CONSTRUCTOR Y DESTRUCTOR

- Sobrecarga de funciones
- Constructor, tipos
- Destructor

Constructor y Destructor

Sobrecarga de funciones (I)

- Definición de funciones con el mismo nombre y distinto número, orden y/o tipo de parámetros.

Ejemplo 3.1

```
1  #include <iostream>
2
3  using namespace std;
4
5  int
6  main(void)
7  {
8     cout << min(1, 2) << endl;
9
10    cout << min(1L, 2L) << endl;
11
12    cout << min(1.1, 2.2) << endl;
13
14    return 0;
15 }
```

Constructor y Destructor

Sobrecarga de funciones (II)

- Sobrecarga de cualquier función, incluso las de la librería estándar: (ejemplo 3.2)

```
1  #include <iostream>
2
3  using namespace std;
4
5
6  int
7  min(int a, int b, int c)
8  {
9    return min(min(a, b), c);
10 }
11
12 long
13 min(long a, long b, long c)
14 {
15   return min(min(a, b), c);
16 }
```

```
18 double
19 min(double a, double b, double c)
20 {
21   return min(min(a, b), c);
22 }
23
24 int
25 main(void)
26 {
27   cout << min(1, 2, 3) << endl;
28
29   cout << min(1L, 2L, 3L) << endl;
30
31   cout << min(1.1, 2.2, 3.3) << endl;
32
33   return 0;
34 }
```

Constructor y Destructor

Sobrecarga de funciones (III)

- Error típico: funciones con mismo nombre, mismos parámetros y distinto valor de retorno. (Ejemplo 3.3)

```
1  #include <iostream>
2
3  using namespace std;
4
5  int
6  calcula(int a, float b) {
7    return (int) (a * b);
8  }
9
10 float
11 calcula(int a, float b) {
12   return (float) (a * b);
13 }
14
```

```
15 int
16 main(void)
17 {
18   cout << calcula(1, 2.2) << endl;
19
20   cout << calcula(1, 2.2) << endl;
21
22   return 0;
23 }
```

Salida ejemplo 3.3

```
1  ejem7.cc: In function 'float calcula(int, float)':
2  ejem7.cc:9: new declaration 'float calcula(int, float)'
3  ejem7.cc:5: ambiguates old declaration 'int calcula(int, float)'
```

Constructor y Destructor

Constructor.

- **Definición**. Función miembro especial de una clase invocada al crear un objeto de esa clase.
- **Objetivo**: iniciar todo lo necesario para el funcionamiento correcto del objeto.
- Declaración:
  - Mismo nombre que la clase.
  - No devuelven ningún tipo de dato.
  - Suelen estar en la parte pública.
  - Pueden haber varios constructores.
  - Si no se define alguno → El compilador da uno por defecto.

Constructor y Destructor

Constructor por defecto.

- Constructor sin parámetros.

Ejemplo 3.4

```
1  class TCoordenada {
2    public:
3      TCoordenada();
4
5      void setX(int);
6      void setY(int);
7      void setZ(int);
8
9      int getX(void);
10     int getY(void);
11     int getZ(void);
12
13     void Imprimir(void);
14
15   private:
16     int x, y, z;
17 };
```

Ejemplo 3.6

```
1  #include <iostream>
2
3  using namespace std;
4
5  #include "tcoordenada.h"
6
7  int main(void) {
8    int i;
9    TCoordenada p1;
10
11   p1.Imprimir();
12   cout << endl;
13
14   return 0;
15 }
```

Ejemplo 3.5

```
1  TCoordenada::TCoordenada() {
2    x = y = z = 0;
3  }
```

(0, 0, 0)

Constructor y Destructor

Otros constructores.

- Cuantos queramos con parámetros.

Ejemplo 3.7

```
1  class TCoordenada {
2    public:
3      TCoordenada();
4      TCoordenada(int, int, int);
5
6      void setX(int);
7      void setY(int);
8      void setZ(int);
9
10     int getX(void);
11     int getY(void);
12     int getZ(void);
13
14     void Imprimir(void);
15
16   private:
17     int x, y, z;
18 };
```

Ejemplo 3.8

```
1 TCoordenada::TCoordenada(int a, int b, int c) {
2   x = a;
3   y = b;
4   z = c;
5 }
```

```
1 // Constructor por defecto
2 TCoordenada p1;
3
4 // Constructor sobrecargado a partir de tres enteros
5 TCoordenada p2(10, 20, 30);
```

Constructor y Destructor

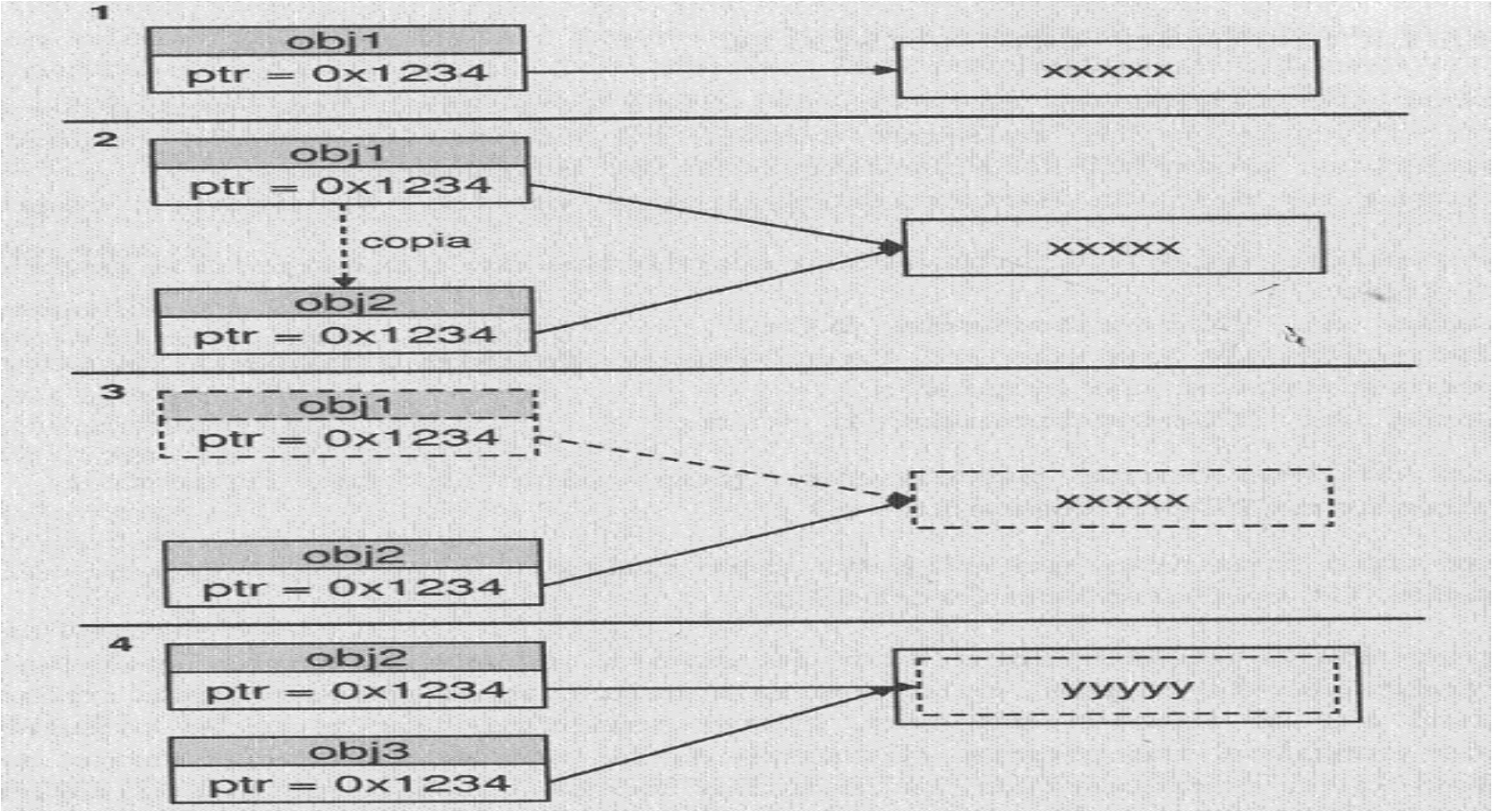
Constructor de copia (I)

- **Definición:** Constructor especial que crea un objeto a partir de otro de la misma clase.
- **Invocaciones:**
  - De forma explícita en la declaración.
  - En el paso por valor de un objeto en una función.
  - En la devolución de un objeto por valor en una función.
- **Si no se declara:**
  - el compilador proporciona uno: copia bit a bit.
  - Posibles problemas si hay uso de memoria dinámica.

Constructor y Destructor

Constructor de copia (II)

- Ejemplo de error:



Constructor y Destructor

Constructor de copia (III)

- Consejo: Crear siempre un constructor de copia.

Ejemplo 3.9

```
1  class TCoordenada {
2    public:
3      TCoordenada();
4      TCoordenada(int, int, int);
5      TCoordenada(const TCoordenada &);
6
7      void setX(int);
8      void setY(int);
9      void setZ(int);
10
11     int getX(void);
12     int getY(void);
13     int getZ(void);
14
15     void Imprimir(void);
16
17   private:
18     int x, y, z;
19 };
```

Ejemplo 3.10

```
1 TCoordenada::TCoordenada(const TCoordenada & c) {
2   x = c.x;
3   y = c.y;
4   z = c.z;
5 }
```

Ejemplo 3.11

```
1  #include <iostream>
2
3  using namespace std;
4
5  #include "tcoordenada.h"
6
7  int
8  main(void)
9  {
10   int i;
11   TCoordenada p1;
12   TCoordenada p2(10, 20, 30);
13   TCoordenada p3(p2);
14
15   p1.Imprimir();
16   cout << endl;
17
18   p1.setX(1);
19   p1.setY(2);
20   p1.setZ(3);
21
22   p1.Imprimir();   ---> (1, 2, 3)
23   cout << endl;
24
25   p2.Imprimir();   ---> (10, 20, 30)
26   cout << endl;
27
28   p3.Imprimir();   ---> (10, 20, 30)
29   cout << endl;
30
31   return 0;
32 }
```

Constructor y Destructor

Constructor de copia (IV)

- El objeto que se pasa por parámetro, ***siempre por referencia***. En caso contrario: error del compilador.

Salida ejemplo 3.13

```
1  In file included from tcoordenada.cc:1:
2  tcoordenada.h:5: invalid constructor; you probably meant 'TCoordenada
3     (const TCoordenada&)'
4  tcoordenada.cc:13: prototype for 'TCoordenada::TCoordenada(TCoordenada)'
5     does not match any in class 'TCoordenada'
6  tcoordenada.h:1: candidates are: TCoordenada::TCoordenada(const
7                                    TCoordenada&)
8  tcoordenada.cc:7:                TCoordenada::TCoordenada(int, int, int)
9  tcoordenada.cc:3:                TCoordenada::TCoordenada()
10 tcoordenada.cc:13: invalid constructor; you probably meant 'TCoordenada
11    (const TCoordenada&)'
12 tcoordenada.cc:13: syntax error before '{' token
13 tcoordenada.cc:15: ISO C++ forbids declaration of 'y' with no type
14 tcoordenada.cc:15: 'c' was not declared in this scope
15 tcoordenada.cc:16: ISO C++ forbids declaration of 'z' with no type
16 tcoordenada.cc:16: 'c' was not declared in this scope
17 tcoordenada.cc:17: syntax error before '}' token
```

Constructor y Destructor

¿Un constructor en la parte privada?

- Sí, si no queremos que sea invocado públicamente.

Ejemplo 3.14

```
1  class TCoordenada {
2    public:
3      TCoordenada();
4      TCoordenada(int, int, int);
5
6      void setX(int);
7      void setY(int);
8      void setZ(int);
9
10     int getX(void);
11     int getY(void);
12     int getZ(void);
13
14     void Imprimir(void);
15
16   private:
17     TCoordenada(const TCoordenada &);
18
19     int x, y, z;
20 };
```

Salida ejemplo 3.14

```
1  tcoordenada.h: In function 'int main()':
2  tcoordenada.h:17: 'TCoordenada::TCoordenada(const TCoordenada&)' is
3     private
4  main.cc:13: within this context
```

Constructor y Destructor

Destructor (I)

- **Definición:** Función miembro de una clase que se invoca automáticamente cada vez que se destruya y libere el objeto.
- **Declaración:** *~nombreclase()*
- Solo un destructor
- Si no hay: el compilador proporciona uno que no hace nada -> Problema con la memoria dinámica.

Constructor y Destructor

Destructor (II)

Ejemplo 3.15

```
1  class TCoordenada {
2    public:
3      TCoordenada();
4      TCoordenada(int, int, int);
5      TCoordenada(const TCoordenada &);
6      ~TCoordenada();
7
8      void setX(int);
9      void setY(int);
10     void setZ(int);
11
12     int getX(void);
13     int getY(void);
14     int getZ(void);
15
16     void Imprimir(void);
17
18   private:
19     int x, y, z;
20 };
```

Ejemplo 3.16

```
1  TCoordenada::~TCoordenada() {
2    x = y = z = 0;
3  }
```

Constructor y Destructor

Forma canónica de una clase.

- Importante definir una clase en su forma canónica.
- Ha de contener como mínimo:
  - Constructor por defecto
  - Constructor de copia
  - Destructor
  - Sobrecarga del operador asignación (=)