UNIVERSITAT DE BARCELONA

Sessió 3. Teoricopràctica

Estructura de Dades
Curs 2020-2021

Grau en Enginyeria Informàtica
Facultat de Matemàtiques i Informàtica,
Universitat de Barcelona

Contingut

1. Exercici Time
2. Exercici Herència
3. Exercici Usuari

Exercici Time

Problema 14 llista de problemes 1

UNIVERSITAT DE BARCELONA

Exercici Time (ex. 14)

- Crea una classe **Time** que permeti guardar l'hora, minut i segon.
- A la implementació de la classe, fareu els següents mètodes:
  - Un getter per a cada atribut
  - Una funció setter per a cada atribut
  - Un constructor per defecte per a la classe
  - Un constructor amb paràmetres per a la classe. Aquest constructor recollirà les dades pels tres atributs
  - Una funció nextSecond que retornar un objecte Time amb un segon més que l'actual
  - Una funció print que mostri per pantalla l'hora, minut i segon en el següent format: hh:mm:ss
- Implementeu un **main.cpp** que crei dues instàncies de Time. La primera d'elles ha d'estar creada estàticament amb el constructor amb paràmetres i la segona d'elles l'heu de crear dinàmicament. Afegiu dues hores a cada instancia i imprimiu per pantalla el seu contingut

Time.h

```
#ifndef TIME_H
#define TIME_H

class Time {
private:
    int hour;    // 0 - 23
    int minute;  // 0 - 59
    int second;  // 0 - 59
public:
    Time(int h = 0, int m = 0, int s = 0);
    int getHour() const;
    void setHour(int h);
    int getMinute() const;
    void setMinute(int m);
    int getSecond() const;
    void setSecond(int s);
    void setTime(int h, int m, int s);
    void print() const;
};
#endif /* TIME_H */
```

Time.cpp

```
#include <iostream>
#include <iomanip>
#include <stdexcept> // Necessari per les excepcions
#include "Time.h"

using namespace std;

Time::Time(int h, int m, int s) {
    setHour(h);
    setMinute(m);
    setSecond(s);
}
```

Time.cpp (cont.)

```
int Time::getHour() const {     return hour;  }

void Time::setHour(int h) {
      if (h >= 0 && h <= 23) {
            hour = h;
      } else {
         throw invalid_argument("Invalid hour! Hour shall be 0-23.");
      }
}
int Time::getMinute() const {    return minute; }

void Time::setMinute(int m) {
    if (m >= 0 && m <= 59) {
    minute = m;
    } else {
    throw invalid_argument("Invalid minute! Minute shall be 0-59.");
}
```

Time.cpp (cont.)

```
int Time::getSecond() const {     return second; }
void Time::setSecond(int s) {
    if (s >= 0 && s <= 59) {
    second = s;
    } else {    throw invalid_argument("Invalid second! Second shall be 0-59.");   }
}

void Time::setTime(int h, int m, int s) {
    setHour(h);
    setMinute(m);
    setSecond(s);
}

void Time::print() const {
    cout << setfill('0');
    cout << setw(2) << hour << ":" << setw(2) << minute << ":"   << setw(2) << second << endl;
}
```

Main.cpp bàsic

```
#include <iostream>
#include <stdexcept> // Per manegar les excepcions
#include "Time.h"
using namespace std;

int main(int argc, char** argv)
{       // Proveu de descomentar el codi i veure què passa
         // Time t2(25, 0, 0); // si fem això el programa acaba abruptament
        // t2.print(); // La resta del programa no s'executarà
        // Solució captar les possibles excepcions
        try {
             Time t1(25, 0, 0); // Saltarà tot el que hi hagi a sota i anirà al catch si l'excepció s'activa
             t1.print();
            // Continua després del try-catch si l'excepció no es produeix
        } catch (invalid_argument& ex) { // necessiteu incloure el include
            cout << "Exception: " << ex.what() << endl;
            // Continue tdesprés del try-catch
        }
    cout << "Next statement after try-catch" << endl;
   return 0;
}
```

Main.cpp Avançat (I)

```
int main(int argc, char** argv)
{
        Time t1(1, 2, 3); // declara un objecte estàtic
        t1.print(); // usa el print amb l'objecte i digues quina es la sortida
        Time* ptrT1 = &t1;  // declara un punter i assigna l'adreça de l'objecte estàtic creat abans
        (*ptrT1).print(); // usa el print amb l'objecte i digues quina es la sortida
        ptrT1->print(); // usa el print amb l'objecte i digues quina es la sortida
        // RECORDA QUE: anObjectPtr->member és el mateix que  (*anObjectPtr).member

        Time& refT1 = t1; // crea a un objecte Time anomenat refT1 un alias de t1
        refT1.print(); // usa el print amb l'objecte i digues quina es la sortida

        Time* ptrT2 = new Time(4, 5, 6); // Crea dinàmicament un objecte
        ptrT2->print(); // usa el print amb l'objecte i digues quina es la sortida
        delete ptrT2; // allibera la memoria de l'objecte dinàmic
return 0;
}
```

Main.cpp Avançat (II)

```
int main(int argc, char** argv)
{     // Declara un array estàtic de 2 d'objectes Time anomenat tArray1 inicialitzats amb el constructor per defecte
       Time tArray1[2];
       tArray1[0].print();    tArray1[1].print(); // usa el print i imprimeix els dos objectes i digues quina es la sortida
       // Declara un array estàtic de 2 d'objectes Time anomenat tArray2 inicialitzats amb el constructor per paràmetres
      Time tArray2[2] = {Time(7, 8, 9), Time(10)};
      tArray2[0].print();   tArray2[1].print(); // usa el print i imprimeix els dos objectes i digues quina es la sortida
       // Declara un array dinàmic de 2 d'objectes Time anomenat ptrTArray3 inicialitzats amb el constructor per defecte
      Time* ptrTArray3 = new Time[2];
       ptrTArray3[0].print();  ptrTArray3[1].print(); // usa el print i imprimeix els dos objectes i digues quina es la sortida
      delete[] ptrTArray3; // allibera l'espai de l'array
    // Declara un array dinàmic de 2 d'objectes Time anomenat ptrTArray4 inicialitzats amb el constructor per paràmetres
      Time* ptrTArray4 = new Time[2]{Time(11, 12, 13),Time(14)};
      ptrTArray4->print();   (ptrTArray4 + 1)->print(); // usa el print i imprimeix els dos objectes i digues quina es la sortida
      delete[] ptrTArray4; // allibera l'espai de l'array
   return 0;
}
```

Exercici Herència

Quina és la sortida per pantalla del següent main?

Exercici herència

A.h

```
#include <iostream.h>

class A{

public:
   A();
   ~A();
   void    QuiSoc();
   virtual void QuiSocTambe();
};
```

Exercici herència

A.cpp

```
#include "A.h”
using namespace std;

A::A(){ cout<<"Constructor de A"<<endl;}

A::~A(){cout<<"Destructor de A"<<endl;}

void A::QuiSoc(){ cout<<"Soc A"<<endl; }

void A::QuiSocTambe(){
   cout<<"Soc A tambe"<<endl;
}
```

Exercici herència

B.h

```
#include <iostream.h>

class B:public A{
public:
   B();
   ~B();
   void QuiSoc();
   void QuiSocTambe();
};
```

Exercici herència

B.cpp

```
#include "B.h”
using namespace std;

B::B(){    cout<<"Constructor de B"<<endl;
}
 B::~B(){    cout<<"Destructor de B"<<endl;
}
void B::QuiSoc(){cout<<"Soc B"<<endl;
}
void B::QuiSocTambe(){cout<<"Soc B tambe"<<endl;
}
```

Exercici herència

```
void main(){

A *p1;
B *p2;
A *p3;

 cout<<"Prova amb el punter p1"<<endl;
    p1= new A;
    p1->QuiSoc();
    p1->QuiSocTambe();
    delete p1;
    cout<<endl;

    cout<<"Prova amb el punter p2"<<endl;
    p2 = new B;
    p2->QuiSoc();
    p2->QuiSocTambe();
    delete p2;
    cout<<endl;

    cout<<"Prova amb el punter p3"<<endl;
    p3= new B;
    p3->QuiSoc();
    p3->QuiSocTambe();
    delete p3;
    cout<<endl;
}
```

Exercici herència: Sortida per pantalla

```
Prova amb el punter p1
Constructor de A
Soc A
Soc A tambe
Destructor de A
```

```
Prova amb el punter p2
Constructor de A
Constructor de B
Soc B
Soc B tambe
Destructor de B
Destructor de A
```

```
Prova amb el punter p3
Constructor de A
Constructor de B
Soc A
Soc B tambe
Destructor de A
```

Exercici Usuari

Problema 15 a la llista de problemes 1

Enunciat

Definiu un nou projecte NetBeans i definiu la classe Usuari

La classe Usuari té dos atributs: La identificació de l'usuari (conegut com ID) i el nombre d'Identificació personal (conegut com a PIN). Per a efectes d'aquest exercici, es considera vàlid un ID que tingui una longitud mínima de 6 caràcters; i en el cas del PIN el valor vàlid serà qualsevol nombre major que zero.

Implementeu els **següents mètodes a la classe**:

- Una funció setter per introduir les dues dades membre.
- Una funció setter per a cada dada membre.
- Un constructor per defecte per a la classe.
- Un constructor amb paràmetres per a la classe. Aquest constructor recollirà les dades per tots els atributs.
- Una funció print que mostri per pantalla les dades membre de l'objecte.

A més a més, implementeu un **main.cpp** que faci les següents accions:

- Definir i inicialitzar un primer objecte estàticament amb el constructor per defecte
- Definir i inicialitzar un segon objecte estàticament amb el constructor amb paràmetres
- Definir i inicialitzar un tercer objecte dinàmicament amb el constructor per defecte
- Definir i inicialitzar un quart objecte dinàmicament amb el constructor amb paràmetres
- Demanar les dades pel teclat per definir valors als objectes construïts amb el constructor per defecte.
- Guardar els quatre objectes en un array estàtic de 4 posicions
- Imprimir la informació de cada objecte de l'array.

Usuari.h

```
#include <iostream>
#include <string>
using namespace std;
#ifndef USUARI_H
#define USUARI_H
class Usuari{
    private:
        int pin;
        string id;
    public:
        Usuari();
        Usuari(string id, int pin);
        void setData(string id, int pin);
        void setId(string id);
        void setPin(int pin);
        void print();
};
#endif /* USUARI_H */
```

Usuari.cpp

```
#include "Usuari.h"
Usuari::Usuari() {
    //Default construncto
    this->setId("Alicia");
    this->setPin(12345); // Best password ever
}
Usuari::Usuari(string id, int pin) {
    this->setData(id, pin);
}
void Usuari::setData(string id, int pin){
    this->setId(id);     this->setPin(pin);
}
void Usuari::setId(string id){
    if(id.size() >= 6){         this->id = id;
    }else{     cout << "Incorrect id" << endl;
    }
}
void Usuari::setPin(int pin) {     this->pin = pin; }

void Usuari::print() {     cout << "The user: " << this->id << " has the pin: " << this->pin << endl; }
```

Main.cpp

```
#include "Usuari.h"
using namespace std;
int main(){
    Usuari usr1;
    Usuari usr2("Lolita", 341234);
    Usuari *usr3 = new Usuari();
    Usuari *usr4 = new Usuari("Antoni", 654534);
    cout << "Intro the new id for the default constructor objects " << endl;
    string id;
    cin >> id;
    cout << "Intro the new pin for the default constructor objects " << endl;
    int pin;
    cin >> pin;
    usr1.setData(id,pin);      usr3->setData(id,pin);
    Usuari usuaris[4];
    usuaris[0] = usr1;      usuaris[1] = usr2;
    usuaris[2] = *usr3;      usuaris[3] = *usr4;

    for(int i=0; i<4; i++){         usuaris[i].print();     }
    delete usr3;
    delete usr4;
    return 0;
}
```