Type equivalence

- if ~~one~~ an operand of one type can be substituted for another in an expn, without coercion.

2 approaches

- Name $\overset{type}{\wedge}$ equivalence
- Structure $\overset{type}{\wedge}$ equivalence

Name type equivalence

- means that two var. have equivalent types if they are defined either in the same declaration or in declarations that use the same type name.

Structure type equivalence

- means 2 var. have equivalent types if their types have identical structure

eg:- ~~typedef stack-element int~~
~~stack-element pop(Stack *);~~
~~int a = pop(myStack);~~

~~typedef~~

```
typedef link struct *cell;
  link next, last;
  struct *cell p;
  struct *cell q, r;
```

- under structural equivalence all variables are type equivalent.
- under name equivalence, next and last are type equivalent and p, q and r are type equivalent.

Types of name equivalence:
- strict name equivalence
- loose name equivalence.

```
typedef stack-element int
stack-element pop(stack *);
 int a = pop(myStack);
typedef fahrenheit int
 typedef celcius int
```

farenheit f;
Celcius c;
c= f

strict name equivalence
- a lang. in which aliased types are considered distinct
∴ it raises a type error for c=f.

loose name equivalence
- in which aliases types are considered equivalent.
∴ allows stack assignment
c, c++ - use loose name equivalence

Ada
uses restrictive form of name equivalence
It provides 2 types of construct
-subtype
-derived type.

Derived type
- new type ie based on some previously defined type which is not equivalent, although it may have identical structure

- Derived types inherit all properties of the parent types.

eg:- type Celsius is new float;
     type Fahrenheit is new float;

Subtype

range constrained version of an existing type
type equivalent with parent type.

subtype Small-type is Integer range 0...99;

Type conversion

explicit req. by the programmer to convert one type to another.

```
eg:- int x
     float y = 6.3;
     x = (int) y;
```

Type Coercion

- implicit conversion from one type to another without informing user

```
int x;
short y;
y = x;
```