I. Programación C++

Estructura de Datos

Marzo, 2019

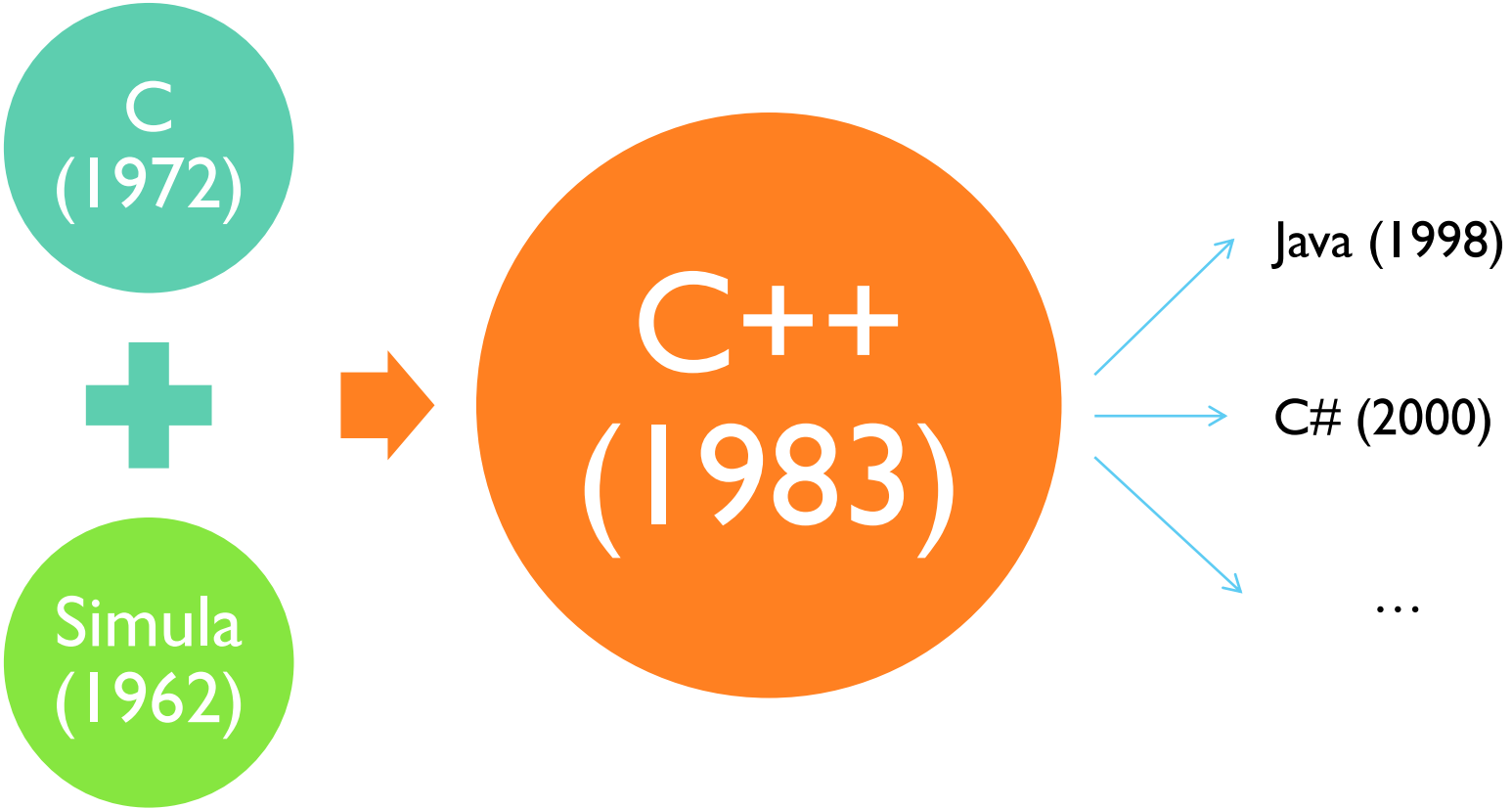
Contenidos

a) Aspectos generales de C++
b) Modelo de programación en C
c) Abstracción
d) Programación a gran escala
e) Biblioteca estándar C++

Contenidos

a) **Aspectos generales de C++**
b) Modelo de programación en C
c) Abstracción
d) Programación a gran escala
e) Biblioteca estándar C++

a) Aspectos generales de C++



a) Aspectos generales de C++

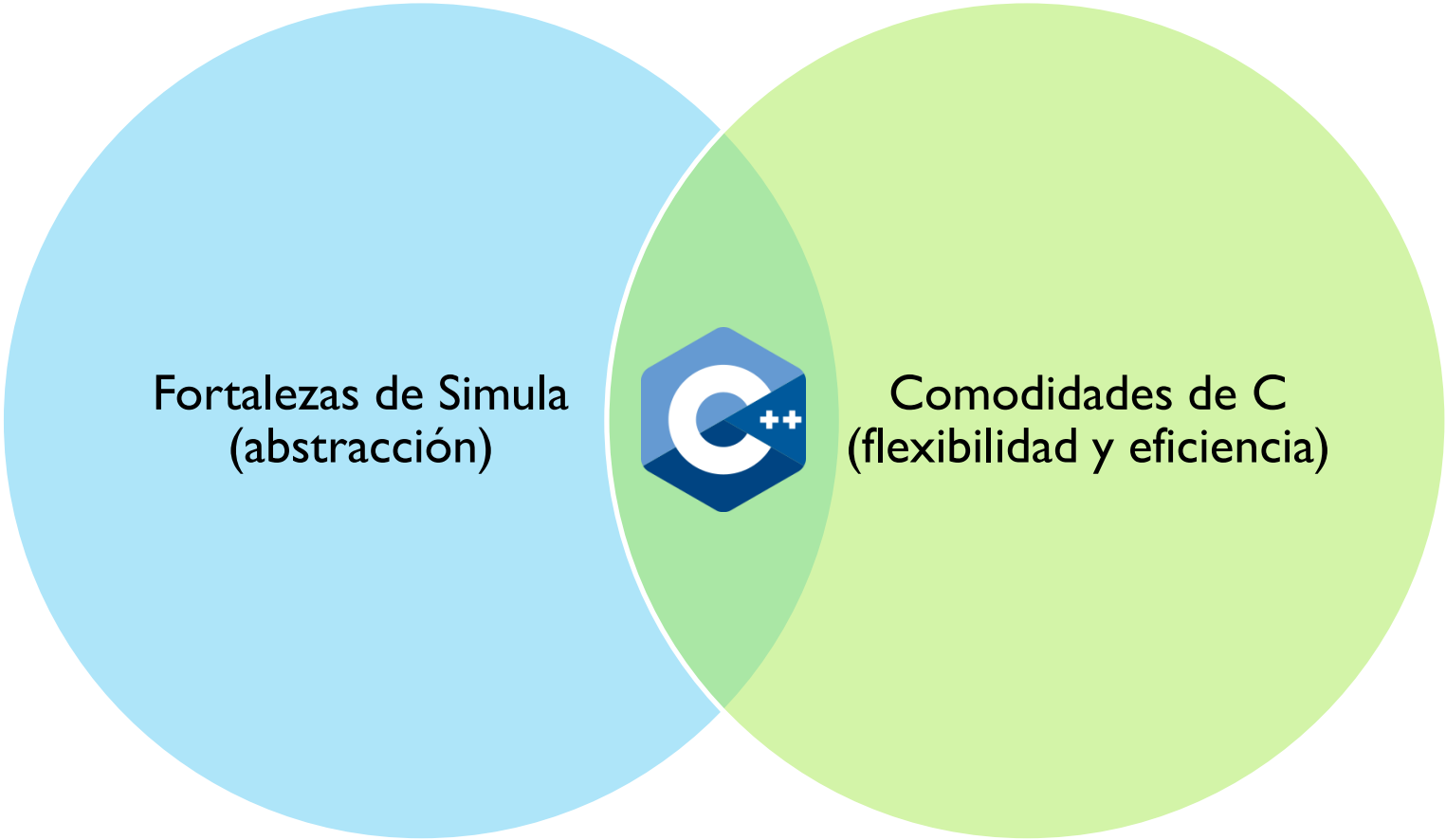
- Es un LP (**L**enguaje de **P**rogramación) de **propósito general**, orientado a:
  - Eficiencia.
  - Abstracción (nuevos tipos de datos).
  - POO (Programación Orientada a Objetos).
  - Programación genérica.

a) Aspectos generales de C++

- Es un LP de **alto nivel** (más cercano a una visión humana) que manipula bastante directamente los recursos de hardware (→ eficiente)

- 1998: Estándar Internacional (ISO) (ANSI)
  - Bien definido.
  - Completa documentación de su diseño y evolución.

a) Aspectos generales de C++



a) Aspectos generales de C++

- Los mecanismos de abstracción de C++ fueron diseñados específicamente para ser aplicados a tareas de programación que demanden el más alto nivel de eficiencia y flexibilidad.

a) Aspectos generales de C++

Objetivos de diseño

- Hacer la programación más disfrutable.
- Ser mejor que C.

Reglas generales de C++

- Evolución guiada por problemas reales.
- No buscar la perfección (porque es un LP de propósito general).
- Ser útil AHORA.
- Programadores de C++:
  - Darles una ruta de transición.
  - Darles soporte exhaustivo.
  - No forzar a las personas.

a) Aspectos generales de C++

Reglas de soporte al diseño

- Facilidad para organizar programas.
- Decir lo que quieres decir.

Reglas de lenguaje técnico

- Buen soporte para tipos nativos y creados por el usuario.
- Importa la sintaxis.
- Evitar dependencias de orden
  (algunas no se pueden evitar, como el uso de variables sin declarar).

a) Aspectos generales de C++

Reglas de soporte de programación de bajo nivel

- Compatible con C.
- No pagas lo que no usas
  (una nueva característica no puede disminuir la eficiencia de programar sin esa característica).

Contenidos

a) Aspectos generales de C++

b) Modelo de programación en C

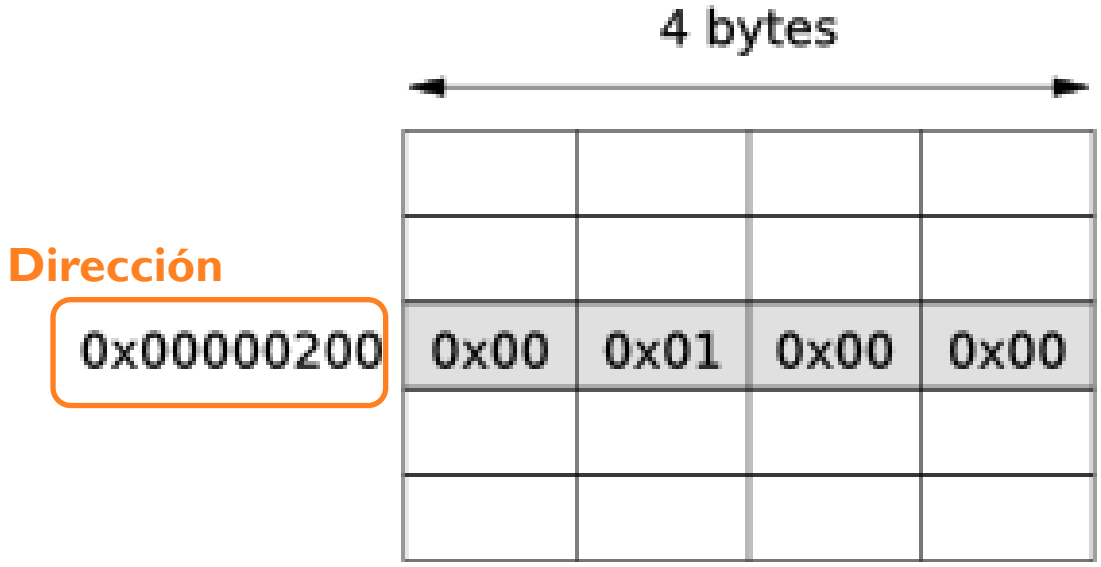
c) Abstracción

d) Programación a gran escala

e) Biblioteca estándar C++

b) Modelo de programación en C

- **Memoria**: secuencia de palabras o bytes, indexado por enteros llamados direcciones.



b) Modelo de programación en C

- C es el mejor LP que entrega un modelo de programación más cercano al de la máquina (mapeo más directo entre las instrucciones en el programa y en el hardware).

- Por lo tanto, es relativamente fácil de usar en áreas donde se tiene el beneficio de conocer algo de la máquina real.

b) Modelo de programación en C

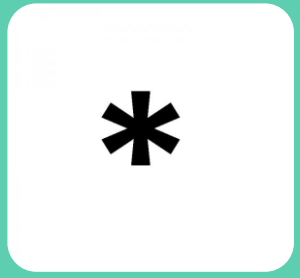
- **Vector:** es una secuencia de espacios de memoria.

```
int v[10];
v[3] = 1;
int x = v[3];
```

¿Valor de "x"? → **R: 1**

b) Modelo de programación en C

- **Punteros**:

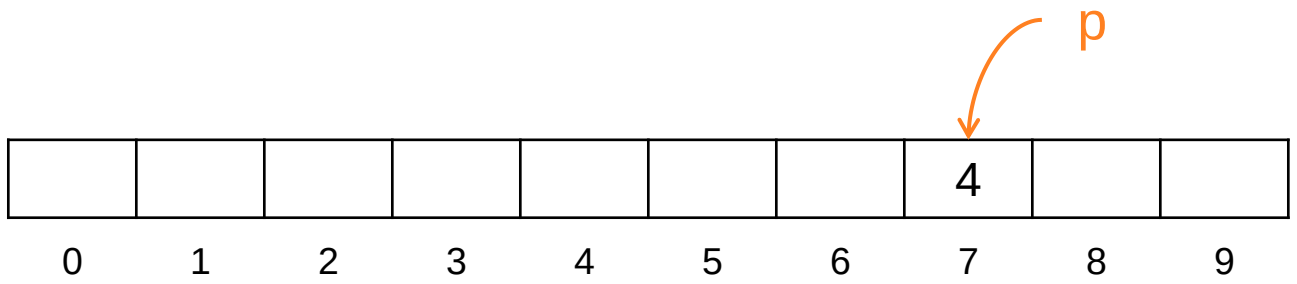


Puntero.
Variable que puede contener una dirección a una posición de memoria.

Dirección en la memoria.
Referencia.

b) Modelo de programación en C

```
int *p; //p es un puntero a un int
p = &v[7]; //p referencia a la dirección de v[7]
*p = 4; //el contenido en la dirección donde apunta p es 4
int y = *p; //y toma el valor de v[7] a través de p
```



Por lo tanto, *p = v[7]

¿Valor de "y"? → **R: 4**

b) Modelo de programación en C

- C++ adoptó esto de C.

- **int* p**: p es un puntero preparado para contener una dirección de una celda de memoria que almacena un entero.

- ***p**: contenido en la dirección del puntero "p".

b) Modelo de programación en C

- Almacenamiento:
  - En C y C++ hay 3 modos de usar la memoria.

Estática

- Se crea una vez y muere al acabar el programa.
- Su dirección (&) no cambia en toda la ejecución.

Automática

- Parámetros de funciones y variables locales.
- Su espacio en memoria se reserva al iniciar un bloque y se libera al salir de él.

Almacenamiento libre

- Objetos con los que hay que llamar explícitamente los comandos ***new*** y ***delete*** para reservar y liberar memoria respectivamente.
- MUY IMPORTANTE el ***delete***.

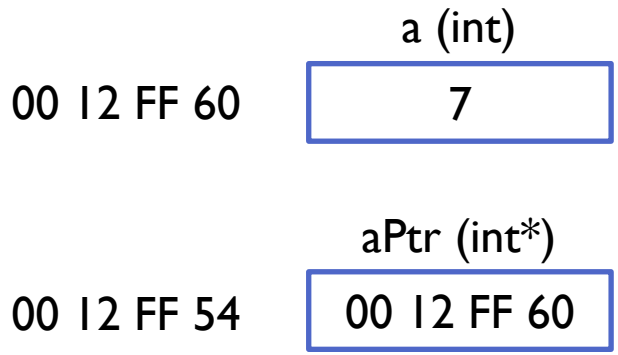
b) Modelo de programación en C

- Se pueden usar contenedores para manejar el almacenamiento libre. Por ejemplo: *String* maneja una secuencia de *chars* en almacenamiento libre.

- En C++ existen los "recolectores de basura" (cuidado).

Ejercicio 1

```
int main{
    int a;
    int* aPtr;
    a = 7;
    aPtr = &a;
}
```



Imprimir:

a) a → 7
b) &a → 00 12 FF 60
c) aPtr → 00 12 FF 60
d) *aPtr → 7
e) &*aPtr → 00 12 FF 60
f) &aPtr → 00 12 FF 54
g) *&aPtr → 00 12 FF 60

Ejercicio 2

```
int main{
    int vect[8];
    for (int i=0; i<8; i++)
        vect[i] = 0;

    int* a = &vect[4];
    *(a+2) = 2;
    a = &vect[*(a+1)];
    *a = 1;
    *(&vect[2] + *(a+6)) = 3;
}
```

¿Cómo queda el vector?

1	0	0	0	3	0	2	0
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]

Ejercicio 3

```
int main{
    int vect[8];
    for (int i=0; i<8; i++)
        vect[i] = 0;

    int* a = &vect[6];
    *(a-2) = 4;
    *(a - *(a-2)) = 5;
    a = &vect[vect[2]-vect[4]];
    *a = 3 + *(a+1);
}
```

¿Cómo queda el vector?

0	8	5	0	4	0	0	0
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]

b) Modelo de programación en C

- **Compile, Link, and Execute**
  - ¿Qué son los archivos .c y .h? ¿Cómo se relacionan entre sí?

- **Type checking**
  - ¿Cuándo se efectúa la revisión de consistencia de tipos?
  - ¿Qué tipos de datos causan error de conversión entre si?

Contenidos

a) Aspectos generales de C++
b) Modelo de programación en C
c) Abstracción
d) Programación a gran escala
e) Biblioteca estándar C++

c) Abstracción

1	• Tipos Concretos
2	• Tipos Abstractos
3	• POO
4	• Programación Genérica
5	• Contenedores

1. Tipos Concretos:

- ¿Qué son?
- ¿Quién los define?
- ¿Qué es una clase? ¿Cuál es la importancia de los constructores?
- ¿Para qué sirve la declaración "friend"?

2. Tipos Abstractos:

- ¿Para qué sirven las clases abstractas?
- ¿Qué significa que una función sea virtual? ¿Qué implica el "=0"?
- ¿Qué relación hay entre clases base y derivadas?
- ¿Cuál es la utilidad de los destructores?

3. POO:

- ¿Qué es la POO?
- ¿Cómo funciona la herencia de clases?
- ¿Qué es RTTI?
- ¿Qué tipo de casteos se pueden hacer?

4. Programación Genérica:

- ¿Para qué sirve la parametrización?
- ¿Qué es un template?
- ¿Qué características debe tener un template?

5. Contenedores:

- ¿Cómo se utilizan los templates?
- ¿Qué es un contenedor?
- ¿En qué condiciones un algoritmo puede operar sobre distintos tipos de elementos?

Contenidos

a) Aspectos generales de C++
b) Modelo de programación en C
c) Abstracción
d) Programación a gran escala
e) Biblioteca estándar C++

d) Programación a gran escala

- ¿Qué es la programación a gran escala?
- ¿Cuál es la utilidad de los namespace?
- ¿Para qué se utilizan las excepciones?
- ¿Qué clase de errores se deben manejar con el lanzamiento de excepciones?
- ¿Cómo funcionan los bloques try/catch?

Contenidos

a) Aspectos generales de C++
b) Modelo de programación en C
c) Abstracción
d) Programación a gran escala
e) Biblioteca estándar C++

e) Biblioteca estándar C++

- ¿Qué es la STL?
- ¿Qué es el flujo I/O?
- ¿Qué provee la librería estándar?
- ¿Qué son los contenedores?

¿Consultas?