Expressions

Objectius:
- **Entendre la divisió entera.**
- **Escriure expressions numèriques.**
- **Entendre què és el codi ASCII.**
- **Indicar si una expressió és errònia i perquè.**
- **Avaluar i escriure expressions Booleanes.**
- **Aplicar el teorema de De Morgan.**

(Objectius: 1.4)

1. La divisió entera

Obre un nou fitxer "**`div.cpp`**" i entra-hi el següent programa:

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

int main()
{
  int a, b;
  cout << "Entra 2 enters: ";
  cin >> a >> b;

  cout << La seva divisió és: " << a / b << endl;
}
```

Ara executa el programa mirant el resultat que dóna per a les següents parelles de valors:

```
10 5
10 3
2 5
```

Què hauries de fer perquè el resultat donés amb decimals??

2. Una funció senzilla

Implementa un programa (**`func.cpp`**) que calculi la funció

$$f(x) = \frac{1+x^2}{1-x^2}$$

és a dir, que et demani el valor de x i mostri el valor de la funció per pantalla.

3. Codi ASCII

Fes un programa (**`ascii1.cpp`**) que et demani un caràcter i després escrigui el seu codi ASCII per pantalla. Comprova que si poses **`'a'`** surti 97, si poses **`'A'`** surti 65 i si poses **`'0'`** surti 48.

4. Majúscules

Fes un programa (**`majusc.cpp`**) que et demani un caràcter i mostri per pantalla el mateix caràcter en majúscules. Mira primer, quina relació hi ha entre els caràcters en minúscules i majúscules perquè aquest programa fa un càlcul més o menys senzill. Per fer-ho pots suposar que el caràcter que et donen és una lletra minúscula. Fes unes quantes proves.

Què passa si li entres al programa una lletra que no sigui minúscula? Pots explicar-ho?

5. Expressions Booleanes

Les expressions que tenen com a tipus resultant un Booleà són molt importants en programació perquè, com veurem més endavant, permeten controlar quines instruccions del programa s'executaran. Obre un fitxer nou de programa anomenat "**`interv.cpp`**" i entra el següent programa:

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

int main()
{
  float x;
  cout << "Entra un nombre real: ";
  cin >> x;
  cout << (x > 0.0 && x < 10.0) << endl;
}
```

I executa'l vàries vegades posant els valors: -20, -1.7, 5.5, 15.2, 21.3, i 100. Què surt per pantalla en cada cas?
Vist això, què creus que surt per pantalla si en C++ escrius la següent instrucció?

```
cout << true << false << true << endl;
```

Modifica ara el programa per tal que surti un **`1`** quan el valor de **`x`** sigui més petit que 0.0 o més gran que 5.0. Comprova com el programa dona el valor correcte en uns quants cassos.

6. Comparació a tres bandes

Quan programem, a vegades podem cometre errors que no són gens evidents. Per exemple, quin error veus en el programa següent? Salva'l com "**`entremig.cpp`**":

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

int main()
{
```

```
  float x, y, z;
  bool entremig;
  cin >> x >> y >> z;
  entremig = x < y < z;
  cout << entremig << endl;
}
```

Primer de tot: què et sembla que fa?
Dóna errors de compilació?
Si suposem que el programa escriu `1` quan el valor de `y` està comprès entre el de `x` i el de `z`, i escriu `0` si no és així, et sembla que el programa funciona bé? Fes unes quantes proves per veure què pot passar...

Substitueix, ara, la línia "`entremig = x < y < z`" per "`entremig = x < y && y < z`". Mira si ara el programa funciona bé. Pots explicar, doncs, quin és l'error que hi havia en el programa? (pels atrevits: podries dir quins valors de `x`, `y` i `z` donaven `0` i `1` en el programa erroni??)

7. De Morgan

Quan neguem una expressió com

```
a < 'Y'
```

apliquem la negació a tota l'expressió posant parèntesis a fora, així

```
!(a < 'Y')
```

però la negació també es pot aplicar a l'operador de dintre.
Si l'expressió inicial vol dir "`a` és menor que `'Y'`", llavors la negació seria "`a` és major o igual que`'Y'`", o sigui

```
a >= 'Y'
```

Fixa't, doncs, en l'expressió següent (suposant que `c` és un `char`), que permet mirar si la variable `c` és una vocal

```
c == 'a' && c == 'e' && c == 'i' && c == 'o' && c == 'u'
```

Així, mentalment, què et sembla, funciona correctament?
Fes un programa que la utilitzi per comprovar-ho i corregeix l'expressió cas que es confirmés que està malament.
Ara, converteix l'expressió amb el teorema de De Morgan, i comprova que funciona correctament tornant a provar el programa.
Sabent això, com traduiries la següent expressió (si suposem que `a` és un `int`)?

```
!(a < 10 || a > 20)
```

8. Encerta la combinació

Entra el programa següent (`xngu.cpp`):

```
#include <iostream>
```

```
using namespace std;

int main()
{
  char c;
  int i, j;
  bool b;
  cin >> c >> i >> j >> b;
  bool xngu = ( int(c) >= 48 && c <= '9' ) ||
              ( ( i - j ) == 5 && i < 0 && b ) ||
              ( ( i < -10 && j < -10 || i > 10 && j > 10 ) && !b );
  cout << xngu << endl;
}
```

Sota quines condicions el programa escriu '1'?? Escriu la solució com un comentari al codi del programa per entregar-la.
Fes totes les proves necessàries i modifica l'expressió si ho necessites per entendre cada part per separat.