a 7a. ábrán látható golyó helyére képzelt *kettős forrás* áramvonalait láthatjuk. A 7b. és a 7c. ábrán látható áramvonal-spektrum összegezéséből megkapjuk a 7a. ábrán látható áramvonalakat. Ezt az állítást egy magyarázattal kvalitatíve igazolhatjuk. Az áramló folyadéknak az a része amely nekiütközik a golyónak, visszafordul és egy kitérő áramlást végez. A kitérő áramlásban résztvevő folyadékrészek a gömb bal oldaláról, a golyó megkerülésével átáramlanak a jobb oldalra, vagyis olyan áramvonalak mentén haladnak amilyen áramvonalakat a 7c. ábrán látható *kettős forrás* szolgáltat. Így belátható, hogy a 7b. és a 7c. ábrán látható-áramvonal rendszer összegezéséből megkapjuk a 7a. ábrán feltüntetett áramvonal-spektrumot.

A stacionárius áramlások terét leíró **v**=**v**(x,y,z) sebességfüggvény egy vektoriális egyenlet, ezért az áramlási tér is vektor tér, melynek az áramvonalai ugyancsak irányított vektorvonalak.

Megfigyelhető a hasonlóság a gravitációs, magnetosztatikus és elektrosztatikus erőterek erővonal spektrumai és a megfelelő áramlási terek áramvonalai között. Az 5. ábrán látható pontszerű forrás áramvonalai tökéletesen megegyeznek a pozitív ponttöltés elektromos erővonalrendszerével, míg a 6. ábrán látható kettős forrás áramlási vonalai az elektromos dipólus erővonal rendszerével egyeznek meg.

A 7a. ábrán látható áramvonal-spektrum elektromos megfelelője az az erővonal rendszer amely akkor áll elő, ha egy szigetelő anyag homogén elektromos erőterébe behelyezünk egy szigetelő anyagból készült gömböt, melynek a permittivitása kisebb a golyót körülvevő közegénél.

A két vektortér között fennálló hasonlóság lehetőséget nyújt a hasonlósági modellek módszerének az alkalmazására. Ami azt jelenti, hogy ha az egyik vektortérben elvégeztük kísérleti úton az erővonal vagy az áramvonal-spektrum felvételét, tudunk következtetni a hasonlósági modell alapján a másik vektortér megfelelő spektrumára. Így mindig azt a kísérletet végezhetik el, amelyik könnyebben kivitelezhető, vagy kevésbé költséges.

Puskás Ferenc

Karakterek ábrázolása a számítógépen

Adatok ábrázolása elképzelhetetlen valamilyen kódrendszer – karakterkészlet, jelkészlet – megléte nélkül. A programozási nyelvek tervezésénél is az első lépések egyike a jelkészlet meghatározása. A korai programozási nyelvek általában csak az angol ábécé betűit, a számjegyeket és néhány speciális karaktert (pl. zárójelek, műveleti jelek stb.) engedtek meg a lexikális elemekben. Napjainkban egyre nagyobb az igény arra, hogy az egyes nemzeti karakterek használatát is megengedjék az egyes programozási nyelvek, így használhassunk például „á” vagy „é” betűket az azonosítókban stb. Sőt az ábécé szerinti rendezés is engedje meg a nemzeti karakterek használatát.

A számítógépek megjelenésekor nem volt egy szabványos karakter-kódolási rendszer. Minden gépgyártó saját szabványt használt, amely hatalmas kompatibilitási problémákhoz vezetett, nem is beszélve a számítógépek közötti kommunikáció lehetetlenségéről. Az 1950-es években több mint 60 különböző módon ábrázolták a karaktereket.

Az ASCII táblázat

1963-ra nyilvánvalóvá vált egy egységes kódolási rendszer bevezetésének szükségessége. Az Amerikai Szabványügyi Hivatal (ANSI – *American National Standard Institute*) két éves munkával bevezette az ASCII (*American Standard Code for Information Interchange*) szabványkódot az információcsere megvalósítására.

Kezdetben az ASCII szabvány 128 karaktert kódolt 7 biten, 33 vezérkarakter és 95 nyomtatható karakter ábrázolásával. Később 8 bitesre bővült a szabvány, így lehetőség nyílt 256 karakter kódolására, amelyek között megjelentek az egyes nemzeti karakterek is. Ezt a második 128 karaktert a Windows nemzeti kódlapok kialakítására használja.

A Unicode szabvány

A személyi számítógépek rohamos elterjedése, a grafikus felületű operációs rendszerek megjelenése felerősítette azt a zűrzavart, amely a karakterek azonosításában már létezett. A különböző billentyűzetkiosztásokba, a betűtípusokba (fontokba) az egyes karakterek – főleg a speciális nemzeti karakterek – a lehető legkülönbözőbb módon kerültek bele. Az 1990-es évek elejére nyilvánvalóvá és szükségszerűvé vált egy új kódoló rendszer kidolgozása a karakter-táblák számára. 1991-ben az Apple és a Xerox cégek kezdeményezésére létrejött a *Unicode Consortium*, amelynek az volt a feladata, hogy kidolgozzon egy mindenki számára elfogadható kódkiosztást a világ elterjedtebb írásrendszerei számára. A Unicode szabvány (jelen pillanatban a 3.0-ás ajánlásnál tart) 16 biten ábrázolja a karaktereket, így 65 536 karakter azonosítására alkalmas. Az első 128 karakter egybeesik az ASCII táblával, az efölötti karaktereket pedig szegmensekre osztották, amelyek a különböző írásrendszereket tartalmazzák. Így egy nyelv szerinti kódtábla megállapításához két információra van szükségünk: a nyelvre és az ehhez tartozó Unicode-szegmensre. Ezek után már csak egy olyan billentyűzetmeghajtóra van szükség, amely megfelelteti egymásnak a karaktereket és a billentyűket. Az egyes Unicode-szegmenseket külön fontállományban tárolják, hogy ne kelljen túl nagy méretű állományokkal dolgozni – egyszerre úgysem használjuk a világ összes írásjelét!

A Unicode 3.0-ás szabvány jelenleg 49 194 karaktert tartalmaz, s így megvalósít közel 100 írásrendszert. Kiterjed a betűrendes, szótagos és ideografikus írásrendszerekre, beleértve a legtöbb latin ábécét használó nyelvet, a cirill, görög, thaiföldi ábécéket, a közel- és távol-keleti írásjeleket. A szabvány tartalmaz továbbá 8515 karaktert egyéni célokra, esetleges továbbfejlesztésekre.

A Unicode szabvány azért született meg, hogy egy *egyetemes*, *hatékony*, *egységes* és *egyértelmű* karakterkészlet terjedjen el a gyakorlatban.

Egyetemesség: a készlet annyira terjedelmes kell legyen, hogy felölelje mindazon írásjeleket, amelyekre valószínűleg szükség lehet.

Hatékonyság: egyszerű szöveget, mely rögzített hosszúságú írásjelekből épül fel, könnyű kezelni, elemezni, az alkalmazás nem kell speciális karakterekre figyeljen.

Egységesség: a rögzített hosszúságú írásjelek használata megkönnyíti a rendezést, keresést, ábrázolást, a szöveg szerkesztését.

Egyértelműség: bármely 16 bites érték mindig ugyanazt az írásjelt (karaktert) ábrázolja.

Minden Unicode írásjegyet 16 bit hosszúságon ábrázoltak. A visszafelé történő kompatibilitás miatt a Unicode szabvány leírja az UTF-8-as kódolást is, mely segítségével megvalósítható a veszteségmentes átalakítás Unicode írásjegyek és a 8-bites karakterek között. Az UTF-16-os kódolással pedig a Unicode szabvány újabb 1 000 000 írásjegy ábrázolására bővült ki. Amikor egy írásjegy az U+0000 – U+FFFF halmazon kívül

értelmezett, az UTF-16-os kódolással két 16 bites szekvenciára bomlik le. Az UTF-32 kódolás 32 biten ábrázolja a karaktereket.

A 65 536-os határ túllépésének okai:

- az írásjegyek kódjainak kijelölése blokkonként történik, így mindegyik blokkban van olyan kód, amely sohasem kerül felhasználásra;
- az olyan karakterek sokasága, melyeket összetett karakterekként elő lehetne állítani, de a létező leképezések régebbi karakterkészletekkel ezt lehetetlenné teszik;
- a távol-keleti ideogramok óriási száma;
- a még fel nem vett írásmódok (archaikusak is) nagy száma.

A Unicode szabványban az írásjegyek logikai sorrendben vannak tárolva: a kiolvasás sorrendjében. Számos nyelv esetén (pl. héber, arab stb.) jobbról-balra történik az olvasás. Mivel a logikai kiindulópont a legbaloldalibb írásjegy, ez a karakter lesz az első a Unicode szövegben.

A Unicode egyesít olyan karaktereket, amelyek több nyelvben is szerepelnek, így az azonos kinézetű írásjegyek ugyanazt a kódot kapják. Ezáltal több, mint 130 000 kínai, japán, illetve koreai ideogram összevonódott, és mindössze 27 786 Han kód került tényleges lefoglalásra.

Például a magyar „ö”-n, „ü”-n szereplő pontok, a diarézis és bizonyos rendszerekben a kétszeres deriváltként jegyzett jelek összevonásra kerültek, így alakult ki az U+0308-as kóddal rendelkező, COMBINING DIAERESIS névvel rendelkező „¨” karakter. Azonban a visszafelé történő kompatibilitás miatt nem minden karaktert vontak össze, például az OMEGA („Ω”) és az Ohm („Ω”) mértékegység jele külön szerepelnek.

Programozási nyelvek Unicode támogatottsága

Talán a *Java* a legismertebb programozási nyelv, amely támogatja a Unicode-ot. A `char` és a `string` típusok 16 bites Unicode karakterekre épülnek. A megjegyzések és változók nevei, az azonosítók és a teljes Java forrásszöveg Unicode szerint van ábrázolva. A változók és sztringek viszont nincsenek nominalizálva, így az „ö” és az „ö” ugyanúgy jelenik meg, holott az egyik LATIN SMALL LETTER O WITH DIAERESIS, a másik pedig LATIN SMALL LETTER O és a COMBINING DIAERESIS dinamikus összetevése.

C, illetve *C++* programozási nyelvben lehetőség van az UTF-8, UTF-16 és UTF-32 kódolásra. *Visual C++* esetén a `_UNICODE` szimbólum deklarálása után a `TCHAR` makrók `wchar_t`-vé fejlődnek, így képesek a Unicode támogatottságra. Amikor a `_MBCS` szimbólumot deklaráljuk, a makrókban használt sztring-függvények több bájtos karakterek kezelésére is képessé válnak.

A *Visual Basic* a karaktersorozatokat Unicode karakterekként kezeli. Az `AscW()` és a `StrConv()` függvények kezelni tudják a Unicode-os karaktereket.

A *Borland Delphi* programozási nyelvben a Unicode írásjegyek kezelésére létezik a `WideChar`, `WideString` típus, átalakításra pedig számos függvény: `UnicodeToUtf8`, `Utf8ToUnicode`, `Utf8ToAnsi`, `AnsiToUtf8`, `Utf8Encode`, `Utf8Decode` stb., azonban a standard komponensek nem támogatják a Unicode sztringek használatát, más komponenscsomagokat, pl. a *TntControls* kell használni erre a célra.

Az adatbázisok területén is növekvő a Unicode iránti érdeklődés, az *Oracle* és a *Sysbase* például már évek óta tagja a Unicode Consortium-nak.

Kovács Lehel