Arreglos y Apuntadores

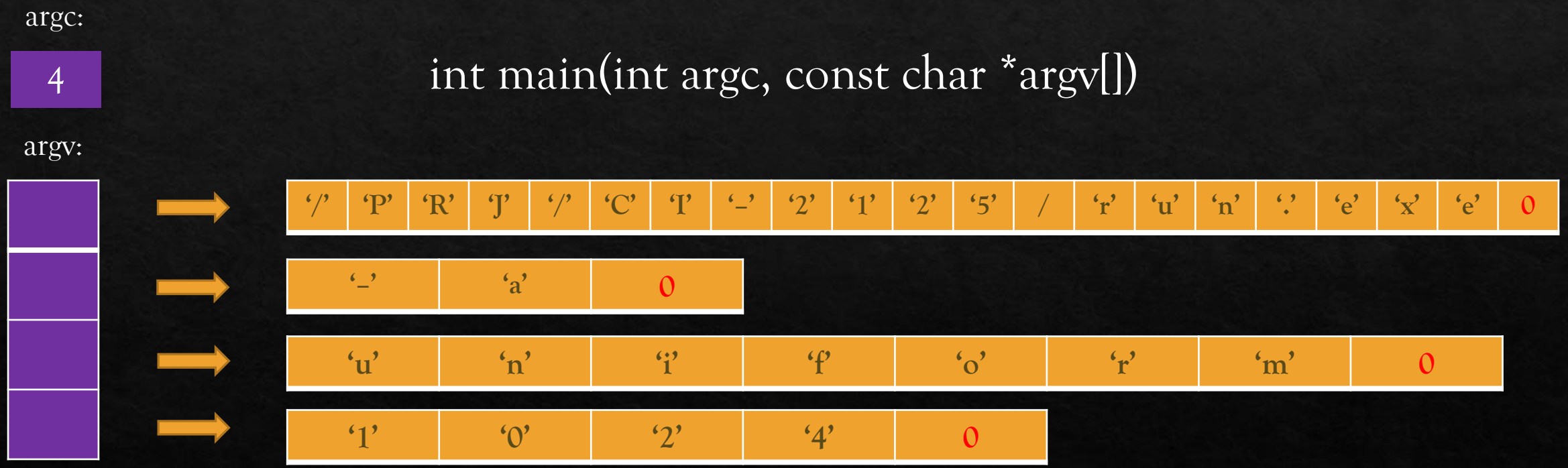
De nivel principiante ...
... a nivel intermedio

Arreglos

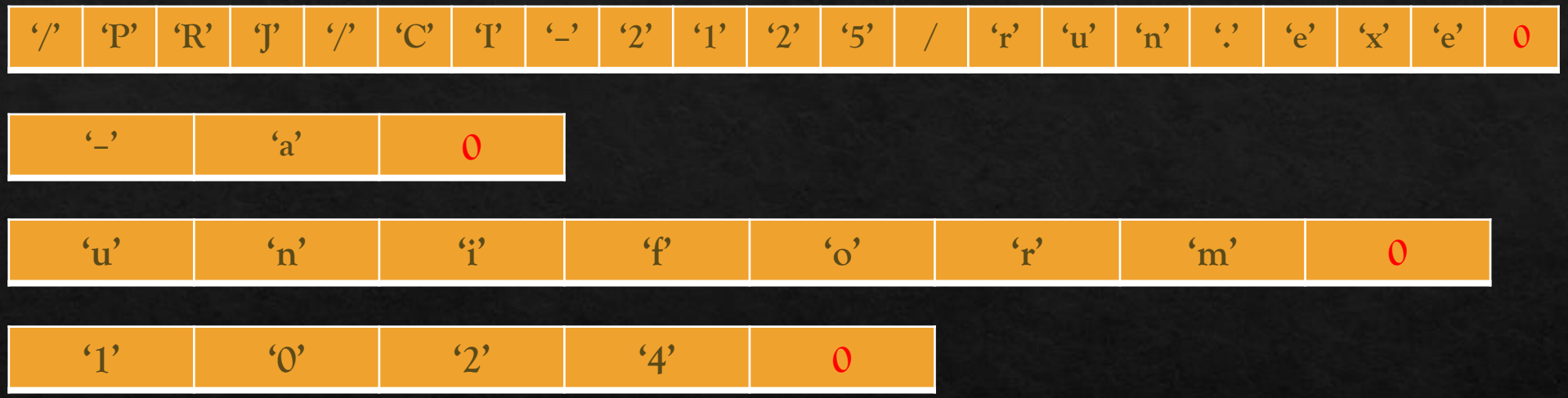
Arreglos

./run -a uniform 1024

int main(int argc, const char *argv[])



Cadenas de Caracteres – “Strings”



strlen(const char *s);
strcmp(const char *s1, const char *s2);
strcpy(char *dst, const char *src);
strcat(char *dst, const char *src);

strcpy & strcat

char *abre = “abre”;

char *latas = “latas”;

strcpy(abre, latas) ➔ ?

strcpy(latas, abre) ➔ ?

strcat(abre, latas) ➔ ?

strcat(latas, abre) ➔ ?

Mapa de la Memoria – Memory Layout

0000 < C	Sistema de Operación	Operating System	
C < D	Código del Usuario	User Code	Código de máquina (ejecutable), no modificable Immutable machine language code
D < H	Data Estática	Static Data	Estructuras de datos de tamaño fijo Fixed size data
H < F	Data Dinámica	Runtime Heap	Donde se crean estructuras de datos dinámicas Where dynamic data structures are allocated
…		… (free) …	…
…	…	… (free) …	…
…	…	… (free) …	…
…		… (free) …	…
F < S	Pila de Ejecución	Runtime Stack	Hace posible la magia de la recursión Where the recursion magic is enabled
R < Ω	?	Under Stack OS ++	In the Apple][and many other late 70s computers Ω was 0xFFFF

Pero ... es una simplificación

- El mapa anterior es una simplificación muy burda de la realidad actual
  - Debido al multiprocesamiento y le implementación de “memoria virtual”
- Sin embargo, dicha simplificación tiene méritos importantes:
  1. Corresponde al modelo típico de sistemas mono proceso sin memoria virtual
  2. Es consistente con la ***ilusión*** creada (para cada proceso) en un sistema con memoria virtual
  3. Provee un modelo mental ***adecuado*** para la programación de aplicaciones

- Si el tiempo y los estudiantes lo permiten:
  1. Podemos mostrar un modelo mucho más cercano a la realidad
  2. Pero no vamos a hacer ningún laboratorio donde la diferencia es relevante
  3. Si esto les apasiona, consideren una maestría en una de las áreas de computación

Pila de Ejecución

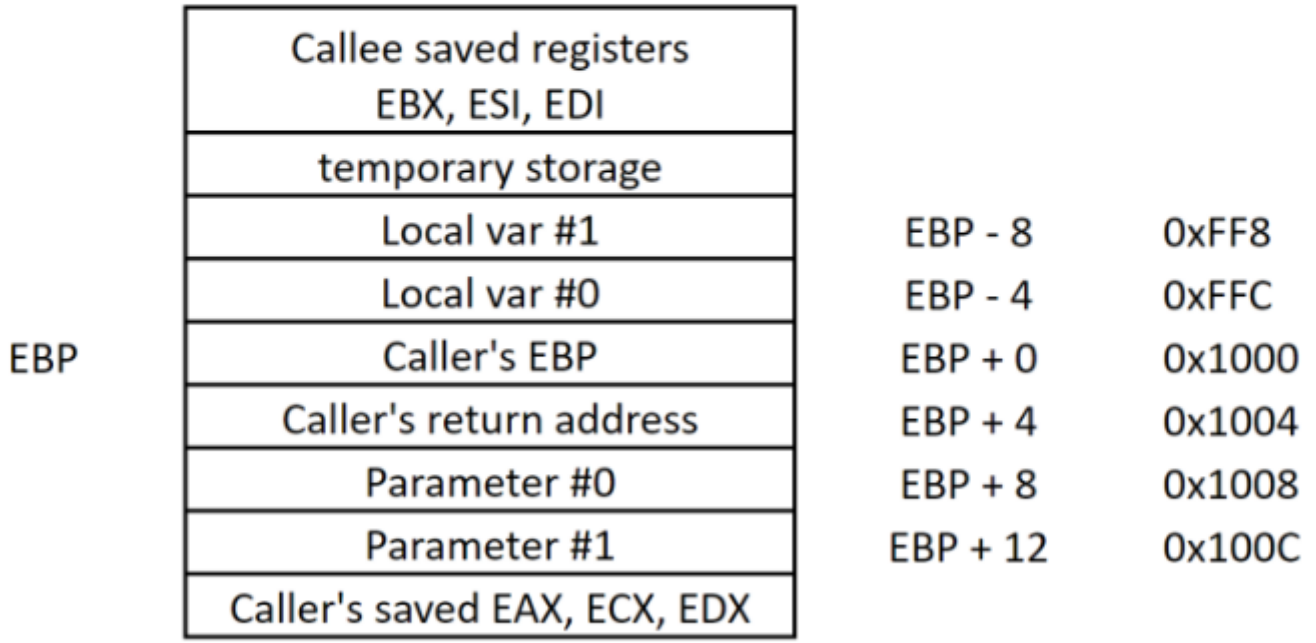
```
int suma(int x, int y, int z) {
  int t = x + y;
  return t + z;
}

int main() {
  …
  int w = 8;
  suma(32, w, 2);
  …
}
```

		t
		Cadena estática Cadena dinámica Dirección de retorno
	32	x
	8	y
	2	z

Pila de Ejecución

Now, let us analyze a stack frame on a typical x86 for a C function as follows:

```
1  int function(int param0, int param1)
2  {
3      int var0;
4      int var1;
5      // do some computation here
6      return var0;
7  }
```



EBP: Base pointer register: Indicating the origin of current stack frame. Most of other locations are accessed using offsets to EBP register.

Arreglos

```
__ f(int n, int *x) {
  ... x[i]   ⇔ *(x + sizeof(int) * i)
}


__ m(...) {
  ...
  int a = { 4, 1, 7, 3, 0, 6, 2, 5 };
  ...
  ... f(8, a) ...
  ...
}
```

Arreglos

```
__ f(int n, int x[]) {
  ... g(n, x);
}


__ m(...) {
  ...
  int a = { 4, 1, 7, 3, 0, 6, 2, 5 };
  ...
  ... f(8, a) ...
  ...
}
```

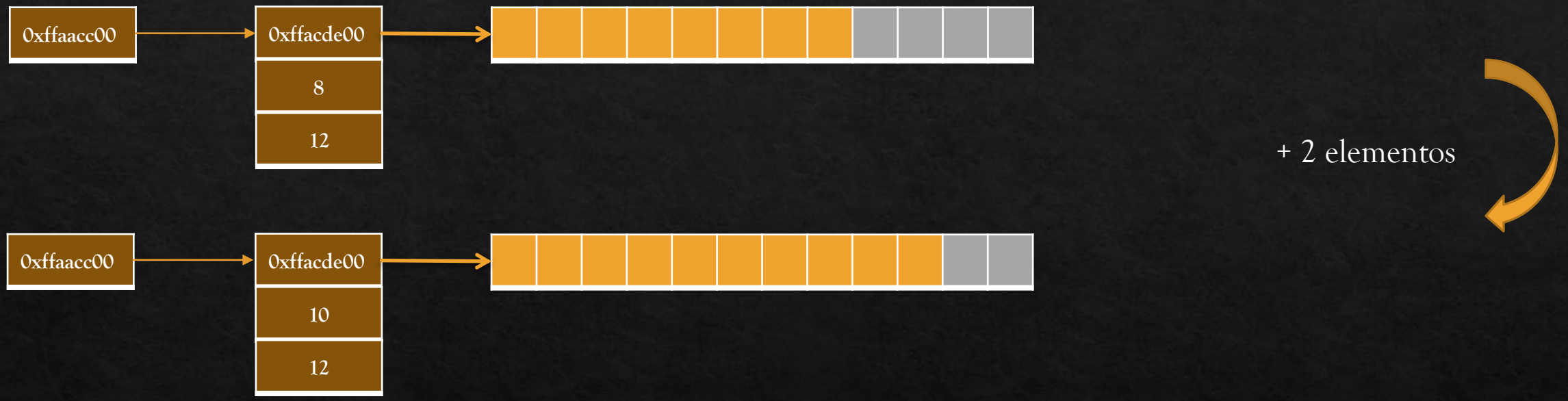
¿Que pasa si f es una función que busca la posición del mínimo en x?
¿Que pasa si f es una función que ordena x?
¿Que pasa si f llama a g pasando x como argumento?

Arreglos Dinámicos

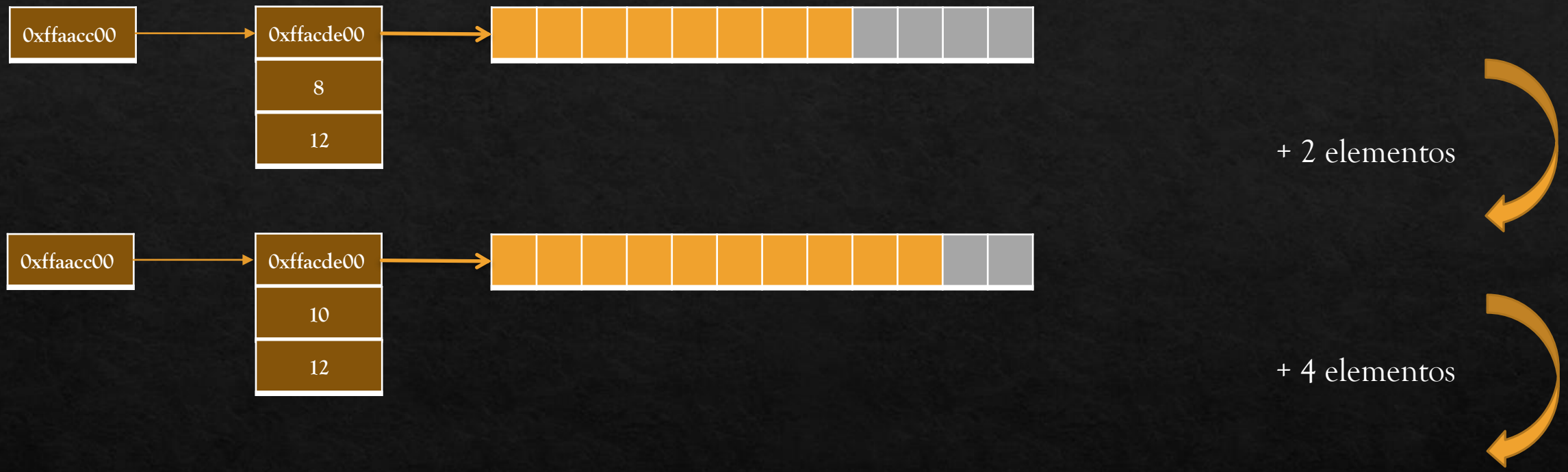
Arreglos Dinámicos

+ 2 elementos

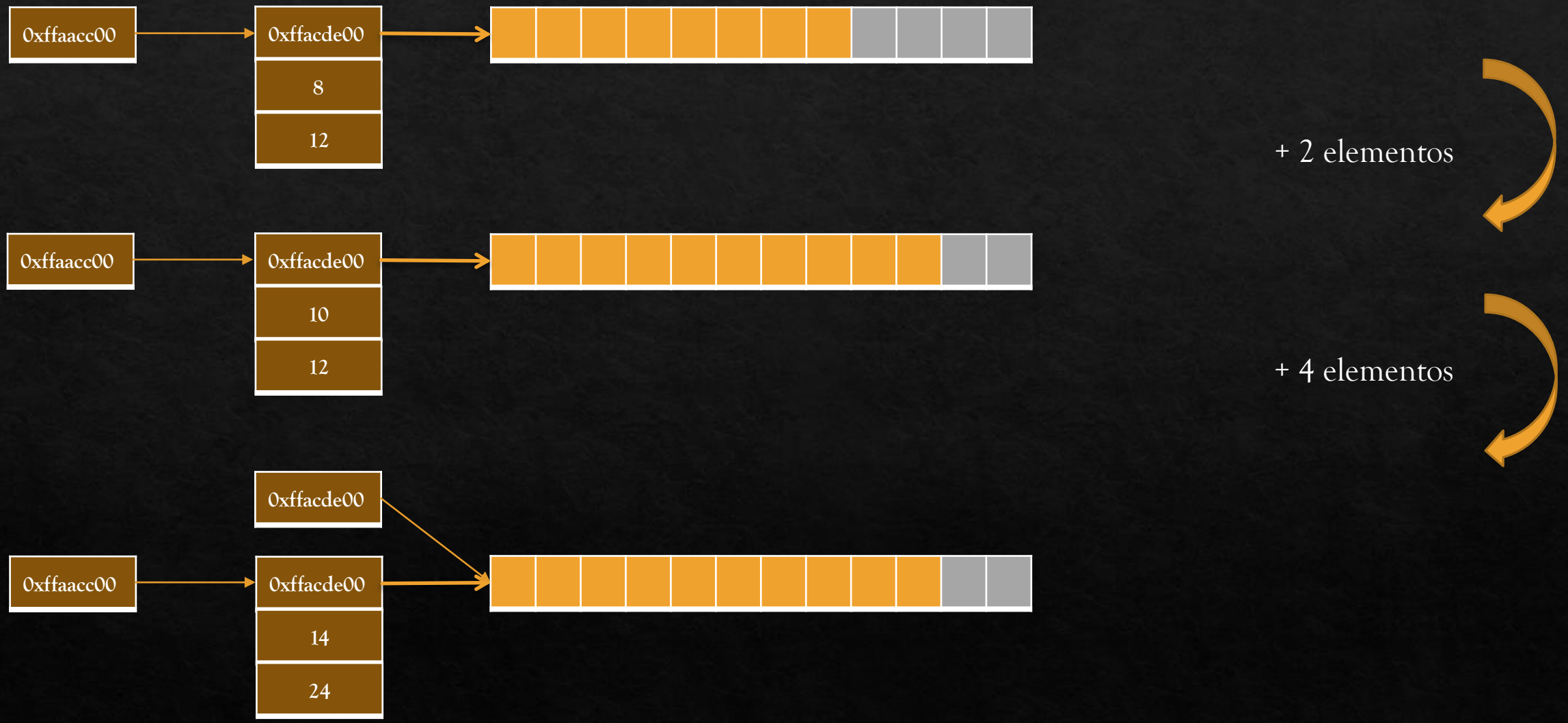
Arreglos Dinámicos



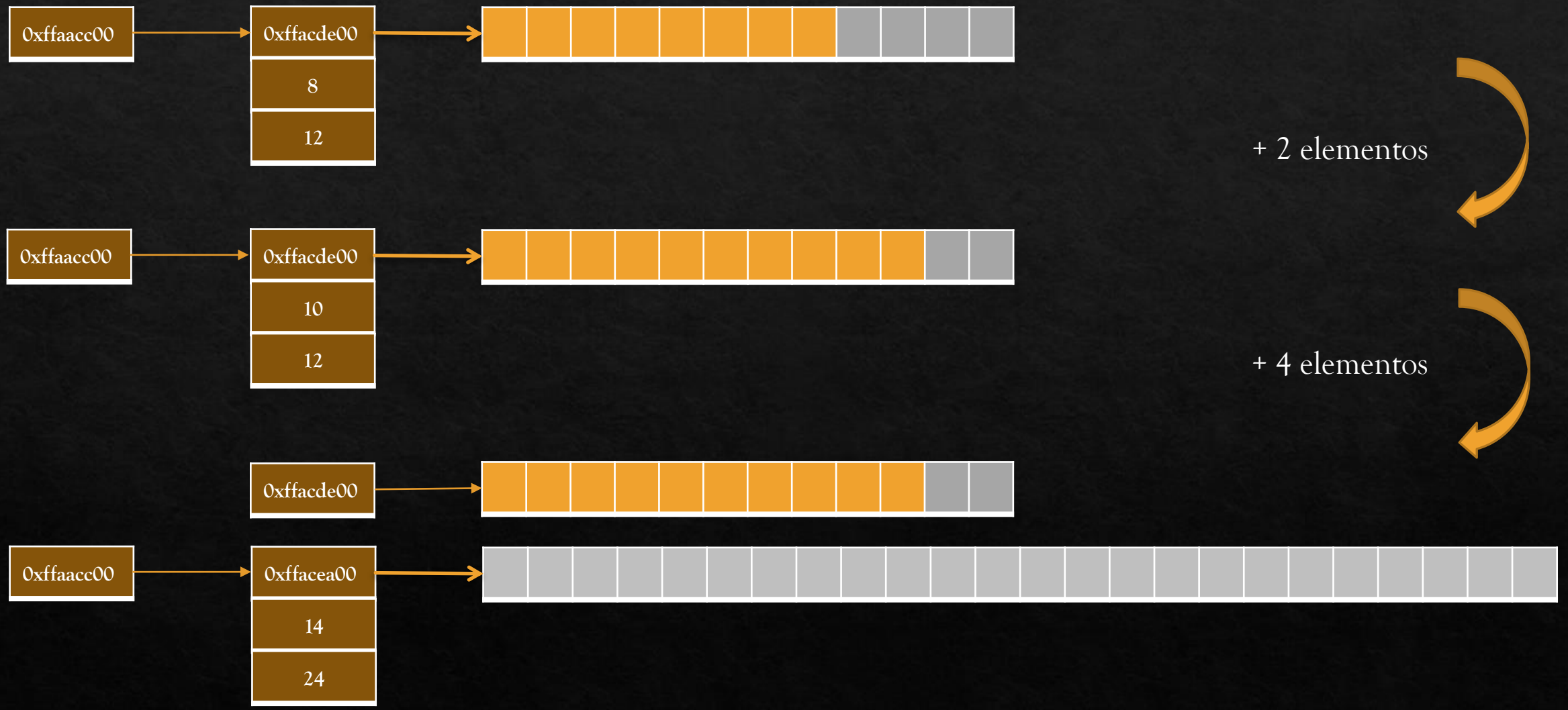
Arreglos Dinámicos



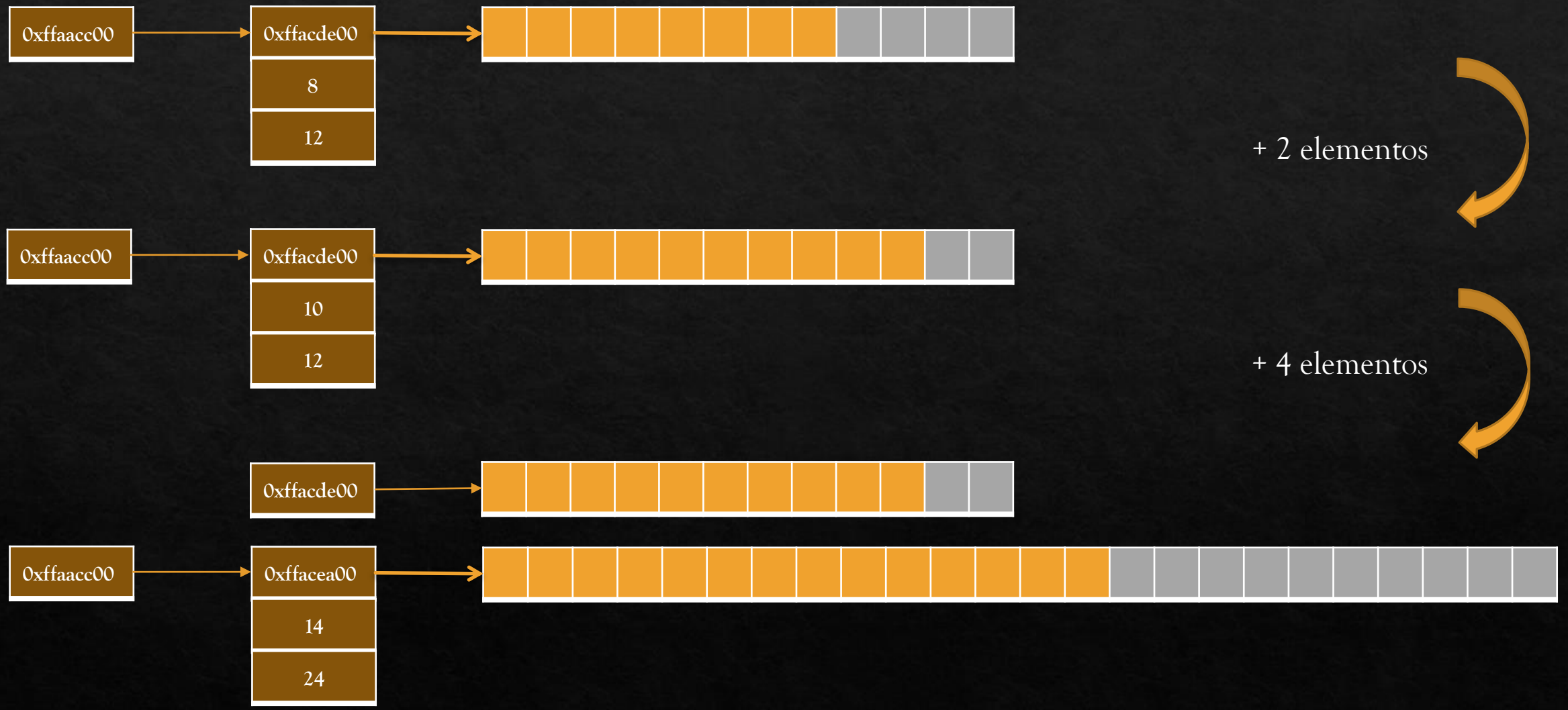
Arreglos Dinámicos



Arreglos Dinámicos

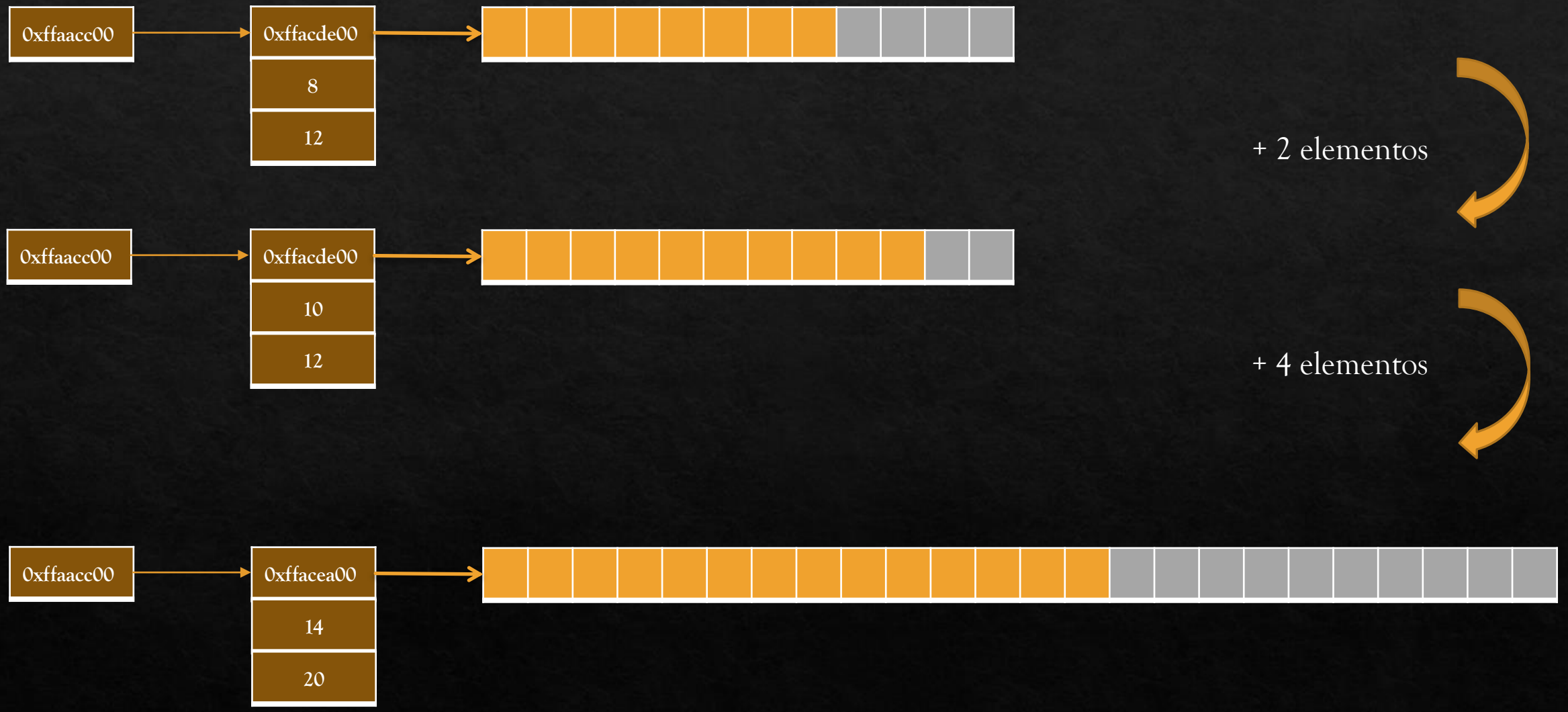


Arreglos Dinámicos

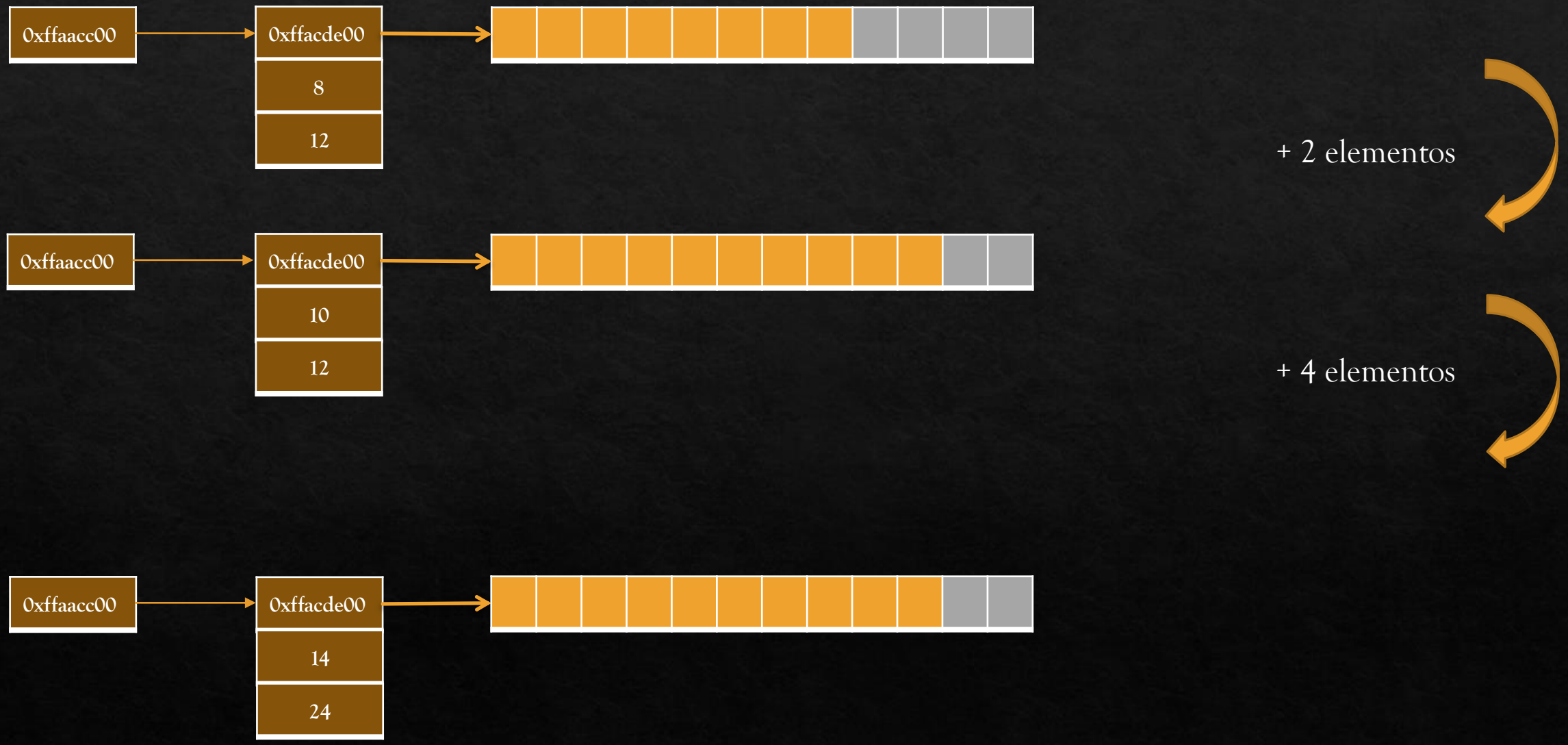


Arreglos Dinámicos

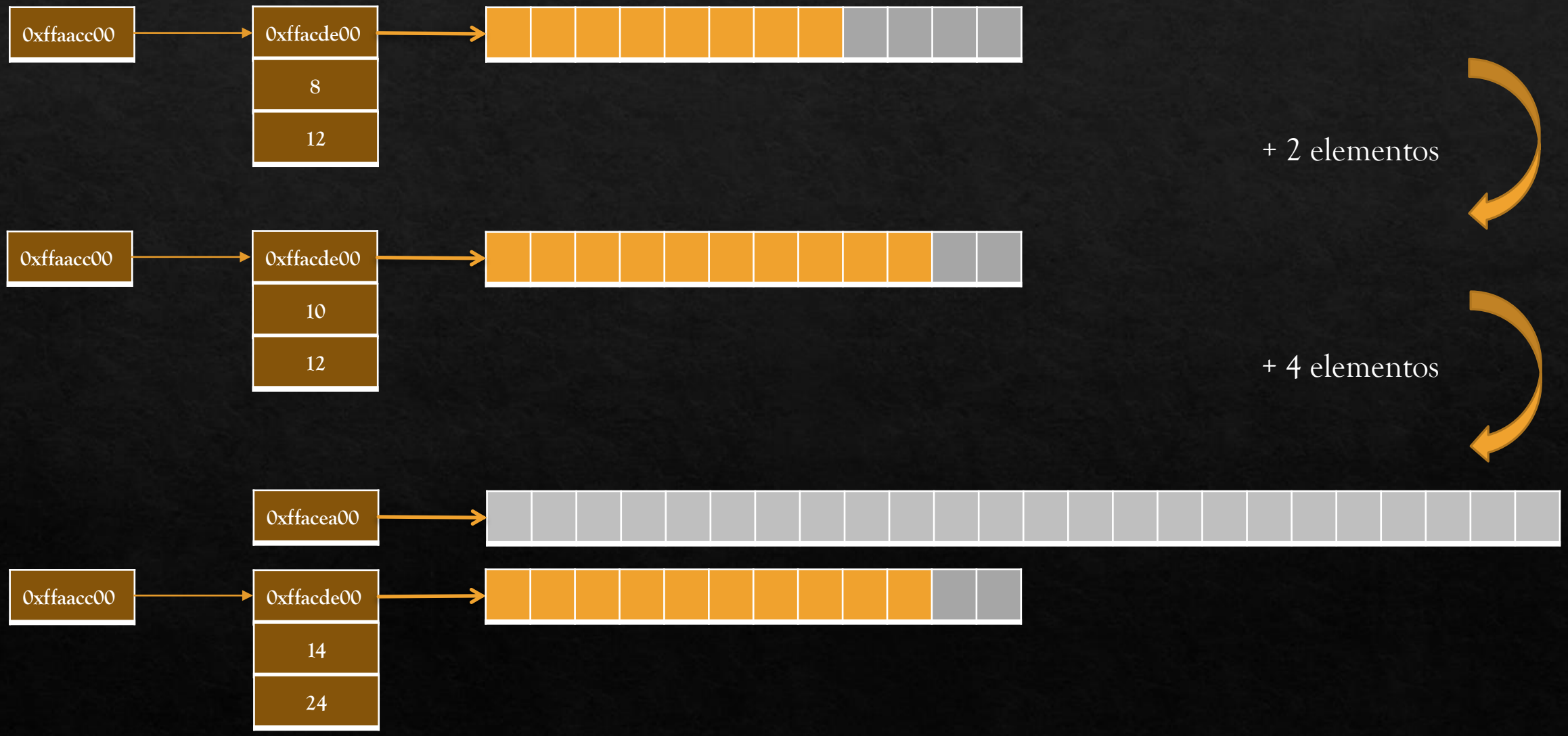
Arreglos Dinámicos



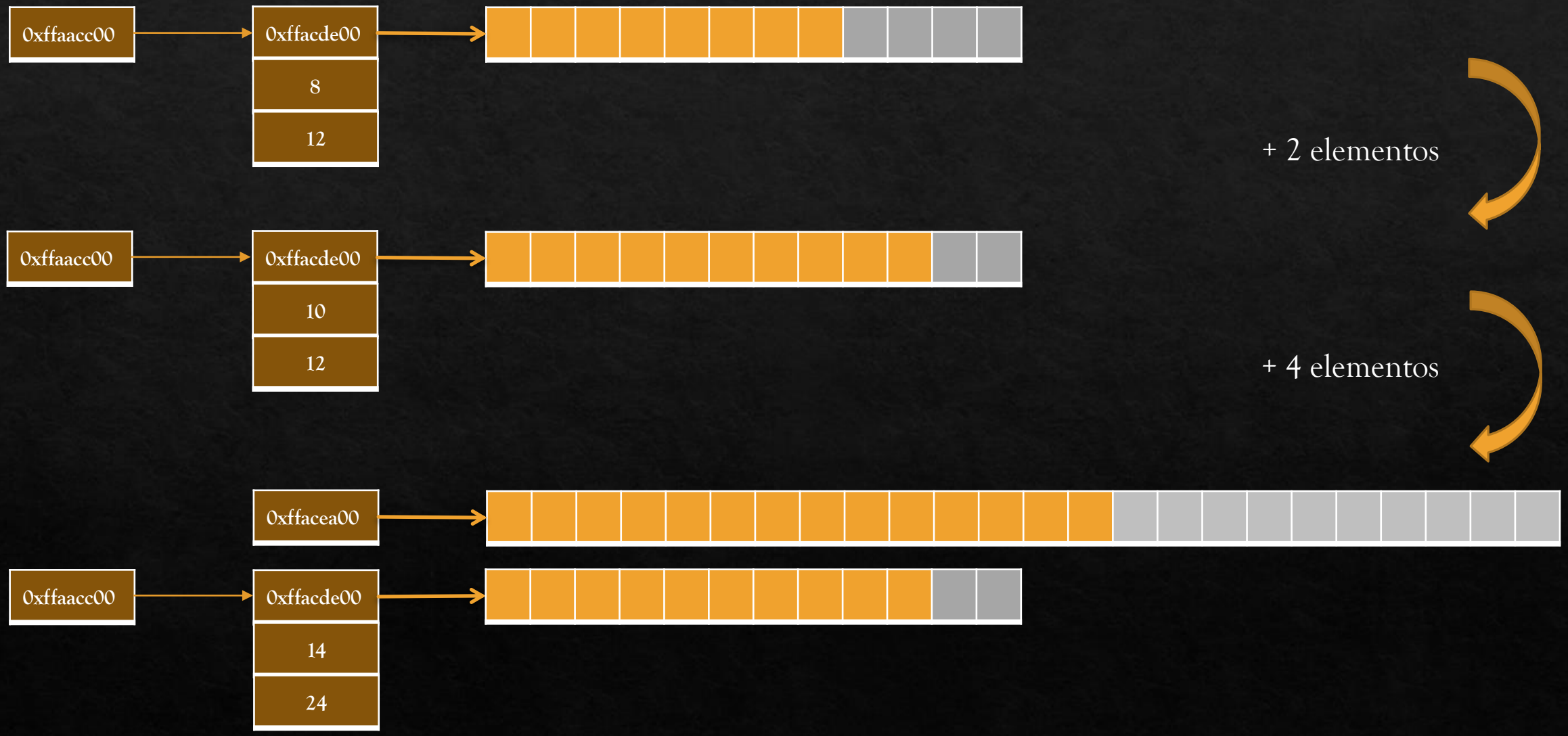
Secuencia Alternativa - Arreglos Dinámicos



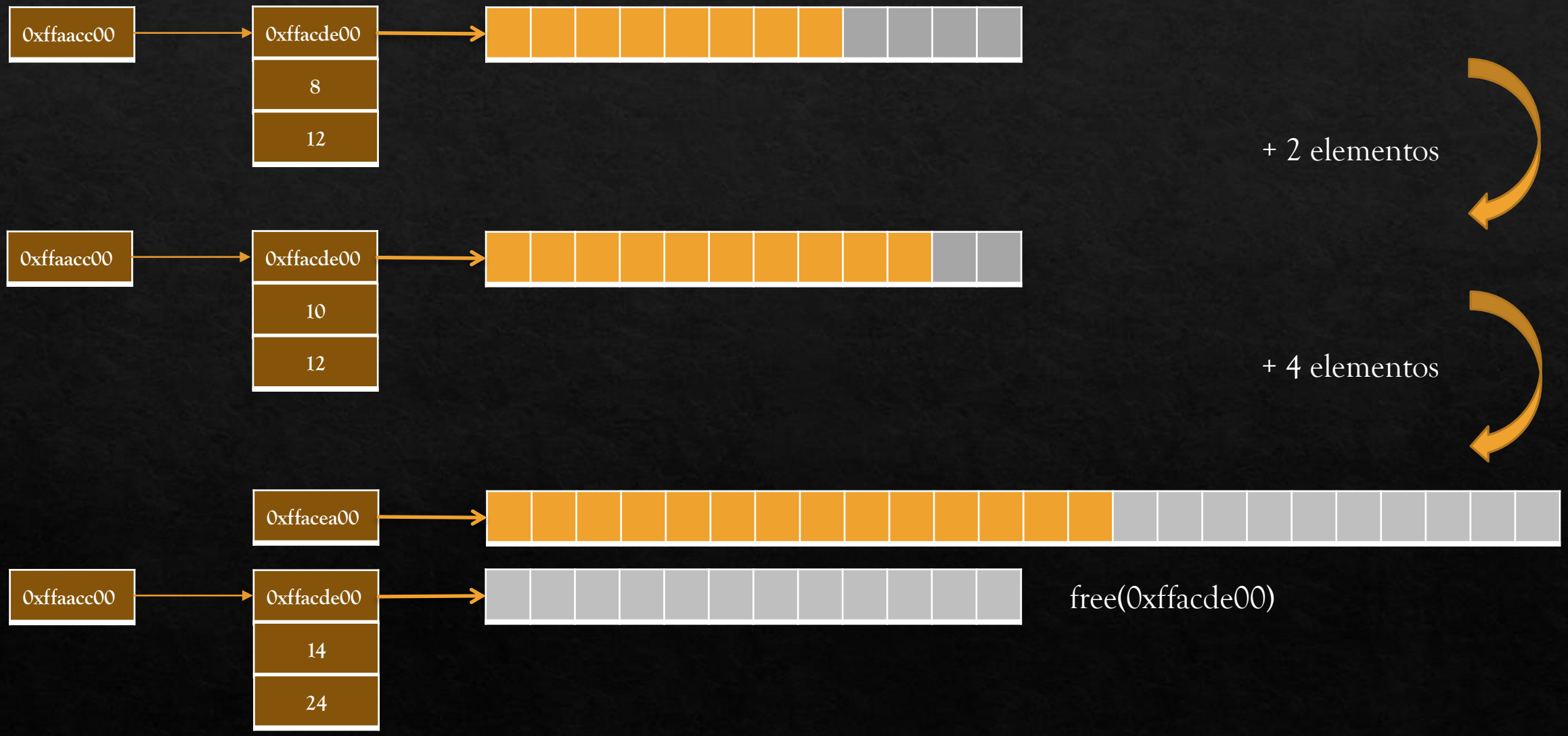
Secuencia Alternativa - Arreglos Dinámicos



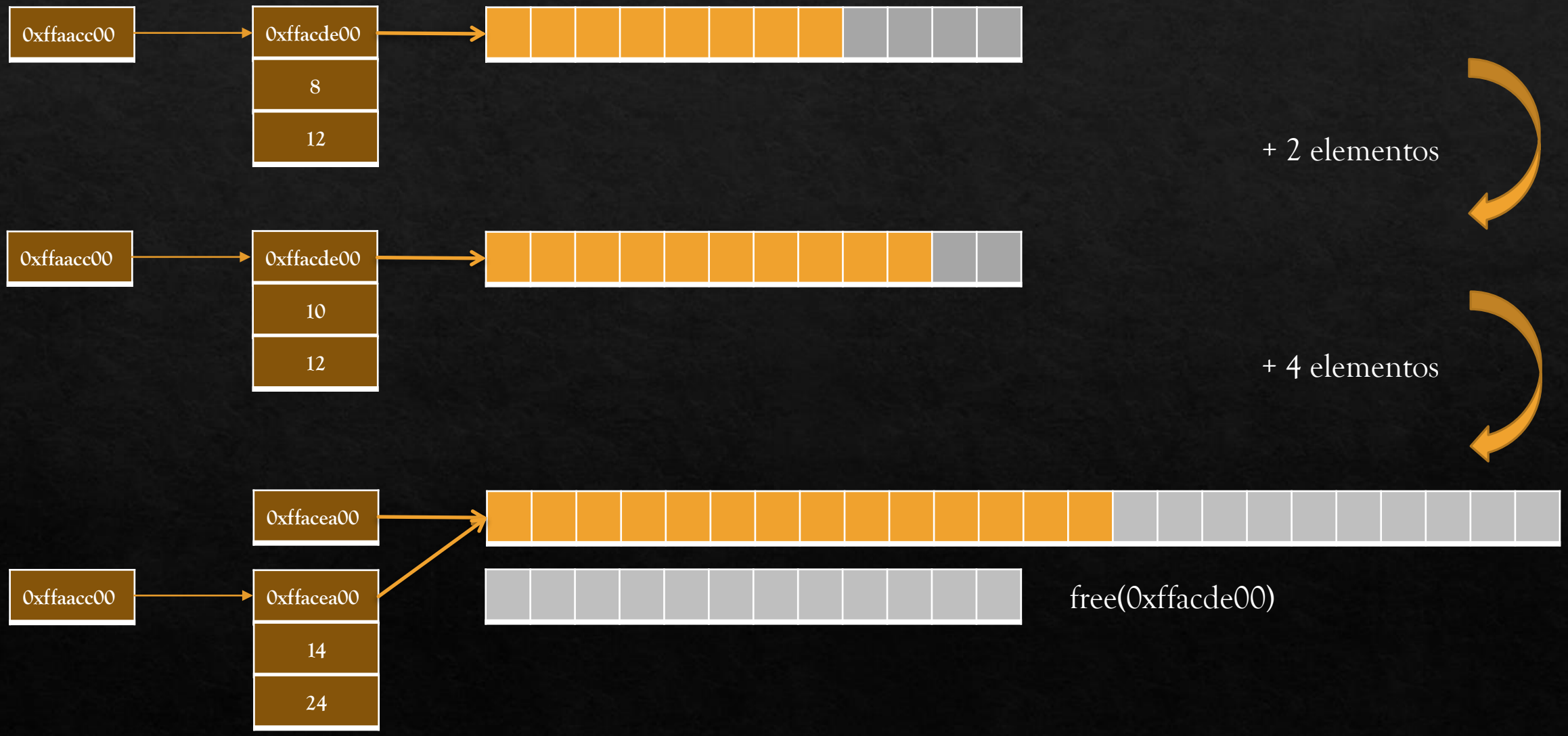
Secuencia Alternativa - Arreglos Dinámicos



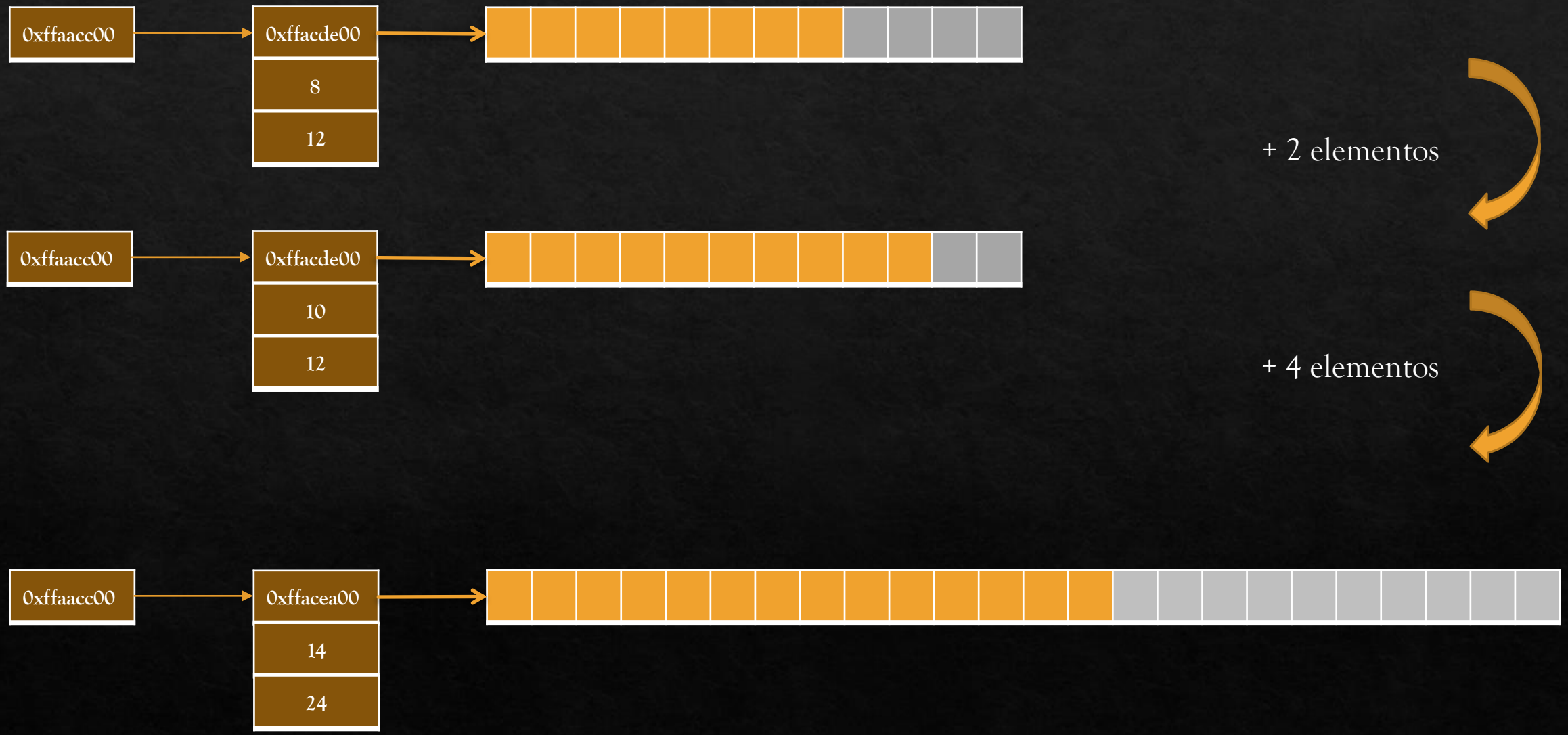
Secuencia Alternativa - Arreglos Dinámicos



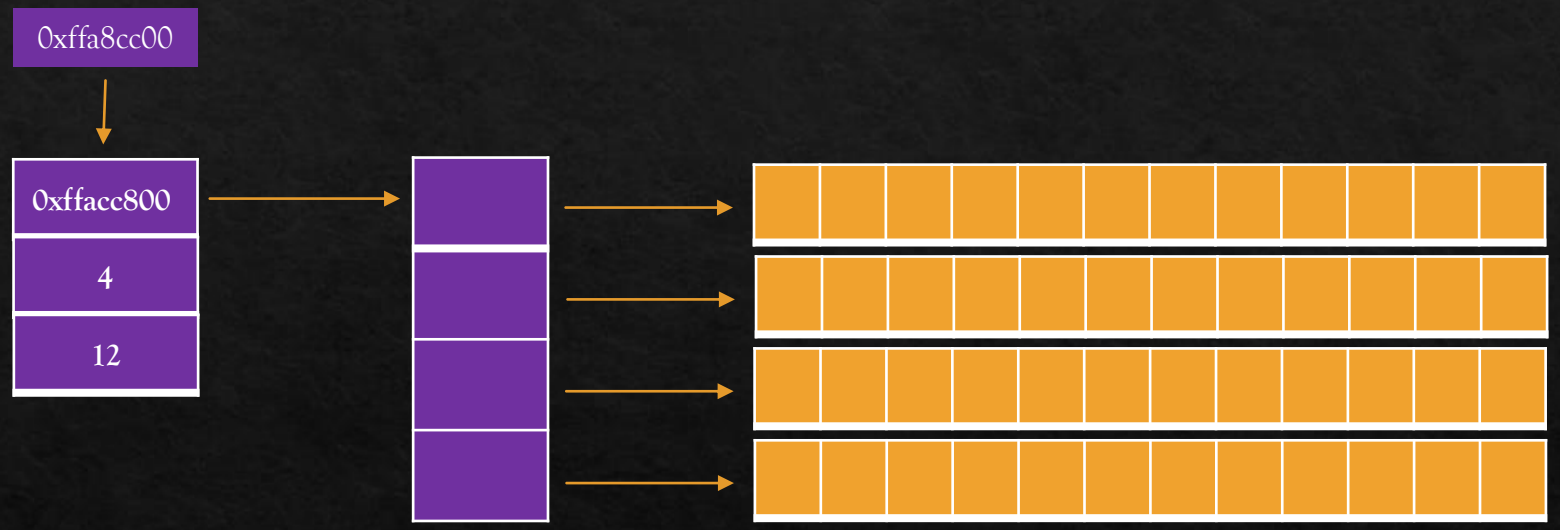
Secuencia Alternativa - Arreglos Dinámicos



Secuencia Alternativa - Arreglos Dinámicos



Matrix



Especificadores de Formato (printf)

Character	Description
%	Prints a literal % character (this type doesn't accept any flags, width, precision, length fields).
d, i	int as a signed integer. %d and %i are synonymous for output, but are different when used with scanf for input (where using %i will interpret a number as hexadecimal if it's preceded by 0x, and octal if it's preceded by 0.)
u	Print decimal unsigned int.
f, F	double in normal (fixed-point) notation. f and F only differs in how the strings for an infinite number or NaN are printed (inf, infinity and nan for f; INF, INFINITY and NAN for F).
e, E	double value in standard form (*d.ddde±dd*). An E conversion uses the letter E (rather than e) to introduce the exponent. The exponent always contains at least two digits; if the value is zero, the exponent is 00. In Windows, the exponent contains three digits by default, e.g. 1.5e002, but this can be altered by Microsoft-specific _set_output_format function.
g, G	double in either normal or exponential notation, whichever is more appropriate for its magnitude. g uses lower-case letters, G uses upper-case letters. This type differs slightly from fixed-point notation in that insignificant zeroes to the right of the decimal point are not included. Also, the decimal point is not included on whole numbers.
x, X	unsigned int as a hexadecimal number. x uses lower-case letters and X uses upper-case.
o	unsigned int in octal.
s	null-terminated string.
c	char (character).
p	void* (pointer to void) in an implementation-defined format.
a, A	double in hexadecimal notation, starting with 0x or 0X. a uses lower-case letters, A uses upper-case letters.[5][6] (C++11 iostreams have a hexfloat that works the same).
n	Print nothing, but writes the number of characters written so far into an integer pointer parameter. In Java this prints a newline.[7]

https://cplusplus.com/reference/cstdio/fprintf/
https://cplusplus.com/reference/cstdio/fscanf/

Si *f* es de tipo float y *d* es de tipo double, *fscanf*(*file*, "%f", &*f*) y *fscanf*(*file*, "%lf", &*d*) son correctos

Calendario: C23 Estándar

C23 - cppreference.com

Revised C23 Schedule **WG 14 N 2984**

Start	Event	Days	Completed
2022-05-16	Committee meeting (virtual)	5	2022-05-20
	Deadline for documents (final version of proposals)		2022-06-17
2022-07-18	Committee meeting (virtual)	5	2022-07-22
2022-07-25	Action items complete from meeting	7	2022-08-01
2022-08-01	Editor revises draft	7	2022-08-08
2022-08-08	Editorial review	7	2022-08-15
2022-08-15	Editor readies document for CD	7	2022-08-22
2022-08-22	CD ballot	91	2022-11-21
	Deadline for documents		2022-12-09
2023-01-09	Ballot resolution/Committee meeting (hybrid?)	5	2023-01-13
2023-01-16	Action items complete from meeting	7	2023-01-23
2023-01-23	Editor revises draft	7	2023-01-30
2023-01-30	Editorial review	14	2023-02-13
2023-02-13	Editor readies document for DIS	7	2023-02-20
2023-02-20	ISO editing (submit DIS ballot)	56	2023-04-17
2023-04-17	DIS ballot	182	2023-10-16
2023-10-16	Contingency	14	2023-10-30
2023-10-30	Editor prepares IS	14	2023-11-13