FACULTAD DE INFORMÁTICA

Arrays y cadenas de caracteres

TALLER

Programación orientada a objetos — Unidad 4

Autor: Luis Hernández Yáñez

FdI UCM

Cuestiones

Sobre los arrays (*resolución individual*)

¿Qué es lo que hace mal el siguiente programa?

```
#include <iostream>
using namespace std;
#define MAX 100

int main() {
  int lista[MAX];
  for(int i = 1; i <= MAX; i++) lista[i] = i;
  for(int i = 1; i <= MAX; i++) cout << lista[i] << endl;

  return 0;
}
```

¡¡¡ Y, sin embargo, parece que funciona !!!
(aunque lo mejor es no fiarse)

FdI UCM

Cuestiones

¿Qué es lo que muestra el siguiente programa?

```
#include <iostream>
using namespace std;
#define MAX 10
typedef int Lista[MAX];

void muestra(Lista);
void acumula(Lista);

int main() {
  Lista lista;
  for(int i = 0; i < MAX; i++) lista[i] = i;
  muestra(lista);
  acumula(lista);
  muestra(lista);

  return 0;
}

void muestra(Lista array) {
  for(int i = 0; i < MAX; i++) cout << array[i] << " ";
  cout << endl;
}

void acumula(Lista array) {
  for(int i = 0; i < MAX; i++) array[i] += 2;
}
```


FdI UCM

Composición de tipos

Cómo pasa el tiempo (*resolución en grupo*)

Resulta muy habitual que en los programas se tengan que manejar instantes de tiempo (fechas y horas). Por esto, os proporciono dos clases: **Fecha** y **Hora**. La primera implementa el tipo **Fecha** que permite manipular fechas. La segunda implementa el tipo **Hora** que permite manipular instantes de tiempo de un día.

A continuación se muestran las interfaces de esas clases.

Antes de realizar el ejercicio que se plantea tras las definiciones de las clases, jugad un poco con ellas para comprobar cómo funcionan.

La clase **Fecha**

```
// Clase Fecha - Archivo de cabecera "fecha.h"

#ifndef fecha_h // Evitar inclusiones múltiples
#define fecha_h

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

class Fecha {
public:
  Fecha(int = 1, int = 1, int = 1900);
    // Constructor (día, mes, año)
  Fecha(const Fecha&);
  Fecha& operator=(const Fecha&);
  ~Fecha();

  int dia() const;
  int mes() const;  // Accedentes
  int anio() const;
```

(*continúa*)


```
  void dia(int);
  void mes(int);    // Mutadores
  void anio(int);

  long int operator-(Fecha) const;
    // Días transcurridos entre la fecha y la proporcionada
  Fecha operator+(long int) const; // Fecha más días
  Fecha operator-(long int) const; // Fecha menos días
  Fecha& operator+=(long int); // Abreviatura para la suma
  Fecha& operator-=(long int); // Abreviatura para la resta
  Fecha& operator++(); // Día siguiente
  Fecha& operator++(int); // (variante postfija)
  Fecha& operator--(); // Día anterior
  Fecha& operator--(int); // (variante postfija)

  bool operator==(Fecha) const;
  bool operator!=(Fecha) const;
  bool operator<(Fecha) const;   // Operadores relacionales
  bool operator<=(Fecha) const;
  bool operator>(Fecha) const;
  bool operator>=(Fecha) const;
```

(*continúa*)


```
  bool bisiesto() const;
  string diaSemana() const; // Devuelve el día de la semana
  string completa() const;
    // Devuelve la fecha en formato largo:
    // <día de la semana> <día> de <nombre del mes> de <año>
    // Por ejemplo: sabado 1 de abril de 2000
  string reducida() const;
    // Devuelve la fecha en formato corto:
    // <día>/<mes>/<año> (día y mes con 2 dígitos)
    // Por ejemplo: 01/04/2000
  void leer();

private:
  // No es necesario conocer esta sección para usar la clase
};

#endif
```


La clase **Hora**

```
// Clase Hora - Archivo de cabecera "hora.h"

#ifndef hora_h // Evitar inclusiones múltiples
#define hora_h

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

class Hora {
public:
   Hora(int = 0, int = 0, int = 0);
     // Constructor (horas, minutos, segundos)
  Hora(const Hora&);
  Hora& operator=(const Hora&);
  ~Hora();
  Hora(long int); // Otro constructor:construye la hora
    // desde la medianoche pasados los segundos indicados.
```

(*continúa*)

Composición de tipos

```
    void horas(int);
    void minutos(int);   // Mutadores
    void segundos(int);

    int horas() const;
    int minutos() const; // Accedentes
    int segundos() const;

    long int operator-(Hora) const;
      // Segundos transcurridos entre la hora y la proporcionada
    Hora operator+(long int) const; // Hora más segundos
    Hora operator-(long int) const; // Hora menos segundos
    Hora& operator+=(long int); // Abreviatura para la suma
    Hora& operator-=(long int); // Abreviatura para la resta
    Hora& operator++(); // Un segundo más
    Hora& operator++(int); // (versión postfija)
    Hora& operator--(); // Un segundo menos
    Hora& operator--(int); // (versión postfija)
```

(*continúa*)

Composición de tipos

```
    bool operator==(Hora) const;
    bool operator!=(Hora) const;
    bool operator<(Hora) const;    // Operadores relacionales
    bool operator<=(Hora) const;
    bool operator>(Hora) const;
    bool operator>=(Hora) const;

    string hora() const; // Devuelve la hora en una cadena
    void leer();

  private:
    // No es necesario conocer esta sección para usar la clase
  };

  #endif
```


Composición de tipos

Si queremos trabajar en los programas con instantes históricos, necesitamos las dos clases anteriores, ya que se ha de conocer la hora concreta de la fecha concreta en que se produce cada evento.

Aunque podríamos mantener la fecha y la hora por separado, resulta mucho más adecuado crear otra clase, **FechaHora**, que se encargue de implementar los instantes históricos. Así, por ejemplo, cuando se incremente la hora y llegue a ser 00:00:00, se pasará automáticamente al día siguiente.

Así pues, lo que se pide es desarrollar esa nueva clase **FechaHora**. A continuación se muestra la interfaz de la clase que se quiere que implementéis.
De lo que os tenéis que encargar vosotros es de todo lo que falta.

Composición de tipos

Interfaz deseada para la clase **FechaHora**

```
// Clase FechaHora - Archivo de cabecera "fechahora.h"

#ifndef fechahora_h // Evitar inclusiones múltiples
#define fechahora_h

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

class FechaHora {
public:
  FechaHora(); // Constructor predeterminado
  FechaHora(const FechaHora&); // Constructor de copia
  FechaHora& operator=(const FechaHora&); // Op. asignación
  ~FechaHora(); // Destructor

  // Otros constructores:
  FechaHora(Fecha);
  FechaHora(Hora);
  FechaHora(Fecha, Hora);
```

(*continúa*)

Composición de tipos

```
    // Accedentes y mutadores:
    Fecha fecha() const;
    Hora hora() const;
    void fecha(Fecha);
    void hora(Hora);
    // Operadores aritméticos:
    long int operator-(FechaHora) const;
    // Segundos entre la FechaHora y la proporcionada
    long int operator-(Hora) const;
    // Segundos entre la hora y la proporcionada
    long int operator-(Fecha) const;
    // Días entre la fecha y la proporcionada
    FechaHora operator+(long int) const; // + segundos
    FechaHora operator-(long int) const; // - segundos
    FechaHora& operator++(); // Un segundo más
    FechaHora& operator--(); // Un segundo menos
    // Operadores relacionales:
    bool operator==(FechaHora) const;
    bool operator!=(FechaHora) const;
```

(*continúa*)

Composición de tipos

```
    bool operator<(FechaHora) const;
    bool operator<=(FechaHora) const;
    bool operator>(FechaHora) const;
    bool operator>=(FechaHora) const;

    // Otros métodos:
    long int aDias(long int); // Segundos a días
    long int aSegundos(long int); // Días a segundos
    string diaSemana() const;
    string completa() const;
      // Cadena con la fecha completa seguida de la hora
    string reducida() const;
      // Cadena con la fecha reducida seguida de la hora
    void leer();

  private:
    // Esto es cosa vuestra (es parte de la implementación)
  };

  #endif
```


Composición de tipos

Algunas indicaciones

- ✓ Necesitaréis una constante que indique el número de segundos por día (86.400). Será un atributo de clase (compartido).
- ✓ Al sumar (restar) un número de segundos hay que calcular el número de días completos y quedarse con los segundos restantes, para luego proceder a sumar (restar) los días a la fecha y sumar (restar) los segundos restantes a la hora.
- ✓ Al incrementar (decrementar) hay que tener en cuenta que se puede cambiar de fecha.

En el mundo real

Para una biblioteca (*resolución en grupo*)

Crea una clase **`Libro`** que modele la información que se mantiene en una biblioteca sobre cada libro: **`título`**, **`autor`** (usa la clase **`Persona`**), **`ISBN`**, **`páginas`**, **`edición`**, **`editorial`** , **`lugar`** (ciudad y país) y **`fecha`** de edición (usa la clase **`Fecha`** anterior). La clase se debe ajustar a la Forma canónica ortodoxa y proporcionar los siguientes servicios: accedentes y mutadores, método para leer la información y método para mostrar la información. Este último método mostrará la información del libro con este formato:

```
Título: C++ Cómo programar
2a. edición
Autor: Deitel, Harvey M.
ISBN: 0-13-528910-6
Prentice Hall, Méjico (Méjico), martes 16 de noviembre de 1999
1130 páginas
```

Prueba la clase en una función **`main()`**.