Intro a C

Arreglos

With 💜 by @vichoeq & @KnowYourselves

Contenedores Indexados

Tanto C como **Python** tienen contenedores indexados de información,
C tiene arreglos y **Python** tiene listas.

Arreglos v/s Listas

```c
int A[6] = {0, 5, 3, 2, 1, 7};
printf("%d, %d\n", A[0], A[2]);
```

```python
A = [0, 5, 3, 2, 1, 7]
print(f"{A[0]}, {A[2]}")
```

```c
int A[6];
```

No existe en Python 😢

Arreglos v/s Listas

?

```
A = [1, "Hello", 5, "World!"]
B = [1.642, 12, "Hi", "Ciao"]
C = ["Muchos", "Tipos!", 154]
```

Arreglos v/s Listas

❌ 😱

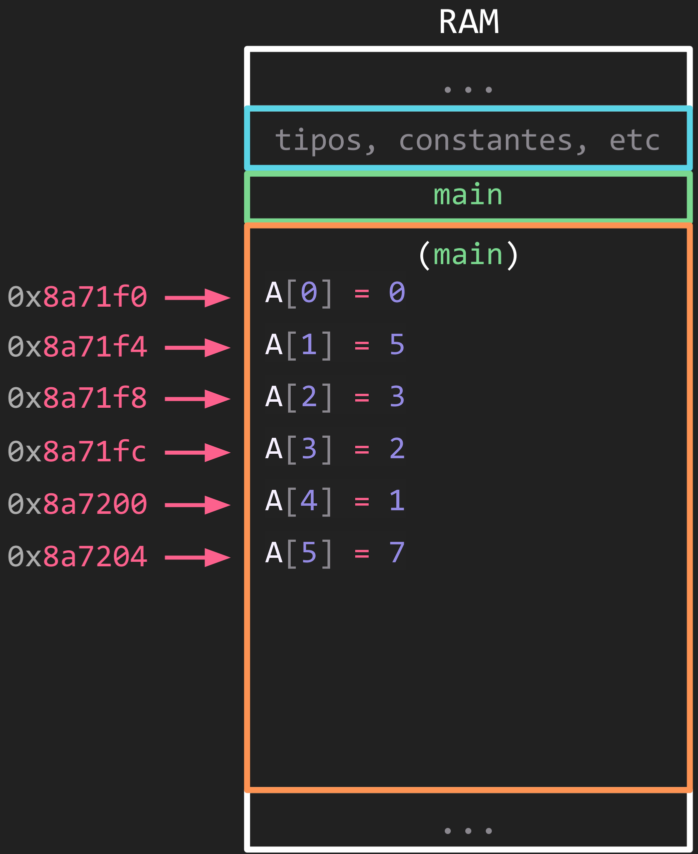
```
A = [1, "Hello", 5, "World!"]
B = [1.642, 12, "Hi", "Ciao"]
C = ["Muchos", "Tipos!", 154]
```

Arreglos y STACK

Arreglos y STACK

```c
int A[6] = {0, 5, 3, 2, 1, 7};
printf("%p, %d\n", A, sizeof(int));
```

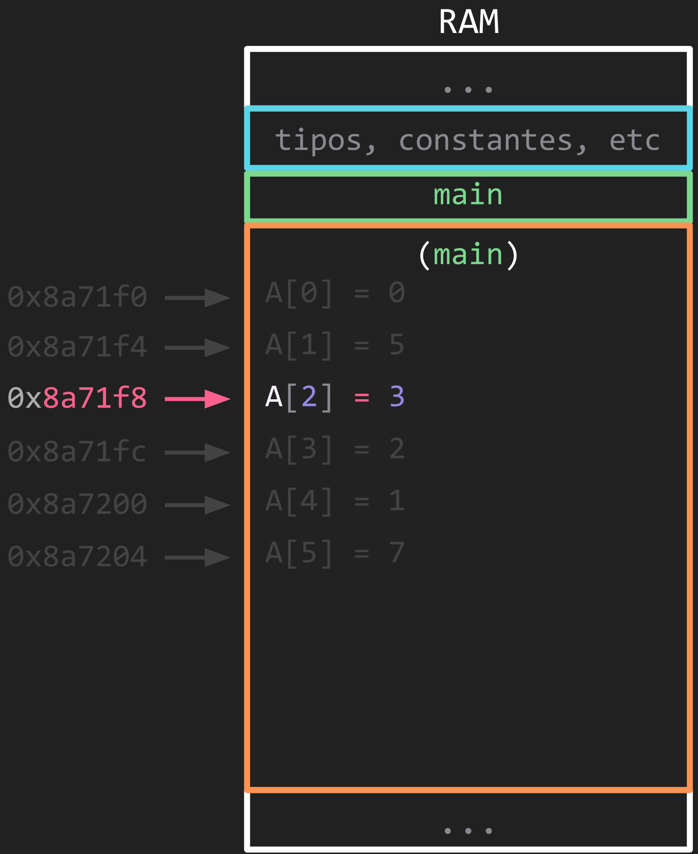
```
$ gcc main.c -o main
$ ./main
0x8a71f0, 4
```



Arreglos y STACK

```c
int A[6] = {0, 5, 3, 2, 1, 7};
printf("%p: %d\n", &A[2], A[2]);
```

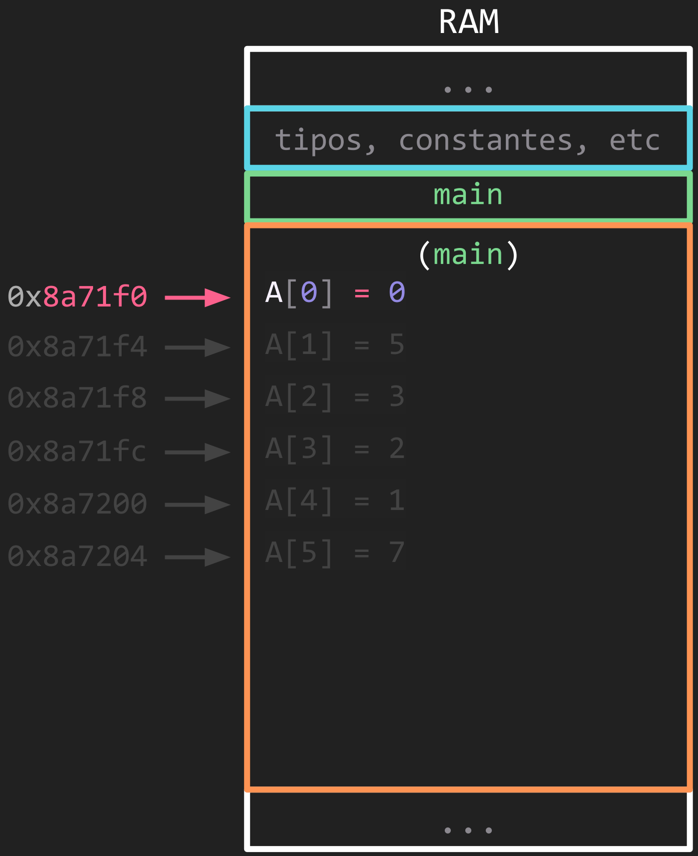
```
$ gcc main.c -o main
$ ./main
0x8a71f8: 3
```



Arreglos y STACK

```c
int A[6] = {0, 5, 3, 2, 1, 7};
printf("%p\n", A);
printf("%p\n", &A);
printf("%p\n", &A[0]);
```

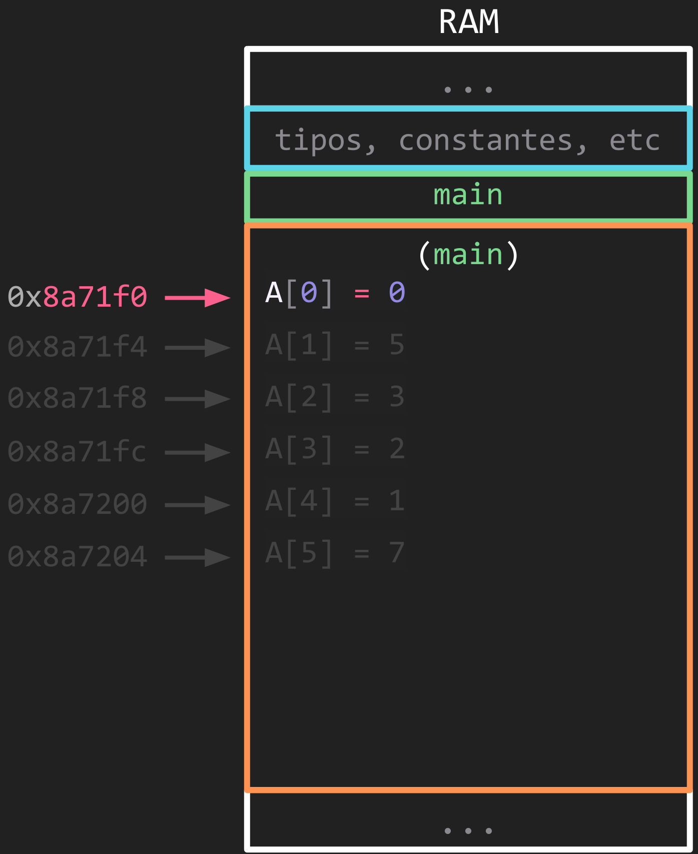
```
?
```



Arreglos y STACK

```c
int A[6] = {0, 5, 3, 2, 1, 7};
printf("%p\n", A);
printf("%p\n", &A);
printf("%p\n", &A[0]);
```

```
$ gcc main.c -o main
$ ./main
0x8a71f0
0x8a71f0
0x8a71f0
```



Indexación

Indexación = Aritmética de punteros

```
type A[]
```

Indexación = Aritmética de punteros

$$\begin{tikzcd}
\textit{type}\ \texttt{A[]} \arrow[d] \\
\texttt{A[i] =}
\end{tikzcd}$$

Indexación = Aritmética de punteros

$$\begin{tikzcd} \texttt{\textit{type} A[]} \arrow[d] \\ \texttt{A[i] = *(A + i)} \end{tikzcd}$$

Indexación = Aritmética de punteros

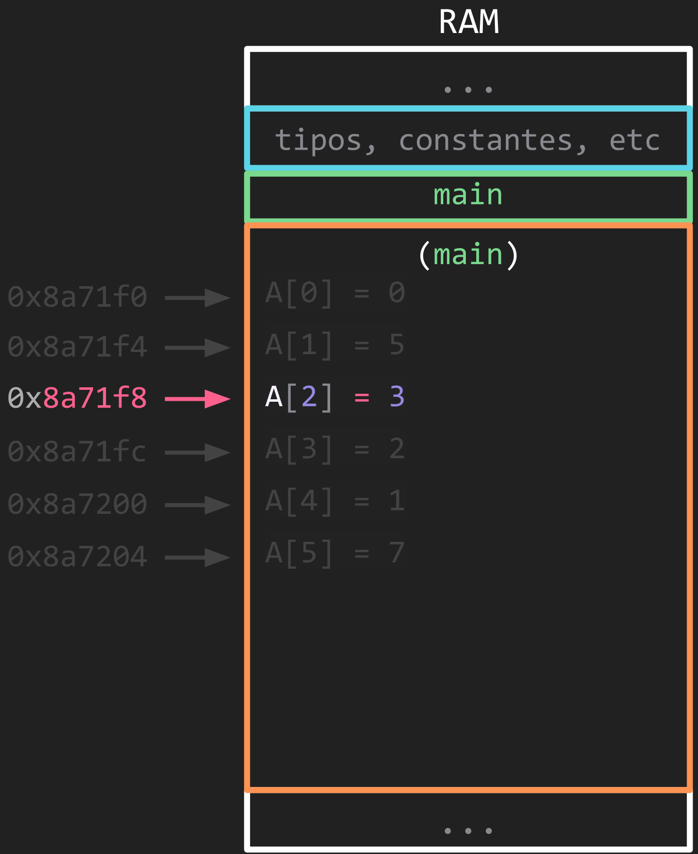
```
type A[]
   ↓
A[i] = *(A + i)

A[i] = i[A]
```
😱

Indexación

```c
int A[6] = {0, 5, 3, 2, 1, 7};
printf("%p\n", &A[2]);
printf("%p\n", A + 2);
printf("%p\n", &2[A]);
```

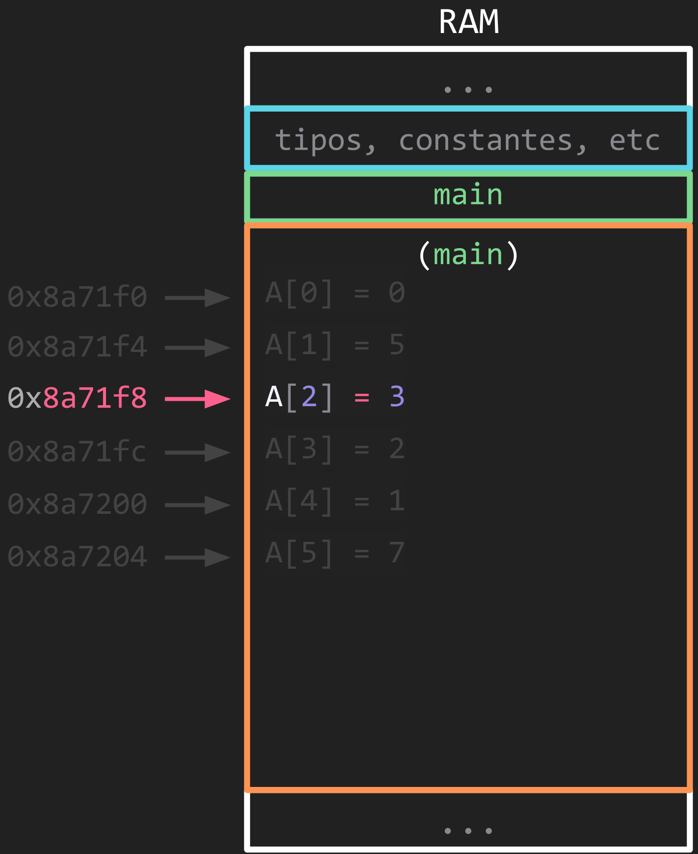
```
$ gcc main.c -o main
$ ./main
0x8a71f8
0x8a71f8
0x8a71f8
```



Indexación

```c
int A[6] = {0, 5, 3, 2, 1, 7};
printf("%d\n", A[2]);
printf("%d\n", *(A + 2));
printf("%d\n", 2[A]);
```

```
$ gcc main.c -o main
$ ./main
3
3
3
```



Índices - ☢ WARNING! ACHTUNG! PELIGRO! ☢

```c
int A[3] = {1, 2, 3};
printf("%d\n", A[-1]);
printf("%d\n", A[3]);
```

```
$ gcc main.c -o main
$ ./main
-1283455584
1145768448
```

```python
A = [1, 2, 3]
print(A[-1])
print(A[3])
```

```
$ gcc main.c -o main
$ ./main
3
IndexError: list index out of range
```

Iteración

Iteración: Arreglos v/s Listas

```c
for (int i = 0; i < 6; i+=1)
{
  printf("%d\n", F[i]);
}
```

```python
for i in range(6):
    print(F[i])
```

```c
int i = 0;
while (i < 6)
{
  printf("%d\n", F[i]);
  i += 1;
}
```

```python
i = 0
while i < 6:
    print(F[i])
    i += 1
```

¿for elem in array?

❔

¿for elem in array? 😭

❌

Índices - ☢ WARNING! ACHTUNG! PELIGRO! ☢

```c
int A[2] = {1, 2};
for (int i = 0; i < 100; i++)
{
  printf("%d ", A[i]);
}
```

```
$ gcc main.c -o main
$ ./main
1 2 -915650048 -1821377856 -931133664
32601 -939385961 32601 3 0 -859349320
32767 32768 3 -931133782 32601 235...
```

```python
A = [1, 2]
for i in range(100):
    print(A[i])
```

```
$ gcc main.c -o main
$ ./main
1
2
IndexError: list index out of range
```

Índices - ☢ WARNING! ACHTUNG! PELIGRO! ☢

```c
int A[2] = {1, 2};
for (int i = 0; i < 100; i++)
{
  A[i] = 'c';
}
```

```
$ gcc main.c -o main
$ ./main
*** stack smashing detected ***: <unknown>
terminated
Aborted (core dumped)
```

C nos permite escribir en espacios inválidos de memoria, pero afortunadamente algunos sistemas operativos lo evitan.

Arreglos de Arreglos

Arreglos de Arreglos v/s Listas de Listas

```c
int A[3][2] = {{1, 2}, {3, 4}, {5, 6}};
printf("%d, %d", A[0][1], A[2][0]);
```

```python
A = [[1, 2], [3, 4], [5, 6]]
print(f"{A[0][1]}, {A[2][0]}")
```

```c
int A[3][2];
```

No existe en Python 😢

Arreglos de Arreglos v/s Listas de Listas

?

```
A = [["Distinto"], ["Tamaño", "!"]]
B = [[1, 2], [3, 4, 5], ["f", "b"]]
C = [[1.563], [1.241], [1.45, 1.2]]
```

Arreglos de Arreglos v/s Listas de Listas

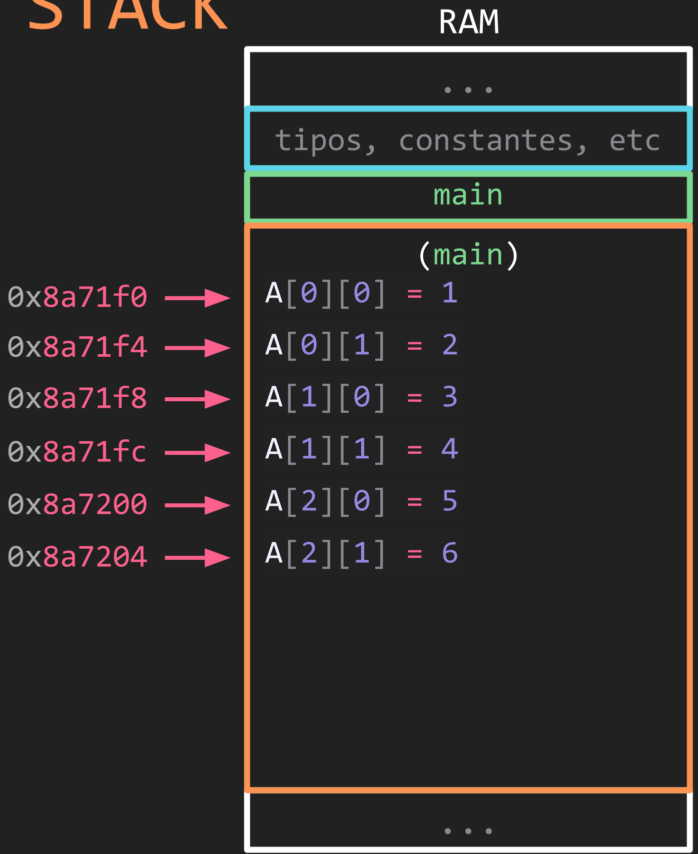
❌ 😱

```
A = [["Distinto"], ["Tamaño", "!"]]
B = [[1, 2], [3, 4, 5], ["f", "b"]]
C = [[1.563], [1.241], [1.45, 1.2]]
```

Arreglos de Arreglos y STACK

```c
int A[3][2] = {{1, 2}, {3, 4}, {5, 6}};
printf("%p\n", A);
```

```
$ gcc main.c -o main
$ ./main
0x8a71f0
```



Arreglos y funciones

Arreglos y funciones

```c
int sum(int* array, int n)
{
  int count = 0;
  for (int i = 0; i < n; i+=1)
  {
    count += array[i];
  }
  return count;
}

int A[5] = {1, -2, 3, 4, 1};
printf("SUM = %d\n", sum(A, 5));
```

Como un arreglo no es más que un puntero a su primer elemento, podemos pasarlo por "referencia" a una función.

Arreglos y funciones

```c
int A[5] = {1, -2, 3, 4, 1};
printf("SUM = %d\n", sum(A, 5));
```

```
$ gcc main.c -o main
$ ./main
SUM = 7
```

Esto nos permite facilitar mucho la sintáxis.

Arreglos y funciones

```c
void multiply_by(int* array, int n, int x)
{
  for (int i = 0; i < n; i+=1)
  {
    array[i] *= x;
  }
}

int A[3] = {1, -2, 3};
multiply_by(A, 5, 3);
printf("%d, %d, %d\n", A[0], A[1], A[2]);
```

También podemos modificar una arreglo en una función.

Arreglos y funciones

```c
int A[3] = {1, -2, 3};
multiply_by(A, 3, 3);
printf("%d, %d, %d\n", A[0], A[1], A[2]);
```

```
$ gcc main.c -o main
$ ./main
3, -6, 9
```

Esto nos permite facilitar mucho la sintáxis.

¡Muchas Gracias!

With 💜 by @vichoeq & @KnowYourselves