

Nevek szintaxisa, kapcsolata a nyelv védett szavaival; változók jellemzői: név, cím, érték, típus, élettartam, hatókör; program entitások kapcsolása tulajdonságokkal: a kötés koncepciója; változók jellemzése: (statikus, verem dinamikus, explicit és implicit halom dinamikus); típus ellenőrzés, típus kompatibilitás; hatókör és élettartam, hivatkozási környezet

Karakterkészlet: Minden nyelv definiálja a saját karakterkészletét. A karakterkészletek között lényeges eltérések lehetnek, de a programnyelvek általában a karaktereket a következő módon kategorizálják: betűk, számjegyek, vagy egyéb karakterek. Minden programnyelvben betű az angol ABC 26 nagybetűje. A nyelvek továbbá betű kategóriájú karakternek tekintik gyakran az `_`, `$`, `#`, `@` karaktereket is. Ez viszont sokszor implementációfüggő.

Szimbolikus nevek - Azonosító: Olyan karaktersorozat, amely betűvel kezdődik, és betűvel vagy számjeggyel folytatódhat. Arra való, hogy a program írója a saját programozói eszközeit megnevezze vele, és ezután ezzel hivatkozzon rá a program szövegében bárhol. A hivatkozási nyelvek általában nem mondanak semmit a hosszáról, az implementációk viszont értelemszerűen korlátozzák azt.

Szabályos C azonosítók: „x”, „almafa”, „hallgato_azonosito”, „SzemelyNev”

Kulcsszó (alapszó, fenntartott szó, védett szó, foglalt szó): Olyan karaktersorozat (általában azonosító jellegű felépítéssel), amelynek az adott nyelv tulajdonít jelentést, és ez a jelentés a programozó által nem megváltoztatható. Nem minden nyelv (pl. FORTRAN, PL/I) ismeri ezt a fogalmat. Az utasítások általában egy-egy jellegzetes kulcsszóval kezdődnek, a szakmai szleng az utasítást gyakran ezzel nevezi meg (pl. IF-utasítás). Minden nyelvre nagyon jellemzőek a kulcsszavai. Ezek gyakran hétköznapi angol szavak vagy rövidítések. Az alapszavak soha nem használhatók azonosítóként.

C-ben például alapszavak: „if”, „for”, „case”, „break”

Standard azonosító: Olyan karaktersorozat, amelynek a nyelv tulajdonít jelentést, de ez az alapértelmezés a programozó által megváltoztatható, átértelmezhető. Általában az implementációk eszközeinek (pl. beépített függvények) nevei ilyenek. A standard azonosító használható az eredeti értelemben, de a programozó saját azonosítóként is felhasználhatja. A C-ben például standard azonosító a NULL.

A változó olyan programozási eszköz, amelynek 4 komponense van: név, attribútumok, cím, érték.

Név: egy azonosító. A program szövegében a változó mindig a nevével jelenik meg, az viszont bármely komponenst jelentheti. Szemléltethetjük úgy a dolgokat, hogy a másik három komponenst a névhez rendeljük hozzá.

Attribútumok: olyan jellemzők, amelyek a változó futás közbeni viselkedését határozzák meg. Az eljárásorientált nyelvekben (általában a típusos nyelvekben) a legfőbb attribútum a típus, amely a változó által felvehető értékek körét határolja be. Változóhoz attribútumok deklaráció segítségével rendelődnek. A deklarációnak különböző fajtáit ismerjük. (*Explicit deklaráció:* A változó teljes nevéhez kell az attribútumokat megadni. A nyelvek általában megengedik, hogy egyszerre több változónévhez ugyanazokat az attribútumokat rendeljük hozzá. *Implicit deklaráció:* A programozó végzi, betűkhöz rendel attribútumokat egy külön deklarációs utasításban. Ha egy változó neve nem

szerepel explicit deklarációs utasításban, akkor a változó a nevének kezdőbetűjéhez rendelt attribútumokkal fog rendelkezni, tehát az azonos kezdőbetűű változók ugyanolyan attribútumúak lesznek. *Automatikus deklaráció:* A fordítóprogram rendel attribútumot azokhoz a változókhoz, amelyek nincsenek explicit módon deklarálva, és kezdőbetűjükhöz nincs attribútum rendelve egy implicit deklarációs utasításban. Az attribútum hozzárendelése a név valamelyik karaktere (gyakran az első) alapján történik.)

Változó címkomponense: a tárnak azt a részét határozza meg, ahol a változó értéke elhelyezkedik. A futási idő azon részét, amikor egy változó rendelkezik címkomponenssel, a **változó élettartamának** hívjuk. Egy változóhoz cím rendelhető az alábbi módokon:

Statikus tárkiosztás: A futás előtt eldől a változó címe, és a futás alatt az nem változik. Amikor a program betöltődik a tárba, a statikus tárkiosztású változók fix tárhelyre kerülnek.

Dinamikus tárkiosztás: A cím hozzárendelését a futtató rendszer végzi. A változó akkor kap címkomponenst, amikor aktivizálódik az a programegység, amelynek ő lokális változója, és a címkomponens megszűnik, ha az adott programegység befejezi a működését. A címkomponens a futás során változhat, sőt vannak olyan időintervallumok, amikor a változónak nincs is címkomponense.

Programozó által vezérelt tárkiosztás: A változóhoz a programozó rendel címkomponenst futási időben. A címkomponens változhat, és az is elképzelhető, hogy bizonyos időintervallumokban nincs is címkomponens. Három alapesete van:

- A programozó abszolút címet rendel a változóhoz, konkrétan megadja, hogy hol helyezkedjen el.
- Egy már korábban a tárban elhelyezett programozási eszköz címéhez képest mondja meg, hogy hol legyen a változó elhelyezve, vagyis relatív címet ad meg. Lehet, hogy a programozó az abszolút címet nem is ismeri.
- A programozó csak azt adja meg, hogy mely időpillanattól kezdve legyen az adott változónak címkomponense, az elhelyezést a futtató rendszer végzi. A programozó nem ismeri az abszolút címet.

Mindhárom esetben lennie kell olyan eszköznek, amivel a programozó megszüntetheti a címkomponenst. A programozási nyelvek általában többféle címhozzárendelést ismemek, az eljárás-orientált nyelveknél általános a dinamikus tárkiosztás. A változók címkomponensével kapcsolatos a többszörös tárhivatkozás esete. Erről akkor beszélünk, amikor két különböző névvel, esetleg különböző attribútumokkal rendelkező változónak a futási idő egy adott pillanatában azonos a címkomponense és így értelemszerűen az értékkomponense is. Így ha az egyik változó értékét módosítjuk, akkor a másiké is megváltozik. A korai nyelvekben (pl. FORTRAN, PL/I) erre explicit nyelvi eszközök álltak rendelkezésre, mert bizonyos problémák megoldása csak így volt lehetséges. A szituáció viszont előidézhető (akár véletlenül is) más nyelvekben is, és ez nem biztonságos kódhoz vezethet.

Típusellenőrzés: eljárás, ami fordítási időben (statikus ellenőrzés) vagy végrehajtási időben (dinamikus ellenőrzés) ellenőrzi a típuskényszerítés szabályait, és szükség esetén végrehajtja a előírt művelet(ek)et. A típuskényszerítés hatására a program az aktuális változót/memóriacím alatti területet a típusnak megfelelően fogja értelmezni, vagyis ekkor dől el, hogy milyen jellegű adattal is dolgozik.