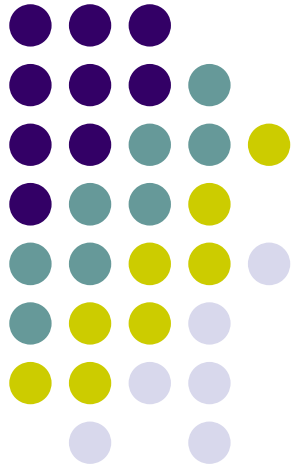
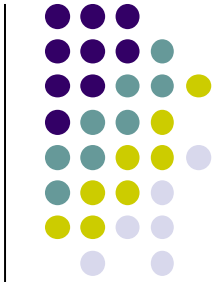
Bindings
Definiciones y Declaraciones



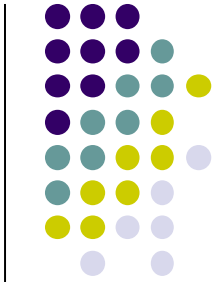
Definiciones y Declaraciones



Definición: Produce vinculaciones entre identificadores y entidades creadas en la propia definición.

Declaración: Produce vinculaciones entre identificadores y entidades ya creadas o que todavía lo serán.

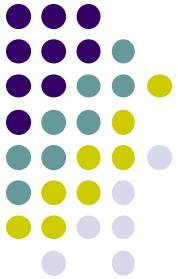
Definiciones y Declaraciones



- Lenguajes pueden restringir momentos para Definiciones y Declaraciones.
  - Pascal tiene un espacio reservado antes del bloque para variables;
  - Versiones iniciales de C: permite dentro del bloque, pero como primera instrucción;
  - Java/C++: donde quiera, inclusive en el for.

```
void f() {
    int a = 1;
    a = a + 3;
    int b = 0;
    b = b + a;
}
```

Declaración de Constantes



- En C

```
const float pi = 3.14;
#define pi 3.14
```

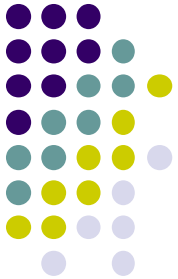
  - Posibilidad de alterar *consts* creó cultura de uso de #define, pero este no tiene ámbito ni tipo.

- En JAVA

```
final int const1  = 9;
static final int const2  = 39;
final int const3 = (int)(Math.random()*20);
static final const4 = (int)(Math.random()*20);

final int j;
Constructor () {
    j = 1;
}
```

Definiciones y Declaraciones de Tipos



- Definiciones Tipos en C

```
struct fecha {
   int d, m, a;
};
```

```
union angulo {
   int grados;
   float rad;
};
```

```
enum dia_util {
    lun, mar, mie,
   jue, vie
};
```

- Declaraciones Tipos en C

```
struct fecha;
typedef union angulo curvatura;
typedef struct fecha aniversario;
```

Definiciones y Declaraciones de Variables

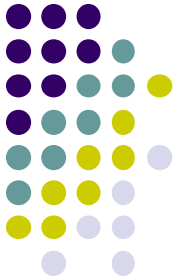
- Definiciones de Variables en C

```
int k;
union angulo ang;
struct fecha d;
int *p, i, j, k, v[10];
```

- Definiciones con Inicialización

```
int i = 0;
char coma = ',';
float f, g = 3.59;
int j, k, l = 0, m=23;
```

Definiciones y Declaraciones de Variables



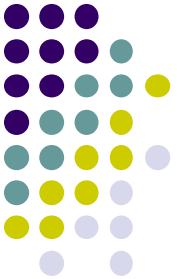
- Definiciones con Inicialización Dinámica

```
void f(int x) {
    int i;
    int j = 3;
    i = x + 2;
    int k = i * j * x;
}
```

- Definiciones con Inicialización en Variables Compuestas

```
int v[3] = { 1, 2, 3 };
```

Definiciones y Declaraciones de Variables



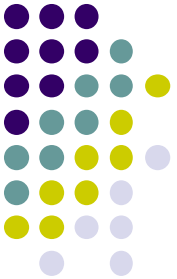
- Declaración de Variables en C

```
extern int a;
```

- Declaración de Variables y C++

```
int r = 10;
int &j = r;
j++;
```

Definiciones y Declaraciones de Subprogramas



- Definición de Subprogramas en C

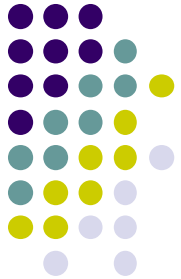
```
int suma (int a, int b) {
    return a + b;
}
```

- Declaración de Subprogramas en C

```
int incr (int);
void f(void) {
    int k = incr(10);
}
int incr (int x) {
    x++;
    return x;
}
```

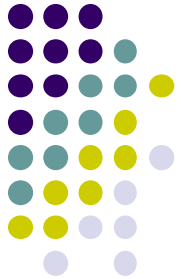
Un compilador que hiciese um pre-procesamiento podria dispensar la declaración de incr. C no lo hace, por eso la “forward reference”. En Java ella no es necesaria.

Composición de Definiciones



- Definiciones pueden ser compuestas a partir de otras definiciones o a partir de ellas mismas. Pueden ser compuestas o recursivas

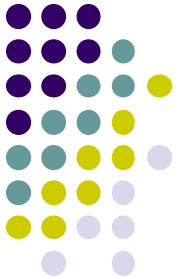
Definiciones Compuestas Secuenciales



- Utilizan definiciones establecidas anteriormente en el programa. Ejemplo en C:

```
struct funcionario {
    char nombre [30];
    int matricula;
    float salario;
};
struct empresa {
    funcionario listafunc [1000];
    int numfunc;
    float facturamiento;
};
int m = 3;
int n = m;
```

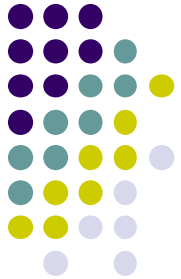
Definiciones Compuestas Secuenciales



- Definiciones Secuenciales en ML

```
val par = fn (n: int) => (n mod 2 = 0)
val negacion = fn (t: bool) => if t then false
  else true
val impar = negacion o par
val juego = if x < y then par else impar
```

Definiciones Compuestas Recursivas



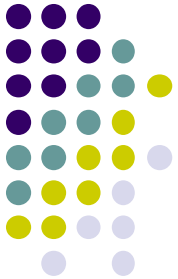
- Definición Recursiva de Función en C

```
float potencia (float x, int n) {
   if (n == 0) {
         return 1.0;
   } else if (n < 0) {
         return 1.0/ potencia (x, -n);
   } else {
         return x * potencia (x, n - 1);
   }
}
```

  - Tipo Recursivo en C

```
struct lista {
   int elemento;
   struct lista * siguiente;
};
```

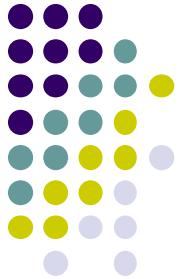
Definiciones Compuestas Recursivas



- Definiciones Mutuamente Recursivas en C

```
void segunda (int);
void primera (int n) {
   if (n < 0) return;
   segunda (n - 1);
}
void segunda (int n) {
   if (n < 0) return;
   primera (n - 1);
}
```

Definiciones Compuestas Recursivas



- Error en Definición de Función strcmp en C

```
int strcmp (char *p, char *q) {
     return !strcmp (p, q);
}
```

  - La idea era llamar el strcmp de la biblioteca, pero acaba llamando a el mismo recursivamente;
  - Java posee super.método() para exiplicitar el no uso de la recursividad;
- Explicitación de Recursividad en Función ML

```
val rec mdc = fn ( m:int, n: int) = >
   if m > n then mdc (m – n, n)
   else if m < n then mdc (m, n – m)
   else m
```