Pràctica 1. Programació orientada a objectes amb C++. Herència i polimorfisme

1. Introducció

***Objectius**:* Familiaritzar-se amb el llenguatge C++ i amb NetBeans 8.2. Entendre els conceptes d'herència i polimorfisme i la seva realització en C++.

Temes de teoria relacionats amb la pràctica: Prob1. C++ i TADs

Les pràctiques es faran amb l'IDE NetBeans 8.2 i la versió 11 de C++.

2. Enunciat

Exercici 1. Crea un programa **main.cpp** que demani el nom a l'usuari, el saludi pel seu nom i li demani una opció de les mostrades a un menú. Aquest menú l'anirem ampliant durant la pràctica, de moment, només tindrà dos opcions; "*Sortir*" o "*Benvinguda*". Si escollim *Sortir* el programa finalitzarà i si escollim *Benvinguda*, el programa ens donarà la benvinguda personalitzada a l'assignatura d'estructura de dades i ens tornarà a mostrar el menú per tornar a escollir una opció.

Per exemple, el programa hauria de mostrar el següent:

```
Hola, com et dius?
Alicia
Hola, Alicia, que vols fer?

1. Sortir
2. Benvinguda
2
Benvingut/da a l'assignatura d'estructura de dades Alicia
```

Per fer-ho cal:

a. Al main, definir les variables per guardar el nom, l'opció escollida (integer) i un array de strings per emmagatzemar les diferents opcions del nostre programa. Aquest últim ara serà: `string arr_options[] = {"Sortir", "Benvinguda"};`
b. Preguntar el nom i agafar-lo fent servir les comandes `cout` i `cin` de C++. Caldrà fer l'include de la llibreria *iostream* i el namespace corresponent.
c. Definir una estructura `do{...}while(option != 0),` que saluda a l'usuari, mostra les opcions del menú amb un bucle `for` i agafa l'opció l'escollida per l'usuari, comprovant que aquesta sigui una opció vàlida.

Un cop creat aquest programa executa'l. Prova de posar un punt de control i fes el debug. Quins tipus de fitxers tens a la carpeta del teu ordinador del projecte? A què correspon cada tipus?

Exercici 2. Copia l'anterior main i fem algunes modificacions i ampliacions.

a. Amplia el menú amb l'opció "Redefinir el nom", que modifica el nom de l'usuari entrat a l'inici del programa.
b. El tipus de dades Vector té moltes 'built-in funcions' com ara vector::size() que ens ajuden a tractar-lo. Transformeu l'array de strings arr_options[] a un vector de strings i utilitzeu aquest per mostrar el menú utilitzant la funció vector::size() per saber quan s'ha de parar d'iterar.
c. Un cop funcioni el canvi anterior, agafeu la part de mostrar el menú i la comprovació de que l'opció entrada és correcte i traslladeu tot el codi a un mètode que retorna l'opció escollida.
d. Al main, un cop recollida l'opció escollida per l'usuari, mitjançant una estructura switch, fer una de les tres opcions següents: 1 sortir, 2 Benvinguda o 3 Redefinir el nom.

Digues els dubtes que hagis tingut i explica com els has solucionat.

Exercici 3. Defineix una classe `Circle` amb un mètode `getArea` al fitxer **Circle.cpp** (i el corresponent **Circle.h**) i el programa **main.cpp** de l'exercici 2 canviant les opcions del menú per "*Sortir*" i "*Introduir cercle*". "*Introduir cercle*" cridarà des del switch a un mètode que demanarà a l'usuari les dades d'un cercle (el seu radi), crearà l'objecte i mostrarà la seva àrea per pantalla. Un cop hagi acabat tornarà a mostrar el menú.

A més a més, crea un comptador de cercles al main i passa'l per referència a la funció "*Introduir cercle*" per que incrementi el valor del punter que rep.

Per exemple, el programa hauria de mostrar el següent:

```
*** Inici ***

Hola, que vols fer?

1. Sortir
2. Introduir cercle

2

Cercle número 1

Radi?

1

L'àrea d'aquest cercle és de 3.14

Hola, que vols fer?
```

1. Sortir
2. Introduir cercle

2

Cercle número 2

Radi?

2

L'àrea d'aquest cercle és de 12.56

Hola, que vols fer?

1. Sortir
2. Introduir cercle

1

*** Fi ***

Un cop acabat el codi anterior, pots millorar la classe **circle.cpp** per a què si el valor del radi és 0 o negatiu enviï una excepció i es mostri per pantalla el text "Atenció: aquest valor no és acceptat". El programa `main.cpp` haurà d'interceptar aquesta excepció i tornar a mostrar el menú.

Al laboratori us explicarem amb més detalls excepcions. A continuació us mostrem un exemple d'una funció per comparar dos nombres. En el vostre cas, l'excepció s'ha de llançar (`throw`) a la classe **circle.cpp**. Noteu que s'ha d'incloure una llibreria: `stdexcept`. Aquesta llibreria inclou moltes excepcions, una d'elles és `invalid_argument`. Vegeu tota la informació a: http://www.cplusplus.com/reference/stdexcept/

Noteu que quan el valor d'a o b és negatiu, es llança l'excepció amb el missatge que es vulgui.

```
#include <stdexcept> // aquest include es necessari al vostre codi

int compare( int a, int b ) {
    if ( a < 0 || b < 0 ) {  // condició per decidir llançar l'excepció
        throw std::invalid_argument( "received negative value" );// llançar-la
    }
}
```

A continuació teniu el tros del codi main on es crida al mètode `compare` i per recollir l'excepció es fica dins d'una estructura `try … catch`. El `try` conté el tros de codi que crida al mètode que pot llançar l'excepció i el `catch` recull l'excepció si es produeix. És molt important que veieu que SEMPRE es recull l'excepció per referència (per això té un & a l'objecte e).

```
try {
    compare( -1, 3 ); // Aqui vindrà el vostre codi del main
}
catch( const std::invalid_argument& e ) {
    // codi de gestió de l'excepció, en aquest exercici: missatge per consola
}
```

Quan implementeu el control de nombres negatius, per exemple, el programa hauria de mostrar el següent:

```
Cercle número 2

Radi?

-2

Atenció: aquest valor no és acceptat      // el produeix la classe Circle

S'ha produït una excepció                 // el produeix el main

Hola, que vols fer?

1. Sortir
2. Introduir cercle
```

Exercici 4. Defineix la classe `Ellipse` amb un mètode `getArea()` al fitxer **Ellipse.cpp**, **Ellipse.h**. La classe Ellipse guardarà el radi petit R1 i el radi gran R2 (ambdós de tipus float). Modifica la funció de "*Introduir cercle*" de l'exercici anterior per la funció "*Afegir figura*", què afegirà un cercle o una el·lipse segons com sigui l'entrada de l'usuari i mostrarà la seva àrea. Per crear un cercle l'usuari haurà d'introduir el parell (C radi), on **C** indica que la figura és un cercle i radi és un valor flotant positiu. Per crear una el·lipse l'usuari haurà d'entrar la tripleta (E R1 R2), on **E** indica que la figura és una el·lipse i **R1** i **R2** són valors flotants positius. Recorda tractar les excepcions per valors d'entrada negatius o cero per cada figura i portar un comptador per ambdues figures.

Crea també una altre opció en el menú ("*Glossari de figures*") per veure quants cercles i el·lipses s'han creat des de l'inici del programa.

Per exemple, el programa hauria de mostrar el següent:

```
*** Inici ***

Hola, que vols fer?

1. Sortir
2. Afegir figura
3. Glossari de figures

2

Entra les dades de la teva figura ( tipus[C o E] data1 data2[buit si el tipus es C] )

E 3 2     (o pogués ser C 2 )

L'àrea d'aquesta el·lipse és de 18.84     (si fos C 2, ens donaria 12.56)

Hola, que vols fer?
```

```
1. Sortir
2. Afegir figura
3. Glossari de figures

3

Tens 0 cercles i 1 el·lipses
```

Exercici 5. Fes una opció més en el menú de l'exercici anterior per tal de fer que llegeixi les dades d'un fitxer i les vagi processant fins a arribar al final. Recorda seguir tractant les excepcions i portar el comptador de les figures per mostrar-ho si es demana l'opció de l'exercici anterior "*Glossari de figures*".

Per exemple, en el fitxer hi haurà la següent informació:

```
C 3

E 5 6

E 1 2

C 8

E 4 4
```

On **C** fa referència a un Cercle i **E** a una El·lipse. Tenint en compte el fitxer anterior, el programa ha de mostrar per pantalla els següents resultats:

```
L'àrea d'aquest Cercle és de 28.26

L'àrea d'aquest el·lipse és de 94.2

L'àrea d'aquest el·lipse és de 6.28

L'àrea d'aquest Cercle és de 200.96

L'àrea d'aquest el·lipse és de 50.24
```

Exercici 6. Crea un projecte amb diverses classes relacionades amb geometria:

a) El cercle és el cas particular de l'el·lipse amb radi = R1 = R2, on R1 i R2 són els radis gran i petit de l'el·lipse definits en els exercicis anteriors. Reutilitza el que puguis del projecte anterior, conservant les quatre opcions de menú que ja tenim, però fes que la classe `Circle` hereti de la classe `Ellipse`, class Circle: public Ellipse. Fes que el constructor de cada classe al cridar-se imprimeixi per pantalla "Sóc el constructor del Cercle/El·lipse", i pensa que quan estiguis creant un cercle cridaràs també al constructor de l'el·lipse que no rep paràmetres (inicialitza R1=R2=0 per defecte). Assegura't que quan facis un cercle s'executen els dos constructors.

b) Comprova que totes les funcions del menú funcionen correctament inclosa la de llegir les figures, fixa't que per cada una es mostrin per pantalla els corresponents constructors.
c) Fes que el mètode `getArea()` de l'el·lipse sigui virtual.

Exercici 7. Fes una còpia del projecte anterior i agrega-hi la classe **EllipseContainer**, que serà un vector de com a màxim 10 figures que poden ser tan el·lipses com cercles, amb els següents mètodes:

a. `void addEllipse (Ellipse *)` que afegirà al vector la figura que se li passi per paràmetre, recorda que d'aquesta manera també podràs afegir cercles.
b. `float getAreas()` que retornarà la suma de les àrees dels cercles i el·lipses contingudes en el vector. Per recórrer el vector usa un iterador.
c. En el **main.cpp** crea un objecte `EllipseContainer` i afegeix cada figura que estiguis creant tant directament com des del fitxer, quan escollis l'opció corresponent en el menú.
d. Fes una opció més al menú per calcular la suma de totes les àrees i mostrar-la.
e. Comprova que es criden tots els constructors i destructors de les classes. Per comprovar-ho, posa un missatge al destructor de cada classe, per saber quins destructors es criden en tot moment al acabar el programa. Per esborrar cadascuna de les figures hauràs de fer un càsting dinàmic de Circle o Ellipse:

```
if (dynamic_cast<Circle*>(Ellipse[i]))

delete (Circle *)( Ellipse [i]);
```

A partir del què has observat respon:

a. Què ens permet fer l'herència que no podríem fer altrament?
b. Que passa si `getArea()` de la classe Ellipse no és virtual? Perquè?
c. Perquè els constructors i destructors els hem de cridar a les classes derivades i no a la classe base?
d. Es podria fer que `getArea()` fos un mètode de la classe pare "Ellipse"?
e. Anomena els membres de dades definits en els teus programes i digues a quina classe pertanyen. Explica les decisions de visibilitat de cadascun
f. L'iterador que recorre les figures, quant s'incrementa cada cop? Perquè?

3. Lliurament

A partir de la descripció del problema, es demana:

- Implementar els exercicis en C++ i usar el **NetBeans 8.2 amb C++ versió 11**. Lliurar el codi C++ corresponent als vostres exercicis en una carpeta anomenada *codi*, amb una subcarpeta per a cada exercici. Els comentaris de cada exercici (observacions, decisions preses i resposta a les preguntes plantejades, si n'hi ha) els fareu com a comentari al fitxer `main.cpp` de cada exercici.

Com a màxim el dia del lliurament es penjarà en el campus virtual un fitxer comprimit **en format ZIP** amb el nom del grup (A, B, C, D, E o F), el nom i cognoms de la persona i el número de la pràctica com a nom de fitxer. Per exemple, **GrupA_BartSimpson_P1.zip**, on P1 indica que és la "pràctica 1". El fitxer ZIP inclourà: la carpeta amb tot el codi.

Els criteris per acceptar la pràctica són:

- La pràctica ha de funcionar en la seva totalitat.
- La pràctica ha de ser orientada a objectes.

IMPORTANT: La còpia de la implementació de la pràctica implica un zero a la nota de pràctiques de l'assignatura per les persones implicades (tant la que ha copiat com la que ha deixat copiar).

4. Planificació

Per aquesta pràctica disposeu de dues setmanes però tingueu en compte que només teniu una sessió de laboratori amb el vostre professor de pràctiques.

Els professors us proposem la següent planificació:

- **Setmana 1 (del 11 de febrer al 17 de febrer)** → treball autònom a casa per cada alumne
  - Implementació de l'exercici 1 a l'exercici 4
  - No fareu lliurament aquesta setmana, només la última en aquesta pràctica
- **Setmana 2 (del 18 de febrer al 24 de febrer)** → laboratori amb el professor
  - Implementació de l'exercici 3 a l'exercici 7
  - Cal presentar el codi de tots els exercicis

El lliurament final de la pràctica serà el **dia 24 de febrer de 2019** al campus virtual.