

Rekord típusok

A direkt szorzat és az unió típuskonstrukciókat számos nyelvben az ún. rekord típusok segítségével valósítják meg.

2005.03.02.

1

Direkt szorzat

- Ha adott két típus, S és T, direkt szorzatukat $S \times T$ jelöli.
 - $S \times T = \{(x,y) \mid x \in S; y \in T\}$
 - Ez általánosítható több halmazra is:
 $S_1 \times S_2 \times \dots \times S_n$
- A COBOL, Pascal, Ada stb. rekordjait, az Algol68, C, C++ struktúráit a direkt szorzat terminusaival érthetjük meg.

2005.03.02.

2

```
record
  <name1> : <type1>;
  <name2> : <type2>;
  ...
  <namek> : <typek>;
end
```

- A komponenseket a rekord mezőinek hívják.
- Az alapművelet a komponens kiválasztás.

2005.03.02.

3

Lehet-e paramétere a típusnak?
Van-e kezdő értékadás a mezőkre?
Van-e teljes rekordra vonatkozó értékadás?
Van-e rekord konstans?
Hogyan működik a kiválasztás művelet?

2005.03.02.

4

Pascal
type rektipnev = record
 mnev₁ : típus₁;
 ...
 mnev_n : típus_n;
end;

- változó deklarálása:
rek: rektipnev;
- hivatkozás ponttal: rek.mnev₁
- csak mezőnkénti értékadás lehetséges

2005.03.02.

5

- Speciális utasítás, aminek a segítségével a rekord mezőire közvetlenül tudunk hivatkozni:

```
type Date= record
  Year: Integer;
  Month: 1..12;
  Day: 1..31;
end;
var R1,R2: Date;
begin
  R1:=R2; {értékadás megengedett} ...
  with R1 do begin
    Year := 2000; Month:=2; Day:=29;
    Year := R2.Year;
  end;
```

2005.03.02.

6

C++

```
struct strnev {  
    tipus1 mnev1;  
    ...  
    tipusn mnevn;  
};
```

- változó deklarálása:
strnev x;
- hivatkozás ponttal: x.mnev₁

2005.03.02.

7

- A tömbökre használt jelölés alkalmazható struktúrákra is. pl.:

```
struct address{  
    long number;  
    char* street;  
    char* town;  
    int zip;  
}  
address a={1, „Korkeakoulunkatu”, „Tampere”,  
33720}
```

- Az inicializálásra a konstruktorok (később...) jobban használhatóak.
- Értékadás megengedett, de az egyenlőségvizsgálat nem előre definiált. A felhasználó definiálhat operátorokat rá (később...).

2005.03.02.

8

Ada-95

```
type Complex is record  
    Re: Float;  
    Im: Float;  
end record;
```

- változó deklarálása:
C: Complex;
- a komponenseire
C.Re, vagy C.Im segítségével
hivatkozhatunk.
- Általában megengedett az értékadás is:
C1, C2: Complex; C1:=C2;

2005.03.02.

9

A rekord diszkriminánsa(i)

- a típus paramétere
- több diszkriminánsa is lehet egy típusnak
- a rekord diszkrimináns diszkrét típusú

```
type Szöveg( Hossz: Natural ) is record  
    Érték: String(1 .. Hossz) := (others=>' ');  
    Pozíció: Natural := 0;  
end record ;
```

2005.03.02.

10

- Speciális rekord fajta a *limited record*. Sem egyenlőségvizsgálat, sem értékadás nem definiált rá:

```
type Arek is limited record  
    X: Integer:=0;  
end record;  
A,B: Arek;  
begin  
    A.X:=1; --OK. a komponens nem limited  
    A:=(2); -- hiba, az egész rekord limited  
    B:=A; -- hiba, az egész rekord limited  
    B.X:=A.X; --OK. a komponens nem limited  
    if A=B then A.X:=2; end if;  
    -- hiba, az egész rekord limited  
    hiba, az egész rekord limited, nincs  
    egyenlőségvizsgálat
```

2005.03.02.

11

Bizonyos programozási nyelvekben –
pl. SmallTalk, Eiffel, Java - nincs rekord
típus, a tervezők az osztályok használatát
javasolják helyette.

C#: a rekord (struct) érték típus, az osztály
(class) referencia típus.

2005.03.02.

12

Uniók és variáns rekordok

- A variáns rekordokat olyan objektumok reprezentálására használjuk, melyeknek van néhány (de nem az összes) közös tulajdonsága. Az unió ennek speciális esete, amikor a közös rész üres.
- A variáns rekordoknak van egy közös része az összes ilyen típusú rekordra, és egy, ami a rekordok egy részhalmazára specifikus.
- A típusérték-halmaz a komponensei típusérték-halmazának az uniója.

2005.03.02.

13

„Választó” típusműveletek

- A programozási nyelvek a megbízhatóság különböző szintjén támogatják ezt az adatszerkezetet.
- Az unió típusnak van egy speciális, *tag*-nek nevezett komponense, és egy kiválasztási mechanizmusa, ami megadja a tag különböző értékeinek megfelelő alstruktúrákat. Ha a tag-et tárolja a rekord, és az alkomponensek elérhetősége ennek aktuális értékétől függ, akkor ez egy „megkülönböztetett” (*discriminated*) unió, különben ez egy „szabad” (*free*) unió.

2005.03.02.

14

Unió (Variáns rekord):

- Meg lehet-e állapítani, hogy a rekord melyik változat szerint lett kitöltve?
- Ki lehet-e olvasni a kitöltéstől eltérő változat szerint??

2005.03.02.

15

```
case <tag-name> : <tag-type> of
  <const1> : ( <fields1>);
  <const2> : ( <fields2>);
  ...
  <constv> : ( <fieldsv>);
```

2005.03.02.

16

- A szabad uniók esetében a tag mezők használata opcionális, és a fordító nem ellenőrzi a kiválasztott mező és a tárolt érték konzisztenciáját. Ez a nyelv típusrendszerét megbízhatatlanná teszi.
- A megkülönböztetett unió esetében fontos kérdés, hogyan lehet új értéket adni a tag mezőnek.
 - A tag értéket beállítja a rekord létrehozásakor.
 - Míg a program képes kell legyen új értéket adni a rekord "normális" komponenseinek, a tag megváltoztatása a rekord szerkezet megváltozását vonja maga után.

2005.03.02.

17

```
Pascal:
type Listptr = ^Listnode;
type Listnode = record
  Next: Listptr;
  case Tag: Boolean of
    False: (Data: char);
    True: (Down: Listptr)
  end;
var p, q: Listptr; ...
p^.Tag := true;
p^.Down := q;
p^.Tag := false;
writeln(p^.Data); {!!!!hiba!!!!}
```

2005.03.02.

18

C++:

```

union typename{
    typ1 field1;
    ...
    typm fieldm;
};

union Fudge{
    int i;
    int* p;
};

int* cheat(int i){
    Fudge a;
    a.i=i;
    return a.p;
}

```

HIBA!

2005.03.02. 19

Ada-95:

```

type Állapot is (Egyedülálló, Házas, Özvegy, Elvált);
subtype Név is String(1..25);
type Nem is (Nő, Férfi);
type Ember (Családi_Áll: Állapot := Egyedülálló) is
    record
        Neve: Név;
        Neme: Nem;
        Születési_Ideje: Dátum;
        Gyermek_Száma: Natural;
        case Családi_Áll is
            when Házas      => Házastárs_Neve: Név;
            when Özvegy     => Házastárs_Halála: Dátum;
            when Elvált      => Válás_Dátuma: Dátum;
        end case;
        Gyerekek_Gondozója:
    end record ;

```

Boolean;

```

    when Egyedülálló => null ;
end case;
end record ;

```

2005.03.02. 20

```

Hugó: Ember(Házas);
    Családi_Áll, Neve, Neme, Születési_Ideje,
    Gyermek_Száma, Házastárs_Neve
Eleonóra: Ember(Egyedülálló);
    Családi_Áll, Neve, Neme, Születési_Ideje,
    Gyermek_Száma
Ödön: Ember(Özvegy);
    Családi_Áll, Neve, Neme, Születési_Ideje,
    Gyermek_Száma, Házastárs_Halála
Vendel: Ember(Elvált);
    Családi_Áll, Neve, Neme, Születési_Ideje,
    Gyermek_Száma, Válás_Dátuma,
    Gyerekek_Gondozója
Aladár: Ember; -- Egyedülálló
    Családi_Áll, Neve, Neme, Születési_Ideje,
    Gyermek_Száma

```

■ **Helytelen** (futási idejű hiba, altípus-megszorítás megsértése):

2005.03.02. 21

Megszorítatlan altípus használata

```

Aladár: Ember; -- alapért. Egyedülálló
Aladár := (Házas, ....);
Aladár := Elek;
Aladár := Hugó;

```

- A szerkezetét megváltoztathatjuk, diszkriminánsostul
- Csak úgy, ha az egész rekord értéket kap egy értékadásban

2005.03.02. 22

Bizonyos programozási nyelvekben – pl. SmallTalk, Eiffel, Java, C# - nincs unió típus, a tervezők az osztályok és az öröklődés használatát javasolják helyette.

2005.03.02. 23

Halmaz

- Mi lehet az eleme?
- Hány eleme lehet?
- Megvannak-e a 'hagyományos' halmazműveletek?

2005.03.02. 24

Pascal

Alaphalmaz: diszkrét típus
Elemek száma: max. 256
Értékek sorszáma csak 0..255 között
type Small_Letters = set of 'a'..'z';
type Digits = set of '0'..'9';
type Day = (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday);
type Days = set of Day;
var A, B : Days;
A:=[Monday, Wednesday] - halmazkonstruktor
üres halmaz: []

2005.03.02.

25

Műveletek:

- értékadás,
- halmazműveletek:
 1. eleme-teszt (in),
 2. az =, <, >, részhalmaz reláció ('<=', '>=')
 3. (a '<' és '>' nem megengedett!)
 4. unió ('+'), differencia ('-').
 5. metszet ('*')

Ez a precedencia-sorrend is.

2005.03.02.

26

Mikor ekvivalens két típus?

- x, y: array[0..9] of integer;
z: array[0..9] of integer;
- Strukturális ekvivalencia esetén:
 - A rekordok mezőnevei is figyelembe vannak véve, vagy csak a struktúrájuk?
 - Számít-e a rekordmezők sorrendje?
 - Tömböknél elég-e az indexek számosságának egyenlőnek lenni, vagy az indexhatároknak is egyezniük kell?

2005.03.02.

27

```
Dimensions = RECORD
  Breadth: REAL;
  Length: REAL;
END;
```

```
Complex = RECORD
  RealPart: REAL;
  ImPart: REAL;
END;
```

```
Size: Dimensions;
Root: Complex;
```

Értékül adhatóak egymásnak?

Név szerinti ekvivalencia esetén:

- Deklarálhatók-e egy típushoz típusok, amelyekkel ekvivalens?
- Névtelen tömb- ill. rekordtípusok ekvivalensek-e valamivel?

2005.03.02.

29

Típuskonverziók: Van-e, és hogyan működik ?

- az automatikus konverzió?
- az identitáskonverzió?
- a bővítő konverzió?
- a szűkítő konverzió?
- a toString konverzió?

2005.03.02.

30

Java:

- **identitáskonverzió:**
 - a boolean csak ez szabad
- **bővítő konverzió :**
 - byte to short, int, long, float or double
 - short to int, long, float or double
 - int to long, float or double
 - long to float or double
 - float to double

2005.03.02.

31

Java:

- **szűkítő konverzió :**
 - byte to char
 - short to byte or char
 - char to byte or short
 - int to byte, short or char
 - long to byte, short, char or int
 - float to byte, short, char, int or long
 - double to byte, short, char, int, long or float

2005.03.02.

32

Változók

- **Változó =**
(név, attribútumhalmaz, hely, érték)
- **Láthatóság, Elérhetőség**
- **Hogyan definiálhatunk változókat és konstansokat?**

2005.03.02.

33

Változók

	szintaxis	példa
Pascal	var <identif>: <type>;	var i : integer;
C++	<type> <identif> [= <value>];	int i = 0;
Java	<type> <identif> [= <value>];	int i = 0;
ADA	<identif>: <type> [= <value>];	I : Integer := 0;
CLU	<identif>: <type> [= <value>];	i : int := 0;
Eiffel	<identif>: [expanded]<type>;	I : INTEGER;

2005.03.02.

34

Konstansok

	szintaxis	példa
Pasc.	const <name> = <value>;	const val = 3;
C++	#define <name> <value> vagy const <type> <name> = <value>;	#define val 3 const int val = 3;
Java	final <type> <name> = <value>;	final int val = 3;
ADA	<name> : constant <type> := <value>;	Val : constant Integer := 3;
CLU	<name> = <value>	x = 3
Eiffel	<name> : <type> is <value>;	A: INTEGER is 3;

2005.03.02.

35

Kifejezések

- **Prefix jelölés**
+ a b
- **Postfix jelölés**
a b +
- **Infix jelölés**
a + b
- **A műveletek precedencia szintjei**

2005.03.02.

36

Pascal

zárójel:	()
függvényhívás	fv(..)
unáris operátorok	not, @, ^, +, -
multiplikatív operátorok	*, /, and, div, mod, shl, shr
additív operátorok	+, - or, xor
relációk	in, <, >, <=, >=, <>
balról jobbra kiértékelés	

2005.03.02.

37

CLU

zárójelek:	()
unáris operátorok	~, -
hatványozás	**
multiplikatív operátorok	*, /, //
additív operátorok	+, -
relációk	<, >, <=, >=, ~=, =, <=, >=, <=, >=
logikai mult.	&, cand
logikai add.	!, cor
balról jobbra kiértékelés	

2005.03.02.

38

Ada95

zárójelek	()
legmagasabb prec. op.	**, abs, not
mult. op.	*, /, mod, rem
unáris add. op.	+, -
Bináris add. op.	+, -, &
relációs op.	=, /=, <, <=, >, >=
logikai op.	and, or, xor, and then, or else
balról jobbra kiértékelés	

2005.03.02.

39

C++

zárójelek	()
scope	::
selection, call, size	., -> [] () sizeof
postf, pref, compl, not, un. add. address	++, --, ~, !, +, & *
of, deref, cre, destroy, cast	new, delete ()
member	.*, ->*
multipl. op.	*, /, %
binary add.	+
shift	<<, >>
relációs	<, <=, >, >=
egyenlőség	==, !=
bitwise AND	&
bitwise excl. OR	^
bitwise OR	
logical AND	&&
logical OR	
cond. expr.	? :
értékdadások	=, *=, /=, %=, +=, -=, >>=, <<=, &=, ^=, !=
throw	throw
comma (sequence)	,

Java

postfix	., [] () ++ --
prefix	++ -- ~ ! + -
constr, cast	new ()
multipl. op.	*, /, %
binary add.	+
shift	<<, >>, >>>
relational	<, <=, >, >= instanceof
equality	==, !=
bitwise AND	&
bitwise excl. OR	^
bitwise OR	
logical AND	&&
logical OR	
cond. expr.	? :
értékdadások	=, *=, /=, %=, +=, -=, >>=, <<=, &=, ^=, !=

Eiffel

select	.
unáris	old strip not + - free unary
free non standard bináris	
hatványozás	^
multipl. op.	*, /, //, \
binary add.	+
relációs	=, /=, <, <=, >, >=
logical AND	and, and then
logical OR	or, or else
logical impl.	Implies
array const.	<<, >>
Semicolon	;

2005.03.02.

42