CARACTERÃ STICAS Y REGLAS DE LOS CONSTRUCTORES

- El nombre del constructor debe ser el mismo que el de su clase
- No debe tener ningun tipo de retorno ni siquiera void
- Una clase puede tener distintos tipos de constructores o ninguno, el compilador asigna uno automÃ¡ticamente a esa clase
- Un constructor predeterminado es aquel que no tiene ningÃºn tipo de parÃ¡metro o posee una lista de parÃ¡metros donde todos ellos son predeterminados

```
// constructor sin parÃ¡metros

class punto {

double x;

double y;

public:

punto();

……

};

// la misma clase con constructor con argumentos

// predeterminados

class punto {

double x;

double y;

public:

punto(double xval=0, double yval=0);

……

};
```

- El constructor de copia permite crear una instancia de clase usando una instancia existente

```
class punto {

double x;

double y;
```

public:

punto();

punto(double xval, double yval);

punto(const punto& pt);

……

};

Ejemplo de construcciones :

punto p1; // llama al constructor predeterminado

punto p2(1.1, 1.3); // llama al constructor con dos

// argumentos

punto p3(p2);

CLASES AMIGAS

En algunos casos es necesario que una funciÃ³n que no es miembro de una clase pueda acceder a los miembros privados de la clase en estos casos se denomina a la funciÃ³n amiga *(friend).*

vector multiplicar(const matriz& m, const vector &v)

{

vector r;

*for(int i=0; i<3; i++) {// r[i]=m[i]*v;*

r.elem(i)=0;

for (int j=0; j<3; j++)

*r.elem(i)+=m.elem(i,j)*v.elem(j);*

}

return r

}

class matriz;

class vector {

float v[4];

//…

friend vector multiplicar(const matriz&,const vector&);

};

class matriz {

vector v[4];

//…

friend vector multiplicar(const matriz&,const vector&);

};

La funciÃ³n *friend* no tiene nada particular excepto el derecho de acceder a la parte privada de una clase.

Una funciÃ³n *friend* se puede declarar tanto en la parte privada como en la pÃºblica de una clase.

La funciÃ³n amiga se declara explÃ–citamente en la clase en la cual es amiga â ´ forma parte de la interfase de esa clase tanto como una funciÃ³n miembro. Se debe tener en cuenta que en la funciÃ³n *friend* no se puede emplear el puntero *this*, por lo tanto se debe referenciar explÃ–citamente el miembro del objeto con el que se estÃ¡ trabajando.

vector multiplicar(const matriz& m, const vector &v)

{

vector r;

for (int i=0; i<3; i++) {

r.v[i]=0;

for (int j=0; j<3; j++)

*r.v[i]+=m.v[i][j]*v.v[j];*

}

return r

}

Una funciÃ³n miembro de una clase puede ser amiga de otra

class X {

//…

void f();

};

class Y {

//…

friend void X::f();// la funciÃ³n miembro f de X es

// amiga de Y

};

Para declarar que todas las funciones miembros de una clase son amigas de otras tenemos :

class X {

…

friend class Y; // todas las funciones miembros de Y

… // son amigas de X

};

Universidad TecnolÃ³gica Nacional - Santa Fe - Departamento Sistemas -

Curso : Desarrollos de ProgramaciÃ³n en C++