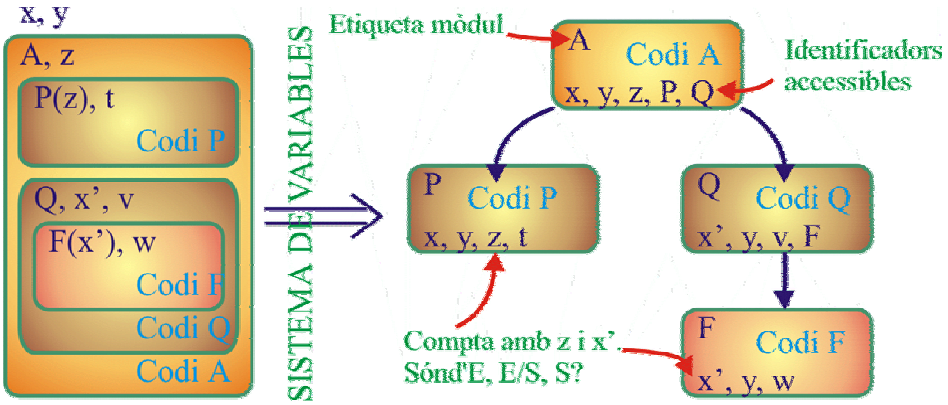
5.5 Variables i pas de paràmetres (I)

- Variables globals i variables locals.
  - Identificador global → Àmbit tot el programa. És a dir, es pot utilitzar en tots els mòduls. Estan declarades en nivells alts.
  - Identificador local al mòdul → Quan el seu ús queda restringit al mòdul. Són les declarades en cada mòdul. El seu àmbit és en el mòdul i/o en els mòduls 'fills'.
  - El terme identificador fa referència tant a noms de funcions/procediments com de variables.
- Els identificadors que es passen entre mòduls poden tenir diferent pas de paràmetres segons el llenguatge que s'utilitzi. S'ha de vigilar, sobretot, quan les variables globals poden ser modificades en el submòdul.
- El pas de paràmetres estableix la comunicació entre procediments i subprocediments



5.5 Variables i pas de paràmetres (II)

- Àmbit de les variables en el C.
  - En C hi ha variables locals (dintre de procediments) i globals (definides abans del *main()*).
  - S'ha d'anar en compta amb les variacions de les variables globals dintre dels procediments.
- Exemple: Segons on es defineix la variable A la sortida és diferent.

```
① ← int A=0;
void procediment(void);
void main()
{
      ② ← int A = 1;
      printf("1. A = %i\n", A);
      procediment();
      A += 3;
      printf("2. A = %i \n", A);
}
void procediment(void)
{
      ③ ← int A = 7;
      A += 1;
      printf("3. A = %i \n", A);
}
```

En ① → El resultat serà: 1. A = 0.
3. A = 1.
2. A = 4.
El procediment modifica A → A és variable global.

Només en ② → Error de compilació: procediment() no té definida A.

En ② i en ③ → 1. A = 1.
3. A = 8.
2. A = 4.
A és diferent variable en procediment() i en main().
En ocupar posicions de memòria diferents emmagatzemen diferents valors.

En ① i en ③ → 1. A = 1.
3. A = 8.
2. A = 3.
Passa com en el cas anterior.

5.5 Variables i pas de paràmetres (III)

- Tipus de variables en C
  - Constants → No canvien en tot el programa.
    És aconsellable l'ús de variables globals constant.
  - Automàtiques → Per defecte, qualsevol variable definida en una funció es considera automàtica.
    Totes les emprades fins ara han estat automàtiques.
  - Externes → Variable global llur definició apareix en una altra part del programa (o en un altre arxiu font del programa)
  - Estàtiques → Variable local, però que té vida mentre duri el programa.
    La variable estàtica declarada en una funció manté el seu valor entre diferents invocacions de la funció.
  - De registre → Indica al compilador que s'ha d'emmagatzemar en un registre (d'accés més ràpid que la memòria) de la CPU.

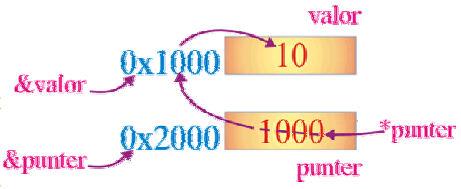
  - Exemple:
    - `const float PI = 3.141592;` Constant.
    - `(auto) int valor;` Variable automàtica.
    - `extern char caracter;` Variable externa.
    - `static int fusible;` Variable estàtica.
    - `register int comptador;` Variable registre.

5.5 Variables i pas de paràmetres (IV)

Emmagatzament de variables en memòria

- Fins ara s'ha vist que les variables es declaren i es defineixen (s'usen).
- Quan es declara no es fa res més que reservar l'espai de memòria que cal per al tipus de variable declarada.
- Exemples:
  - `int total;` → Reserva 2 bytes per a un enter al que s'accedeix mitjançant la constant total
  - `float volt;` → Reserva 4 bytes per a un real al que s'accedeix mitjançant la constant volt
- A partir d'ara, per passar paràmetres, sovint parlarem de la posició de memòria de la variable i del contingut de la variable. O, dit més normalment, d'adreces i apuntadors.
- Per parlar de l'adreça d'una variable emprarem l'operador d'adreça &. Per parlar del contingut de la variable emprarem l'operador d'indirecció *.

- Així, en l'exemple següent:
  - valor i punter són dos identificadors que referencien dues posicions de memòria.
  - Les adreces que referencien són: &valor=0x1000 i &punter=0x2000
  - Quan es fa valor=10 s'assigna el valor (contingut) 10 a la posició de memòria referenciada per la variable valor.
  - Si ara es defineix la variable punter com a punter=&valor, s'està dient que a la *posició de memòria referenciada per punter hi posarem l'adreça de la posició de memòria referenciada per valor*.
  - En conseqüència la variable punter apunta a la posició de memòria 0x1000 i *punter=10 (llegeixi's com el contingut de la variable referenciada per l'apuntador *punter.



5.6 Pas per paràmetre valor

- Els pas de paràmetres més coneguts són el pas per valor i el pas per referència. En C el pas per apuntador sol substituir el pas per referència.
- Característiques del pas de paràmetres per valor:
  - Emprat en llenguatges com C, Pascal, Algol, ...
  - Correspondència (entre procediment i subprocediment) posicional.
  - Els paràmetres són tractats com a variables locals.
  - Valors inicials obtinguts per còpia en la correspondència posicional.
  - No retorna informació (el valor de la variable no ha canviat) en el retorn del subprograma.

Pseudocodi

```
//Exemple: Elevar al cub
Real cub (Real n)
Inici
    Real x
    Escriure (“Entra numero: “)
    Llegir(x)
    Escriure(cub(x))
Fi
Real cub(Real n)
Inici
    n=n^3
    Tornar(n)
Fi cub
```

Diagrama de flux

Codificació C

```
//Exemple: Elevar al cub
float cub(float n);       //Declaració funció cub
void main()               //Funció principal
{
    float x;
    printf("Entra numero: ");
    scanf("%f", &x);
    printf(“%6.2f ^3 es %6.2f.\n", x, cub(x));
}
 float cub(float n)       //definició funció cub
{                         //n es el valor formal
    n = n*n*n;
    return(n);
}
```


5.7 Pas per paràmetre per referència (I)

- Característiques del pas per paràmetre referència:
  - La unitat que crida passa l'adreça del paràmetre variable. És una referència a la posició de memòria de la variable.
  - Així el subprogama pot modificar el contingut de la variable passada. En tornar al procediment que ha fet la crida, es manté el valor modificat (en el subprograma) de la variable.
  - Es poden considerar variables d'E/S.
  - És un pas directe en FORTRAN, COBOL, Pascal, ... És més delicat en C.
  - En pseudocodi s’indica posant la paraula var al davant de la declaració del tipus de paràmetre.
  - Exemple:

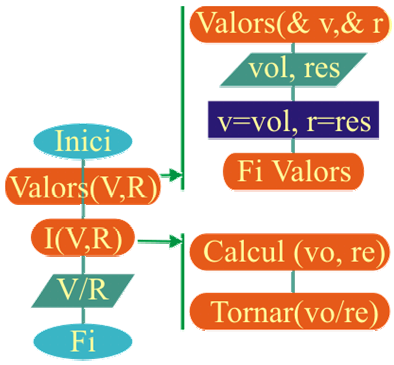
```
...
A ← 5
B ← 7
proc_1(A, 18, B*3+4)              //Crida a procediment
...

procediment proc_1 (var Enter x, y, z)   //x=5, y=18, z=25
Inici
    ...
    x = x+2
    ...
Fi proc_1                         //En retornar, A = 7, i B = 7
```

5.7 Pas per paràmetre per referència (II)

- En C/C++ s'ha de fer passant l'adreça de les variables o passant un apuntador. En aquest exemple es veu el pas per referència recollint les adreces de les variables &.

```
//Càlcul del corrent donats V i R
#include <stdio.h>
void Valors(float&, float&);          //Procediment amb pas d'adreces
float Calcul(float, float);           // Funció: retorna un valor float
void main()
{
      float V, R, I;                  //Variables locals
      Valors(V, R);                   //La crida passa les adreces
      I = Calcul(V, R);               //Càlcul d'I
      printf("El corrent de V/R = %2.1f/%2.1f = %2.1f\n",V, R, I);
}
void Valors(float& v, float& r)       // Es treballa amb les adreces de R i V
{
      float vol, res;                 v i r recullen les adreces passades de V
      printf(" Dona V i R: ");        i R a on s'hi posaran  els nous valors!
      scanf("%f%f", &vol, &res);
      v = vol;                        vol i v (res i r) són dues constants referenciant
      r = res;                        a la mateixa posició de memòria!
}
float Calcul(float vo, float re)      //Paràmetres volt i resist passats per valor
{
      return(vo/re);
}
```

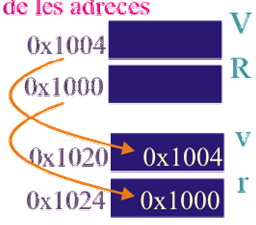


5.8 Pas de paràmetres amb apuntador

- L'ús d'apuntadors és la forma més usual de pas de paràmetres en C. En aquest cas es pot observar com es faria en l'exemple anterior.

```
//Càlcul del corrent donats V i R
#include <stdio.h>
void Valors(float*, float*);          //Procediment amb pas d'adreces
float Calcul(float, float);           // Funció: retorna un valor float
void main()
{
      float V, R, I;                  //Variables locals
      Valors(&V, &R);                 //La crida passa les adreces
      I = Calcul(V, R);               //Càlcul d'I
      printf("El corrent de V/R = %2.1f%2.1f = %2.1f\n",V, R, I);
}
void Valors(float* v, float* r)       // Es treballa amb les adreces de R i V
{                                     Tot i que es pot fer sense emprar les
      float vol, res;                 variables intermitges vol i res
      printf(" Dona V i R: ");            scanf("%f%f", &(*v), &(*r)) =
      scanf("%f%f", &vol, &res);          scanf("%f%f", v, r),
      *v = vol;                       per claretat, és millor emprarles.
      *r = res;
}
float Calcul(float vo, float re)      //Paràmetres volt i resist passats per valor
{
      return(vo/re);
}
```

1. Valors(&V, &R)
Pas de les adreces



2. *v=vol; *r=res;
Pas dels continguts de vol i res a V i R emprant els apuntadors v i r.