Intro a C
Strings

With 💜 by @vichoeq & @KnowYourselves

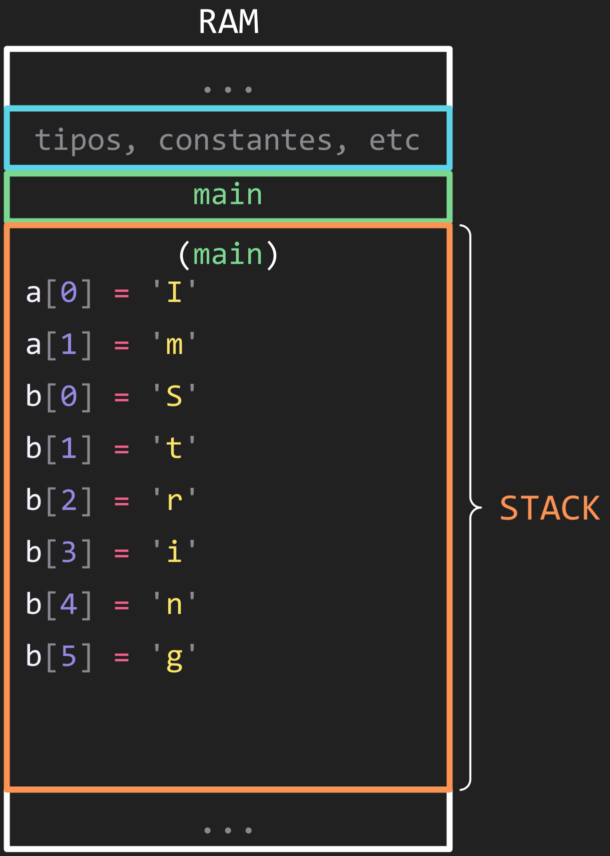
Strings en C

No existen los strings en C como un *tipo* definido. Lo que hace el lenguaje es trabajar con arreglos de *char*.

Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

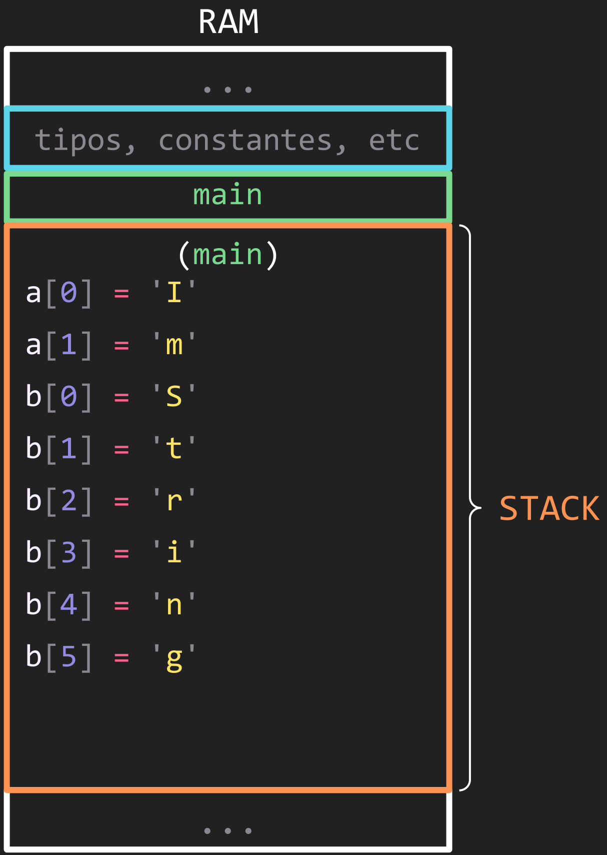
```
?
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

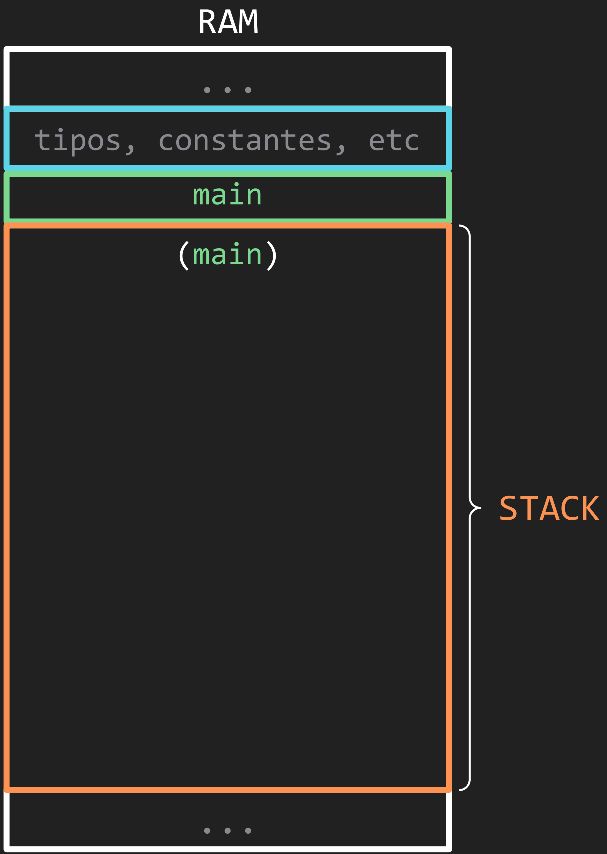
```
$ gcc main.c -o main
$ ./main
ImString String
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

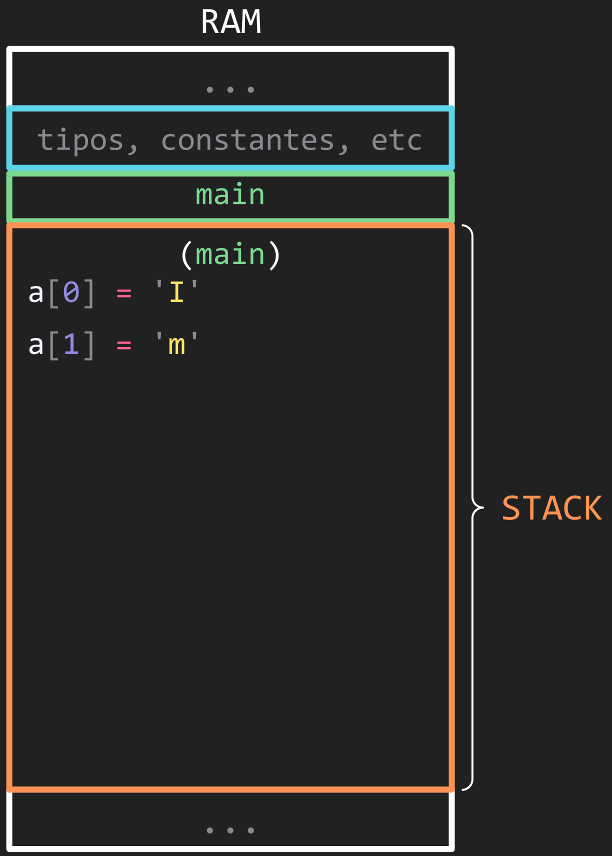
```
$ gcc main.c -o main
$ ./main
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

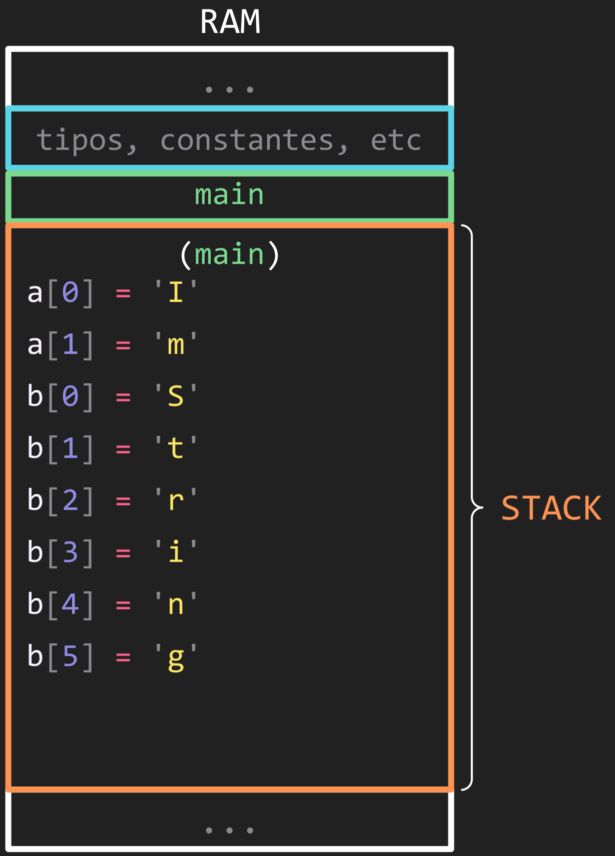
```
$ gcc main.c -o main
$ ./main
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

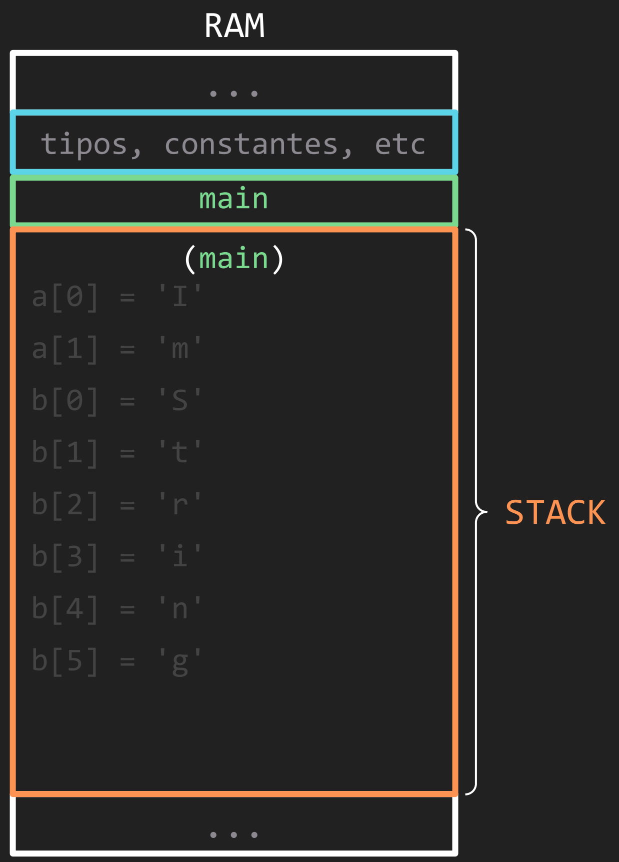
```
$ gcc main.c -o main
$ ./main
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

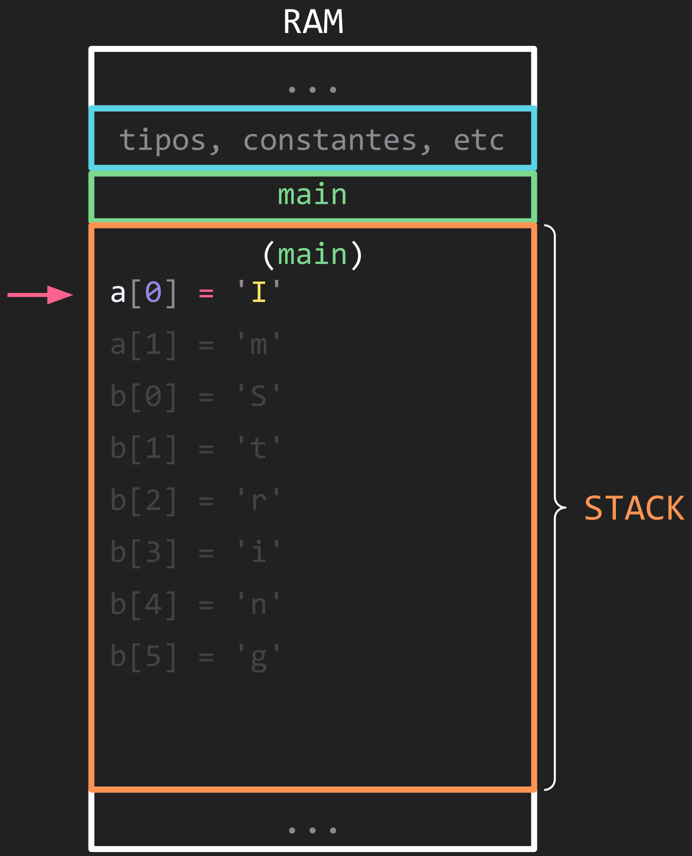
```
$ gcc main.c -o main
$ ./main
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

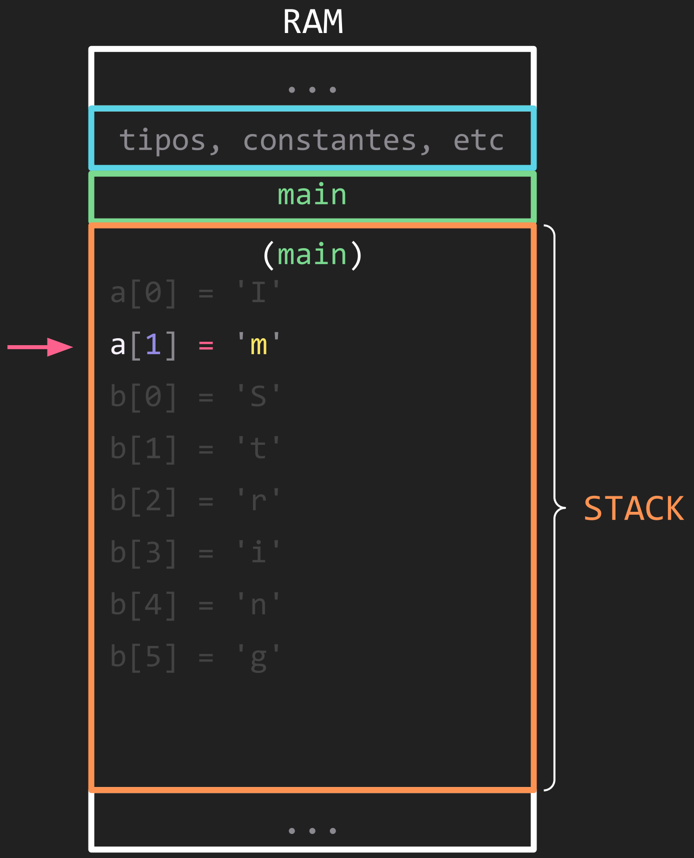
```
$ gcc main.c -o main
$ ./main
I
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

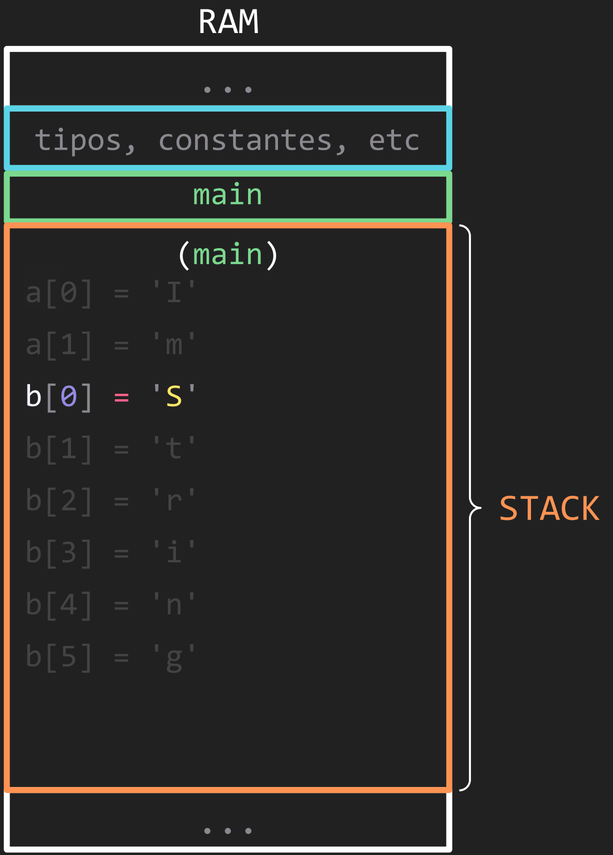
```
$ gcc main.c -o main
$ ./main
Im
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

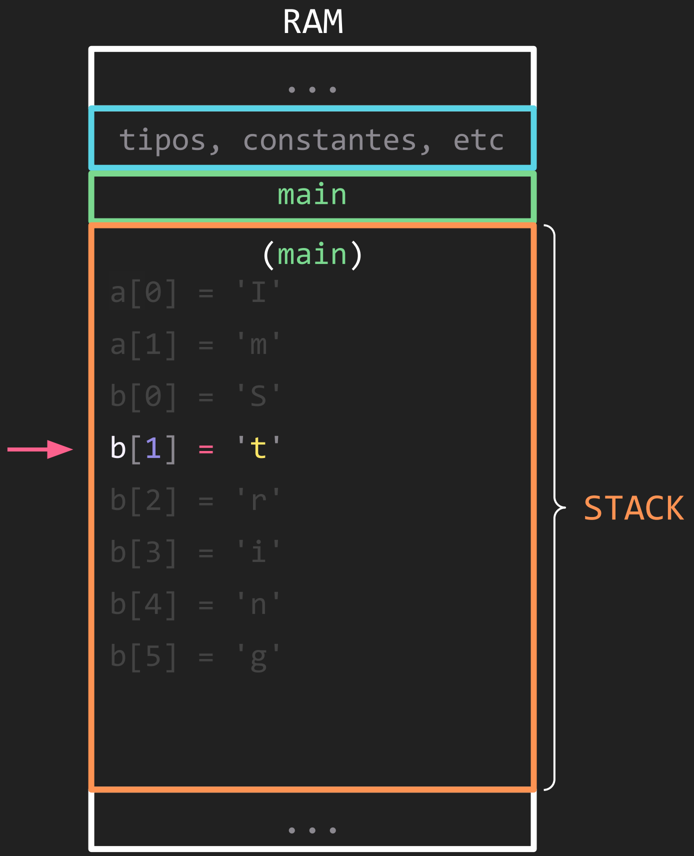
```
$ gcc main.c -o main
$ ./main
ImS
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

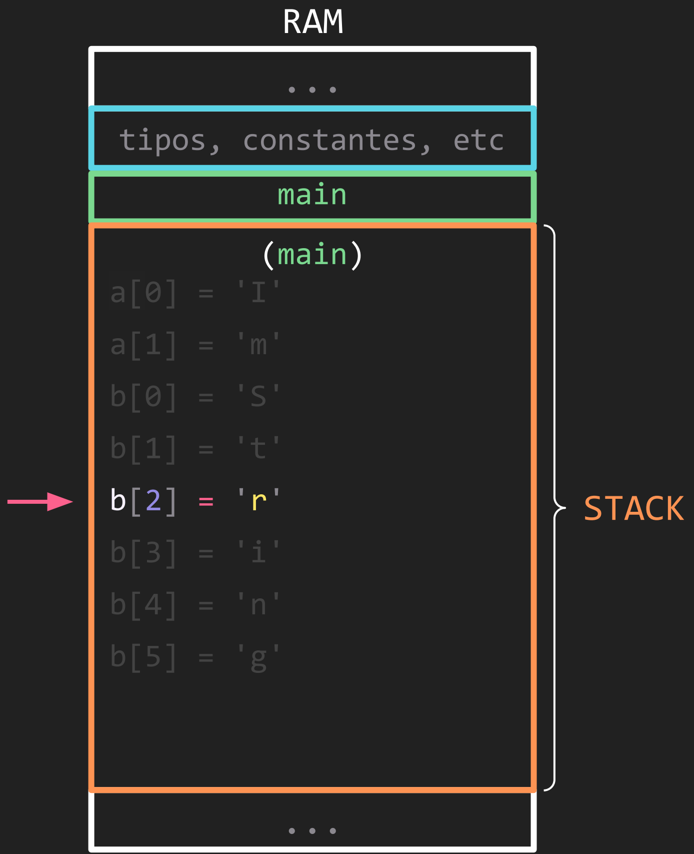
```
$ gcc main.c -o main
$ ./main
ImSt
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

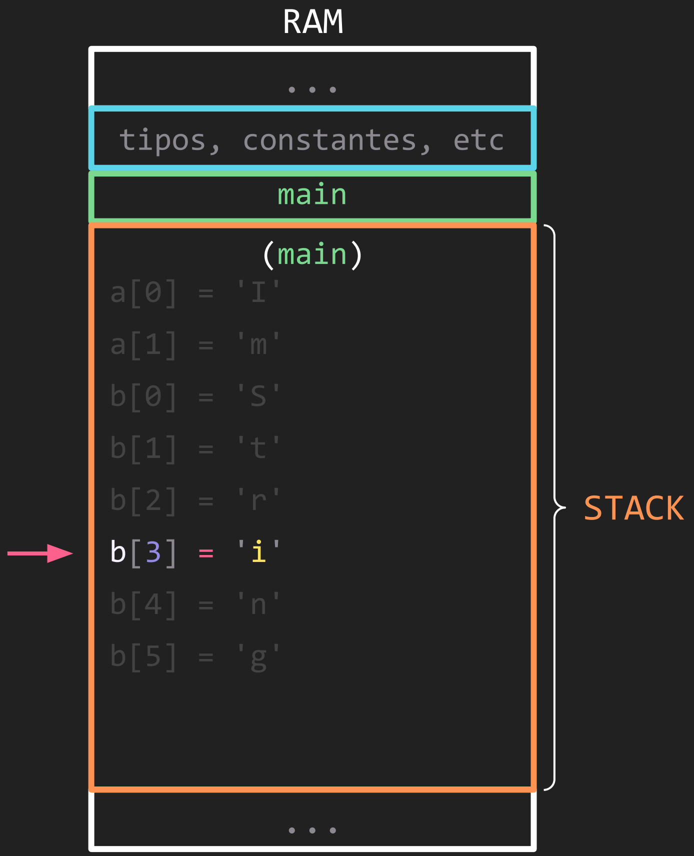
```
$ gcc main.c -o main
$ ./main
ImStr
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

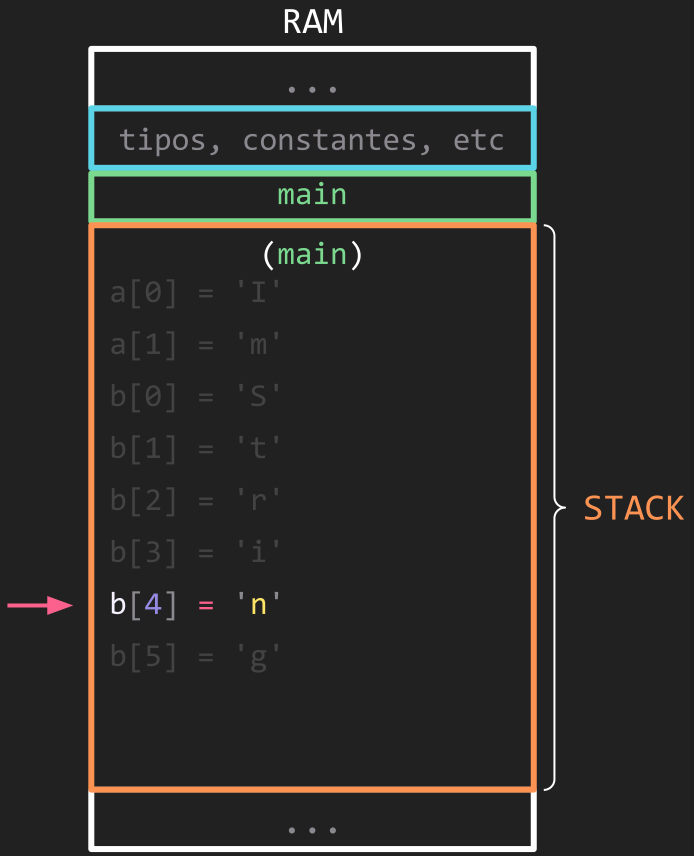
```
$ gcc main.c -o main
$ ./main
ImStri
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

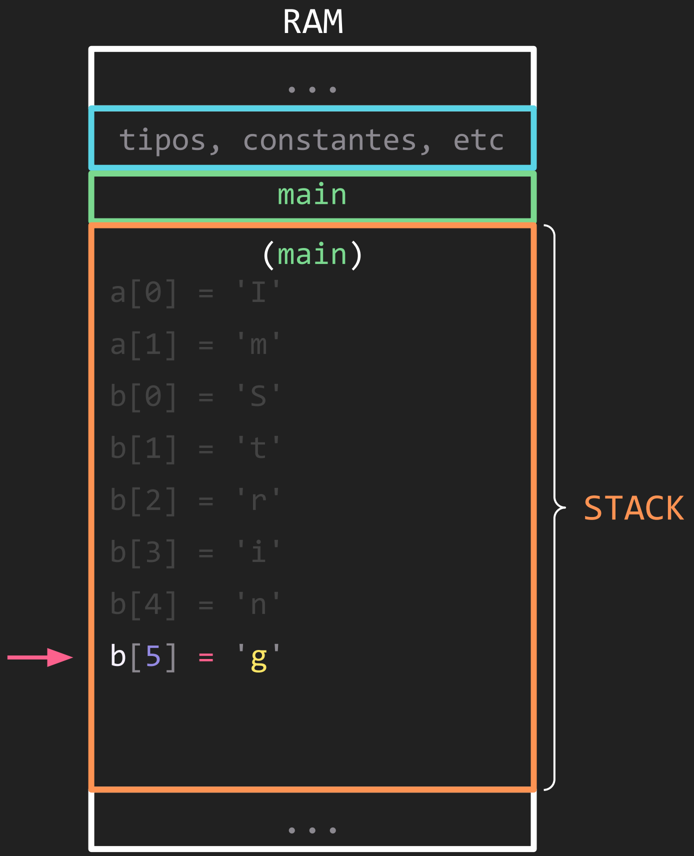
```
$ gcc main.c -o main
$ ./main
ImStrin
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

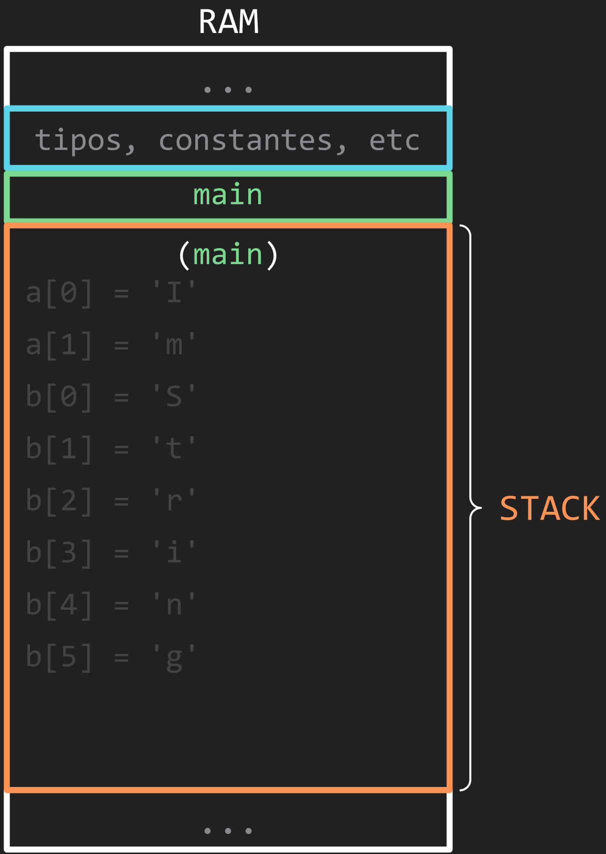
```
$ gcc main.c -o main
$ ./main
ImString
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

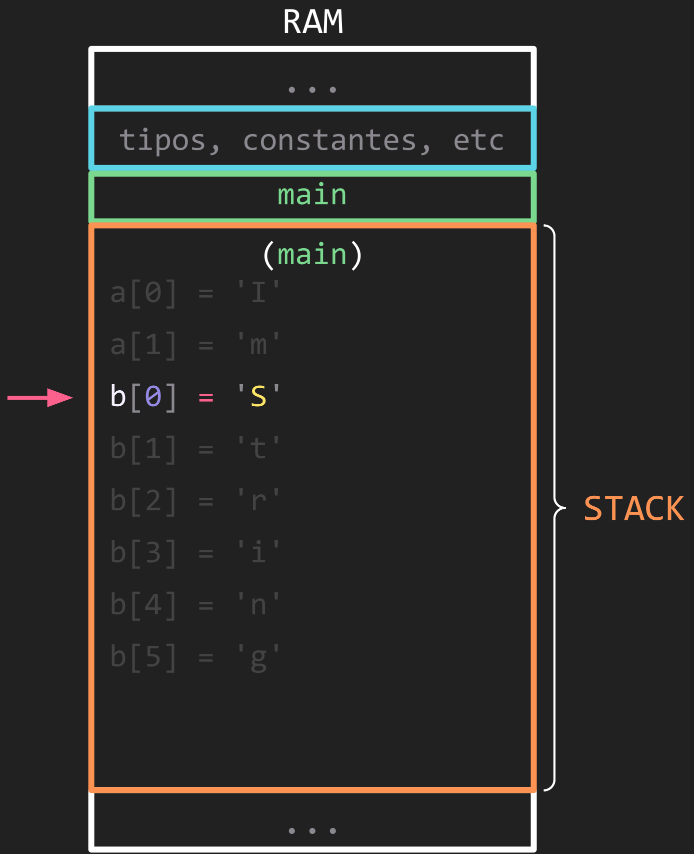
```
$ gcc main.c -o main
$ ./main
ImString
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

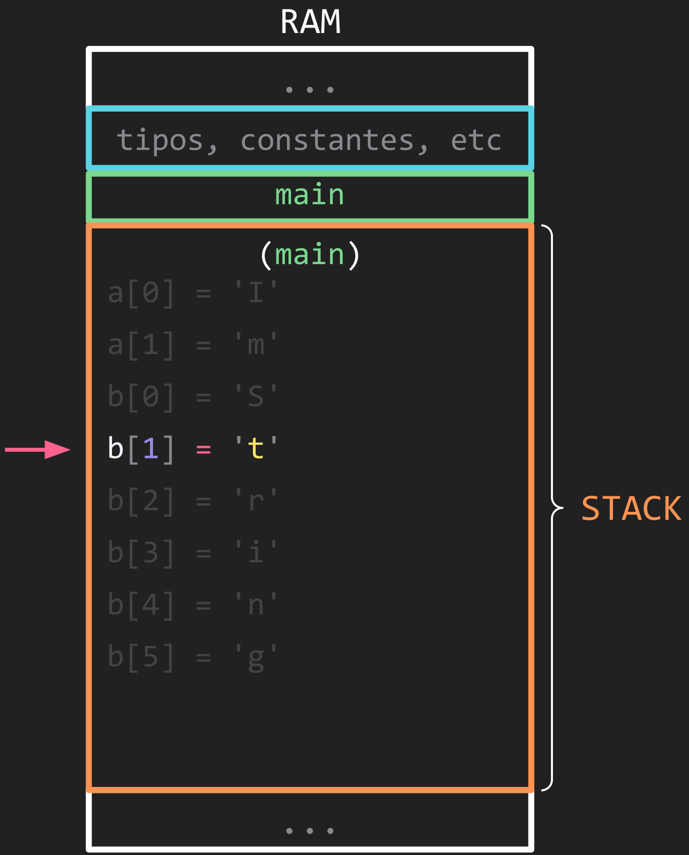
```
$ gcc main.c -o main
$ ./main
ImString S
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

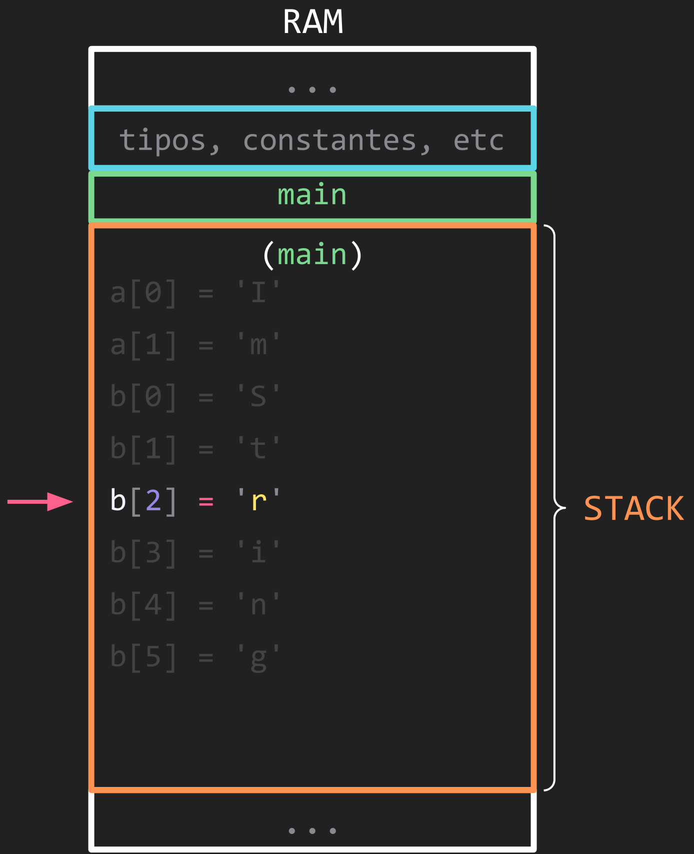
```
$ gcc main.c -o main
$ ./main
ImString St
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

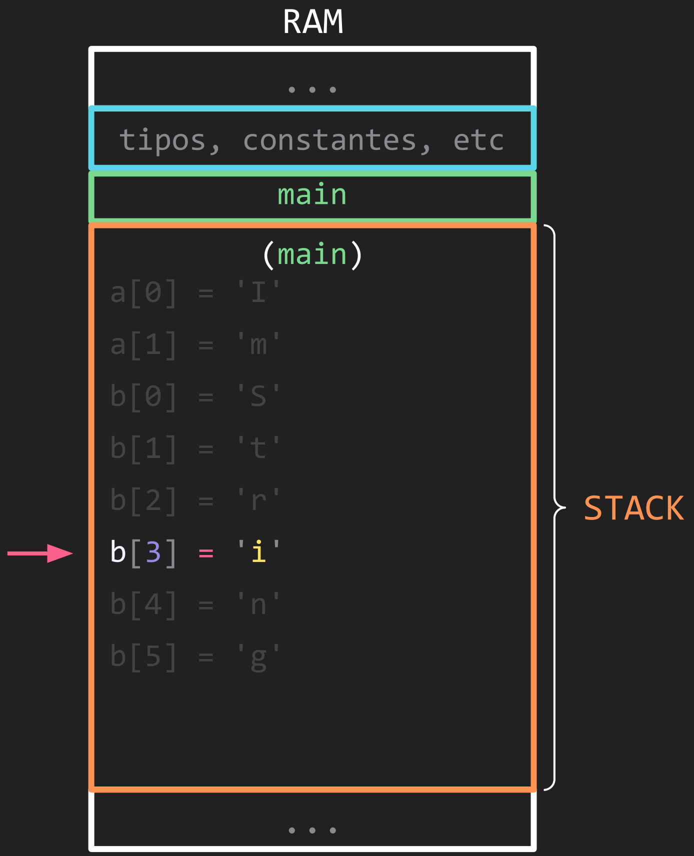
```
$ gcc main.c -o main
$ ./main
ImString Str
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

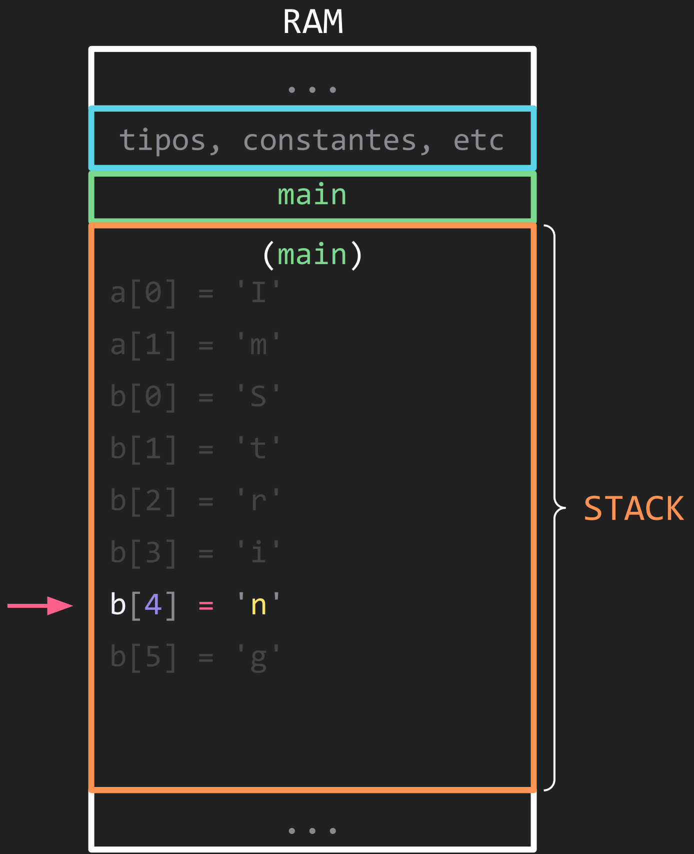
```
$ gcc main.c -o main
$ ./main
ImString Stri
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

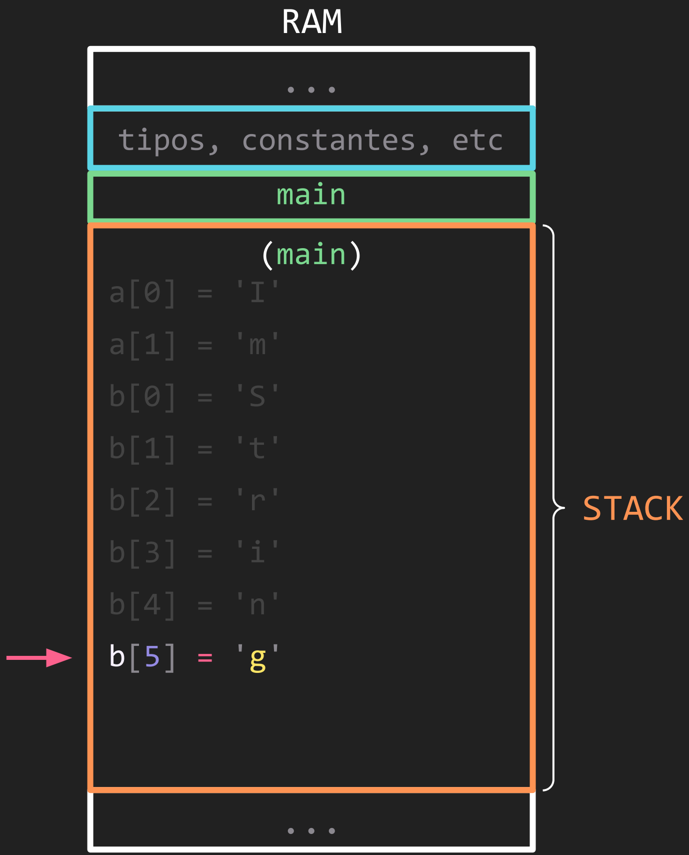
```
$ gcc main.c -o main
$ ./main
ImString Strin
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

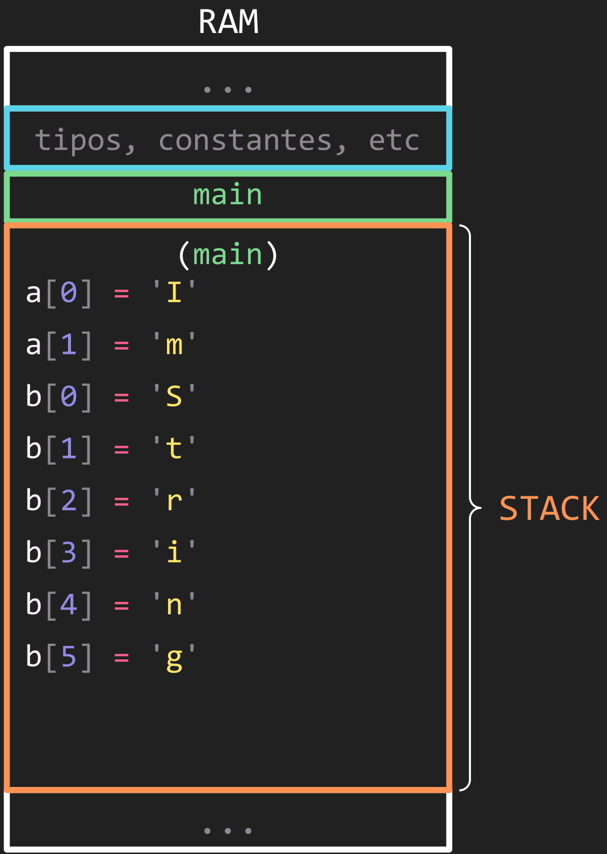
```
$ gcc main.c -o main
$ ./main
ImString String
```



Strings en C

```c
char a[2] = {'I', 'm'};
char b[6] = {'S', 't', 'r', 'i', 'n', 'g'};
printf("%s %s", a, b);
```

```
$ gcc main.c -o main
$ ./main
ImString String
```



Null Terminator

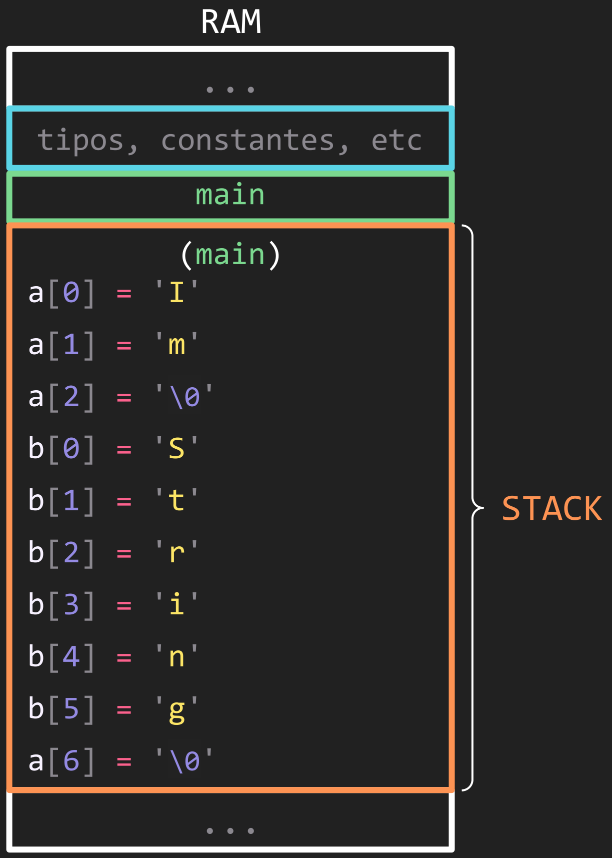
Null Terminator

Para poder recorrer correctamente un string, se le agrega un *char* especial al final, `'\0'`.

Null Terminator Explicito

```c
char a[3] = {'I', 'm', '\0'};
char b[7] = {'S', 't', 'r', 'i', 'n', 'g', '\0'};
printf("%s %s", a, b);
```

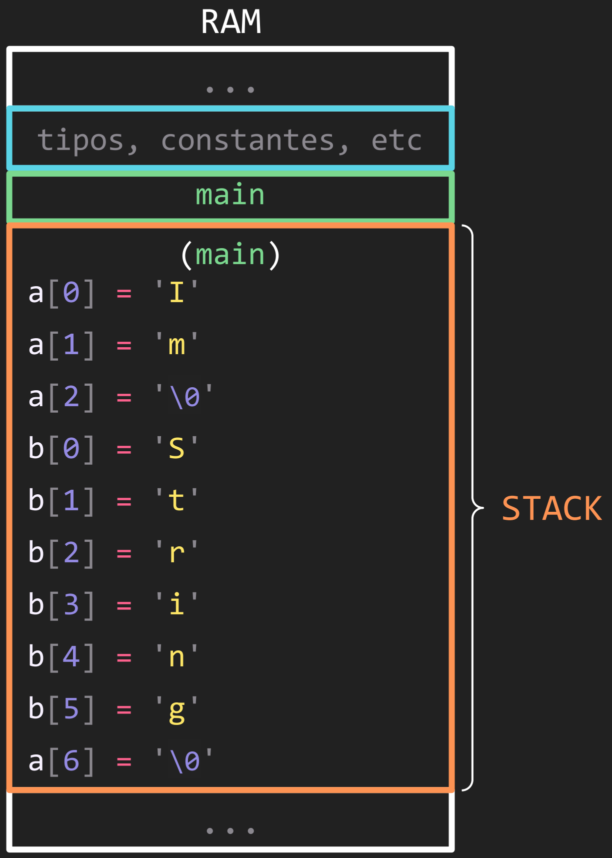
```
?
```



Null Terminator Explicito

```c
char a[3] = {'I', 'm', '\0'};
char b[7] = {'S', 't', 'r', 'i', 'n', 'g', '\0'};
printf("%s %s", a, b);
```

```
$ gcc main.c -o main
$ ./main
Im String
```



Null Terminator Implícito

En C un texto entre comillas dobles se conoce como string literal y el compilador automáticamente le pone null terminator.

```
"Hola" = "Hola\0"
```

Null Terminator Implícito

✗

```c
char a[4] = "Hola";
char b[4] = "como";
char c[6] = "estas?";
printf("%s %s %s\n", a, b, c);
```

```
$ gcc main.c -o main
$ ./main
Holacomoestas? comoestas? estas?
```

✓

```c
char a[5] = "Hola";
char b[] = "como";
char* c = "estas?"; // CONST
printf("%s %s %s\n", a, b, c);
```

```
$ gcc main.c -o main
$ ./main
Hola como estas?
```

Strings y funciones

Strings y funciones

```c
bool contains(char* string, int n, char c)
{
  for (int i = 0; i < n; i++)
  {
    if (string[i] == c) return true;
  }
  return false;
}

char *string = "Hello World!";
printf("%d ", contains(string, 12, '!'));
printf("%d\n", contains(string, 12, 'x'));
```

Como los strings son arreglos de *char*, se pueden utilizar en funciones de la misma forma.

Strings y funciones

```c
char *string = "Hello World!";
printf("%d ", contains(string, 12, '!'));
printf("%d\n", contains(string, 12, 'x'));
```

```
$ gcc main.c -o main
$ ./main
1 0
```

Esto nos permite facilitar mucho la sintáxis.

Strings y funciones

```c
void replace(char* string, int n, char from, char to)
{
  for (int i = 0; i < n; i+=1)
  {
    if (string[i] == from) string[i] = to;
  }
}

char string[] = "Sorry";
replace(string, 5, 'r', 'w');
printf("%s\n", string);
```

También podemos modificar un string en una función, solo que no puede ser un *char** literal

Strings y funciones

```c
char string[] = "Sorry";
replace(string, 5, 'r', 'w');
printf("%s\n", string);
```

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

Esto nos permite facilitar mucho la sintáxis.

Argumentos de Consola

Recordando Hello World

```c
#include <stdio.h>

int main(int argc, char** argv)
{
  printf("Hello world!\n");
  return 0;
}
```

- *argc*:
  Cantidad de argumentos
- *argv*:
  Argumentos (arreglo de strings)

Imprimiendo Argumentos

```c
int main(int argc, char** argv)
{
  printf("Recibi %d argumentos:\n", argc);
  for (int i = 0; i < argc; i++)
    printf("\t%s\n", argv[i]);
  return 0;
}
```

```
$ gcc main.c -o main
$ ./main Hello World!
Recibi 3 argumentos:
	./main
	Hello
	World!
```

Recordar:

- El primer elemento de argv siempre es el nombre del ejecutable
- Los elementos de argv siempre son strings

Guardando Argumentos

```c
int main(int argc, char** argv)
{
  char* string = argv[1];
  int number = argv[2];
  printf("%s %d\n", string, number);
  return 0;
}
```

```
?
```

Como argv es un arreglo, podemos guardar su contenido en variables.

Guardando Argumentos

```c
int main(int argc, char** argv)
{
  char* string = argv[1];
  int number = argv[2];
  printf("%s %d\n", string, number);
  return 0;
}
```

```
$ gcc main.c -o main
$ ./main cien 100
cien -1034752593
```

Estamos igualando un *char** (argv[2]) a un *int* (number).

💀

atoi - ASCII to *int*

```c
int main(int argc, char** argv)
{
  char* string = argv[1];
  int number = atoi(argv[2]);
  printf("%s %d\n", string, number);
  return 0;
}
```

```
$ gcc main.c -o main
$ ./main cien 100
cien 100
```

Dentro de <stdlib.h>, se encuentra la función atoi que recibe un string ASCII y retorna un *int*.

atof - ASCII to *float*

```c
int main(int argc, char** argv)
{
  char* string = argv[1];
  float number = atof(argv[2]);
  printf("%s %f\n", string, number);
  return 0;
}
```

```
$ gcc main.c -o main
$ ./main test 3.56
test 3.560000
```

Dentro de <stdlib.h>, se encuentra la función atof que recibe un string ASCII y retorna un *float*.

¡Muchas Gracias!

YOU HAVE C
YOU HAVE A LIBRARY FOR A CASTLE & A LIBRARY FOR THE PRINCESS - CHARGE!
YOU RESCUE THE PRINCESS HER DOG, HER ENTIRE WARDROBE & EVERYTHING SHE HAS EVER EATEN...
FUCK - DID I FORGET A NULL-TERMINATOR ?
MART VIRKUS '16
toggl

With 💜 by @vichoeq & @KnowYourselves