PARADIGMAS DE PROGRAMACIÓN
Licenciatura en Informática
Programador Universitario

UNIDAD III
PARADIGMAS DE PROGRAMACIÓN
Programación Orientada a Objetos
Clases en C++

TAD implementado como una clase

Una clase es un **tipo definido por el usuario**; es decir, un marco que permite crear objetos que tienen su misma **estructura de datos o atributos** y **comportamientos u operaciones** comunes

Clases en C++

- **class**
- **struct** ⇨ sirven para implementar TAD
- **union**

El nombre de la clase se puede usar para **declarar objetos** de la misma

Duración estática, local o dinámica

2

Especificadores

De Acceso a miembros

Terminan con **:** y pueden aparecer varias veces en una definición de clase

La interfaz de una clase se puede dividir en

- **public** — Visibles a todos los usuarios de la clase
- **protected** — Visibles a la propia clase y a sus subclases
- **private** — Visibles únicamente a la propia clase

Modificadores de acceso

- **const** — Indica que su valor no admite modificaciones
- **volatile** — Indica que su valor puede ser modificado fuera del control del compilador

3

Especificadores

De Almacenamiento de miembros

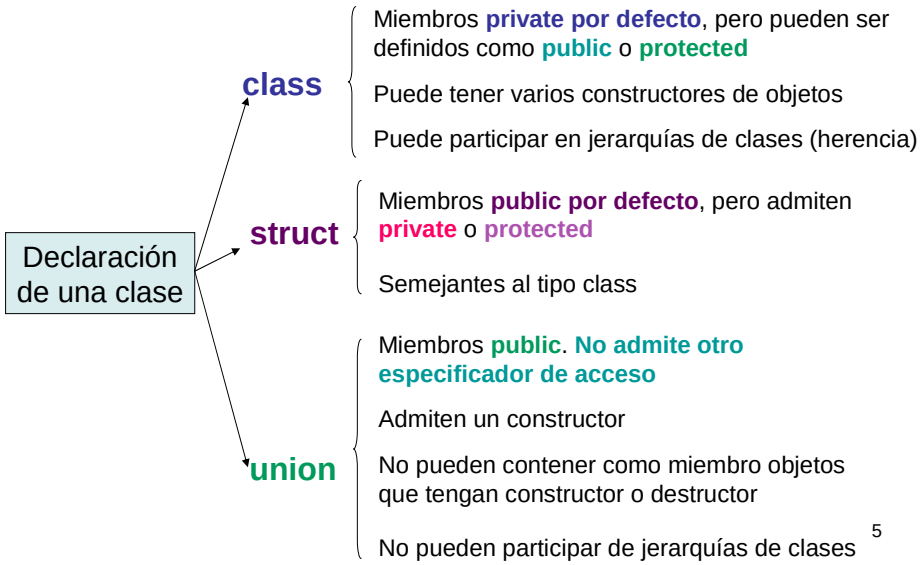
- **static** — Especifica que es un miembro de clase y no de objeto

De Almacenamiento de clase

- **extern** — Indica que la clase tiene visibilidad de todos los archivos que componen el programa y que el enlazado de todos los identificadores de la clase es externo
- **static** — La visibilidad queda restringida a ese archivo, con enlazado interno

4

Declaración de una clase



5

Declaración de una clase

class → Definición de la clase

- Se escribe entre **{ }**
- Contiene la lista completa de miembros de la clase (atributos y métodos)
- Termina con **;**

- Los **miembros datos** de la clase se describen en una lista que puede incluir datos de cualquier tipo
- **No puede ser un miembro dato de la misma clase**, aunque si un puntero o referencia a ella
- La declaración de la clase contiene los **prototipos** de la funciones miembros
- Para definir cada función, fuera del alcance de la clase, se debe usar el **operador de resolución de alcance ::**
- Un identificador no puede ser simultáneamente un miembro dato y una función miembro

```
class Nodo {
  private:
    item dato;
    Nodo* siguiente;
  public:
    Nodo(item, Nodo*);
    item getDato();
    Nodo* getSiguiente();
};
```

```
item Nodo::getDato(){
   return dato;
}
```

6

Declaración de una clase

class

- Funciones miembros especiales
  - **Constructor** — tiene el mismo nombre de la clase
  - **Destructor** — tiene el mismo nombre de la clase precedida por el carácter ~ (tilde)
- El nombre de la clase se convierte en un especificador de tipo
- Cuando se crea un objeto de una clase, **automáticamente**, se llama al constructor de la clase que inicializa a cada miembro dato
- La implementación de una clase está oculta a sus clientes
- C++ permite **declaraciones de clases incompletas** o **forward** y efectuar referencias al nombre de esa clase (con punteros a la misma) antes de que la clase haya sido definida totalmente

7

Declaración incompleta de una clase

```
class X; // Definición incompleta.
         // Exige una declaración posterior

void f(X *a);

class Y {
    X *pX;  // objeto puntero de la clase X
    ....
};

class X{    // definición completa
    ....
};
```

Declaración incompleta de una clase

Empleado.h

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

class Empresa;

class Empleado{
    Empresa *empleador;
    string nombre;
    long int DNI;
public:
    Empleado(string nom, long int doc, Empresa *empresa);
    string getNombre();
    long int getDNI();
    string nombreEmpresa();
};
```

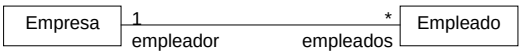
NO INCLUIR EMPRESA.H

Empresa.h

```
#include <iostream>
#include "Empleado.h"
using namespace std;

class Empresa{
    string nombreEmpresa;
    Empleado *empleados[100];
    int indice;
public:
    Empresa(string nombre);
    void listarEmpleados();
    string getNombre();
};
```


Declaración incompleta de una clase



Empleado.cpp

```
#include <iostream>
#include "Empleado.h"
#include "Empresa.h"
using namespace std;

string Empleado::nombreEmpresa(){
    return this->empleador->getNombre();
}
//..
```

Empresa.cpp

```
#include <iostream>
#include "Empresa.h"
using namespace std;

void Empresa::listarEmpleados(){
    for(int i=0;i<indice;i++){
        cout<<this->empleados[i]->getNombre()<<endl;
    }
};
//...
```

Declaración de Objetos

Los objetos de una clase **se declaran del mismo modo** que los objetos de cualquier tipo de datos y se crean en tiempo de ejecución, con la estructura definida en la clase

Pueden ser:

- Asignados
- Pasados como parámetros a funciones
- Devueltos como valor de retorno de funciones
- Etc.

```
clas1 obj1, *obj2, &obj3=obj1,obj4[10];

clas1 fun(clas1);    // prototipo de función (fun) con argumento
                     // y retorno de la clase clas1.

obj1 = fun(obj4[i]);     // asignación de un objeto y llamada a
                         // función con objeto como parámetro.
```

Alcance de clase y acceso a miembros de clase

Alcance de clase → Incluye los nombres de variables y funciones declaradas en ella, y los nombres de datos y funciones miembros de la misma

Acceso a miembros de clase → Operador de selección → Directo • Indirecto →

A. Dentro del alcance de la clase → Los miembros se acceden directamente referenciados por su nombre

B. Fuera del alcance de la clase → Los miembros se referencian a través del nombre/referencia/puntero de un objeto de la clase

12

Alcance de clase y acceso a miembros de clase

```
Racional Racional::multiplicar(Racional Q){
   Racional aux;
   aux.num = num * Q.num;
   aux.den = den * Q.den;
   escribir();
   Q.escribir();
   return aux;
}
```

A. Dentro del alcance de la clase

```
class Racional {
    int num , den;
  public:
    Racional(int a, int b);
    Racional multiplicar(Racional Q);
    void setNumerador(int num);
    int getNumerador();
    void escribir(){  cout<<num <<"/"<<getDenominador(); <<endl; }
};
```

```
Void main() {
   Racional R1(1,2), &R2 = R1, *R3 = &R1;
   R1.escribir();
   R2.setNumerador(4);
   R3->escribir();
   //R1.num = 5; //Error
}
```

B. Fuera del alcance de la clase

Alcance de clase y acceso a miembros de clase

OBSERVACIONES

- Si una función miembro define una variable del mismo nombre que una variable con alcance de clase, ésta última queda oculta por la variable con alcance de función, dentro del alcance de la función.

- Se puede tener acceso a variables ocultas mediante el uso del **operador de resolución de alcance**:

```
Binario    <nombre_de_clase>::variable_de_instancia
Unario     ::variable global
```

```
void Racional::setNumerador (int num){
   Racional::num = num;
}
```

14

Como usar datos y funciones miembros

Miembros Privados/Protegidos SOLO los pueden manipular → **Funciones miembros**, **Amigos** de la clase

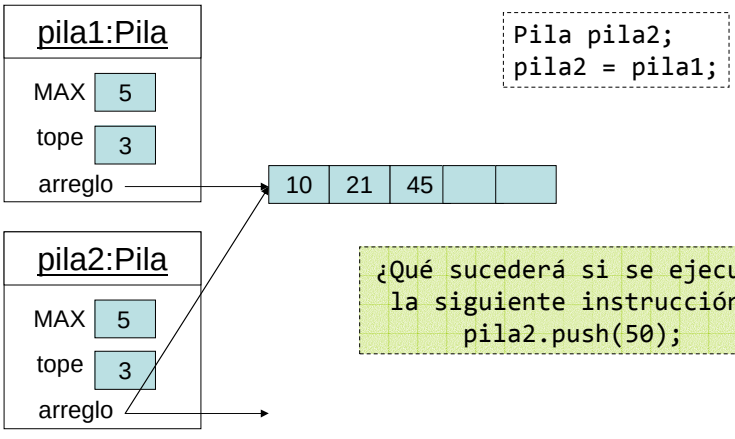
Getters y Setters → **Funciones miembros públicas** que permiten a los clientes de la clase obtener o definir los valores de los datos miembros privados/protegidos

Operador de asignación (=)

- Asignar un objeto a otro objeto del mismo tipo
- Realiza una **copia miembro a miembro**, o copia por omisión
- Se puede sobrecargar para que lleve a cabo una asignación adecuada de los miembros del objeto

15

Operador de asignación (=)



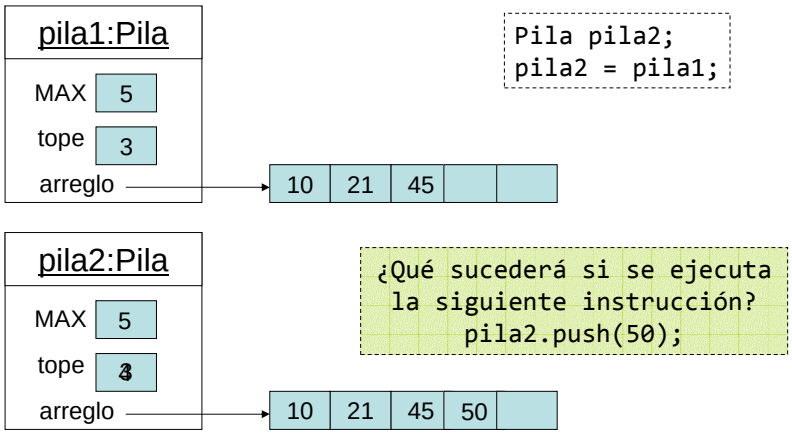

```
Pila pila2;
pila2 = pila1;
```

```
¿Qué sucederá si se ejecuta
la siguiente instrucción?
pila2.push(50);
```

REGLA → Cuando el objeto contiene atributos que poseen reserva dinámica de memoria, se DEBE sobrecargar el operador =

16

Operador de asignación (=)



17

Como usar datos y funciones miembros

Referencia a un objeto → **Pseudónimo del objeto**, y por lo tanto puede ser usada al lado izquierdo de una asignación

```
int & maximo(int &x, int &y);
maximo(x, y) = 0;
```

Mala Práctica de Programación

Hacer que una función miembro pública de una clase retorne una referencia a un dato miembro privado

```
class Racional {
       int numerador, denominador;
public:
       int & getDenominador(){
         return denominador;
       }
};
```

```
main() {
  Racional f(3, 4);
  f.getDenominador()=0;
  …
}
```

Ejemplo – ADT Persona

Atributos
- DNI
- Nombre
- Apellido
- Fecha de Nac.
- Profesión

Persona
-dni -nombre -apellido -fechaNacimiento -profesion
+listarDatosPersona() +calcularEdad(): int +obtenerProfesion(): string

Operaciones
- Calcular la edad de una persona
- Listar los atributos de una persona
- Obtener la profesión de una persona

19

Ejemplo – Clase Persona en C++

Persona.h

```
class Persona {
  // variables
   long dni;
   string nombre, apellido;
   fecha fechaNacimiento;
   string profesion;

  public:
  // métodos
     Persona (long, string, string, fecha, string);//constructor
     void listarDatosPersona();
     int calcularEdad();
     string obtenerProfesión() { return profesion; }
     ~Persona(); //destructor
  };  //Fin de la clase
```

Variables

Métodos

20

Ejemplo – Clase Persona en C++

```
Persona:: Persona (long doc, string nomb, string ape,
                                    fecha nac, string prof) {
    dni = doc;
    nombre = nomb;
    apellido = ape;
    fechaNacimiento = nac;
    profesion = prof;
}

void Persona:: listarDatosPersona(){
    cout<<"DNI = "<<dni<<endl;
    cout<<"Nombre = "<<nombre<<endl;
    cout<<"Apellido = "<<apellido<<endl;
    cout<<"Fecha de Nacimiento = "  <<fechaNacimiento.getDia()<<"/"
    <<fechaNacimiento.getMes()<<"/"<<fechaNacimiento.getAnio()<<endl;
    cout<<"Profesión = "<<profesion<<endl;
}
```

Persona.cpp

21

Ejemplo

```
class Persona {
  // Declaraciones de las variables de instancia o clase.
   long dni;
   string nombre, apellido;
   fecha fechaNacimiento;
   string profesion;
  public:
  // métodos
     Persona (long, string, string, fecha, string);
     //constructor
     void listarDatosPersona();
     int calcularEdad();
     string obtenerProfesión();
     ~Persona(); //destructor
  };  //Fin de la clase
```

La **implementación** de la clase es la visión interna del diseñador de la clase.

La **interfaz** de una clase proporciona el punto de vista del usuario de la clase.

22

Ejemplo

```
#include <iostream>
#include "Persona.h"
using namespace std;

int main() {
Fecha fNac(16,1,2014);
Persona p1(123456,"Santiago", "Acosta",fNac, "Estudiante");
p1.listarDatosPersona();
return 0;
}
```

Salida

```
DNI = 123456
Nombre = Santiago
Apellido = Acosta
Fecha de Nacimiento = 16/1/2014
Profesión = Estudiante
```

23