Pràctica 1. Programació orientada a objectes amb C++. Conceptes bàsics de C++, Classes i objectes, Herència i polimorfisme

1. Introducció

***Objectius**:* Familiaritzar-se amb el llenguatge C++ i amb Visual Studio Code. Entendre els conceptes bàsics de C++ , classes i objectes, encapsulació d'objectes i la seva realització en C++.

Temes de teoria relacionats amb la pràctica: Tema 2. C++ i TADs
Les pràctiques es faran amb l'IDE Visual Studio Code i la versió 11 de C++.

2. Enunciat

Aquesta pràctica consta de 4 exercicis i estan ordenats segons la seva dificultat.

Exercici 1.

Crea un programa **mainExercici1.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 tres opcions; "*Sortir*", "*Benvinguda*" o "*Redefinir el nom*". Si escollim *Sortir* el programa finalitzarà, 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, si escollim *Redefinir el nom*, el programa llegeix per consola un nou nom i modifica el nom de l'usuari entrat a l'inici del programa i ens tornarà a mostrar el menú per tornar a escollir una opció.
A continuació teniu un exemple de l'entrada/sortida del codi.

```
Hola, com et dius? Anna
Hola Anna, que vols fer?
1. Sortir
2. Benvinguda
3. Redefinir nom
2
Benvingut/da a l'assignatura d'estructura de dades Anna
Hola Anna, que vols fer?
1. Sortir
2. Benvinguda
3. Redefinir nom
3
Hola, com et dius? Marta
Hola Marta, que vols fer?
1. Sortir
2. Benvinguda
3. Redefinir nom
1
Fins a la propera Marta
```

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", "Redefinir el nom"};
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 != 1), 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.
d. Un cop funcioni el pas 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.
e. 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.
   Un cop creat aquest programa executa'l. Prova de posar un punt de control i fes el debug.

Exercici 2.

Defineix una classe `Estudiant` amb dos mètodes: `getEdat i print` al fitxer **Estudiant.cpp** (i el corresponent **Estudiant.h**). Al programa **mainExercici2.cpp** les opcions del menú seran: "*Sortir*" i "*Informar Estudiant*".

"*Informar estudiant*" cridarà des del switch a un mètode que demanarà a l'usuari les dades d'un estudiant (el seu nom, el seu any de naixement i el número d'assignatures en les que està matriculat), crearà l'objecte i farà un `print` i mostrarà l'edat de l'estudiant. El mètode `print` imprimirà per pantalla les dades introduïdes per l'estudiant. També haurà de mostrar per pantalla l'edat de l'estudiant calculada utilitzant el seu any de naixement i l'any actual. Un cop hagi acabat tornarà a mostrar el menú.
A més a més, crea un comptador d'estudiants al `main` i passa'l per referència a la funció "*Informar estudiant*" per que incrementi el valor del punter que rep. Per exemple, el programa hauria de mostrar el següent:

```
Hola, que vols fer?
1. Sortir
2. Informar estudiant
2
Estudiant: 1
Nom? Pere
Any naixement? 1990
Assignatures? 5
Estudiant(Nom ==> Pere, Naixement ==> 1990, Assignatures ==> 5)
Edat del nou estudiant: 32

Hola, que vols fer?
1. Sortir
2. Informar estudiant
2
Estudiant: 2
```

```
Nom? Anna
Any naixement? 1997
Assignatures? 6
Estudiant(Nom ==> Anna, Naixement ==> 1997, Assignatures ==> 6)
Edat del nou estudiant: 25

Hola, que vols fer?
1. Sortir
2. Informar estudiant
1
Fins a la propera
```

El càlcul de l'edat de l'estudiant requereix l'any actual que es pot obtenir utilitzant les funcions C++ o, per fer-ho més fàcil, es permet utilitzar una constant a la classe per definir-ho. No tindrem en compte si l'estudiant ja ha fet els anys o no, assumirem que posarem els anys que tindrà al finalitzar l'any actual.

Un cop acabat el codi anterior, pots millorar la classe **Estudiant.cpp** per a què si el valor de l'any de naixement és 0, un nombre negatiu o superior a l'any actual enviï una excepció i es mostri per pantalla el text "Any de naixement incorrecte". El programa `mainExercici2.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 **Estudiant.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 ex).

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

Quan implementeu el control de l'any de naixement, per exemple, el programa hauria de mostrar el següent:

```
Hola, que vols fer?
1. Sortir
2. Informar estudiant
2
Estudiant: 1
Nom? Joan
Any naixement? 2150
Assignatures? 3
Any de naixement incorrecte

Hola, que vols fer?
1. Sortir
2. Informar estudiant
```

Exercici 3.

Fes una còpia del mainExercici2.cpp a mainExercici3.cpp. Fes una opció més en el menú de l'exercici (mainExercici3.cpp) 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 persones per mostrar-ho si es demana l'opció de l'exercici anterior "*Resum estudiants*". Per exemple, en el fitxer pot haver la següent informació:

```
Joan 2000 5
Marta 1997 4
Lluis 1999 3
Pere 2001 6
Eva 2002 5
```

Tenint en compte el fitxer anterior, el programa en execució generarà:

```
Hola, que vols fer?
1. Sortir
2. Afegir estudiant
3. Llegir fitxer
4. Resum estudiants
3
Ruta arxiu:
fitxer.txt
Estudiant(Nom ==> Joan, Naixement ==> 2000, Assignatures ==> 5)
Estudiant(Nom ==> Marta, Naixement ==> 1997, Assignatures ==> 4)
Estudiant(Nom ==> Lluis, Naixement ==> 1999, Assignatures ==> 3)
Estudiant(Nom ==> Pere, Naixement ==> 2001, Assignatures ==> 6)
Estudiant(Nom ==> Eva, Naixement ==> 2002, Assignatures ==> 5)

Hola, que vols fer?
1. Sortir
2. Afegir estudiant
```

```
3. Llegir fitxer
4. Resum estudiants
4
Estudiants creats: 5
Hola, que vols fer?
1. Sortir
2. Afegir estudiant
3. Llegir fitxer
4. Resum estudiants
1

Fins a la propera

Fins a la propera
```

Exercici 4. Biblioteca Municipal

Pas 1. Defineix una classe `Llibre` amb els següents atributs privats: `string titol; string autor; string isbn; int anyPublicacio;` i els mètodes: `getters` de cada atribut, un mètode `print` i el `constructor` amb paràmetres per inicialitzar tots els atributs del llibre al fitxer **Llibre.cpp** (i el corresponent **Llibre.h**).

El mètode `print` mostrarà per pantalla totes les dades del llibre.

Pas 2. Defineix una classe `Usuari` amb els següents atributs privats (no es poden modificar els tipus dels atributs):

```
string nom;
string adreca;
string poblacio;
string telefon;
string dni;
int edat;
vector<Llibre> llibres;
```

i els mètodes públics:

```
Usuari (string, string, string, string, string, int);
string getAdreca();
string getPoblacio();
string getNom();
string getTelefon();
string getDNI();
int getEdat();
void print();
void afegeixLlibre(string, string, string, char, string, int);
void eliminaLlibre(string);
void mostraLlibres();
```

al fitxer **Usuari.cpp** (i el corresponent **Usuari.h**).

- El mètode `print` imprimeix totes les dades de l'usuari per consola.
- El mètode `afegeixLlibre` rep totes les dades per crear un llibre i l'afegeix al vector de llibres de l'usuari. Recordeu que totes les dades han d'estar en minúscules.
- El mètode `eliminaLlibre` rep el títol del llibre, i si existeix, l'elimina del vector de llibres. Assumim que no hi haurà dos llibres amb el mateix títol en l'usuari.
- El mètode `mostrarLlibres` imprimirà per pantalla tots els llibres que tingui l'usuari. Una línia per llibre.

Pas 3. Al programa **mainExercici4.cpp** les opcions del menú seran:

```
1. Afegir Usuari
2. Eliminar Usuari
3. Afegir Llibre a un usuari
4. Eliminar Llibre d'un usuari
5. Imprimir usuaris de la biblioteca municipal
6. Imprimir els libres d'un usuari
0. Sortir
```

En el **mainExercici4.cpp** es crearà un vector `vector<Usuari> usuaris_biblio;` i es farà la implementació de les funcionalitats de la biblioteca municipal. La implementació de cada funcionalitat es farà un mètode propi dins del mainExercici4.cpp. Els mètodes a implementar són:

- `Afegir usuari`, aquest mètode demana les dades de l'usuari a crear i si es vol afegir un llibre a l'usuari. En cas afirmatiu, es demanarà també les dades del llibre. El mètode ha de comprovar abans d'afegir el nou usuari, que aquest no existeixi ja en el vector d'usuaris. El camp que servirà per garantir que no hi ha elements repetits serà el DNI. En el cas de que l'usuari no existeix al vector d'usuaris, s'afegirà el nou usuari. Si existeix, es donarà el missatge corresponent per consola i no s'afegirà.
- `Eliminar usuari`, aquest mètode a partir del DNI introduït per teclat, eliminarà l'usuari del vector d'usuaris si aquest existeix. En cas que no existeixi, es donarà el missatge corresponent.
- `Afegir llibre a un usuari`, aquest mètode demana el DNI de l'usuari i les dades del llibre que es vol afegir. Si l'usuari, se li afegeix el llibre a la seva llista de llibres. Si l'usuari no existeix, s'ha de retornar el missatge corresponent.
- `Eliminar llibre d'un usuari`, és un mètode que a partir del DNI de l'usuari i el títol del llibre, s'esborra el llibre de l'usuari si aquest DNI existeix dins del vector d'usuaris i el llibre també existeix dins del vector de llibres de l'usuari.
- `Imprimir usuaris biblioteca municipal`, mostra totes les dades dels usuaris de la biblioteca municipal. Un usuari per línia.
- `Imprimir els llibres d'un usuari`, és un mètode que a partir del DNI d'un usuari, si aquest existeix, es mostra les dades de tots els seus llibres.
- `Sortir`, és un mètode per indicar per consola que s'està tancant l'aplicació de la biblioteca municipal.

Lliurament

A partir de la descripció dels exercicis, es demana:

- Implementar els exercicis en C++ i usar el **Visual Studio Code 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 `mainExerciciX.cpp` de cada exercici. **Recordeu que també heu de posar el nom i cognoms de l'autor del codi als diferents fitxers .h i .cpp. A l'inici del cada fitxer.**

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 perfectament, és a dir, no s'accepten codis que no compilen i executen.
- 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).

3. Planificació

Per fer aquesta pràctica disposeu de tres setmanes. Els professors us proposem la següent planificació:

- **Setmana 1 (del 9 al 13 de febrer)** → Laboratori (**Lab 1**)
  - Treball al laboratori sobre exercicis bàsics de C++ i exercici 1
  - Implementació a casa de l'exercici 2
- **Setmana 2 (del 16 al 20 de febrer)** → Laboratori (**Lab 2**)
  - Dubtes de la implementació de l'exercici 2
  - Implementació de l'exercici 3
  - Implementació a casa de l'exercici 4
- **Setmana 2 (del 16 al 20 de febrer)** → Laboratori (**Lab 3**)
  - Dubtes de la implementació de l'exercici 4

Val a dir que és molt important implementar els exercicis d'aquesta pràctica, ja que serveixen per aprendre la sintaxi de C++, entendre que són classes i objectes i com treballar amb fitxers a C++. Aquests conceptes són bàsics per implementar la resta de pràctiques de l'assignatura.

Aquesta pràctica no s'avalua amb el codi implementat, **es farà un qüestionari durant el Lab4** (cadascú amb el grup que tingui assignat la setmana del 2 al 6 de Març) on entraran tots els conceptes vistos a la pràctica. La no assistència a la prova implica un NP a la pràctica 1.

El lliurament final de la pràctica serà el **dia 1 de Març de 2026** al campus virtual. Entregueu tots els exercicis que tingueu implementats.