Tétel 2

Lexikális egységek

A lexikális egységek a program szövegének azon elemei, melyeket a fordító a lexikális elemzés során felismer és tokenizál (közbenső formára hoz). Fajtái a következők:

– többkarakteres szimbólum

– szimbolikus név

– címke

– megjegyzés

– literál

Többkarakteres szimbólumok

Olyan (jellemzően egyéb karakterekből álló) karaktersorozatok, amelyeknek a nyelv tulajdonít jelentést és ezek csak ilyen értelemben használhatók. Nagyon gyakran a nyelvben operátorok, elhatárolók lehetnek. Például a C-ben többkarakteres szimbólumok a következők: ++, --, &&, /*, */.

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.

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/Ő) ismeri ezt a fogalmat. Az utasítások általában egy-egy jellegzetes kulcsszóval kezdődnek, a szakmai szleng az utasítást gyakran ezzel nevezi meg (pl. ŐF-utasítás). Minden nyelvre nagyon jellemzőek a kulcsszavai. Ezek gyakran hétköznapi angol szavak vagy rövidítések. Az alapszavak soha nem használhatók azonosítóként. A C-ben például alapszavak a következők: 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.

Az eljárásorientált nyelvekben a végrehajtható utasítások (l. 4. fejezet) megjelölésére szolgál, azért, hogy a program egy másik pontjáról hivatkozni tudjunk rájuk. Bármely végrehajtható utasítás megcímkézhető. A címke maga egy speciális karaktersorozat, amely lehet előjel nélküli egész szám, vagy azonosító. Eléggé általános, hogy a címke az utasítás előtt áll és tőle kettőspont választja el. Az Adában viszont a címke az utasítás előtt a << és >> többkarakteres szimbólumok között szerepel.

A megjegyzés egy olyan programozási eszköz, amely segítségével a programban olyan karaktersorozat helyezhető el, amely nem a fordítónak szól, hanem a program szövegét olvasó embernek. Ez olyan

magyarázó szöveg, amely a program használatát segíti, működéséről, megírásának körülményeiről, a felhasznált algoritmusról, az alkalmazott megoldásokról ad információt. A megjegyzést a lexikális elemzés során a fordító ignorálja.

Literálok (Konstansok) A literál olyan programozási eszköz, amelynek segítségével fix, explicit értékek építhetők be a program szövegébe. A literáloknak két komponensük van: típus és érték. A literál mindig önmagát definiálja. A literál felírási módja (mint speciális karaktersorozat) meghatározza mind a típust, mind az értéket. Példák: Prog pdf, 21. Old

Adattípusok

Az adatabsztrakció első megjelenési formája az adattípus a programozási nyelvekben. Az adattípus maga egy absztrakt programozási eszköz, amely mindig más, konkrét programozási eszköz egy komponenseként jelenik meg. Az adattípusnak neve van, ami egy azonosító. A programozási nyelvek egy része ismeri ezt az eszközt, más része nem. Ennek megfelelően beszélünk típusos és nem típusos nyelvekről. Az eljárásorientált nyelvek típusosak. Egy adattípust három dolog határoz meg, ezek:

- tartomány - Az adattípusok tartománya azokat az elemeket tartalmazza, amelyeket az adott típusú konkrét programozási eszköz fölvehet értékként.
- műveletek - Az adattípushoz hozzátartoznak azok a műveletek, amelyeket a tartomány elemein végre tudunk hajtani.
- reprezentáció - Minden adattípus mögött van egy megfelelő belső ábrázolási mód. A reprezentáció az egyes típusok tartományába tartozó értékek tárban való megjelenését határozza meg, tehát azt, hogy az egyes elemek hány bájtra és milyen bitkombinációra képződnek le.

Minden típusos nyelv rendelkezik beépített (standard) típusokkal. Saját típust úgy tudunk létrehozni, hogy megadjuk a tartományát, a műveleteit és a reprezentációját. Szokásos, hogy saját típust a beépített és a már korábban definiált saját típusok segítségével adjuk meg.

A *skalár* vagy *egyszerű* adattípus tartománya atomi értékeket tartalmaz, minden érték egyedi, közvetlenül nyelvi eszközökkel tovább nem bontható. A skalár típusok tartományaiból vett értékek jelenhetnek meg literálként a program szövegében.

A *strukturált* vagy *összetett* adattípusok tartományának elemei maguk is valamilyen típussal rendelkeznek. Az elemek egy-egy értékcsoportot képviselnek, nem atomiak, az értékcsoport elemeihez külön-külön is hozzáférhetünk. Általában valamilyen absztrakt adatszerkezet programnyelvi megfelelői. Literálként általában nem jelenhetnek meg, egy konkrét értékcsoportot explicit módon kell megadni.

Egyszerű típusok

Minden nyelvben létezik az *egész* típus, sőt általában egész típusok. Ezek belső ábrázolása fixpontos. Az egyes egész típusok az ábrázoláshoz szükséges bájtok számában térnek el és nyilván ez határozza meg a tartományukat is. Néhány nyelv ismeri az előjel nélküli egésztípust, ennek belső ábrázolása előjel nélküli (direkt).

Alapvetőek a *valós* típusok, belső ábrázolásuk lebegőpontos. A tartomány itt is az alkalmazott ábrázolás függvénye, ez viszont általában implementációfüggő.

Az egész és valós típusokra közös néven, mint *numerikus* típusokra hivatkozunk. A numerikus típusok értékein a numerikus és hasonlító műveletek hajthatók végre.

A *karakteres* típus tartományának elemei karakterek, a karakterlánc vagy sztring típuséi pedig karaktersorozatok. Ábrázolásuk karakteres (karakterenként egy vagy két bájt, az alkalmazott kódtáblától függően), műveleteik a szöveges és hasonlító műveletek.

Egyes nyelvek ismerik a *logikai* típust. Ennek tartománya a hamis és igaz értékekből áll, műveletei a logikai és hasonlító műveletek, belső ábrázolása logikai. Speciális egyszerű típus a felsorolásos típus.

A *felsorolásos* típust saját típusként kell létrehozni. A típus definiálása úgy történik, hogy megadjuk a tartomány elemeit. Ezek azonosítók lehetnek. Az elemekre alkalmazhatók a hasonlító műveletek.

Egyes nyelvek értelmezik az egyszerű típusok egy speciális csoportját a, *sorszámozott* típust. Ebbe a csoportba tartoznak általában az egész, karakteres, logikai és felsorolásos típusok. A sorszámozott típus tartományának elemei listát (mint absztrakt adatszerkezetet) alkotnak, azaz van első és utolsó elem, minden elemnek van megelőzője (kivéve az elsőt) és minden elemnek van rákövetkezője (kivéve az utolsót). Tehát az elemek között egyértelmű sorrend értelmezett. A tartomány elemeihez kölcsönösen egyértelműen hozzá vannak rendelve a 0, 1, 2, ... sorszámok. Ez alól kivételt képeznek az egész típusok, ahol a tartomány minden eleméhez önmaga, mint sorszám van hozzárendelve.

Egy sorszámozott típus esetén mindig értelmezhetők a következő műveletek:

- ha adott egy érték, meg kell tudni mondani a sorszámát, és viszont
- bármely értékhez meg kell tudni mondani a megelőzőjét és a rákövetkezőjét

A sorszámozott típus az egész típus egyfajta általánosításának tekinthető. Egy sorszámozott típus altípusaként lehet származtatni az *intervallum* típust.

Összetett típusok

Az eljárásorientált nyelvek két legfontosabb összetett típusa a tömb (melyet minden nyelv ismer) és a rekord

A tömb típus a tömb absztrakt adatszerkezet megjelenése típus szinten. A tömb statikus és homogén összetett típus, vagyis tartományának elemei olyan értékcsoportok, amelyekben az elemek száma azonos, és az elemek azonos típusúak. A tömböt, mint típust meghatározza:

- dimenzióinak száma
- indexkészletének típusa és tartománya
- elemeinek a típusa

Egyes nyelvek a többdimenziós tömböket úgy képzelik el, mint olyan egydimenziós tömbök, amelyek elemei egydimenziós tömbök. Többdimenziós tömbök reprezentációja lehet sor- vagy oszlopfolytonos. Ez általában implementációfüggő, a sorfolytonos a gyakoribb. Az értékcsoport egyes elemeire a programozási eszköz neve után megadott indexek segítségével hivatkozunk.

Mutató típus

A mutató típus lényegében egyszerű típus, specialitását az adja, hogy tartományának elemei tárcímek. A mutató típus segítségével valósítható meg a programnyelvekben az indirekt címzés. A mutató típusú programozási eszköz értéke tehát egy tárbeli cím, így azt mondhatjuk, hogy az adott eszköz a tár adott területét címzi, az adott tárterületre „mutat". A mutató típus egyik legfontosabb művelete a megcímzett tárterületen elhelyezkedő érték elérése. A mutató típus tartományának van egy speciális eleme, amely nem valódi tárcím (NULL).

Nevesített konstans

A nevesített konstans olyan programozási eszköz, amelynek három komponense van:

- név
- típus
- érték

A nevesített konstanst mindig deklarálni kell.

A program szövegében a nevesített konstans a nevével jelenik meg, és az mindig az értékkomponenst jelenti. A nevesített konstans értékkomponense a deklarációnál eldől, és nem változtatható meg a futás folyamán.

A nevesített konstans szerepe egyrészt az, hogy bizonyos sokszor előforduló értékeket „beszélő" nevekkel látunk el, és ily módon az érték szerepkörére tudunk utalni a szövegben. Másrészt viszont, ha a program szövegében meg akarjuk változtatni ezt az értéket, akkor nem kell annak valamennyi előfordulását megkeresni és átírni, hanem elegendő egy helyen, a deklarációs utasításban végrehajtani a módosítást.

#define név literál

A változó

A változó olyan programozási eszköz, amelynek négy komponense van:

- név
- attribútumok
- cím
- érték

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

Az 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 programozó végzi explicit deklarációs utasítás segítségével. 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 kezdőbetűjű 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.

Az eljárásorientált nyelvek mindegyike ismeri az explicit deklarációt, és egyesek csak azt ismerik. Az utóbbiak általánosságban azt mondják, hogy minden névvel rendelkezőprogramozói eszközt explicit módon deklarálni kell.

A 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.
A 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.

Egy változó értékkomponensének meghatározására a következő lehetőségek állnak rendelkezésünkre:

- *Értékadó utasítás*: Az eljárásorientált nyelvek leggyakoribb utasítása, az algoritmusok kódolásánál alapvető.
- *Input*: A változó értékkomponensét egy perifériáról kapott adat határozza meg.
- *Kezdőértékadás*: Két fajtája van. Explicit kezdőértékadásnál a programozó explicit deklarációs utasításban a változó értékkomponensét is megadja.

Van olyan hivatkozási nyelv, amely az automatikus kezdőértékadás elvét vallja. Ezeknél a nyelveknél, ha a programozó nem adott explicit kezdőértéket, akkor a címkomponens hozzárendelésekor a hivatkozási nyelv által meghatározott bitkombináció kerül beállításra.

Kifejezések

A kifejezések szintaktikai eszközök. Arra valók, hogy a program egy adott pontján ott már ismert értékekből új értéket határozzunk meg. Két komponensük van, érték és típus. Egy kifejezés formálisan a következő összetevőkből áll:

- operandusok: Az operandus literál, nevesített konstans, változó vagy függvényhívás lehet. Az értéket képviseli.
- operátorok: Műveleti jelek. Az értékekkel végrehajtandó műveleteket határozzák meg.
- kerek zárójelek: A műveletek végrehajtási sorrendjét befolyásolják.

Minden nyelv megengedi a redundáns zárójelek alkalmazását. A legegyszerűbb kifejezés egyetlen operandusból áll. Attól függően, hogy egy operátor hány operandussal végzi a műveletet, beszélünk egyoperandusú (unáris), kétoperandusú (bináris), vagy háromoperandusú (ternáris) operátorokról. A kifejezésnek három alakja lehet attól függően, hogy kétoperandusú operátorok esetén az operandusok és az operátor sorrendje milyen. A lehetséges esetek:

- prefix, az operátor az operandusok előtt áll (* 3 5)
- infix, az operátor az operandusok között áll (3 * 5)
- postfix, az operátor az operandusok mögött áll (3 5 *)

Az egyoperandusú operátorok általában az operandus előtt, ritkábban mögötte állnak. A háromoperandusú operátorok általában infixek.

Azt a folyamatot, amikor a kifejezés értéke és típusa meghatározódik, a kifejezés kiértékelésének nevezzük. A kiértékelés során adott sorrendben elvégezzük a műveleteket, előáll az érték, és hozzárendelődik a típus. A műveletek végrehajtási sorrendje a következő lehet:

- A műveletek felírási sorrendje, azaz balról-jobbra.
- A felírási sorrenddel ellentétesen, azaz jobbról-balra.
- Balról-jobbra a precedencia táblázat figyelembevételével.

Az infix alak nem egyértelmű. Az ilyen nyelvek operátoraikat egy precedencia táblázatban adják meg. A teljesen zárójelezett infix kifejezés egyértelmű, kiértékelésénél egyetlen sorrend létezik.

A kiértékelés szempontjából speciálisak azok a kifejezések, amelyekben logikai operátorok szerepelnek. Ezeknél ugyanis úgyis eldőlhet a kifejezés értéke, hogy nem végzem el az összes kijelölt műveletet. Például egy ÉS művelet esetén, ha az első operandus értéke hamis, az eredmény hamis lesz, függetlenül a második operandus értékétől.

A kifejezés típusának meghatározásánál kétféle elvet követnek a nyelvek. Vannak a típusegyenértékűséget és vannak a típuskényszerítést vallók. Ugyanez a kérdéskör fölmerül az értékadó utasításnál is.

A típusegyenértékűséget valló nyelvek azt mondják, hogy egy kifejezésben egy kétoperandusú operátornak csak azonos típusú operandusai lehetnek. Ilyenkor nincs konverzió.

A különböző nyelvek szerint két programozási eszköz típusa azonos, ha azoknál fönnáll a

- deklaráció egyenértékűség: az adott eszközöket azonos deklarációs utasításban, együtt, azonos típusnévvel deklaráltuk.
- név egyenértékűség: az adott eszközöket azonos típusnévvel deklaráltuk (esetleg különböző deklarációs utasításban).
- struktúra egyenértékűség: a két eszköz összetett típusú és a két típus szerkezete megegyezik (például két 10-10 egészet tartalmazó tömbtípus, függetlenül az indexek tartományától).

Utasítások

Az utasítások alkotják az eljárásorientált nyelveken megírt programok olyan egységeit, amelyekkel egyrészt az algoritmusok egyes lépéseit megadjuk, másrészt a fordítóprogram ezek segítségével generálja a tárgyprogramot. Két nagy csoportjuk van: a deklarációs és a végrehajtható utasítások.

A deklarációs utasítások mögött nem áll tárgykód. Ezen utasítások teljes mértékben a fordítóprogramnak szólnak, attól kérnek valamilyen szolgáltatást, üzemmódot állítanak be, illetve olyan információkat szolgáltatnak, melyeket a fordítóprogram felhasznál a tárgykód generálásánál.

Alapvetően befolyásolják a tárgykódot, de maguk nem kerülnek lefordításra. A programozó a névvel rendelkező saját programozási eszközeit tudja deklarálni.

A végrehajtható utasításokból generálja a fordítóprogram a tárgykódot. Általában a magas szintő nyelvek végrehajtható utasításaiból több (néha sok) gépi kódú utasítás áll elő. A végrehajtható utasításokat az alábbiak szerint csoportosíthatjuk:

1. Értékadó utasítás - Feladata beállítani vagy módosítani egy (esetleg több) változó értékkomponensét a program futásának bármely pillanatában.

2. Üres utasítás - Az üres utasítás hatására a processzor egy üres gépi utasítást hajt végre.

3. Ugró utasítás - Az ugró (vagy feltétel nélküli vezérlésátadó) utasítás segítségével a program egy adott pontjáról egy adott címkével ellátott végrehajtható utasításra adhatjuk át a vezérlést. Általánosan használt alakja: GOTO címke

4. Elágaztató utasítások - A kétirányú elágaztató utasítás arra szolgál, hogy a program egy adott pontján két tevékenység közül válasszunk, illetve egy adott tevékenységet végrehajtsunk vagy sem. A nyelvekben meglehetősen általános a feltételes utasítás következő szerkezete: ŐF feltétel THEN tevékenység [ELSE tevékenység] A feltétel egy logikai (vagy annak megfelelő típusú) kifejezés. Kérdés, hogy egy nyelv mit mond a tevékenység megadásáról. A többirányú elágaztató utasítás arra szolgál, hogy a program egy adott pontján egymást kölcsönösen kizáró akárhány tevékenység közül egyet végrehajtsunk. A tevékenységek közötti választást egy kifejezés értékei szerint tehetjük meg. Szintaktikája és szemantikája nyelvenként különböző.

5. Ciklusszervező utasítások - ciklusszervező utasítások lehetővé teszik, hogy a program egy adott pontján egy bizonyos tevékenységet akárhányszor megismételjünk. Egy ciklus általános felépítése a következő: fej, mag, vég. Az ismétlésre vonatkozó információk vagy a fejben vagy a végben szerepelnek. A mag az ismétlendő végrehajtható utasításokat tartalmazza. A ciklusok működésénél megkülönböztetünk két szélsőséges esetet. Az egyik az, amikor a mag egyszer sem fut le, ezt hívjuk üres ciklusnak. A másik az, amikor az ismétlődés soha nem áll le, ez a végtelen ciklus. A programozási nyelvekben a következő ciklusfajtákat különböztetjük meg: feltételes, előírt lépésszámú, felsorolásos, végtelen és összetett ciklus.

6. Hívó utasítás

7. Vezérlésátadó utasítások - A C-ben három vezérlő utasítás van még az eddigiekben tárgyalt végrehajtható utasításokon kívül: CONTŐNUE; Ciklus magjában alkalmazható. A ciklus magjának hátralévő utasításait nem hajtja végre, hanem az ismétlődés feltételeit vizsgálja meg, és vagy újabb ciklusléptésbe kezd, vagy befejezi a ciklust. BREAK; Ciklus magjában vagy többszörös elágaztató utasításban helyezhető el. A ciklust szabályosan befejezteti, illetőleg kilép a többszörös elágaztató utasításból. RETURN [kifejezés].

8. Ő/O utasítások

9. Egyéb utasítások

Programegységek

Az eljárásorientált programnyelvekben a program szövege többé-kevésbé független, szuverén részekre, ún. programegységekre tagolható.

Bizonyos nyelvekben a program fizikailag önálló részekből áll, melyek külön-külön fordíthatók. Ezek a részek mélységében nem strukturáltak. Más nyelvekben a programot egyetlen egységként kell

lefordítani. Ilyenkor a program szövege mélységében strukturálható. A programegységek fizikailag nem függetlenek. Végül az előző kettő kombinációja is elképzelhető. Ezen nyelvekben fizikailag független, de tetszőleges belső struktúrával rendelkező programegységek léteznek.

Az eljárásorientált nyelvekben az alábbi programegységek léteznek:

-alprogram - Az alprogram mint absztrakciós eszköz egy bemeneti adatcsoportot képez le egy kimeneti adatcsoportra úgy, hogy egy specifikáció megadja az adatok leírását, de semmit nem tudunk magáról a tényleges leképezésről. Ismerjük a specifikációt, de nem ismerjük az implementációt. Az alprogram, mint programozási eszköz az újrafelhasználás eszköze. Akkor alkalmazható, ha a program különböző pontjain ugyanaz a programrész megismétlődik. Az alprogram az adott helyeken meghívható, aktivizálható. Ezeket formális paraméterekkel látjuk el, vagyis általánosabban írjuk meg, mint ahogyan az adott helyeken szerepelt volna. Formálisan az alprogram a következőképpen épül fel: fej vagy specifikáció, törzs vagy implementáció, vég. Az alprogram, mint programozási eszköz négy komponensből áll: név, formális paraméter lista, törzs, környezet. Az alprogramoknak két fajtája van: eljárás és függvény. A függvényhívás után normális befejeződést feltételezve a vezérlés a kifejezésbe tér vissza, és tovább folytatódik annak a kiértékelése.

-blokk - A blokk olyan programegység, amely csak másik programegység belsejében helyezkedhet el, külső szinten nem állhat. Formálisan a blokknak van kezdete, törzse és vége. A kezdetet és a véget egy-egy speciális karaktersorozat vagy alapszó jelzi. A törzsben lehetnek deklarációs és végrehajtható utasítások. Ugyanúgy, mint az alprogramoknál, ezek az utasítások vagy tetszőlegesen keverhetők, vagy van külön deklarációs rész és végrehajtható rész. A blokknak nincs paramétere. A blokknak egyes nyelvekben lehet neve. Ez általában a kezdet előtt álló címke. A blokk bárhol elhelyezhető, ahol végrehajtható utasítás állhat. Blokkot aktivizálni vagy úgy lehet, hogy szekvenciálisan rákerül a vezérlés, vagy úgy, hogy GOTO-utasítással ráugrunk a kezdetére. Egy blokk befejeződhet, ha elértük a végét, vagy GOTO-utasítással kilépünk belőle, vagy befejeztetjük az egész programot a blokkban. A blokkot az eljárásorientált nyelveknek csak egy része ismeri. Szerepe a nevek hatáskörének elhatárolásában van.

-csomag

-taszk

Hatáskör

A hatáskör a nevekhez kapcsolódó fogalom. Egy név hatásköre alatt értjük a programszövegének azon részét, ahol az adott név ugyanazt a programozási eszközt hivatkozza, tehát jelentése, felhasználási módja, jellemzői azonosak. A hatáskör szinonimája a láthatóság.

Egy programegységben deklarált nevet a programegység lokális nevének nevezzük. Azt a nevet, amelyet nem a programegységben deklaráltunk, de ott hivatkozunk rá, szabad névnek hívjuk.

Azt a tevékenységet, mikor egy név hatáskörét megállapítjuk, hatáskörkezelésnek hívjuk. Kétféle hatáskörkezelést ismerünk, a statikus és a dinamikus hatáskörkezelést.

A statikus hatáskörkezelés fordítási időben történik, a fordítóprogram végzi. Alapja a programszöveg programegység szerkezete. Ha a fordító egy programegységben talál egy szabad nevet, akkor kilép a tartalmazó programegységbe, és megnézi, hogy a név ott lokális-e. Ha igen vége a folyamatnak, ha nem, akkor tovább lépked kifelé, egészen addig, amíg meg
nem találja lokális névként, vagy el nem jut a legkülső szintre. Ha kiért a legkülső szintre, akkor két eset lehetséges:

- A nyelvek egy része azt mondja, hogy a programozónak minden nevet deklarálni kell. Így, ha egy név nem volt deklarálva, az fordítási hiba.
- A nyelvek másik része ismeri az automatikus deklarációt, és a névhez a fordító hozzárendeli az automatikus deklaráció szabályainak megfelelő attribútumokat. A név ilyenkor tehát a legkülső szint lokális neveként értelmeződik.

Statikus hatáskörkezelés esetén egy lokális név hatásköre az a programegység, amelyben deklaráltuk és minden olyan programegység, amelyet ez az adott programegység tartalmaz, hacsak a tartalmazott programegységekben a nevet nem deklaráltuk újra. A hatáskör befelé terjed, kifelé soha.

Egy programegység a lokális neveit bezárja a külvilág elől. Azt a nevet, amely egy adott programegységben nem lokális név, de onnan látható, globális névnek hívjuk. A globális név, lokális név relatív fogalmak. Ugyanaz a név az egyik programegység szempontjából lokális, egy másikban globális, egy harmadikban pedig nem is látszik.

A dinamikus hatáskörkezelés futási idejű tevékenység, a futtató rendszer végzi. Alapja a hívási lánc. Ha a futtató rendszer egy programegységben talál egy szabad nevet, akkor a hívási láncon keresztül kezd el visszalépkedni mindaddig, amíg meg nem találja lokális névként, vagy a hívási lánc elejére nem ér. Ez utóbbi esetben vagy futási hiba keletkezik, vagy automatikus deklaráció következik be.

Statikus hatáskörkezelés esetén a programban szereplő összes név hatásköre a forrásszöveg alapján egyértelműen megállapítható. Dinamikus hatáskörkezelésnél viszont a hatáskör futási időben változhat és más-más futásnál más-más lehet.

Az eljárásorientált nyelvek a statikus hatáskörkezelést valósítják meg.

Paraméterkiértékelés, átadás

Paraméterkiértékelés alatt értjük azt a folyamatot, amikor egy alprogram hívásánál egymáshoz rendelődnek a formális- és aktuális paraméterek, és meghatározódnak azok az információk, amelyek a paraméterátadásnál a kommunikációt szolgáltatják.

A paraméterkiértékelésnél mindig a formális paraméter lista az elsődleges, ezt az alprogram specifikációja tartalmazza, egy darab van belőle. Aktuális paraméter lista viszont annyi lehet, ahányszor meghívjuk az alprogramot. Tehát az egymáshoz rendelésnél mindig a formális paraméter lista a meghatározó, mindig az aktuális paramétereket rendeljük a formálisakhoz.

Melyik formális paraméterhez melyik aktuális paraméter fog hozzárendelődni? Ez történhet sorrendi kötés vagy név szerinti kötés szerint. Sorrendi kötés esetén a formális paraméterekhez a felsorolás sorrendjében rendelődnek hozzá az aktuális paraméterek: az elsőhöz az első, a másodikhoz a második, és így tovább. Ezt a lehetőséget minden nyelv ismeri, és általában ez az alapértelmezés. A név szerinti kötés esetén az aktuális paraméter listában határozhatjuk meg az egymáshoz rendelést úgy, hogy megadjuk a formális paraméter nevét és mellette valamilyen szintaktikával az aktuális paramétert. Ilyenkor lényegtelen a formális paraméterek sorrendje. Néhány nyelv ismeri.

Hány darab aktuális paramétert kell megadni? Lehetséges, hogy a formális paraméterek száma fix, a formális paraméter lista adott számú paramétert tartalmaz. Ekkor a paraméterkiértékelés kétféle módon mehet végbe:

- Az aktuális paraméterek számának meg kell egyeznie a formális paraméterek számával.
- Az aktuális paraméterek száma kevesebb lehet, mint a formális paraméterek száma. Ez csak érték szerinti paraméterátadási mód esetén lehetséges. Azon formális paraméterekhez,

amelyekhez nem tartozik aktuális paraméter, a formális paraméter listában alapértelmezett módon rendelődik érték.

Lehet olyan eset, amikor a formális paraméterek száma nem rögzített, tetszőleges. Ekkor az aktuális paraméterek száma is tetszőleges. Létezik olyan megoldás is, hogy a paraméterek számára van alsó korlát, tehát legalább ennyi aktuális paramétert szerepeltetni kell.

Mi a viszony a formális és aktuális paraméterek típusai között? A nyelvek egyik része a típusegyenértékűséget vallja, ekkor az aktuális paraméter típusának azonosnak kell lennie a formális paraméter típusával. A nyelvek másik része a típuskényszerítés alapján azt mondja, hogy az aktuális paraméter típusának konvertálhatónak kell lennie a formális paraméter típusára.

A paraméterátadás az alprogramok és más programegységek közötti kommunikáció egy formája. A paraméterátadásnál mindig van egy hívó, ez tetszőleges programegység, és egy hívott, amelyik mindig alprogram. Kérdés, hogy melyik irányban és milyen információ mozog. A nyelvek a következő paraméterátadási módokat ismerik:

- érték szerinti
- cím szerinti
- eredmény szerinti
- érték-eredmény szerinti
- név szerinti
- szöveg szerinti

Az érték szerinti paraméterátadás esetén a formális paramétereknek van címkomponensük a hívott alprogram területén. Az aktuális paraméternek rendelkeznie kell értékkomponenssel a hívó oldalon. Ez az érték meghatározódik a paraméterkiértékelés folyamán, majd átkerül a hívott alprogram területén lefoglalt címkomponensre. A formális paraméter kap egy kezdőértéket, és az alprogram ezzel az értékkel dolgozik a saját területén.

A cím szerinti paraméterátadásnál a formális paramétereknek nincs címkomponensük a hívott alprogram területén. Az aktuális paraméternek viszont rendelkeznie kell címkomponenssel a hívó területén. Paraméterkiértékeléskor meghatározódik az aktuális paraméter címe és átadódik a hívott alprogramnak, ez lesz a formális paraméter címkomponense.

Az eredmény szerinti paraméterátadásnál a formális paraméternek van címkomponense a hívott alprogram területén, az aktuális paraméternek pedig lennie kell címkomponensének. Paraméterkiértékeléskor meghatározódik az aktuális paraméter címe, és átadódik a hívott alprogramnak, azonban az alprogram a saját területén dolgozik, és a futás közben nem használja ezt a címet.

Az érték-eredmény szerinti paraméterátadásnál a formális paraméternek van címkomponense a hívott területén és az aktuális paraméternek rendelkeznie kell érték- és címkomponenssel. A paraméterkiértékelésnél meghatározódik az aktuális paraméter értéke és címe és mindkettő átkerül a hívotthoz. Az alprogram a kapott értékkel, mint kezdőértékkel kezd el dolgozni a saját területén és a címet nem használja. Miután viszont befejeződik, a formális paraméter értéke átmásolódik az aktuális paraméter címére.

Név szerinti paraméterátadásnál az aktuális paraméter egy, az adott szövegkörnyezetben értelmezhető tetszőleges szimbólumsorozat lehet. A paraméterkiértékelésnél rögzítődik az alprogram szövegkörnyezete, itt értelmezésre kerül az aktuális paraméter, majd a szimbólumsorozat a formális

paraméter nevének minden előfordulását felülírja az alprogram szövegében, és ezután fut le az. Az információáramlás iránya az aktuális paraméter adott szövegkörnyezetbeli értelmezésétől függ.

A szöveg szerinti paraméterátadás a név szerintinek egy változata, annyiban különbözik tőle, hogy a hívás után az alprogram elkezd működni, az aktuális paraméter értelmező szövegkörnyezetének rögzítése és a formális paraméter felülírása csak akkor következik be, amikor a formális paraméter neve először fordul elő az alprogram szövegében a végrehajtás folyamán.

Egy adott esetben a paraméterátadás módját az alábbiak döntik el:

- a nyelv csak egyetlen paraméterátadási módot ismer (pl. C)
- a formális paraméter listában explicit módon meg kell adni a paraméterátadási módot (pl. Ada)
- az aktuális és formális paraméter típusa együttesen dönti el (pl. PL/Ő)
- a formális paraméter típusa dönti el (pl. FORTRAN)

Az alprogramok formális paramétereit három csoportra oszthatjuk:

- Input paraméterek: ezek segítségével az alprogram kap információt a hívótól (pl. érték szerinti paraméterátadás).
- Output paraméterek: a hívott alprogram ad át információt a hívónak (pl. eredmény szerinti paraméterátadás).
- Input-output paraméterek: az információ mindkét irányba mozog (pl. érték-eredmény szerinti paraméterátadás).

Absztrakt adattípus

Az absztrakt adattípus olyan adattípus, amely megvalósítja a bezárást vagy információ rejtést. Ez azt jelenti, hogy ezen adattípusnál nem ismerjük a reprezentációt és a műveletek implementációját. Az adattípus ezeket nem mutatja meg a külvilág számára. Az ilyen típusú programozási eszközök értékeihez csak szabályozott módon, a műveleteinek specifikációi által meghatározott interfészen keresztül férhetünk hozzá. Tehát az értékeket véletlenül vagy szándékosan nem ronthatjuk el. Ez nagyon lényeges a biztonságos programozásszempontjából. Az absztrakt adattípus (angol rövidítéssel: ADT – Abstract Data Type) az elmúlt évtizedekben a programnyelvek egyik legfontosabb fogalmává vált és alapvetően befolyásolta a nyelvek fejlődését.

Kivételkezelés

A kivételkezelési eszközrendszer azt teszi lehetővé, hogy az operációs rendszertől átvegyük a megszakítások kezelését, felhozzuk azt a program szintjére. A kivételek olyan események, amelyek megszakítást okoznak. A kivételkezelés az a tevékenység, amelyet a program végez, ha egy kivétel következik be. Kivételkezelő alatt egy olyan programrészt fogunk érteni, amely működésbe lép egy adott kivétel bekövetkezte után, reagálva az eseményre.

A kivételkezelés az eseményvezérlés lehetőségét teszi lehetővé a programozásban.

Operációs rendszer szinten lehetőség van bizonyos megszakítások maszkolására. Ez a lehetőség megvan nyelvi szinten is. Egyes kivételek figyelése letiltható vagy engedélyezhető. Egy kivétel figyelésének letiltása a legegyszerűbb kivételkezelés. Ekkor az esemény hatására a megszakítás bekövetkezik, feljön programszintre, kiváltódik a kivétel, de a program nem vesz róla tudomást, fut tovább. Természetesen nem tudjuk, hogy ennek milyen hatása lesz a program további működésére, lehet, hogy az rosszul, vagy sehogy sem tudja folytatni munkáját.

A kivételeknek általában van neve (amely gyakran az eseményhez kapcsolódó üzenet szerepét játssza) és kódja (ami egy egész szám).