

# U n i t a t 9

## Funcions

Introducció.

9.1 Definició de funcions.

9.2 Declaració de funcions.

9.3 Crida de funcions.

9.3.1 Crida d'arguments per valor.

9.3.2 Crida d'arguments per referència.

9.3.3 Relació entre funcions i vectors/taules.

9.4 Variables globals i locals.

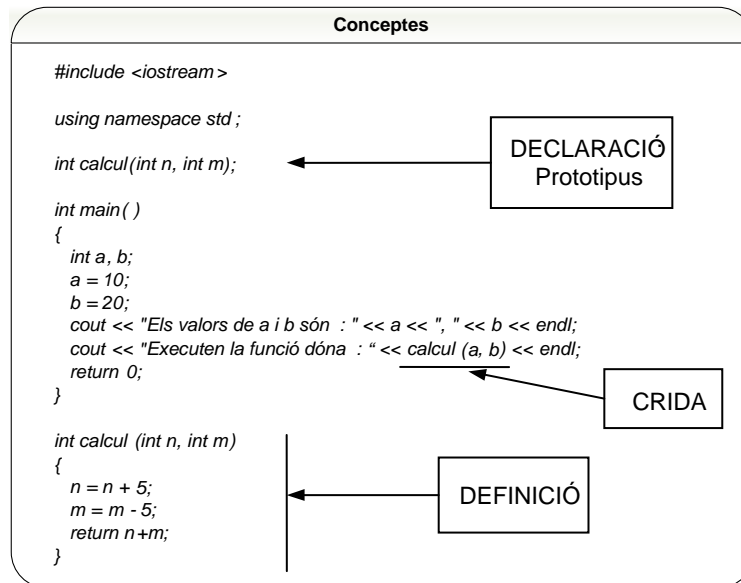
### Introducció.

Podem definir que una funció és un miniprograma dins d'un altre programa. Proporcionen un medi de dividir un projecte gran en mòduls petits més manejables.

Una funció permet agrupar un conjunt de sentències que desenvolupen una tasca determinada. Tota funció pot ésser invocada quan sigui necessària, les vegades que faci falta.

Tots els programes C++ consten, com a mínim, d'una funció, *main* ( ) que és per on comença l'execució del programa.

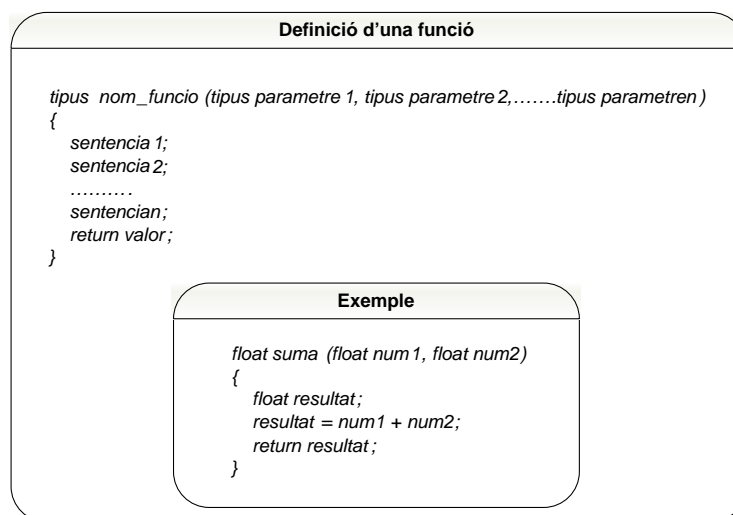
Cada funció realitza una determinada tasca i quan s'executa la sentència *return* o finalitza el codi de la funció, es retorna al punt on va ser cridada pel programa o funció principal.



## 9.1 Definició de funcions.

El compilador de C++ requereix que tota funció sigui definida dins del programa, ja sigui abans o després d'invocar-la. La definició consisteix en escriure totes les sentències que componen la funció.

Una funció es defineix de la següent forma:



Les parts de la funció declarada anteriorment són:

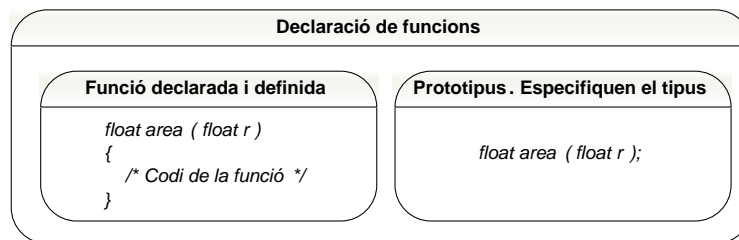
- *tipus*. És el tipus de valor de retorna la funció. Si és del tipus *void* la funció no retornarà cap valor.
- *nom\_funcio*. És l'identificador o nom de la funció.
- *tipus parametre1, tipus parametre2,....* Llista de paràmetres que passem a la funció.
- *sentencia1; sentencia2;...* Es el cos de la funció, on es faran totes les operacions necessàries.
- *return valor;*. Valor que retornarà la funció.

## 9.2 Declaració de funcions.

El compilador de C++ requereix que tota funció sigui declarada abans d'èsser invocada en el programa. La declaració d'una funció pot realitzar-se de dues formes:

- Al mateix temps de la definició.
- Mitjançant un prototipus.

La declaració de funcions permet al compilador comprovar la coherència de tipus quan una funció és invocada.



Els prototipus es col·loquen normalment al principi d'un programa, abans de la definició de la funció *main* ( ).

## 9.3 Crida de funcions.

La crida d'una funció es realitza escrivint el nom o identificador de la funció seguit dels corresponents parèntesis, entre els quals s'inclou els arguments que la funció precisa. El format genèric d'una crida a funció és la següent:

*variable* = *nom\_funcio* ( *parametres* );

*variable*. És la variable on s'emmagatzema el valor de retorn.

*nom\_funcio*. És el nom o identificador de la funció.

*parametres*. Representa el conjunt d'expressions separades per comes que s'assignen en el mateix ordre als paràmetres formals, si la funció els necessita.

Quan una funció necessita rebre dades o arguments per operar, aquests es poden passar a la funció de dos maneres:

- Arguments per valor.
- Arguments per referència.

### 9.3.1 Entrega d'arguments per valor.

Els arguments es copien en el corresponents paràmetres formals de la funció, qualsevol canvi que introdueixi la funció en aquests valors no afecta a les variables utilitzades en la crida, ja que es canvien els valors emmagatzemats en variables locals d'aquesta funció.

## Entrega d'arguments per valor

```
#include <iostream>

using namespace std;

int calcul(int n, int m);

int main()
{
    int a, b;
    a = 10;
    b = 20;
    cout << "Els valors de a i b són : " << a << ", " << b << endl;
    cout << "Executen la funció dóna : " << calcul(a, b) << endl;
    return 0;
}

int calcul(int n, int m)
{
    n = n + 5;
    m = m - 5;
    return n+m;
}
```

En l'exemple anterior, primer assignem a les variables els valors  $a = 10$  i  $b = 20$ , després cridem a la funció `calcul()` amb les variables  $a$  i  $b$  com paràmetres. Dins la funció `calcul()` els paràmetres es diuen  $n$  i  $m$ , i canviem els seus valors, però al retornar a `main()`,  $a$  i  $b$  conserven els seus valors originals.

En realitat, no entreguem les variables  $a$  i  $b$ , sinó que copiem els seus valors a les variables  $n$  i  $m$ .

### 9.3.2 Entrega d'arguments per referència.

A la funció s'entreguen les direccions dels paràmetres. Això li permet modificar els valors d'aquestes variables que s'entreguen.

Els paràmetres per referència, declarats com punters “\*” reben la direcció dels arguments que s'entreguen; i a aquests els precedeix l'operador “&”, excepte els vectors.

## Entrega d'arguments per referència

```
#include <iostream>

using namespace std;

void intercanvi(int *a, int *b);

int main()
{
    int x, y;
    x = 5;
    y = 10;
    cout << "Els valors de x i y són : " << x << ", " << y << endl;
    intercanvi(&x, &y);
    cout << "Executen la funció dóna x i y són : ";
    cout << x << ", " << y << endl;
    return 0;
}

void intercanvi(int *a, int *b)
{
    int aux;
    aux = *a;
    *a = *b;
    *b = aux;
}
```

En l'exemple anterior, primer assignem a les variables els valors  $x = 5$  i  $y = 10$ , després cridem a la funció *intercanvil* ( ) amb les variables  $x$  i  $y$  com paràmetres per referència. Es a dir, entreguem la direcció on es troben les variables. Dins la funció *intercanvi* ( ) els paràmetres són dos punters anomenats  $a$  i  $b$ , que apunten a les direccions  $x$  i  $y$ .

En executar la funció, els valors de les variables  $x$  i  $y$ , s'han intercanviat.

### 9.3.3 Relació entre funcions i vectors/taules.

Els vectors i les taules, normalment s'entreguen i retornen per referència. En les taules multidimensionals cal especificar totes les dimensions excepte la primera, que és opcional.

Hi ha dues maneres d'entregar un vector:

- a) Incloent un punter en els arguments d'entrada.

*tipus nom\_funcio (tipus \*vector, parametre2.....);*

- b) Indicant amb claudàtors que l'argument d'entrada és un vector.

*tipus nom\_funcio (tipus vector[ ], parametre2.....);*

S'entrega a la funció la direcció del primer element del vector.

Hi ha dues maneres d'entregar una taula:

- a) Incloent una taula amb el nº de columnes.

*tipus nom\_funcio (tipus taula [ ][10], parametre2.....);*

- b) Indicant un punter a un vector del tamany del nº de columnes.

*tipus nom\_funcio (tipus (\*taula)[10 ], parametre2.....);*

S'entrega a la funció la direcció del primer element de la taula.

#### Relació funcions i vectors

```
#include <iostream.h>

void per_dos (int vector [ ]);

void main (void)
{
    int sumands [ 3 ] = { 0 };
    cout<<"Introdueixi els 3 números que vulgui doblar:"<<endl;
    cin>>sumands[0]>>sumands[1]>>sumands[2];
    per_dos(sumands);
    cout<<"Els nous valors són "<<endl<<sumands[0]<<endl;
    cout<<sumands[1]<<endl<<sumands[2]<<endl;
}

void per_dos (int vector [ ])
{
    int i;

    for (i=0; i<3; i++)
    {
        vector [ i ] = 2 * vector[ i ];
    }
}
```

#### Relació funcions i taules

```
#include <iostream.h>

void per_dos (int taula [][ 3 ]);

void main (void)
{
    int elements [ 2 ][ 2 ] = { 0 };
    cout<<"Introdueixi els números que vulgui doblar :"<<endl;
    cin>>elements[ 0 ][ 0 ]>>elements[ 0 ][ 1 ];
    cin>>elements[ 1 ][ 0 ]>>elements[ 1 ][ 1 ];
    per_dos(elements);
    cout<<"Els nous valors són "<<endl;
    cout<<elements[ 0 ][ 0 ]<<" "<<elements[ 0 ][ 1 ]<<endl;
    cout<<elements[ 1 ][ 0 ]<<" "<<elements[ 1 ][ 1 ]<<endl;
}

void per_dos (int taula [][ 3 ])
{
    int i , j;

    for (i=0; i<2; i++)
    {
        for (j=0; j<2; j++)
        {
            taula[ i ][ j ] = 2 * taula[ i ][ j ];
        }
    }
}
```

## 9.4 Variables globals i locals.

Les variables utilitzades en una funció poden ser global o local.

Les variables globals són definides fora de la funció, i són accessibles a totes les sentències del programa. El seu àmbit és tot el programa.

Les variables locals són aquelles que es declaren i utilitzen només dins la funció o d'un bloc. Té visibilitat només en el cos de la funció.

```

Variables globals i locals

#include <iostream.h>

int q;           // Variable global

void main(void)
{
    int a;       // Variable local en main

    a = 10;
    q = 0;

    {
        int b;   // Variable local en aquest bloc

        b = 50;
        a = 20;
        q = 1;
    }
}

```

### Exercicis amb funcions.

- Realitzar un programa amb funcions que calculi l'àrea d'un triangle rectangle. La funció principal demanarà a l'usuari les dades de les dimensions de l'alçada i la base. Recordar que l'àrea d'un triangle rectangle és:  $\text{àrea} = (\text{base} * \text{alçada}) / 2$ . [Solució: *9\_1\_calcul\_area\_triangle*].
- Realitzar un programa amb funcions que calculi l'àrea de diferents figures. Un triangle rectangle, un triangle isòsceles, un cercle i un quadrat. La funció principal mostrarà un menú principal per les diferents opcions de càlcul. Segons l'opció s'executarà la funció corresponent. El programa sempre estarà en funcionament fins que l'usuari triï l'opció de sortir del menú principal. Recordar les àrees de les figures:
  - Triangle rectangle:  $\text{àrea} = (\text{base} * \text{alçada}) / 2$
  - Triangle isòsceles:  $\text{àrea} = (\text{costat1} * \text{costat2}) / 2$
  - Cercle:  $\text{àrea} = (3.14 * (\text{radi})^2)$
  - Quadrat:  $\text{àrea} = (\text{costat})^2$

[Solució: *9\_2\_calcul\_areas*].

3. Realitzar un programa amb funcions que rebi com argument de l'usuari una lletra i una cadena de caràcters i mostri per pantalla el número de vegades que la lletra apareix en la cadena. *[Solució: 9\_3\_lletra\_en\_cadena]*.
4. Realitzar un programa de conversió d'euros a pessetes. Demanarem a l'usuari que ens doni 5 quantitats de valors en euros i els emmagatzemarem en una variable. Dissenyarem una funció que modifiqui aquesta variable per convertir aquestes quantitats d'euros a pessetes. Recordar que  $1\text{€}=166.386$  pessetes. *[Solució: 9\_4\_convertidor\_pessetes]*.
5. Realitzar un programa de càlcul científic amb funcions. L'usuari podrà calcular d'un número, el seu quadrat ( $x^2$ ), el seu cub ( $x^3$ ) i qualsevol potència ( $x^y$ ). El programa constarà de les següents característiques:
  - a. Demanar a l'usuari el número per fer el càlcul.
  - b. Mostrar el menú principal amb les opcions: Canviar el número, càlcul del quadrat, càlcul del cub, càlcul potències, sortir.
  - c. El menú sempre ha d'estar visible, fins que l'usuari polsi l'opció de sortir.

*[Solució: 9\_5\_calcul\_cientific]*