



Arrays

Vectores

Universidad Católica Boliviana

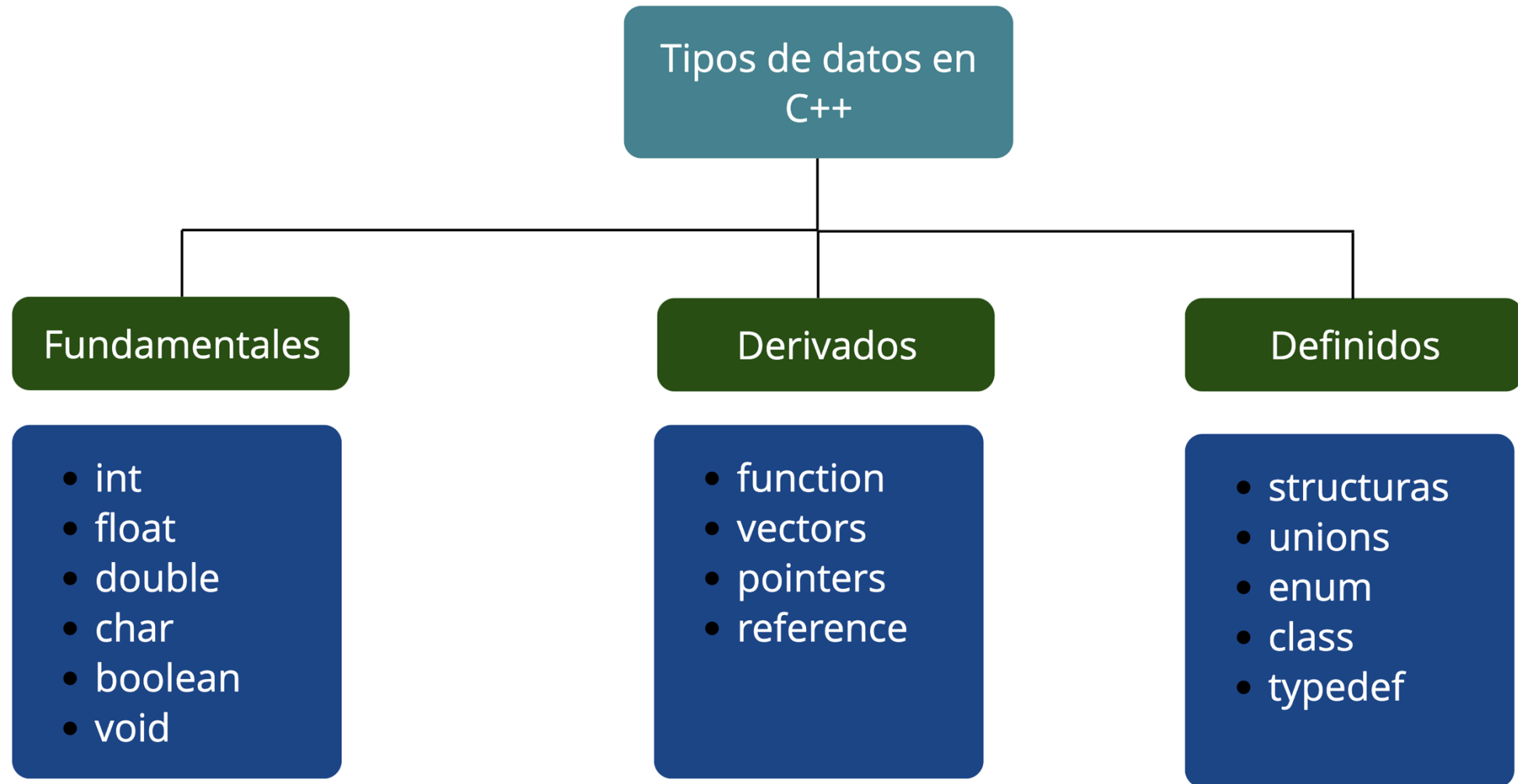
MSc, José Jesús Cabrera Pantoja

Outline

- Introducción a los arreglos (Arrays)
- Vectores en C++
- Algoritmos

Variables: Tipos de datos

ng



Tipos de datos: Fundamentales

Type	Typical Bit Width	Typical Range
char	1byte	-127 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	2bytes	0 to 65,535
signed short int	2bytes	-32768 to 32767
long int	8bytes	-9223372036854775808 to 9223372036854775807
signed long int	8bytes	same as long int
unsigned long int	8bytes	0 to 18446744073709551615
long long int	8bytes	-(2^63) to (2^63)-1
unsigned long long int	8bytes	0 to 18,446,744,073,709,551,615
float	4bytes	
double	8bytes	

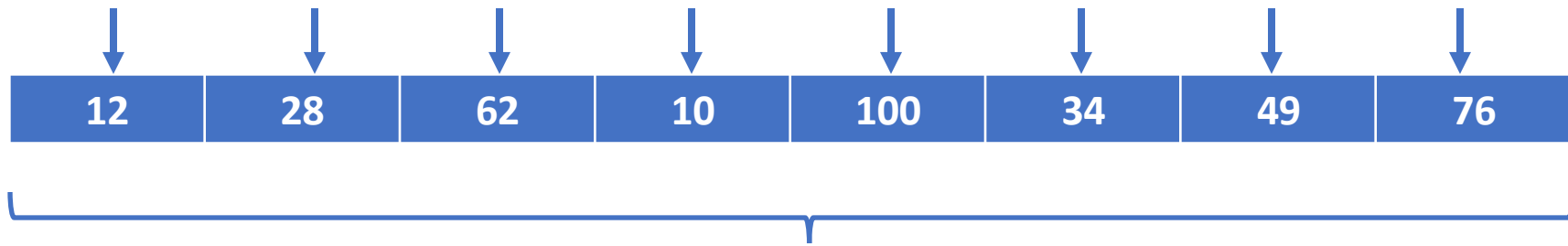
Tipo de datos: Modificaciones

- C++ admite los siguientes modificadores de tipo de datos:
 - long
 - short
 - unsigned
 - signed

Arrays

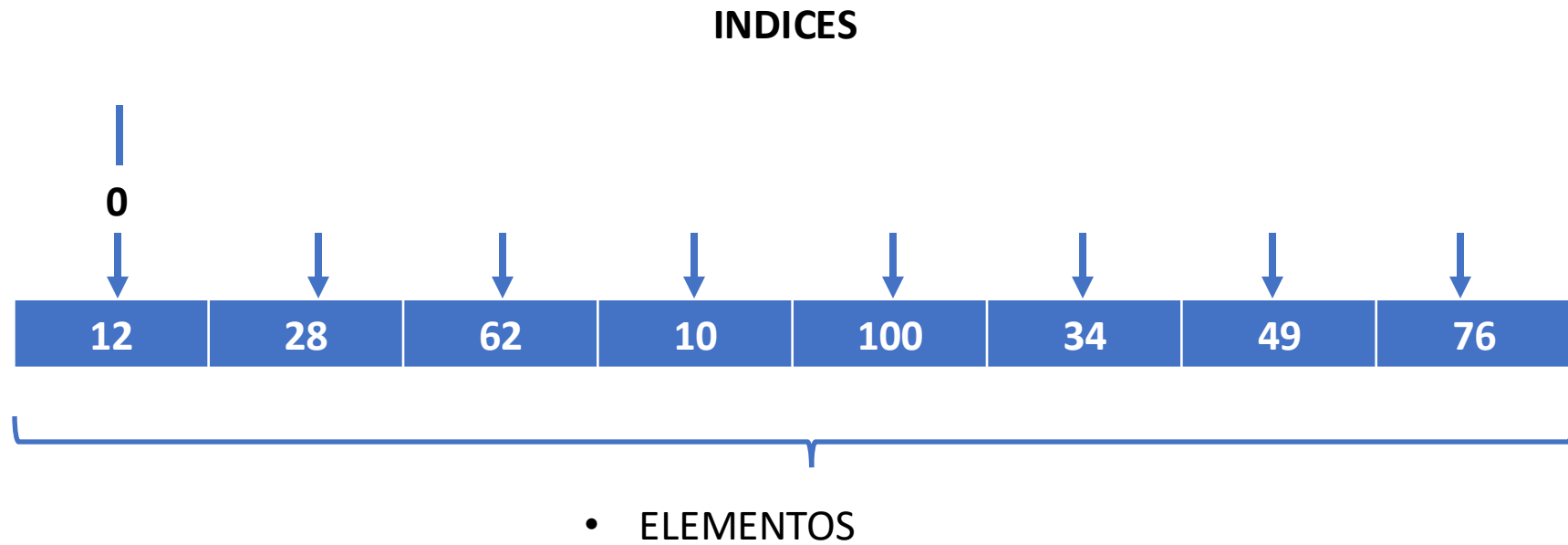
Arrays

- Los arreglos (Arrays)- Además de los valores fundamentales que representan un número, un carácter, una cadena, los contenedores se utilizan en la programación: objetos que almacenan un conjunto completo o, como se suele decir, una matriz de valores.

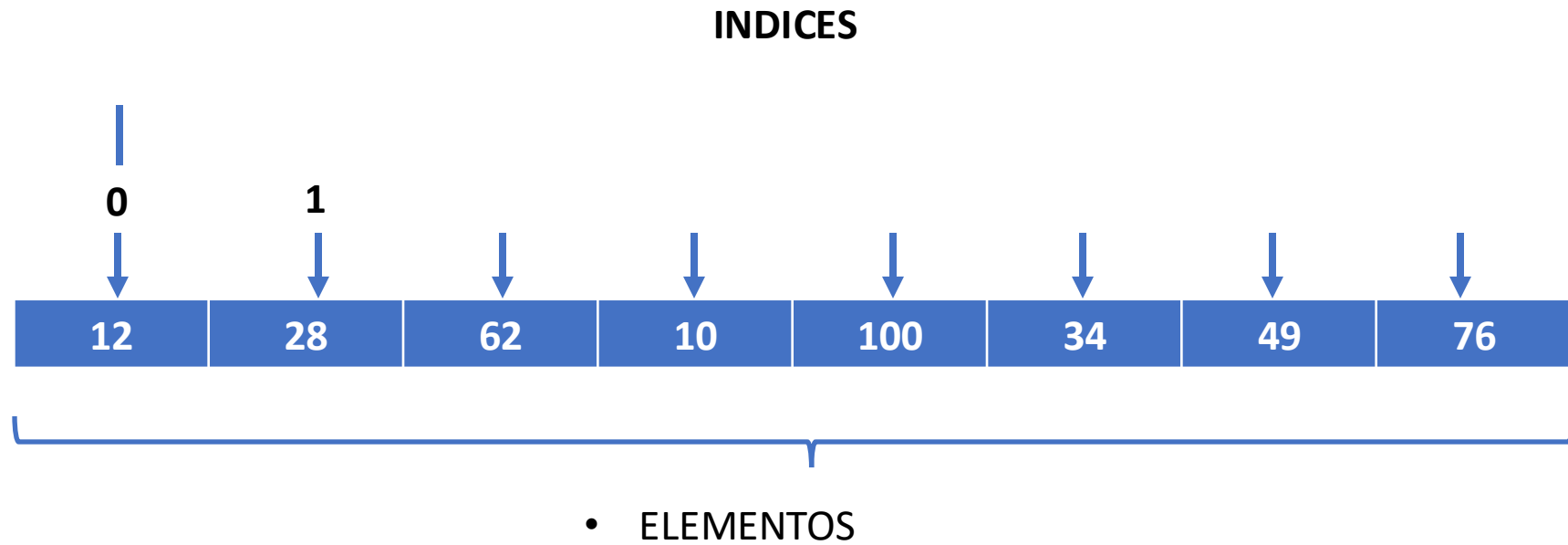


- ELEMENTOS

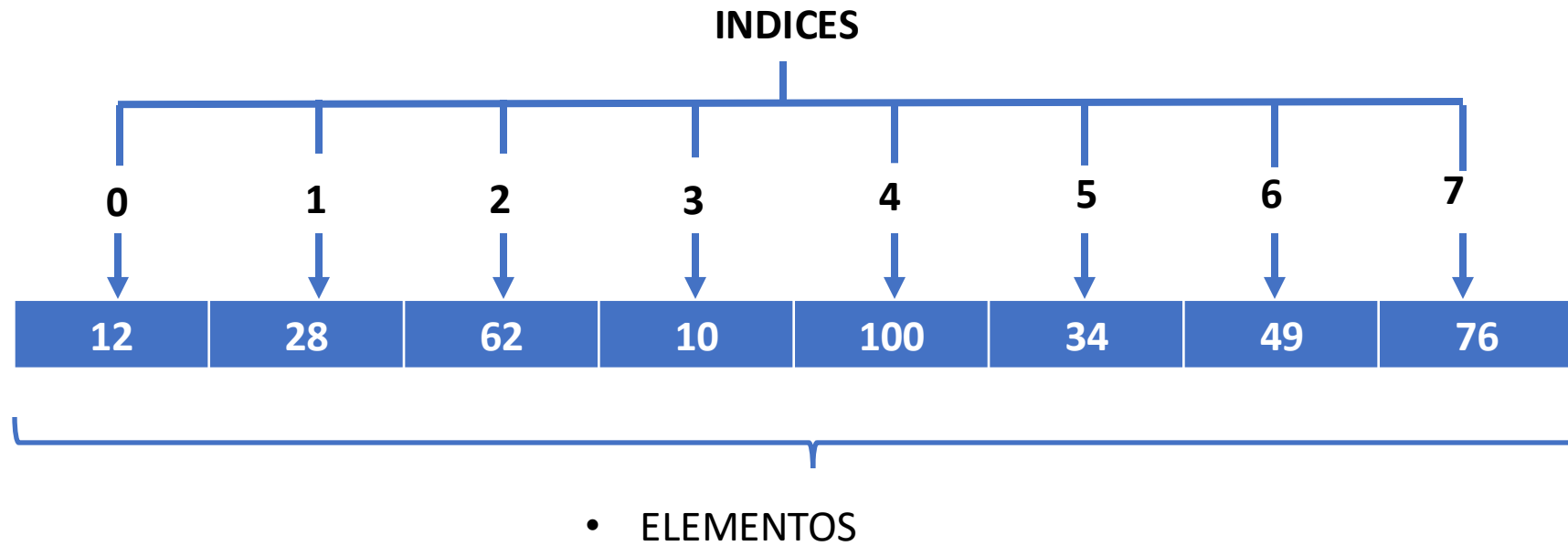
Arrays



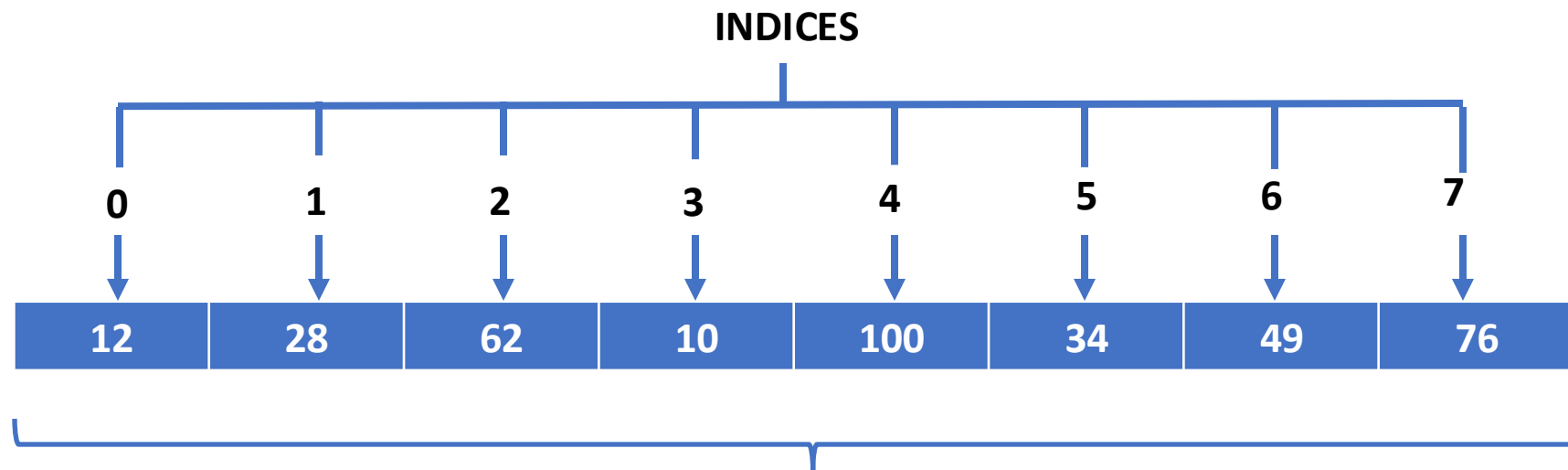
Arrays



Arrays



Arrays



- **TAMANO DEL VECTOR = 8**

Arrays - código

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int arr[] = {1, 2, 3, 4};
7      int size = sizeof(arr) / sizeof(arr[0]);
8
9      cout << arr[0] << endl;
10     cout << arr[1] << endl;
11     cout << arr[2] << endl;
12
13     cout << arr[size] << endl;
14
15     return 0;
16 }
```

Arrays - codigo

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int arr[] = {1, 2, 3, 4};
7      int size = sizeof(arr) / sizeof(arr[0]);
8
9      cout << arr[0] << endl;
10     cout << arr[1] << endl;
11     cout << arr[2] << endl;
12
13     cout << arr[size] << endl;
14
15     return 0;
16 }
```

Decimos que es un array



Arrays - codigo

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int arr[] = {1, 2, 3, 4};
7      int size = sizeof(arr) / sizeof(arr[0]);
8
9      cout << arr[0] << endl;
10     cout << arr[1] << endl;
11     cout << arr[2] << endl;
12
13     cout << arr[size] << endl;
14
15     return 0;
16 }
```


Calculamos el tamaño del array



Arrays - codigo

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int arr[] = {1, 2, 3, 4};
7      int size = sizeof(arr) / sizeof(arr[0]);
8
9      cout << arr[0] << endl;
10     cout << arr[1] << endl;
11     cout << arr[2] << endl;
12
13     cout << arr[size] << endl;
14
15     return 0;
16 }
```


Mostramos el primer
elemento del array



Arrays - codigo

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int arr[] = {1, 2, 3, 4};
7      int size = sizeof(arr) / sizeof(arr[0]);
8
9      cout << arr[0] << endl;
10     cout << arr[1] << endl;
11     cout << arr[2] << endl;
12
13     cout << arr[size] << endl;
14
15     return 0;
16 }
```


Mostramos el segundo
elemento del array



Arrays - código

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int arr[] = {1, 2, 3, 4};
7      int size = sizeof(arr) / sizeof(arr[0]);
8
9      cout << arr[0] << endl;
10     cout << arr[1] << endl;
11     cout << arr[2] << endl;
12
13     cout << arr[size] << endl;
14
15     return 0;
16 }
```

¿Que se mostrara en esta línea?



Vectores

Vector

- Un **vector** es uno de los contenedores estándar en C++.
- Cada contenedor es adecuado para una cierta gama de tareas.
- Hay muchas tareas para las que el vector es adecuado, por lo que se usa con mucha frecuencia.



Vector

- Para trabajar con un vector, incluya la librería `<vector>`.
- Al declarar una variable, los corchetes angulares indican el tipo de elementos que estarán contenidos en el contenedor.
- Por ejemplo,
 - **`vector<int>`**
 - **`vector<string>`**
 - **`vector<char>`**




Vector

- Imagina que quieres almacenar en un programa el número de días de cada mes del año.
- Un vector será útil para esto.
- La cantidad de días es un número, es decir, almacenaremos valores del tipo **int** en el **vector**.
- Declaremos la variable **month_lengths**. Su tipo es **vector<int>**.
- El contenido del vector se escribe entre llaves durante la inicialización:



Vector

- El contenido del vector se escribe entre llaves durante la inicialización:




```
1 #include <vector>
2
3 using namespace std;
4
5 int main() {
6     vector<int> month_lengths = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
7 }
```

Vector

- El contenido del vector se escribe entre llaves durante la inicialización:



```
1 #include <vector>
2
3 using namespace std;
4
5 int main() {
6     vector<int> month_lengths = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
7 }
```



Nombre de
la variable

Vector

- El contenido del vector se escribe entre llaves durante la inicialización:



```
1 #include <vector>
2
3 using namespace std;
4
5 int main() {
6     vector<int> month_lengths = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
7 }
```

Tipo de la
variable

Nombre de
la variable

Vector

- El contenido del vector se escribe entre llaves durante la inicialización:



```
1 #include <vector>
2
3 using namespace std;
4
5 int main() {
6     vector<int> month_lengths = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
7 }
```

Tipo de los
elementos
del vector


Nombre de
la variable

Vector

- El contenido del vector se escribe entre llaves durante la inicialización:



```
1 #include <vector>
2
3 using namespace std;
4
5 int main() {
6     vector<int> month_lengths = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
7 }
```



Elementos del vector, en este caso enteros (int)

Vector

- Como puede ver, al declarar un vector, también se especifica el tipo de sus elementos.
- En C++, el tipo de cada valor, incluido el almacenado en un vector, se conoce de antemano, antes de ejecutar el programa. Esto permite que el compilador decida cómo procesar los valores sin desperdiciar recursos mientras se ejecuta la aplicación.
- Si comparamos tipos de datos con perros de diferentes tamaños y razas, entonces en C++ los perros se ubican por tamaño y raza.

Vector

- Si comparamos tipos de datos con perros de diferentes tamaños y razas, entonces en C++ los perros se ubican por tamaño y raza.



Vector

- Lo que quiere decir que en C++ un vector debe contener elementos del mismo tipo!!!
- No se puede tener un vector con elementos de distintos tipos!



Vector

- Cualquier vector tiene un tamaño: la cantidad de valores en el contenedor.
- El tamaño puede ser de uno o diez millones: C++ puede funcionar de manera eficiente incluso con vectores muy grandes.
- Supongamos también un vector de tamaño 0: se llama **vacío**.

Vector

- Cualquier vector tiene un tamaño: la cantidad de valores en el contenedor.



```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 int main() {
7     vector<int> month_lengths = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
8     cout << month_lengths.size() << endl;
9 }
```

- Cual es la salida del programa?

Vector

- Cualquier vector tiene un tamaño: la cantidad de valores en el contenedor.



```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 int main() {
7     vector<int> month_lengths = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
8     cout << month_lengths.size() << endl;
9 }
```

- Cual es la salida del programa? 12!

Vector

- Cada elemento del vector tiene un **índice**, un número de serie desde el comienzo del vector.
- Además, los elementos están numerados desde **cero**: el índice del primer elemento es **cero** y el índice del último elemento **es uno menos que el tamaño del vector**.

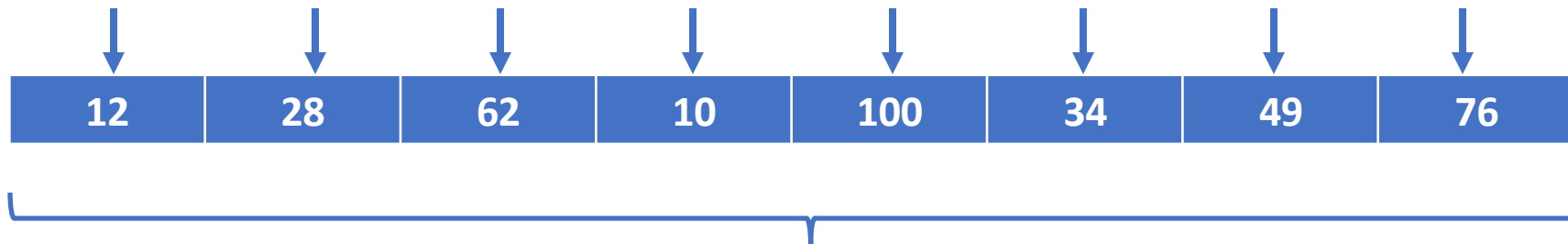


```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 int main() {
7     vector<int> month_lengths = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
8     cout << month_lengths.size() << endl;
9 }
```

Vector



```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 int main() {
7     vector<int> some_numbers = {12, 28, 62, 10, 100, 34, 49, 76};
8     cout << some_numbers.size() << endl;
9 }
```

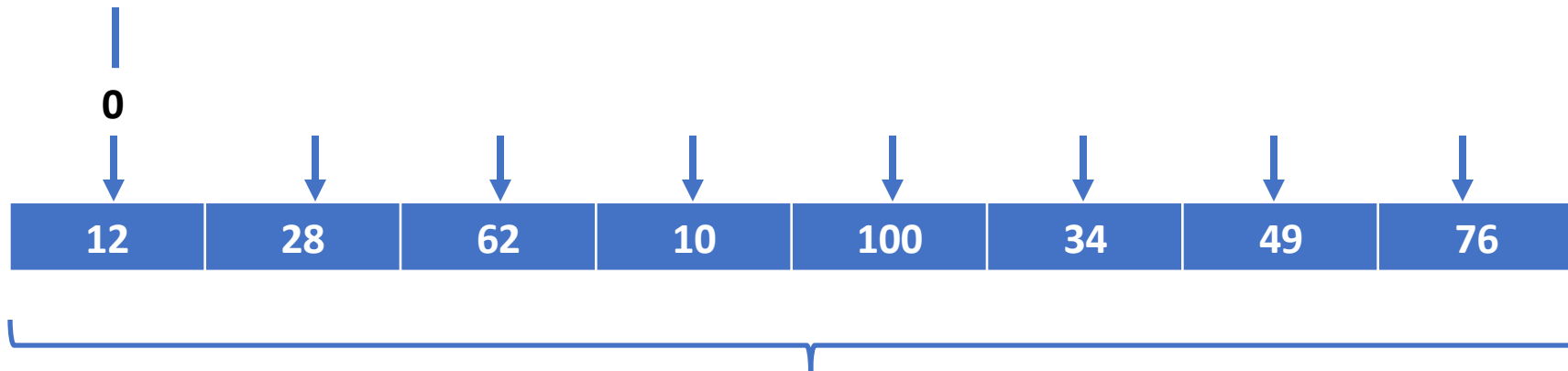


Vector



```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 int main() {
7     vector<int> some_numbers = {12, 28, 62, 10, 100, 34, 49, 76};
8     cout << some_numbers.size() << endl;
9 }
```

INDICES



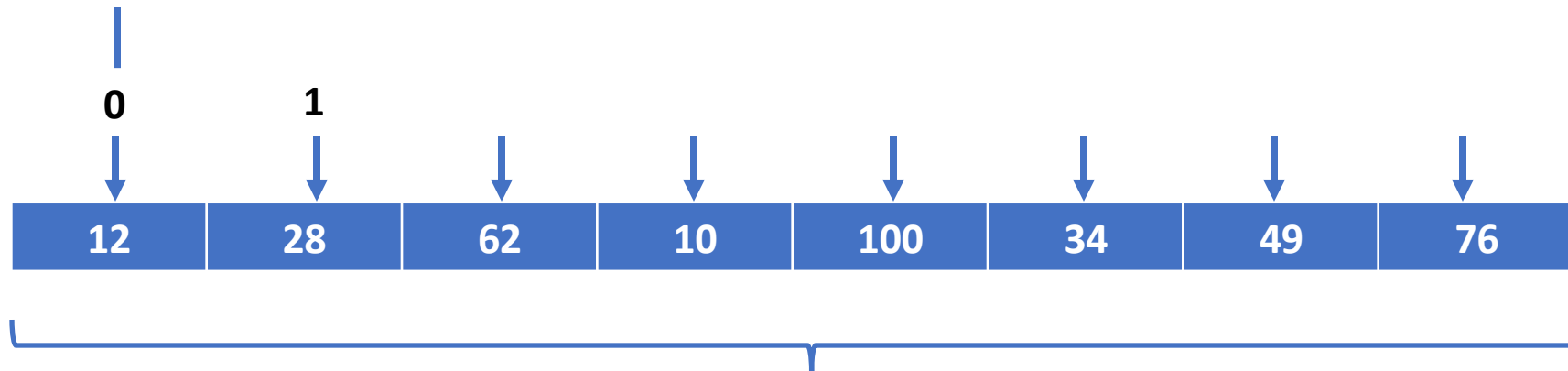
- ELEMENTOS

Vector



```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 int main() {
7     vector<int> some_numbers = {12, 28, 62, 10, 100, 34, 49, 76};
8     cout << some_numbers.size() << endl;
9 }
```

INDICES

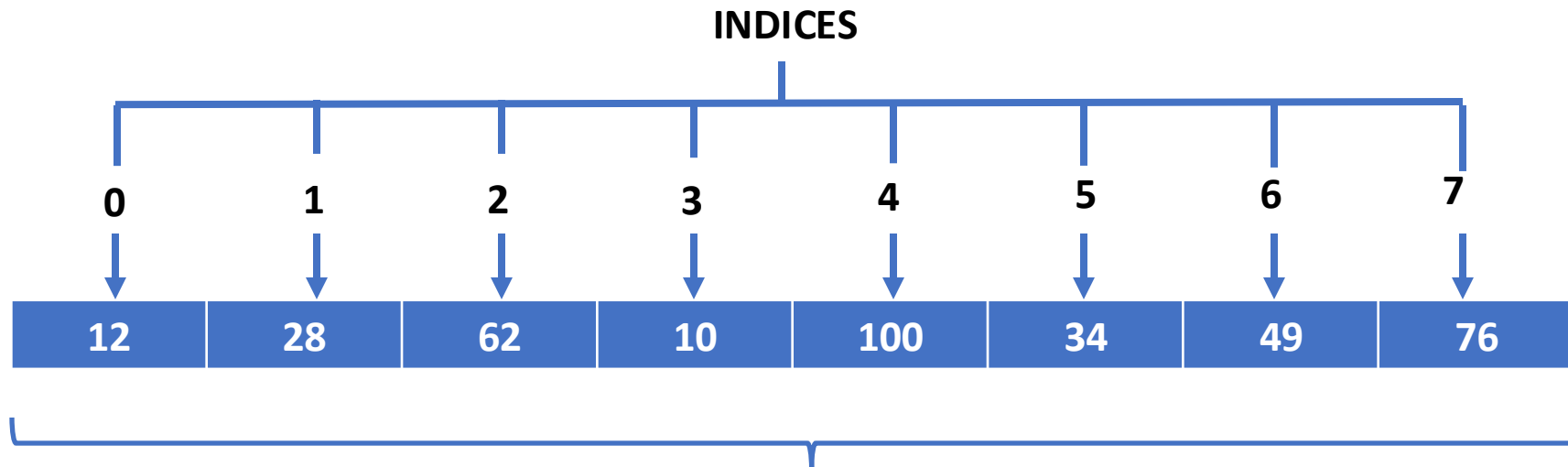


- ELEMENTOS

Vector



```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 int main() {
7     vector<int> some_numbers = {12, 28, 62, 10, 100, 34, 49, 76};
8     cout << some_numbers.size() << endl;
9 }
```

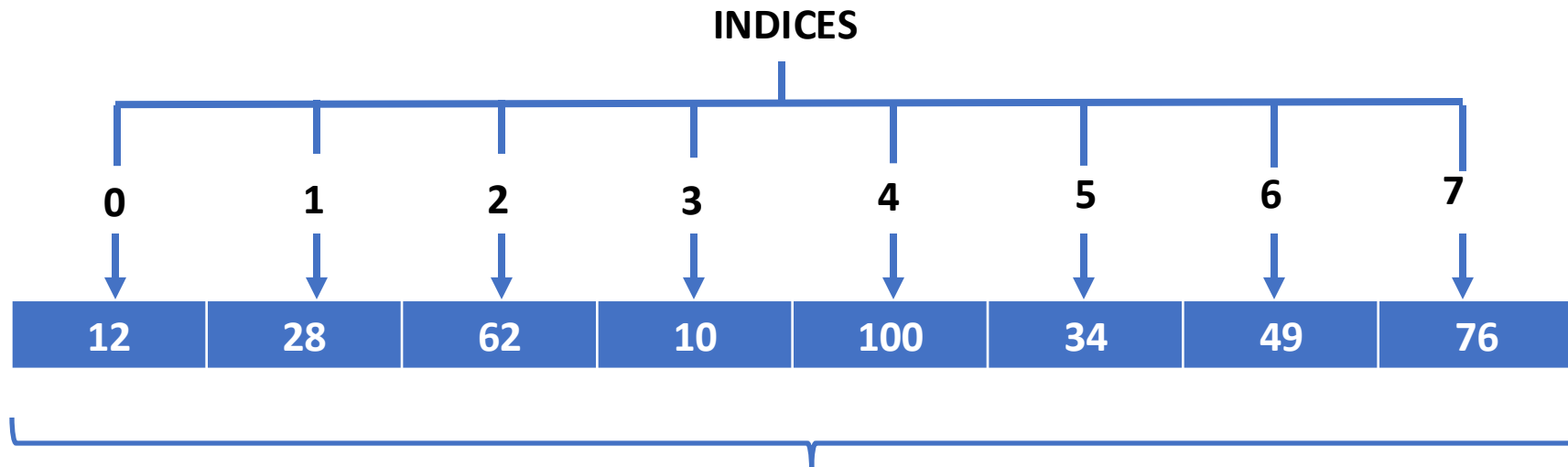


• ELEMENTOS

Vector




```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 int main() {
7     vector<int> some_numbers = {12, 28, 62, 10, 100, 34, 49, 76};
8     cout << some_numbers.size() << endl;
9 }
```



- TAMANO DEL VECTOR = 8

Vector

- Para referirse a un elemento vectorial específico, debe escribir el índice entre corchetes.
- El índice puede ser un número, una variable o una expresión numérica arbitraria:



```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 int main() {
7     vector<int> month_lengths = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
8     cout << month_lengths.size() << endl;
9
10    int month_index;
11    cin >> month_index;
12
13    // Pedir el elemento del vector month-lengths con el indice month_index
14    cout << "There are "s << month_lengths[month_index] << " days"s << endl;
15 }
```


Algoritmos