LLENGUATGES DE PROGRAMACIÓ

BLOC 2 – SEMINARI 1

FUNCIONS
APUNTADORS I
PAS PER REFERÈNCIA

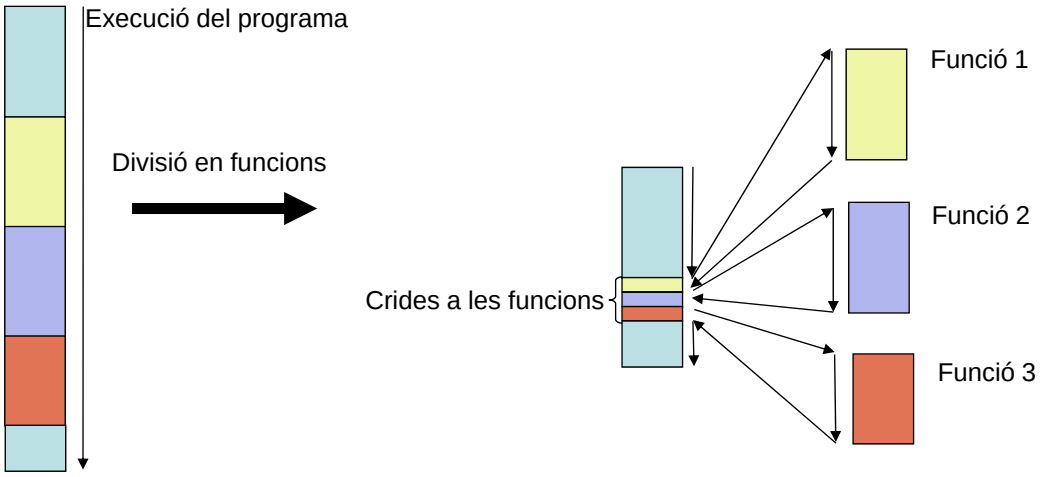
FUNCIONS

Funcions

Idea → Dividir els programes en un conjunt de mòduls més petits i més fàcils de manipular (programar, depurar, reutilitzar, etc...)

Avantatges d'utilitzar funcions:

- Simplificació del codi
- Estructuració i claredat del codi
- Facilitat per la localització d'errors



Funcions

PSEUDOCODI

Funcions → Subalgorismes que retornen algun valor

Procediments → Subalgorismes que no retornen cap valor

LLENGUATGE C

Declaració i definició de funcions

```
<tipus_retorn>   <nom_funció> (<tipus>  <par_1> ,   ... , <tipus>  <par_N>)
{
        <Declaració_variables_locals>;
        <Instruccions>;
        return (<valor_retorn>);
}
```

Crida a la funció

```
<nom_funció>   (<valor_par_1>,  ... , <valor_par_N>);
```

Funcions

PSEUDOCODI	**LLENGUATGE C**
Funcions	Funcions amb: tipus de retorn → int, float, info_alumne **return** al final del a funció
Procediments	Funcions amb: tipus de retorn **void** **sense return** al final del a funció

Funcions

Exemples

```
<tipus_retorn>   <nom_funció> (<tipus>  <par_1> ,   ... , <tipus>  <par_N>)
{
        <Declaració_variables_locals>;
        <Instruccions>;
        return (<valor_retorn>);
}
```

```
float  Calcular_nota (float  nota1, float  nota2)
{
    float nota;

    nota=nota1*0.7+nota2*0.3;

    return nota;
}
```

Funcions

Exemples

```
void   <nom_funció> (<tipus>  <par_1> ,   ... , <tipus>  <par_N>)
{
        <Declaració_variables_locals>;
        <Instruccions>;
}
```

```
void  Imprimir_dades (info_Alumne  alumne)
{
    char nota;

    printf("Dni: %s",alumne.dni);
    printf("Nom: %s",alumne.nom);
    printf("Cognoms: %s",alumne.cognoms);
    printf("Grup: %d",alumne.grup);
    if (alumne.nota_final<5.0)
        nota='S';
    else
        nota='A';
    printf("Nota: %c",nota);
}
```

Recordatori seminari 3:

```
typedef struct Alumne
{
char dni[9];
char nom[25];
char cognoms[50];
int grup;
float nota_examen;
float nota_parcial;
float nota_final;
} info_Alumne;
```

Funcions

```
#include <stdio.h>

typedef struct Alumne
{
char dni[9];
char nom[25];
char cognoms[50];
int grup;
float nota_examen;
float nota_parcial;
float nota_final;
} info_Alumne;


void  Imprimir_dades (info_Alumne  alumne)
{
    char nota;

    printf("Dni: %s",alumne.dni);
    printf("Nom: %s",alumne.nom);
    printf("Cognoms: %s",alumne.cognoms);
    printf("Grup: %d",alumne.grup);
    if (alumne.nota_final<5.0)
        nota='S';
    else
        nota='A';
    printf("Nota: %c",nota);
}
```

Funcions

```
float  Calcular_nota (float  nota1, float  nota2)
{
    float nota;

    nota=nota1*0.7+nota2*0.3;
    return nota;
}


void  main ()
{
    info_Alumne alumne1, alumne2;
    float nota1, nota2;

    ……
    ……
    Imprimir_dades(alumne1);
    ……
    ……
    Imprimir_dades(alumne2);
    ……
    ……
    nota1=Calcular_nota(alumne1.nota_examen,alumne1.nota_parcial);
    nota2=Calcular_nota(alumne2.nota_examen,alumne2.nota_parcial);
    ……
    ……
}
```

Funcions

Variables

Variables globals: Definides fora de qualsevol bloc de funció (inclòs el main()).
Es poden utilitzar a qualsevol funció del programa.

Variables locals: Definides dins del bloc d'alguna funció.
Només es poden utilitzar dins de la funció on estan definides.

Important!!!

En general, les variables globals no s'han d'utilitzar mai.

Utilitzarem variables locals i les passarem a les funcions com a paràmetres.

```
/* Factorial.c     Programa que calcula el factorial d'un número */
#include <stdio.h>

long n;                          /* Variable global */

long factorial ()                /* Declaració i definició de la funció */
{
    long fact;                   /* Variables local de la funció factorial */

    fact = 1;
    for (; n>1; n--) fact = fact*n;     /* Utilitza la variable global n */
    return (fact);                      /* Retorn del resultat de la funció */
}

void main ()
{
    long fact;                   /* Variable local de la funció main */

    printf ("Introdueix un número: ");
    scanf ("%d", &n);                           /* Utilitza la variable global n */
    fact = factorial ();                        /* Crida a la funció */
    printf ("El factorial del número “%d” és: %ld\n", n, fact);
}
```

APUNTADORS

Apuntadors

Fins ara hem vist diferents tipus de dades:

Simples:

- `int`
- `char`
- `float`
- `double`

Compostos:

```
struct <Nom_Struct>{
        <tipus1> <camp1>;
                ⋮
        <tipusN> <campN>;
};
```

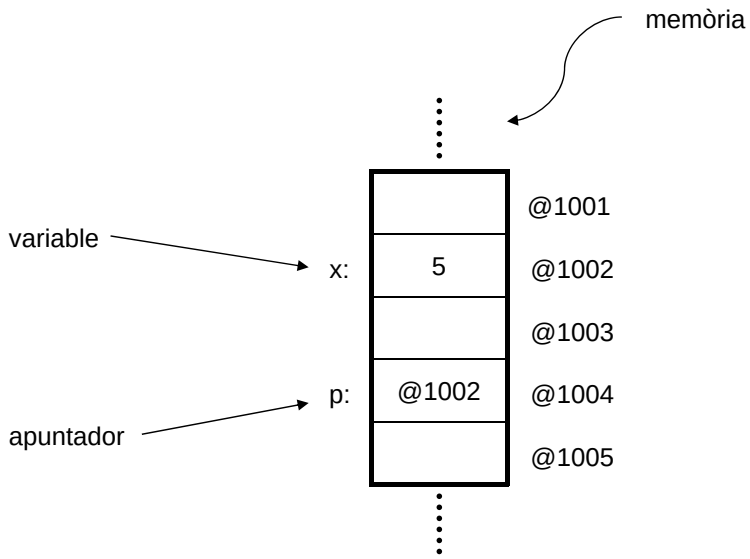
Avui veiem un de nou:

Els APUNTADORS

Apuntadors

Apuntador → Variable que emmagatzema adreces de memòria.

Quan un apuntador emmagatzema l'adreça d'una variable, diem que l'apuntador "apunta" a aquella variable.



Apuntadors

DECLARACIÓ D' APUNTADORS

```
<tipus_del_punter> *<Nom_del_Punter>;
```

Es declara igual que qualsevol variable, però hi afegim l'asterisc "*"

EXEMPLES

```
int *p;
float *PunterReal;
char *P_Ch;
```

L'apuntador `PunterReal`, emmagatzemarà l'adreça d'una variable de tipus `float`.

Apuntadors

OPERADORS

&:

L'operador & (ampersand) → Retorna l'adreça de memòria que ocupa aquesta variable.

```
<Nom_Apuntador> = &<Nom_Variable>;
```

*****:

L'operador * → Retona el contingut de l'adreça de memòria que emmagatzema.

```
<Nom_Variable> = *<Nom_Apuntador>;
```

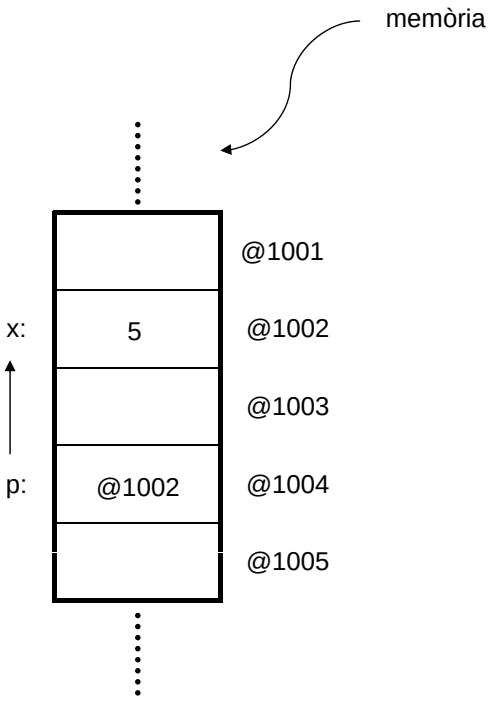
Apuntadors

EXEMPLE

```
int x;
int *p;

p = &x;
x = 5;
```

p apunta a x

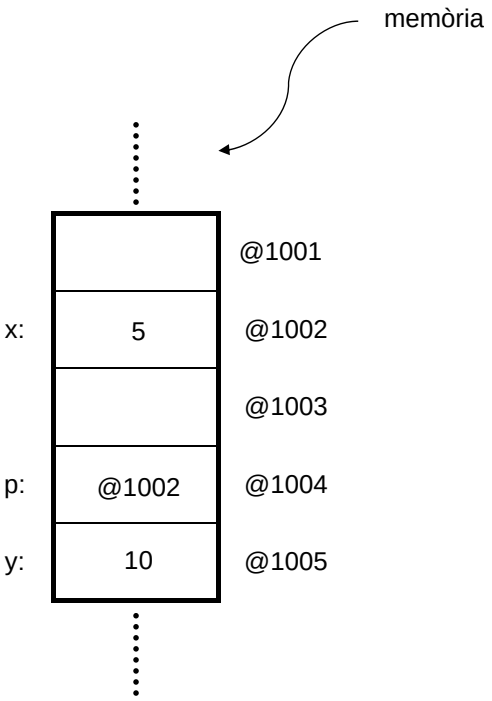
*p és una forma equivalent d'accedir al valor que hi ha dins x, és a dir, 5.



Apuntadors

EXEMPLE

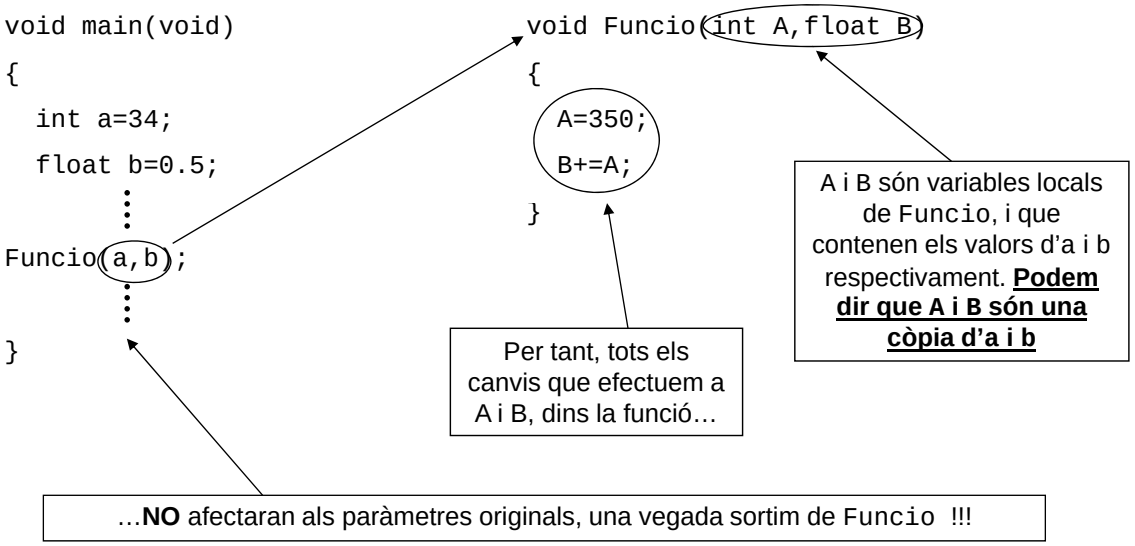
```
int x,y;
int *p;

p = &x;
x = 10;
y = *p;
*p = 5;
```



PAS DE PARÀMETRES PER REFERÈNCIA

Pas de paràmetres per referència

El pas de paràmetres a funcions que havíem vist **anteriorment** era un **pas de paràmetres per valor**.



Després veurem el **pas de paràmetres per referència.**

Pas de paràmetres per referència

LLENGUATGE C

Pas de paràmetres per valor

- A la definició de la funció posem el tipus del paràmetre i el nom.
- Fem la crida a la funció amb les variables que volem passar

Exemple

```
int SumaEnters(int num1, int num2)
{
     return (num1+num2);
}

void main()
{
     int a,b,res;
     ......
     res=SumaEnters(a,b);
}
```

Pas de paràmetres per referència

LLENGUATGE C

Pas de paràmetres per referència

- A la definició de la funció posem el paràmetre com un apuntador
- Fem la crida a la funció amb les adreces de les variables que volem passar
- Si fem canvis al paràmetre per referència també canvia la variable que li hem passat a la funció.

Exemple

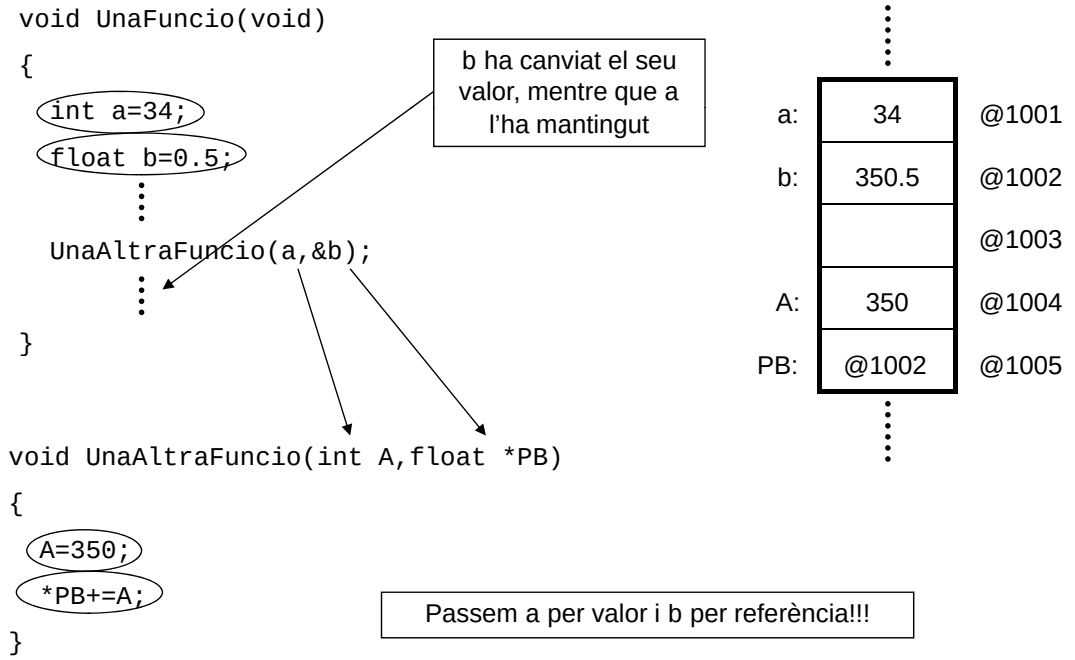
```
void SumaEnters(int num1, int num2, int *resultat)
{
     *resultat=num1+num2;
     num1=0;
}

void main()
{
     int a=7,b=5,res=0;
     ...
     SumaEnters(a,b,&res);
}
```

valors: a=7, b=5, res=0

valors: a=7, b=5, res=12

Pas de paràmetres per referència

EXEMPLE




```
void UnaAltraFuncio(int A,float *PB)
{
 A=350;
 *PB+=A;
}
```

Passem a per valor i b per referència!!!