12- Típus

Típus

A $\mathcal{T} = (\varrho, I, \mathbb{S})$ *hármast típusnak nevezzük, ha*

1. $\varrho \subseteq E^* \times T$ *a reprezentációs függvény (reláció),*
   T *a típusértékhalmaz,*
   E *az elemi típusértékhalmaz,*
2. $I: E^* \to \mathbb{L}$ *típusinvariáns,*
3. $\mathbb{S} = \{S_1, S_2, \dots, S_m\}$, *ahol*
   $\forall i \in [1..m]: S_i \subseteq B_i \times B_i^{**}$ *program,* $B_i = B_{i_1} \times \dots \times B_{i_{m_i}}$ *úgy,*
   hogy $\exists j \in [1..m_i]: B_{i_j} = E^*$ *és* $\nexists j \in [1..m_i]: B_{i_j} = T$

A típus első két komponense az absztrakt típusértékek reprezentációját írja le, míg a programhalmaz a típusműveletek implementációját tartalmazza. Az elemi típusértékhalmaz lehet egy tetszőleges másik típus típusértékhalmaza vagy egy, valamilyen módon definiált, legfeljebb megszámlálható halmaz.

Típusspecifikáció

A $\mathcal{T}_s = (H, I_s, \mathbb{F})$ *hármast típusspecifikációnak nevezzük, ha*

1. H *az alaphalmaz,*
2. $I_s: H \to \mathbb{L}$ *a specifikációs invariáns,*
3. $T_{\mathcal{T}} = \{(\mathcal{T}, x) \mid x \in [I_s]\}$ *a típusértékhalmaz,*
4. $\mathbb{F} = \{F_1, F_2, \dots, F_n\}$ *a típusműveletek specifikációja, ahol*
   $\forall i \in [1..n]: F_i \subseteq A_i \times A_i, A_i = A_{i_1} \times \dots \times A_{i_{n_i}}$ *úgy, hogy* $\exists j \in [1..n_i]: A_{i_j} = T_{\mathcal{T}}$

Az alaphalmaz és az invariáns tulajdonság segítségével azt fogalmazzuk meg, hogy mi az a halmaz, $T_{\mathcal{T}}$*, amelynek elemeivel foglalkozni akarunk, míg a feladatok halmazával azt írjuk le, hogy ezekkel a elemekkel milyen műveletek végezhetők el.*

Invariáns

Az invariáns lényege, hogy ezt a tulajdonságot soha sem sérthetjük meg. Például halmaz típus esetén, nem szabad, hogy megsérüljön az az invariáns tulajdonság, hogy egy halmazban egy elem csak egyszer fordulhat elő.

Reprezentáció

A reprezentáció a típusspecifikáció típusértékhalmazának leképezése a konkrét típusban, amit a reprezentációs függvény ad meg.

Azt, hogy egy típust milyen típusok segítségével, milyen módszerrel, stb. valósítottunk meg, reprezentációnak nevezzük. Például egy verem típust meg lehet valósítani tömb segítségével, de láncolt listával is.

Implementáció

Az implementáció a típusspecifikáció típusműveleteinek megvalósítása a konkrét típus programhalmaza által.

Az implementáció során a típus megvalósításakor a típusértékhalmaz megadását követően definiálni kell a típusműveleteket. Ahogyan a modellben is, a gyakorlatban is az állapottér változásait a program csak a típusműveleteken keresztül végezheti el. Előfordulhat, hogy nincs szükség típusműveletekre, ilyen pl.: a C++ felsorolási típusa (enum).

Primitív típusok

Azok az elemi típusok, melyek típusérték halmaza az egész számok, a természetes számok, a logikai értékek vagy a karakterek halmaza.

Rekord típus

Bővebben a direkszorzat típuskonstrukciókról (122. o.) >>

Egy $T = (s_1 : T_1, \dots, s_n : T_n)$ típust rekordnak nevezünk (ez egy direktszorzat). Az i-edik komponens elérésére bevezetjük a $t \in T$ rekordra a $t.s_i$ jelölést, amelyet értékadásra is használhatunk.

Sorozat típus

Bővebben az iterált típuskonstrukciókról (124. o.) >>

Gyakori és sokféle művelettel implementálható.

Példa

- *lorem -> alsó elem leválasztása.*
- *hiext -> elem beszúrása az utolsó utáni helyre.*
- *lov -> alsó elem lekérdezése.*
- *dom -> elemszám lekérdezése.*

Alkalmazás

- *Szekvenciális fájl feldolgozása.*
- *Sor absztrakt megvalósítása.*

Vektor típus

Véges sok egyforma típusú adat, sorszámozva.

Fontosabb műveletei

- *v[i] –> i. elem elérése.*
- *v.lob és v.hib az alsó és felső index.*

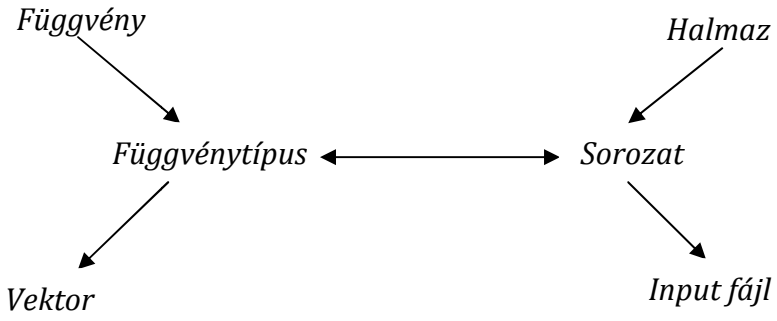
Programozási tételek típus-transzformációi

Bővebben (171. o.) >>

Transzformáció

A transzformáció lényege, hogy az absztrakt programozási tételeket konkrét feladatot megoldó algoritmusokra írjuk át.

Típustranszformációról akkor beszélhetünk, ha az állapottér bizonyos komponenseit valami kapcsolódó típusra cseréljük. Az ábrán látható valamely típuson megoldott program egyszerű szabályok segítségével átvihető egy kapcsolódó típusra:



Vektor esetén

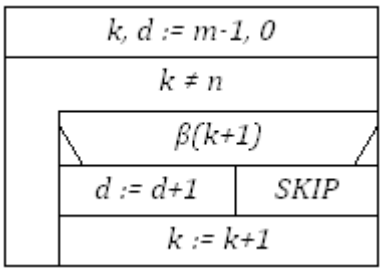
A vektorra az összegzés, számlálás, maximumkeresés, feltételes maximumkeresés, lineáris keresés és logaritmikus keresés egyszerű típustranszformációval visszavezethetőek az intervallumra megfogalmazott változatokra, ha az intervallum alsó- és felső határát jelölő m és n változókat a vektor alsó és felső határát megadó v.lob és v.hib típusműveletekkel helyettesítjük, mind a specifikációban, mind az absztrakt megoldó programban.

Intervallum felett definiált függvényről sorozatra

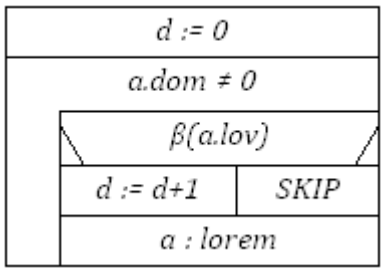
A sorozat nem indexelhető, de erre nincs is szükség, mert a tételek mindig az intervallumok (k+1). elemét vizsgálják. Ezért a tételekben az alábbi helyettesítéseket végezzük el:

- $k \neq n \rightarrow a.dom \neq 0$
- $f(k+1) \rightarrow f(a.lov)$
- $k := k+1 \rightarrow a: lorem$

Példa (Számlálás tétele)



Eredeti programozási tétel



Átírás sorozat típusra

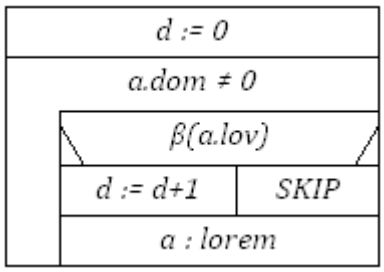
Sorozatról szekvenciális input fájlra

A szekvenciális input fájl sx, dx, x : read tájékoztat az olvasás eredményéről (sx), tárolja a beolvasott értéket (dx), és eltávolítja az olvasott elemet. Fontos, hogy a szekvenciális input fájlról semmi információnk nincs, amíg nem végzünk egy read műveletet, ezért rögtön a program elején kell egy olvasás (ez az előreolvasási technika). Tehát a megfeleltetés:

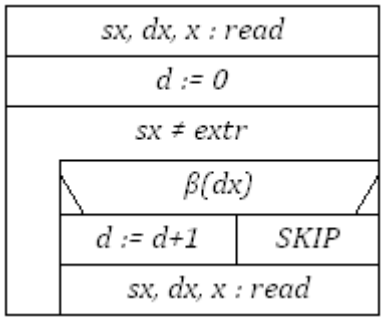
- *(A program elején egy előreolvasás).*
- $a.lov \rightarrow dx$
- $a: lorem \rightarrow sx, dx, x : read$
- $a.dom \neq 0 \rightarrow sx \neq extr$

Az „extr" egy úgynevezett extremális elem, mely általában egy valamilyen sorozatból való olvasás végét jelzi. Tehát extremális elem a beolvasott sorozat végét jelöli, és magában a beolvasott sorozat nem tartalmazza az extremális elemet (kivéve a legvégén).

Példa (Számlálás tétele)



Sorozat típusra



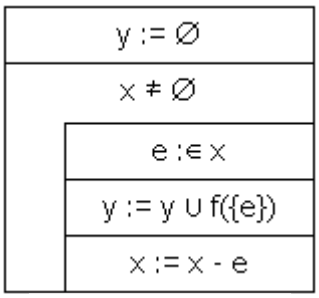
Átírás fájl típusra

Halmazról sorozatra

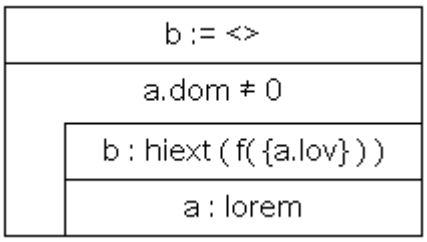
Az a és b sorozatok tagjai felsorolják az x és y halmazok elemeit. A megfeleltetés:

- $y := \emptyset \rightarrow b :=<>$
- $x \neq \emptyset \rightarrow a.dom \neq 0$
- $e: \in x \rightarrow (nincs\ helyettesítés)$
- $e \rightarrow a.lov$
- $y\ \tilde{\cup} \rightarrow b : hiext$
- $x\ \tilde{} \rightarrow a : lorem$

Példa (Egyváltozós egyértékű elemenkénti feldolgozás)



Halmaz típusra



Átírás sorozat típusra

Állapottér transzformáció

Ha egy visszavezetés során azzal szembesülünk, hogy az állapottér különböző komponensei között kell kapcsolatot teremteni, akkor az állapottér transzformációval oldható meg. Ennek során a problémát átfogalmazzuk egy úgynevezett absztrakt állapottérre, ahol már meg tudjuk oldani.

Példa

Adott egy szekvenciális input fájl, mely karaktereket tartalmaz. Kérdés, hogy melyik a leghosszabb összefüggő karaktersorozat?

Vezessünk be egy új, absztrakt állapotot, azt, ahol a szekvenciális input az összefüggő karaktersorozatok hosszainak sorozata. Ezen az állapottéren, a feladatot, a szekvenciális inputra megfogalmazott maximumkeresés programozási tétel megoldja. Az absztrakció pedig elvégezhető egy absztrakt read függvénnyel, amely a szekvenciális inputban szóköztől szóközig olvas, és megszámolja a karaktereket, ami pedig a szekvenciális inputra megfogalmazott számlálás tétel.

Állapottér-transzformáció alkalmazás esetén a feladatot úgy oldjuk meg, hogy az eredeti állapottér helyett egy új (absztrakt) állapotteret választunk, azon megoldjuk a feladatot, majd a megoldásban szereplő műveleteket megvalósítjuk az eredeti téren.

Bővebben (176. o.) >>