Esercizi su Tipi di Dato

Maggio 2022

Esercizio 1

Cosa stampa (indicando i passaggi del ragionamento seguito) il seguente frammento di codice scritto in un linguaggio che ammette parametri per riferimento. Nel codice, `T*` è il tipo "puntatore a un valore di tipo `T`", `T[]` è il tipo "array di elementi di tipo `T`", `print` è una funzione che stampa il valore di qualsiasi sequenza di parametri dati in input, separati da virgola e `&var` e `*var` corrispondono rispettivamente a referenziamento e dereferenziamento di una variable `var`.

```
void foo ( int* x, int* y, int* j ){
  x[ *j ] = *j + 1;
  print( *y );
  x++;
  x[ *j ] = *j;
  print( x[ *j ], *j );
  *j = *j + 1;
}

int[] x[ 10 ];
int i = 1;
x[ 0 ] = 1;
x[ 1 ] = 2;
x[ 2 ] = 3;
foo( x, &x[ i ], &i );
print( x[ i ], i );
```

Esercizio 2

Si consideri il seguente frammento in un linguaggio con tipi statici, dove `f` è una funzione di due argomenti:

```
int i, j;
float y, z;
y = f( i, j );
z = f( y, i );
```

Si fornisca (i) una possibile segnatura per la funzione f e (ii) le ipotesi che occorre fare sul sistema di tipi del linguaggio affinchè la segnatura del tipo dato in (i) sia corretta.

Esercizio 3

Facendo riferimento a uno o più linguaggi di programmazione reali, si fornisca un esempio di due tipi di dato che sono equivalenti strutturalmente ma non per nome. Si forniscano quindi due esempi di tipi che sono compatibili ma non equivalenti.

Esercizio 4

Si consideri il seguente frammento di codice, in un sistema di tipi nominale, dove `A` e `B` sono tipi non confrontabili e valgono le relazioni di sottotipaggio `A <: Top` e `B <: Top` e la notazione `X[? <: T]` e `X[? :> T]` indicano rispettivamente polimorfismo parametrico vincolato covariante e controvariante rispetto al tipo proprio `T`.

```
f( List< A > a, List< B > b ){
 List< B > b = a;         // I1
 List< Top > at = a;      // I2
 List< A > a1 = b;        // I3
 List< ? :> A > a2 = a;   // I4
 List< ? <: B > b1 = a1; // I5
}
```

Si dica quali istruzioni (`I1`, ...) e perchè verrebbero dichiarate non corrette da un type checker.

Esercizio 5

Si consideri la seguente dichiarazione di array multidimensionale

```
int x;
read ( x );
int A [ 10 ][ x ];
```

dove il comando `read(x)` permette di leggere il valore della variabile `x` dall'esterno.

Sappiamo che: un intero è memorizzato su 4 byte; l'array è memorizzato in ordine di riga (raw major), con indirizzi di memoria crescenti (cioè se un elemento è all'indirizzo `i`, il successivo è a `i + 4`, ecc.); il valore di `x` letto è 6.

Come viene memorizzato l'array `A`? Qual'è l'offset dell'elemento `A[2][3]` rispetto all'inizio dell'array? (Si risponda in notazione decimale).

Esercizio 6

Si assumano le seguenti relazioni `A :> B :> C`, dove `:>` è una relazione di sottotipaggio. Sia `a`, `b`, `c` rispettivamente variabili di tipo `A`, `B` e `C`. Siano inoltre `sa` una variabile di tipo `Set[A]` e rispettivamente `sb` e `sc` variabili di tipo `Set[B]` e `Set[C]`, dove il tipo `Set[T]` indica il tipo polimorfo "insieme di elementi di tipo `T`" e ha come operazioni `remove: () -> T` (dato nulla ritorna un valore di tipo `T`) e `add: T -> ()` (accetta un valore di tipo `T` e non ritorna nulla).

Indicare quali istruzioni verrebbero segnate come non corrette da un controllore di un sistema di tipi nominale con supporto al polimorfismo parametrico.

```
a = b
c = b
a = sb.remove()
c = sb.remove()
sa = sb
sb = sa
sc = sb
sb = sb
sa.add( sc )
sa.add( sb.remove() )
b = sb.add( b )
c = sb.remove()
```

Esercizio 7

Sia `A :> B`, dove `:>` è una relazione di sottotipaggio in un sistema di tipi strutturale. Siano inoltre, usando la notazione `variabile: { tipo record }` e `T[]` per indicare array di tipo `T`

```
a: { id: A, name: string }
b: { id: B, name: string, address: string }
c: { name: string, id: B }[]
d: { id: A, address: string, name: string }[]
```

Indicare quali istruzioni verrebbero segnate come non corrette da un controllore di un sistema di tipi strutturale con supporto al sottotipaggio.

```
a = b
b.id = a.id
a.id = b.id
a.name = b.address
c[ 0 ] = b
d[ 0 ] = b
a = d[ 0 ]
c = d
d = c
c[ 0 ] = d[ 0 ]
d[ 0 ] = c[ 0 ]
d[ 0 ] = b
```