Funcions II

Objectius:
- **Detectar les entrades i sortides d'un programa i una funció.**
- **Transformar un programa en una funció.**
- **Implementar una funció directament.**
- **Construir un programa que utilitzi funcions com a subprogrames.**

(Objectius: 2.2.4, 2.2.5, 2.2.6 i 2.2.7)

1. De programa a funció

Per poder crear programes grans (i més complicats) el que es fa és combinar l'execució de "subprogrames". Un **subprograma**, de fet, és un petit programa que utilitzem des d'un altre. Veiem un exemple, el programa següent calcula el valor absolut d'un nombre:

```
#include <iostream>
using namespace std;

int main()
{
  float x;
 cin >> x;
  if ( x < 0.0 ) {
    x = -x;
  }
 cout << x << endl;
}
```

L'important del programa és que *rep un valor d'entrada i mostra per pantalla un valor de sortida*. Això és el que caracteritza el càlcul del valor absolut: "entrem un número y es retorna un resultat". L'entrada està marcada en **vermell** (el cin) i la sortida en **blau** (el cout). Per poder utilitzar aquest programa dins d'un altre, primer l'hem de *transformar* en una funció. La transformació és la següent:

```
float valor_absolut(float a)
{
  if ( a < 0.0 ) {
    a = -a;
  }
  return a;
}
```

La nova funció (**`valor_absolut`**) també té una entrada (el valor de 'x' original) i una sortida (el que hi ha al costat de '**`return`**'), que estan marcats també en vermell i blau, tal com en el programa original. En la funció '**`valor_absolut`**' (el nom descriu de què es tracta) l'entrada la obtenim amb la variable '**`a`**' (que és un *paràmetre*) i la sortida la donem amb el '**`return`**' (que té associat el tipus **`float`**, en blau). El codi

que hi ha a dins de la funció és el mateix que en el programa original, simplement manipula la variable 'a' en comptes de 'x'. Ara la idea és fer servir aquesta funció en un altre programa, que calcularà la fórmula:

$$f(x,y) = \frac{|x|}{|y|}$$

Per fer-ho, escriu el següent codi:

```
#include <iostream>
using namespace std;

int main()
{
  float x, y, res;
  cin >> x >> y;
  res = valor_absolut(x) / valor_absolut(y);
  cout << res << endl;
}
```

Ara inserta en la part marcada en **taronja** la funció **`valor_absolut`**, tal com està a dalt. Tenim, doncs, un programa (el **`main`**), que utilitza un subprograma (el **`valor_absolut`**) per fer el càlcul. Com que el valor absolut es necessita dues vegades, fer una funció que el calcula té l'avantatge de que no s'ha de repetir el càlcul del valor absolut 2 cops, s'implementa només 1 cop i es fa servir tants cops com faci falta.

2. Potències

Fixa't en el següent programa:

```
#include <iostream>
using namespace std;

int main()
{
  float x, y=1;
  int n;
  cin >> x >> n;
  while ( n >= 1 ) {
    y = y * x;

  }
  cout << y << endl;
}
```

Determina primer quin càlcul realitza (i *omple la instrucció que falta* a la zona **taronja**), i mira quants valors rep a l'entrada i quants resultats dóna. Ara transforma el programa en una funció "**`pot`**", tal com hem fet a l'apartat anterior. Utilitza llavors la funció "**`pot`**" per fer un programa que, donat un valor de 'x', calculi la fórmula:

$$f(x) = x^5 - 2x^4 + 0.5x^3 + 5$$

4. Funció `num_xifres`

Transforma el programa *Numero de xifres* que calcula el número de xifres d'un enter (exercici 8 del laboratori de composició iterativa) en una funció "**`num_xifres`**".

5. Obtenir una xifra

Quan volem "declarar" una funció (dir quin nom té, quins paràmetres i quin tipus retorna), escrivim només la seva capçalera:

```
int xifra_N(int x, int n);
```

Amb això n'hi ha prou per saber quins paràmetres rep la funció, quin nom té (com la cridarem) i quin tipus retorna (per obtenir el resultat). Respectant la declaració (o capçalera) de la funció **`xifra_N`**, implementa-la, sabent que el càlcul que realitza és *extreure una xifra d'un número enter*. El resultat no és un **`char`**, sinó un **`int`**. Les xifres de l'enter es numeren des de 1 (la de la més a la dreta) i van creixent segons les desenes. La següent taula mostra el resultat de cridar la funció per a diversos valors d'entrada:

```
xifra_N( 123, 1)    3
xifra_N(7054, 2)    5
xifra_N(3892, 3)    8
xifra_N( 104, 7)    0
xifra_N(  45, 0)    0
```

Fixa't que quan la funció no pot obtenir la xifra que se li demana, retorna **0** (els dos últims cassos).

6. Super-programa

Les dues funcions anteriors ens permeten fer, doncs, el següent programa:

```
#include <iostream>

using namespace std;


// ...


int main()
```

```
{

  int x, n, i;

  bool iguals = true;

  cin >> x;

  n = num_xifres( x );

  i = 1;

  while ( i < n && iguals )
  {

    if ( xifra_N( x, i ) != xifra_N( x, n - i +1 ) )
      iguals = false;
    else
      i++;
  }
  if (iguals) cout << "Si" << endl;
  else cout << "No" << endl;
}
```

Completa el programa inserint les funcions que has fet anteriorment i comprova que funcioni correctament.

Quin càlcul realitza??

7. Declaració de funcions

Volem "declarar" les funcions (donant el nom, paràmetres i tipus de retorn) que s'ajusten a les següents especificacions:
a) Donat un valor numèric sencer determinar si és primer o no.
b) Donats 2 enters retornar el mínim comú múltiple.
c) Donat un enter en base decimal i una base, convertir el numèric decimal a la base donada.
d) Donada una paraula i una lletra averiguar quantes vegades apareix la lletra dins la paraula.
e) Donats 3 enters retornar el màxim o el mínim segons s'indiqui través d'un dels paràmetres.
f) Donades 2 paraules concateneu-les.