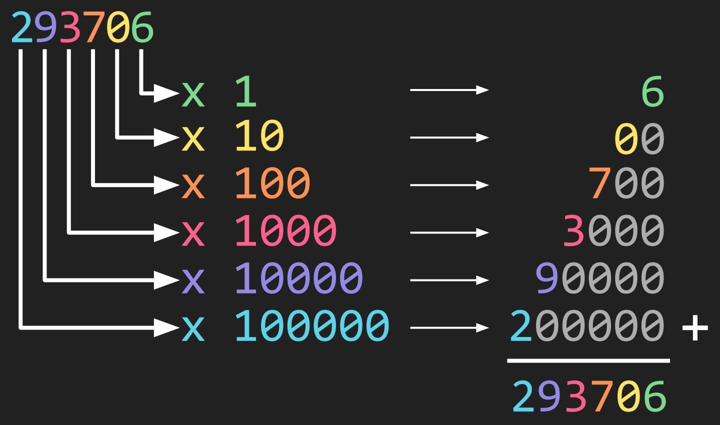
Intro a C
Punteros

With 💜 by @vichoeq & @KnowYourselves

Bases Numéricas #2

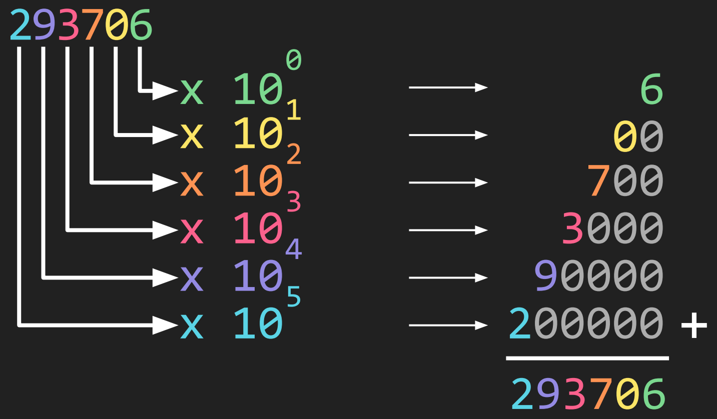
Base Decimal

Los números que usamos normalmente son en base 10



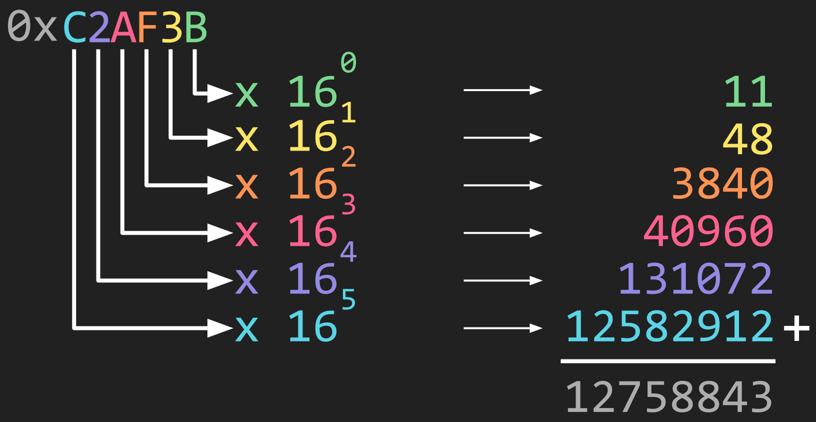
Base Decimal

Los números que usamos normalmente son en base 10



Base Hexadecimal

La base 16 usa los dígitos A, B, C, D, E y F para valores sobre 9



Es muy útil para representar números extremadamente grandes.

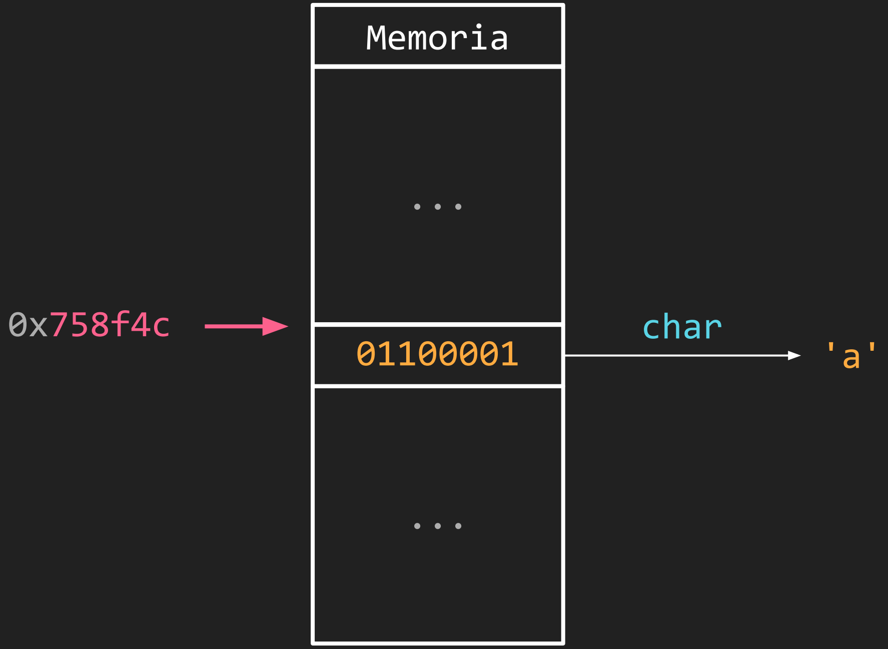
Variables en un programa

Cada variable tiene:

- Una dirección de memoria
- Un tipo
- Un valor

El tipo indica dos cosas:

- Tamaño en memoria
- Interpretación

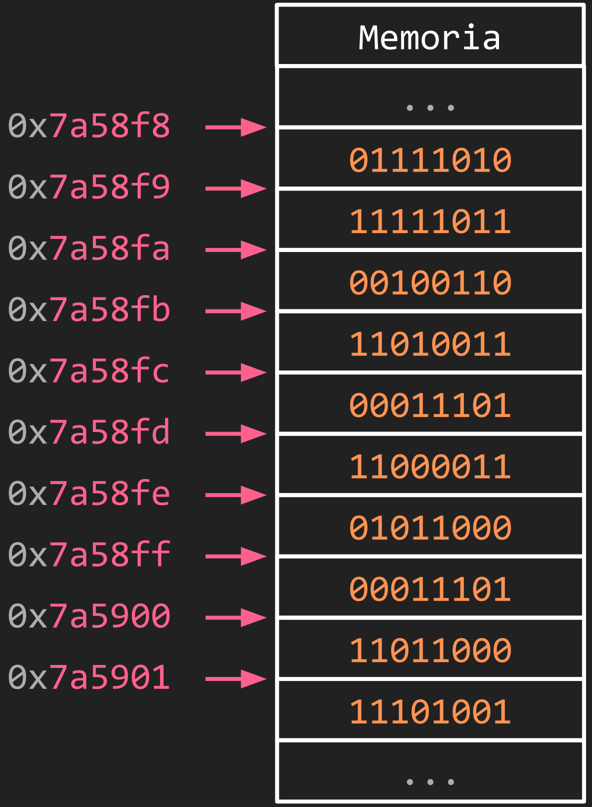


Dirección de memoria

Cada variable tiene:

- Una dirección de memoria
- Un tipo
- Un valor

Puede pensarse en la memoria como una tabla gigante que asocia direcciones a bytes.

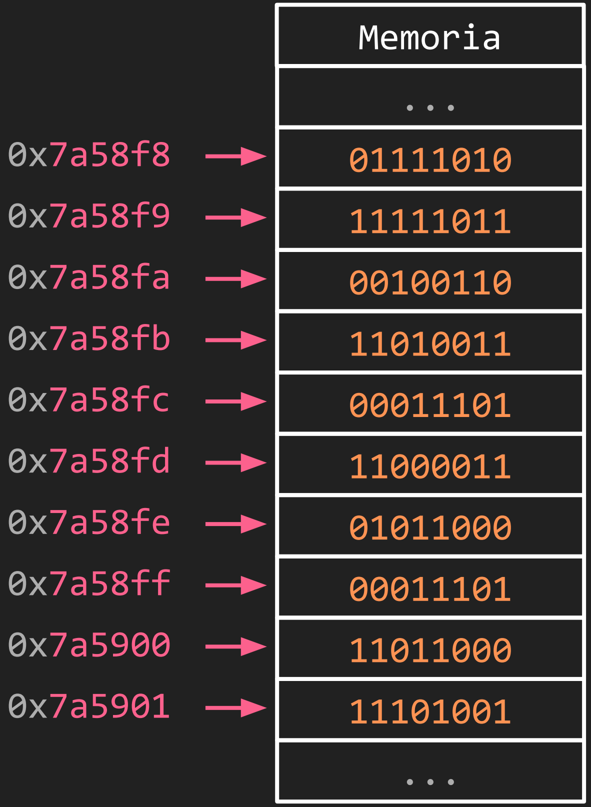


Dirección de memoria

Las direcciones de memoria apuntan al comienzo de un bloque.

Para efectos de los diagramas, las flechas apuntarán al centro del primer valor de un bloque.

Además las direcciones se mostrarán con 6 dígitos hexadecimales en lugar de 12



Punteros

Punteros

En C un puntero es una variable cuyo valor es la dirección de memoria de otra variable.

&var - ¿Dónde está var?

```c
int a = 5;
int b = 7;
printf("%p\n", &a);
printf("%p\n", &b);
```

```
$ gcc main.c -o main
$ ./main
0x7a85f8
0x7a85fc
```

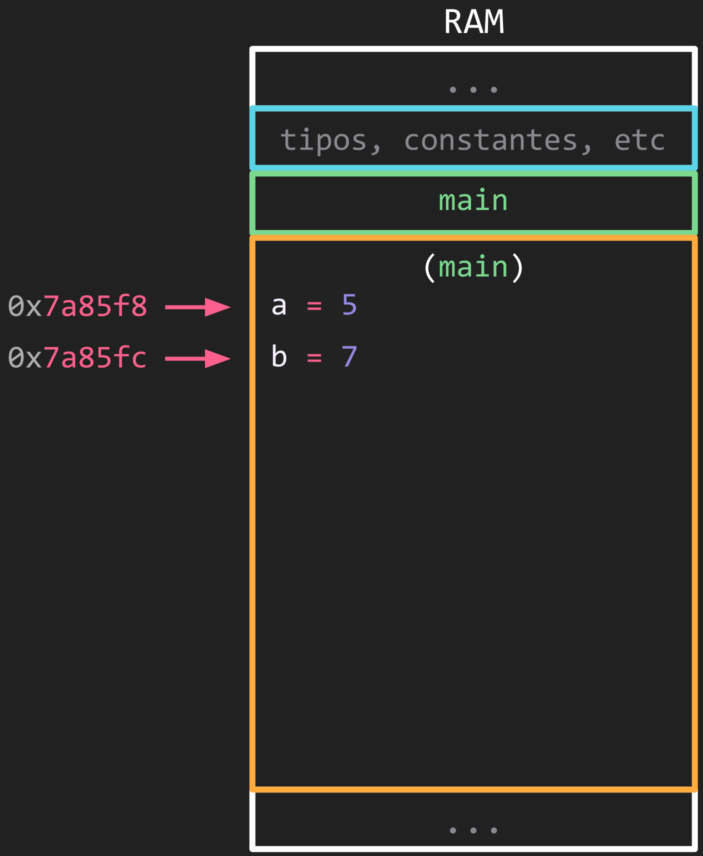
El operador & nos retorna la dirección de memoria de una variable.

Para imprimir: %p

&var - ¿Dónde está var?

```c
int a = 5;
int b = 7;
printf("%p\n", &a);
printf("%p\n", &b);
```

```
$ gcc main.c -o main
$ ./main
0x7a85f8
0x7a85fc
```



*type** - puntero a *type*

```c
int c = 12;
int* d = &c;
printf("%p = %p\n", d, &c);
```

```
$ gcc main.c -o main
$ ./main
0x9b85f8 = 0x9b85f8
```

El símbolo * luego de un *tipo* se refiere a un *tipo* especial, un puntero.

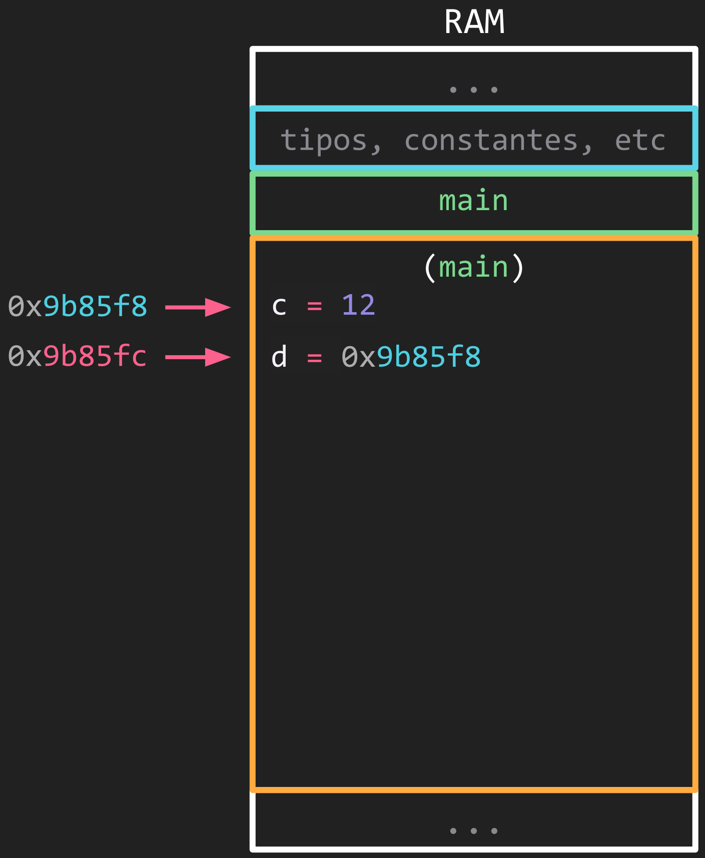
Un puntero es sólo la dirección de memoria a donde apunta.

Tamaño: 64 bits (8 bytes)

*type** - puntero a *type*

```c
int c = 12;
int* d = &c;
printf("%p = %p\n", d, &c);
```

```
$ gcc main.c -o main
$ ./main
0x9b85f8 = 0x9b85f8
```



*ptr - acceder a dirección ptr

```c
int e = 3;
int* f = &e;
*f += 1;
printf("%d\n", e);
```

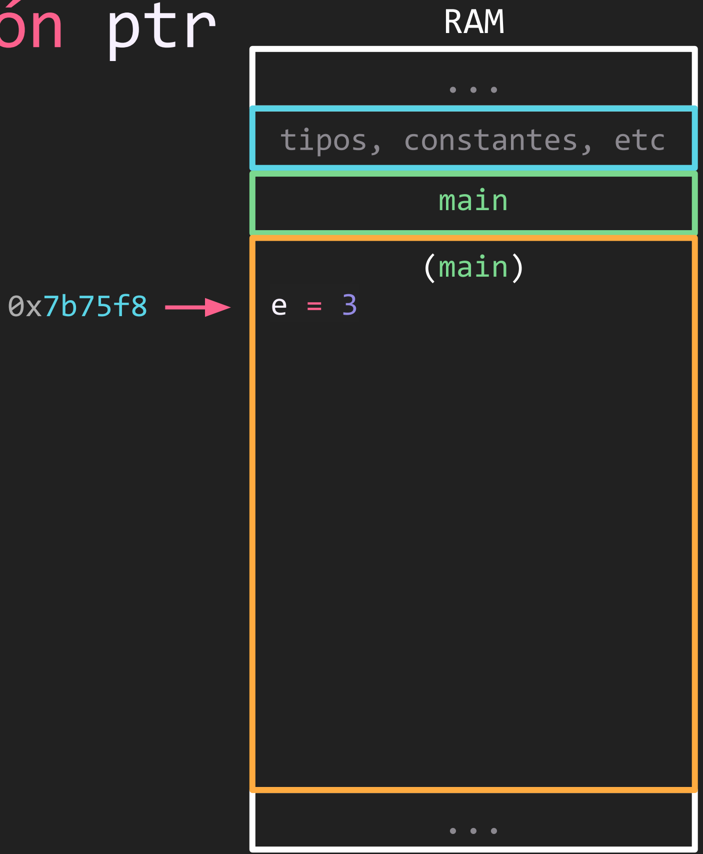
```
?
```

El operador * nos permite acceder a la variable guardada en una dirección de memoria.

*ptr - acceder a dirección ptr

```c
int e = 3;
int* f = &e;
*f += 1;
printf("%d\n", e);
```

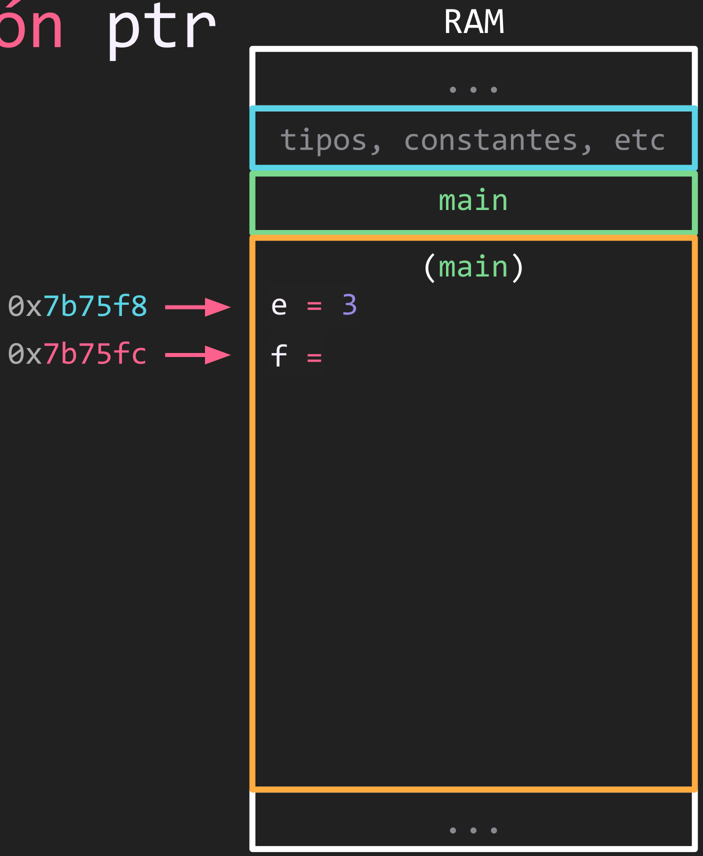
```
$ gcc main.c -o main
$ ./main
```



*ptr - acceder a dirección ptr

```c
int e = 3;
int* f = &e;
*f += 1;
printf("%d\n", e);
```

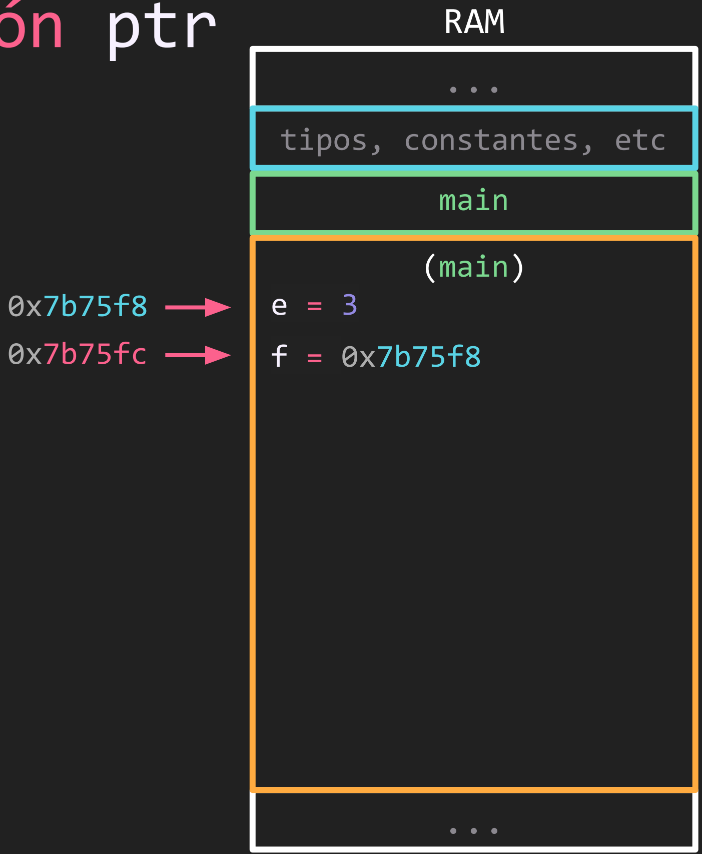
```
$ gcc main.c -o main
$ ./main
```



*ptr - acceder a dirección ptr

```c
int e = 3;
int* f = &e;
*f += 1;
printf("%d\n", e);
```

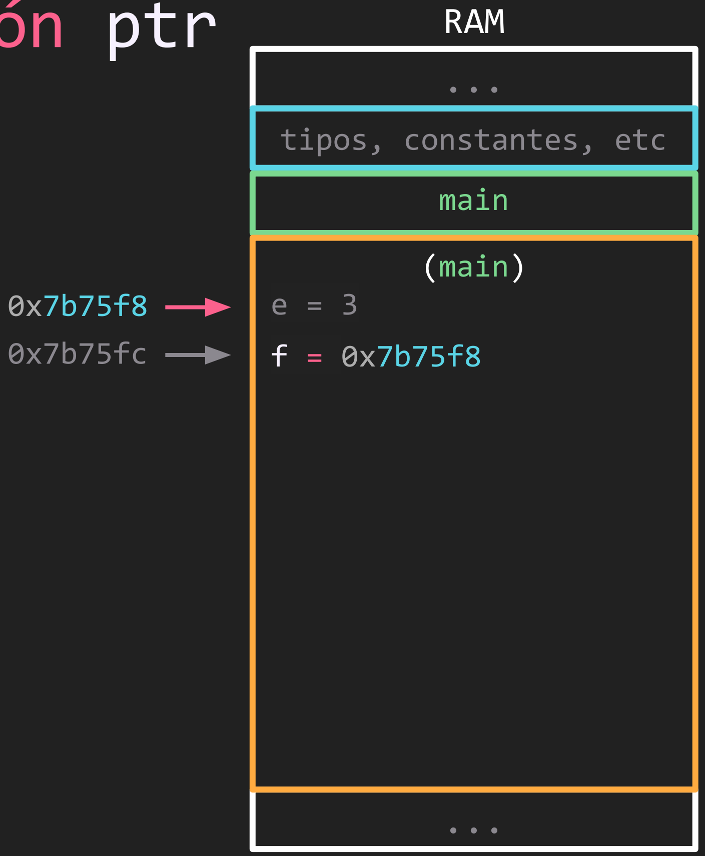
```
$ gcc main.c -o main
$ ./main
```



*ptr - acceder a dirección ptr

```c
int e = 3;
int* f = &e;
*f += 1;
printf("%d\n", e);
```

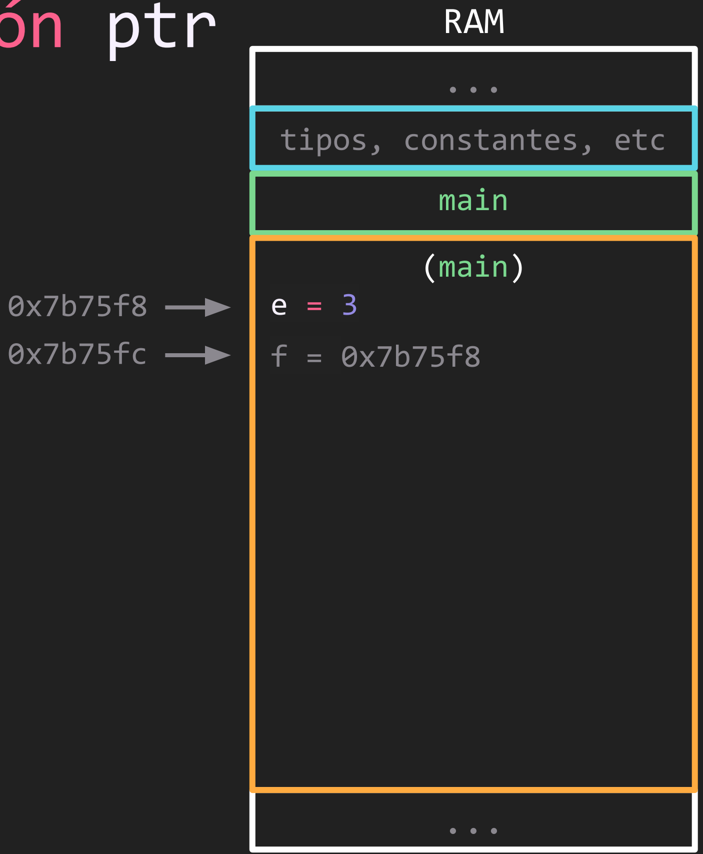
```
$ gcc main.c -o main
$ ./main
```



*ptr - acceder a dirección ptr

```c
int e = 3;
int* f = &e;
*f += 1;
printf("%d\n", e);
```

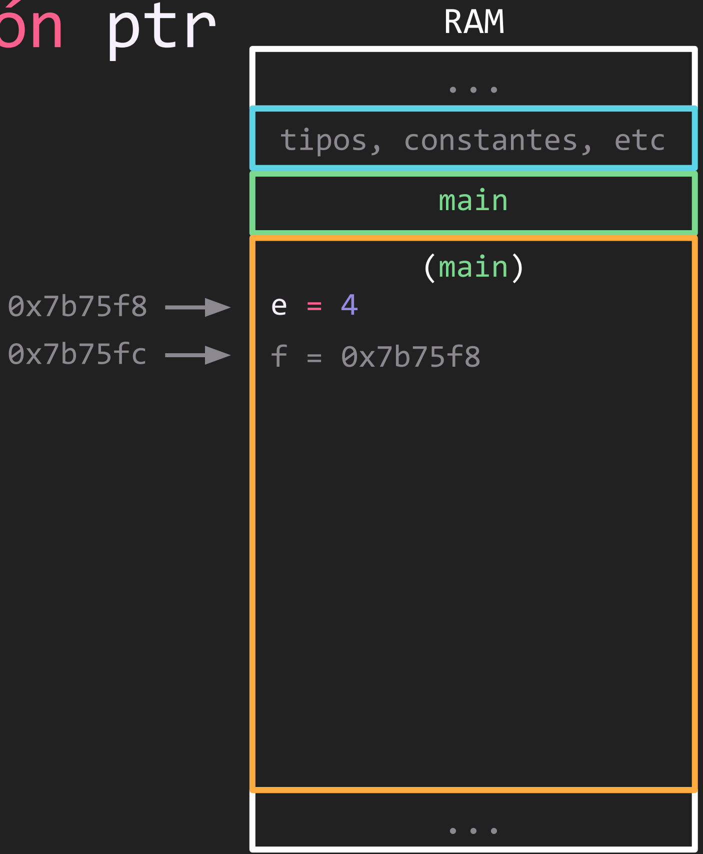
```
$ gcc main.c -o main
$ ./main
```



*ptr - acceder a dirección ptr

```c
int e = 3;
int* f = &e;
*f += 1;
printf("%d\n", e);
```

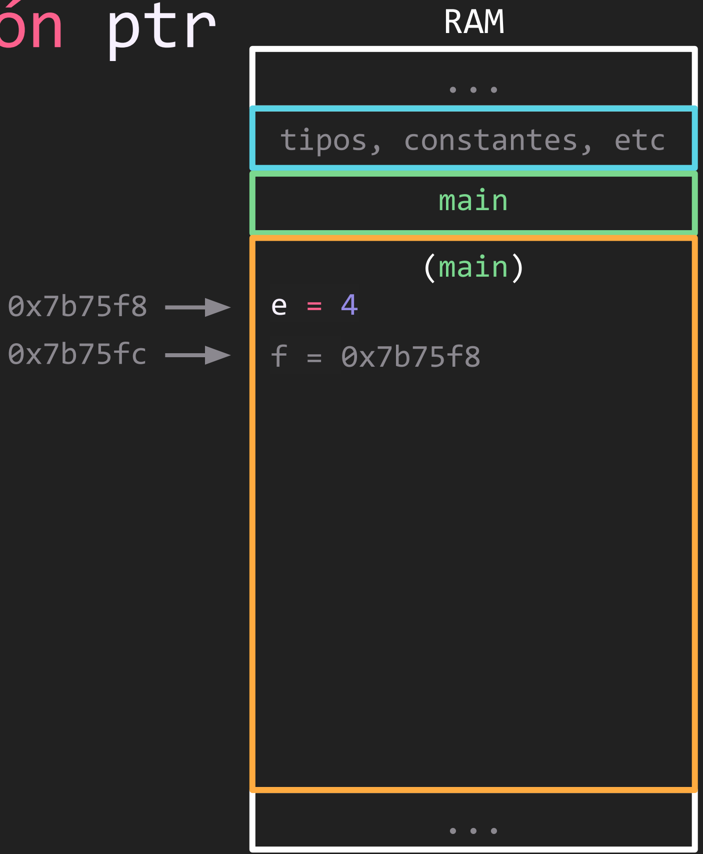
```
$ gcc main.c -o main
$ ./main
```



*ptr - acceder a dirección ptr

```c
int e = 3;
int* f = &e;
*f += 1;
printf("%d\n", e);
```

```
$ gcc main.c -o main
$ ./main
4
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

```
?
```

Ya que son variables, podemos definir punteros a punteros.

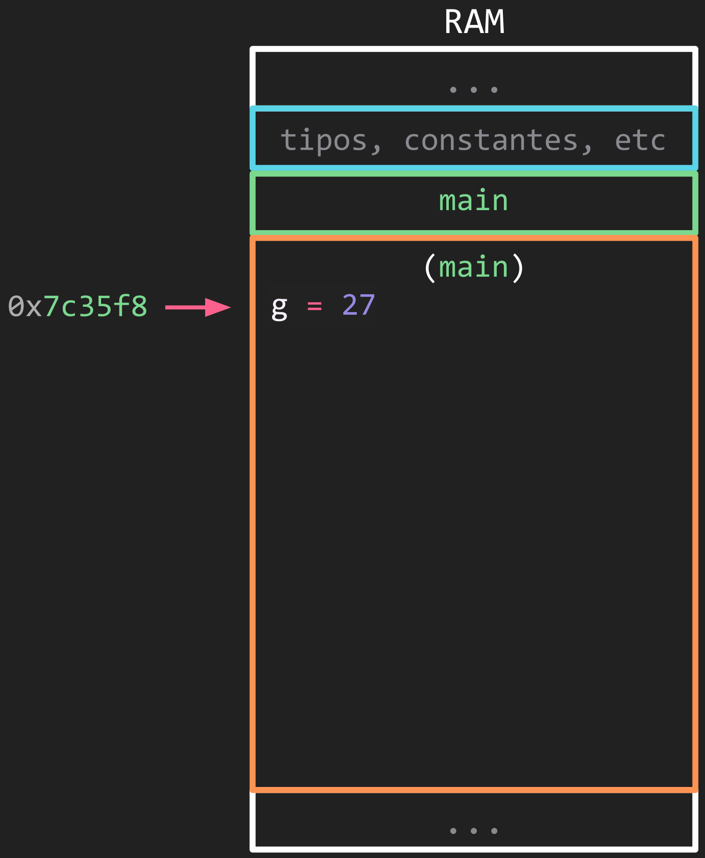
El *tipo* de estos simplemente tiene un * adicional.

Para acceder a ellos también se usa un * adicional.

Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

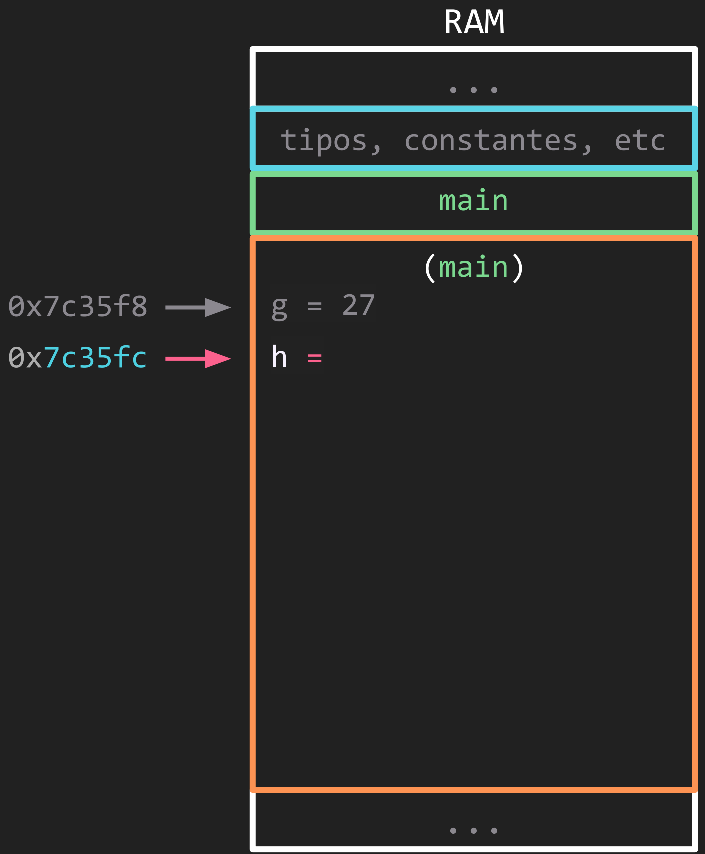
```
$ gcc main.c -o main
$ ./main
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

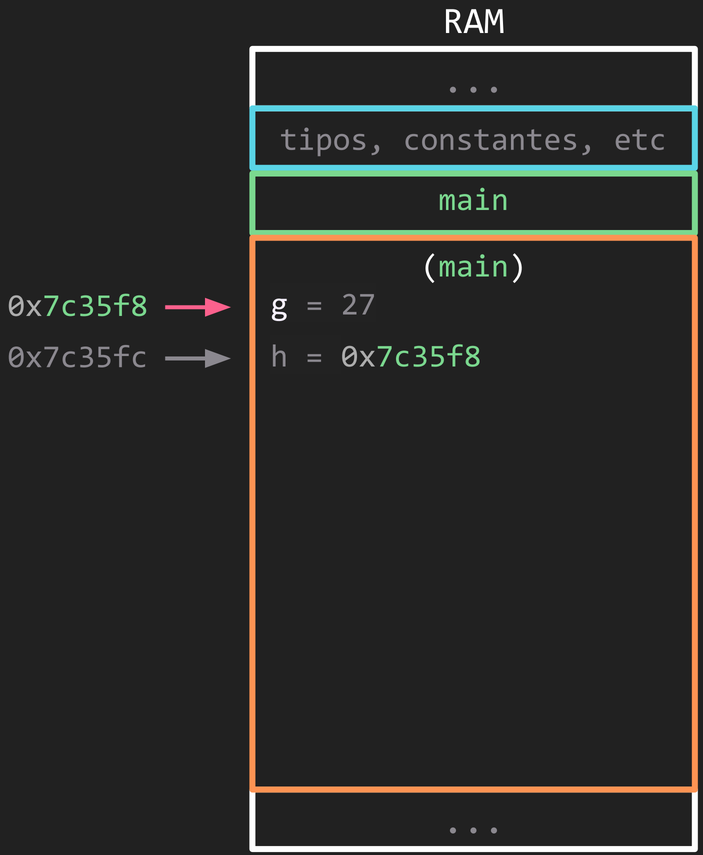
```
$ gcc main.c -o main
$ ./main
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

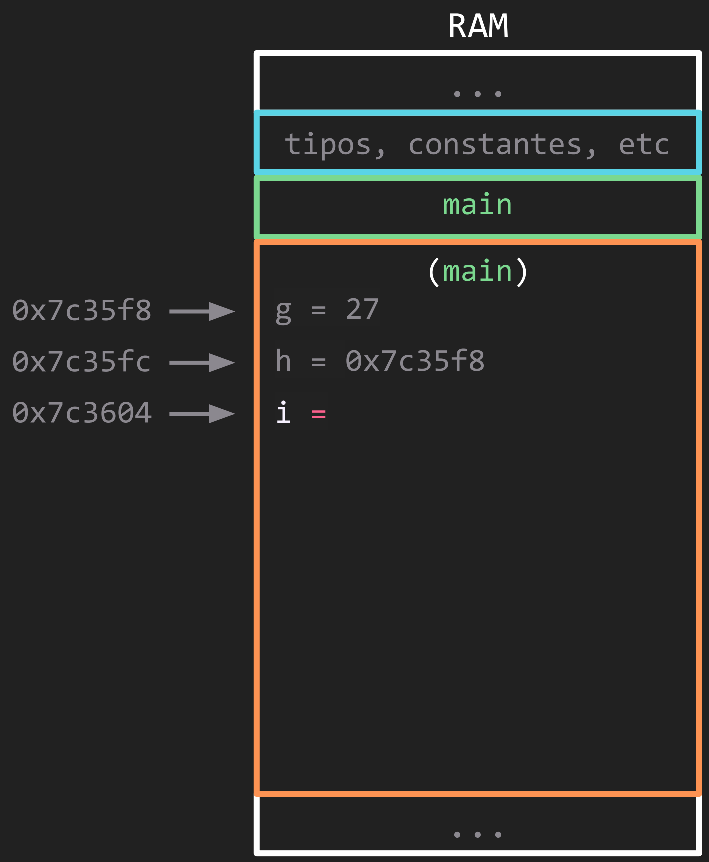
```
$ gcc main.c -o main
$ ./main
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

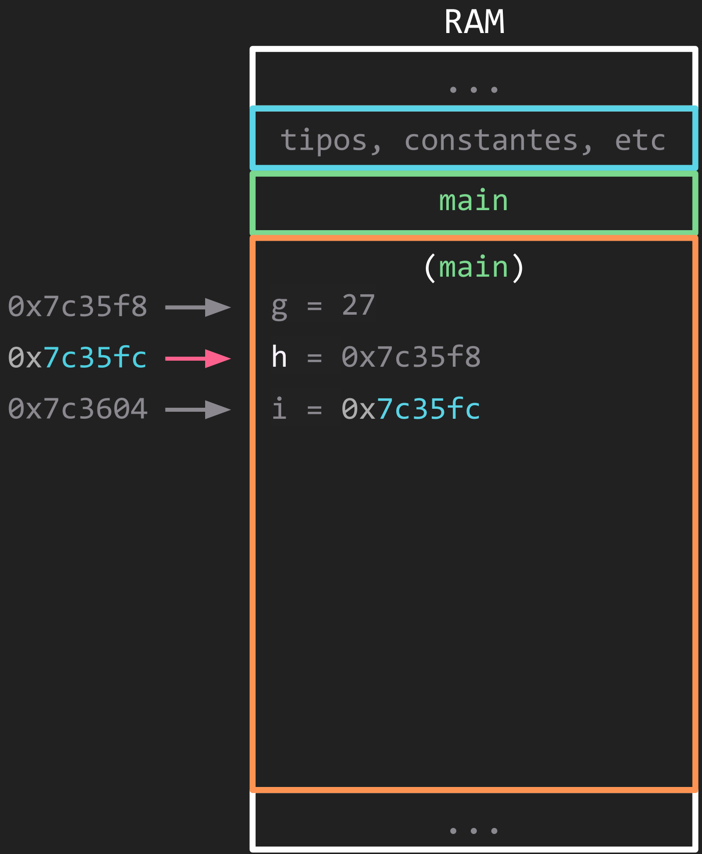
```
$ gcc main.c -o main
$ ./main
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

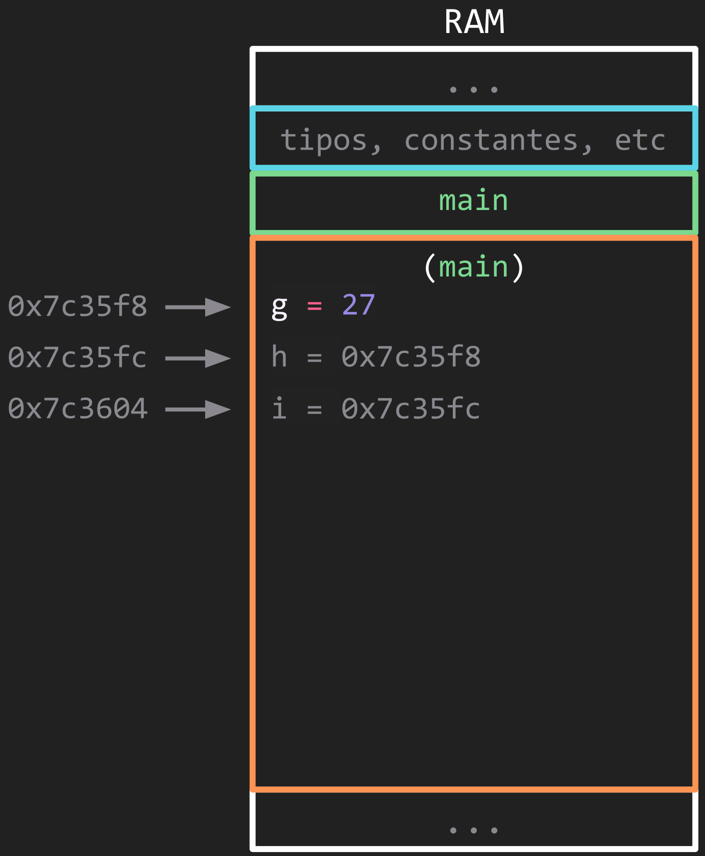
```
$ gcc main.c -o main
$ ./main
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

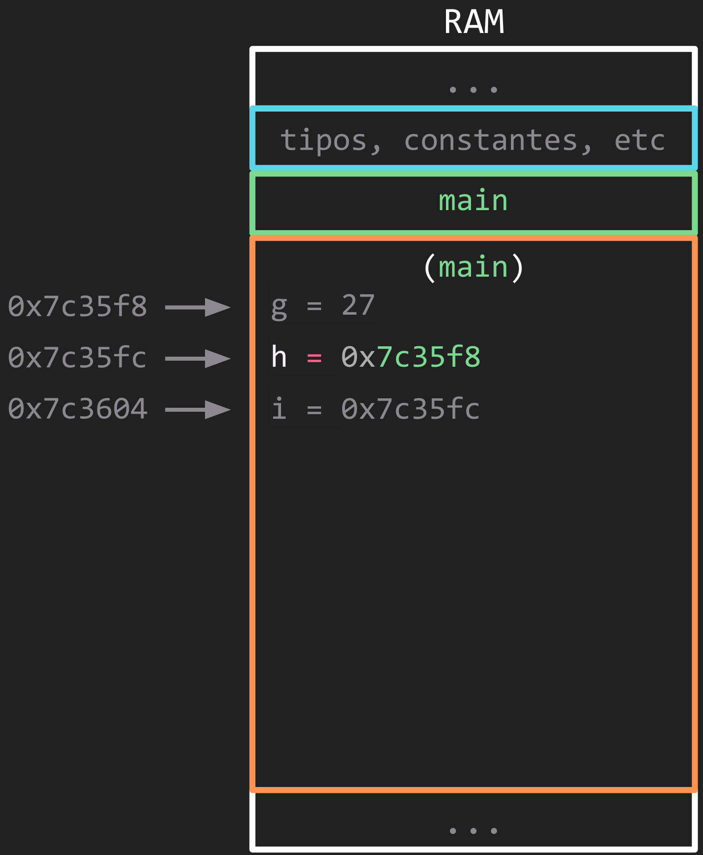
```
$ gcc main.c -o main
$ ./main
27
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

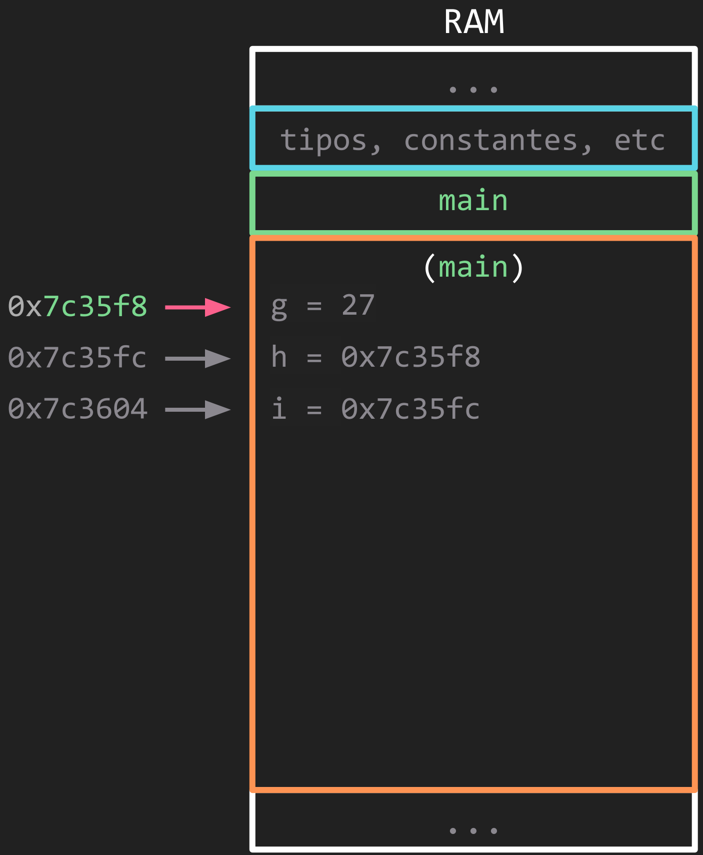
```
$ gcc main.c -o main
$ ./main
27
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

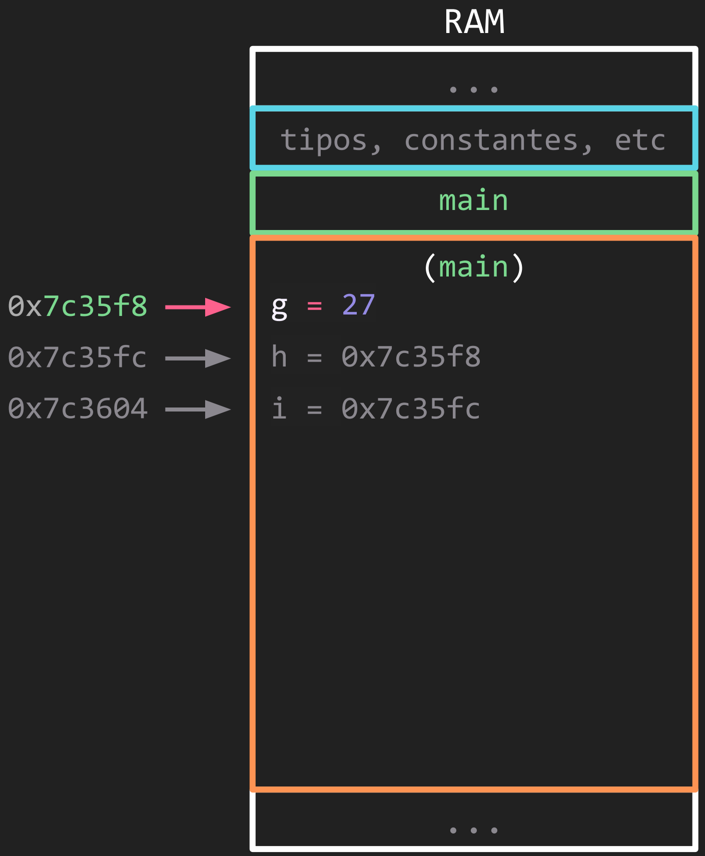
```
$ gcc main.c -o main
$ ./main
27
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

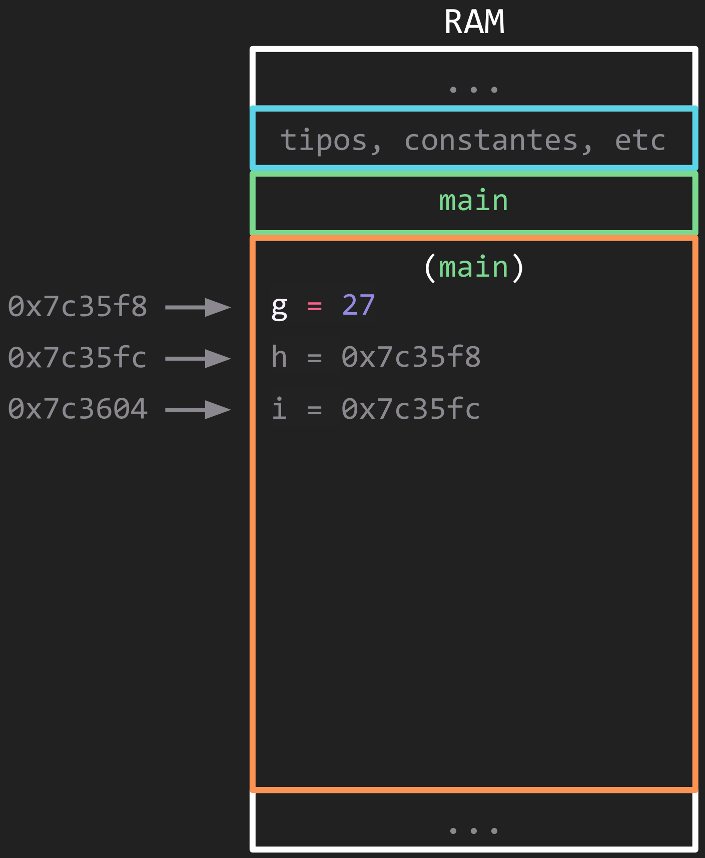
```
$ gcc main.c -o main
$ ./main
27
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

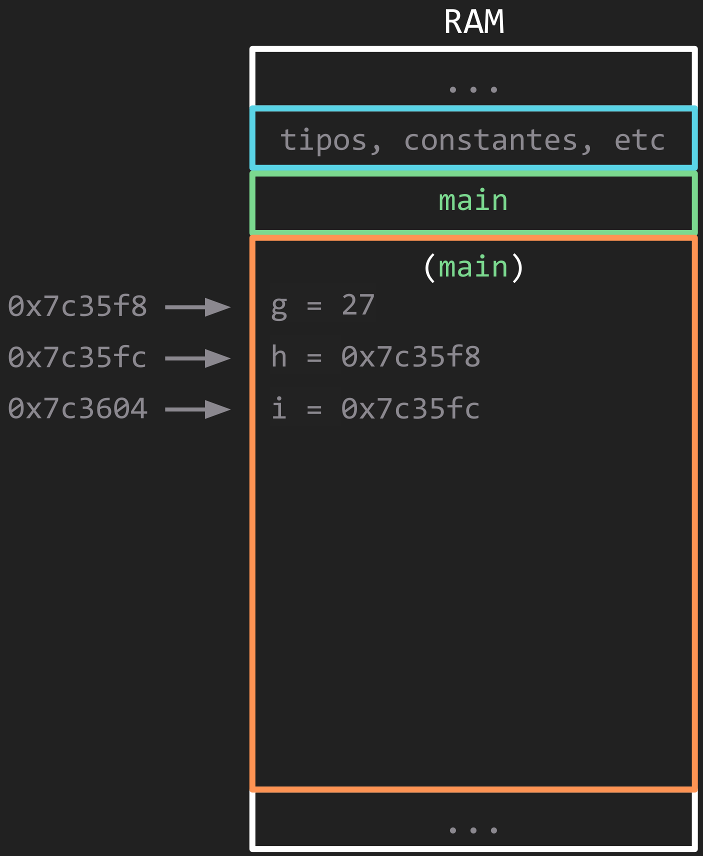
```
$ gcc main.c -o main
$ ./main
27
27
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

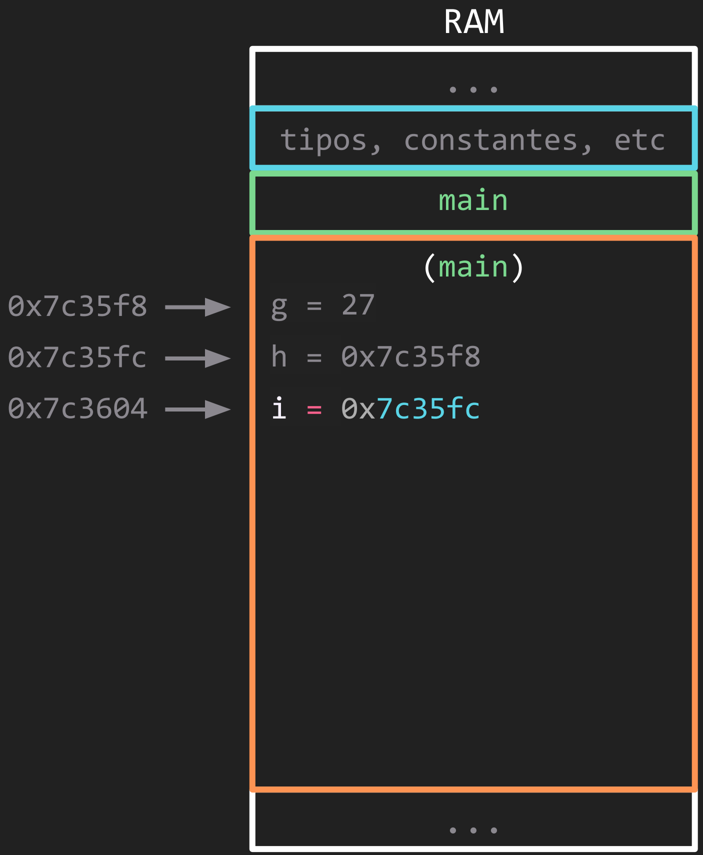
```
$ gcc main.c -o main
$ ./main
27
27
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

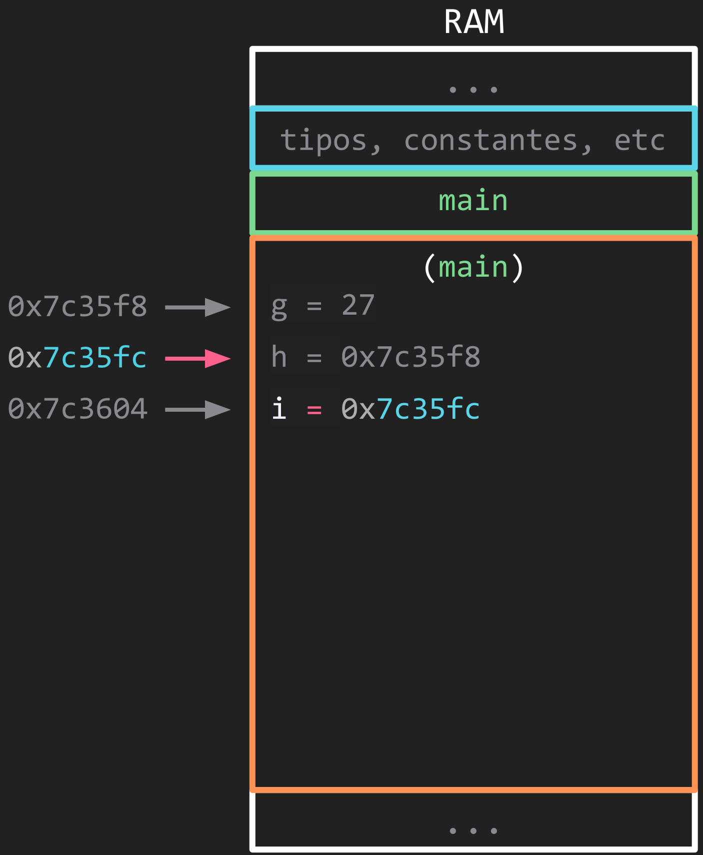
```
$ gcc main.c -o main
$ ./main
27
27
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

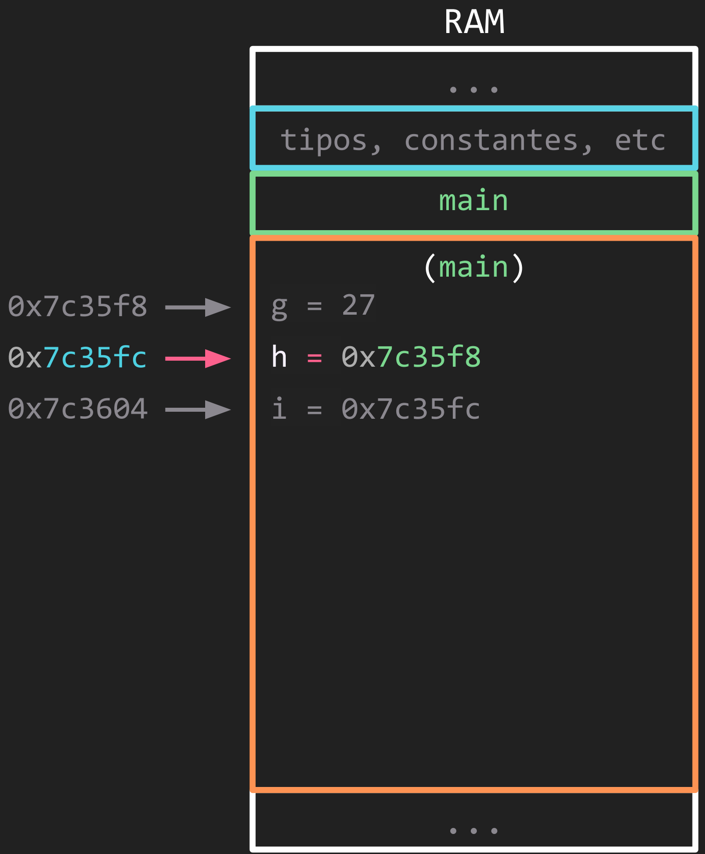
```
$ gcc main.c -o main
$ ./main
27
27
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

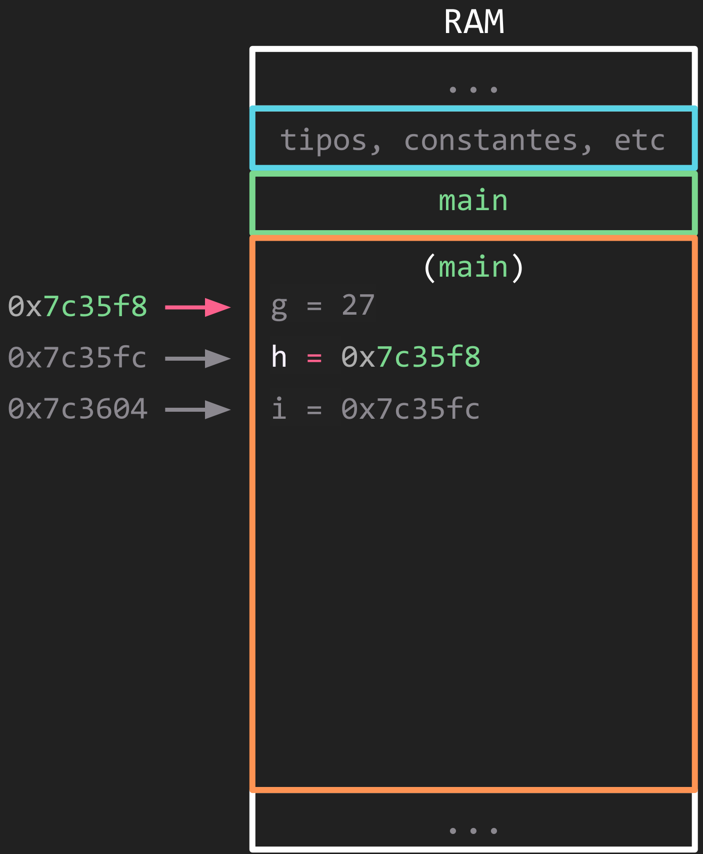
```
$ gcc main.c -o main
$ ./main
27
27
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

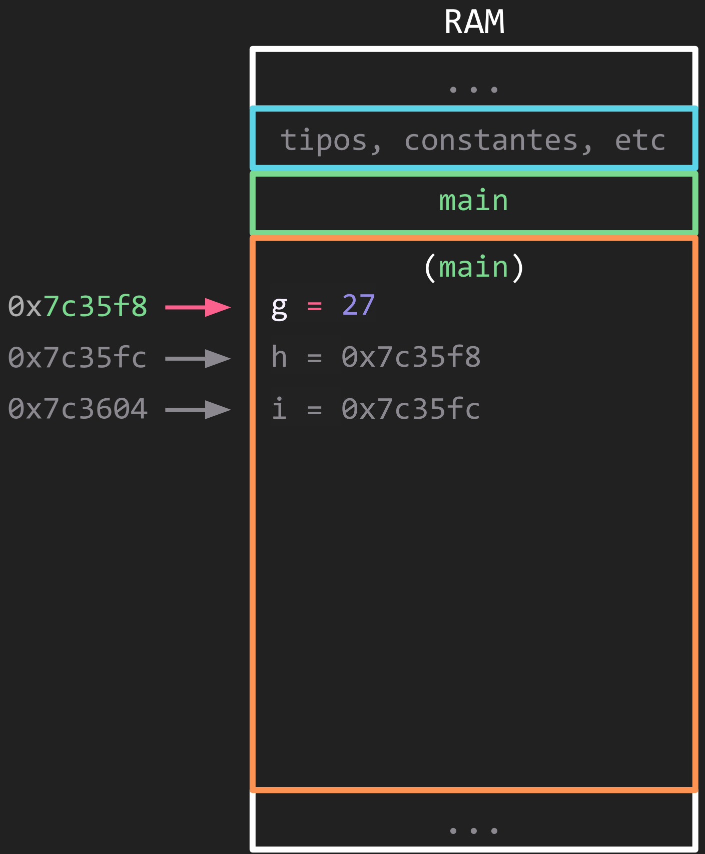
```
$ gcc main.c -o main
$ ./main
27
27
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

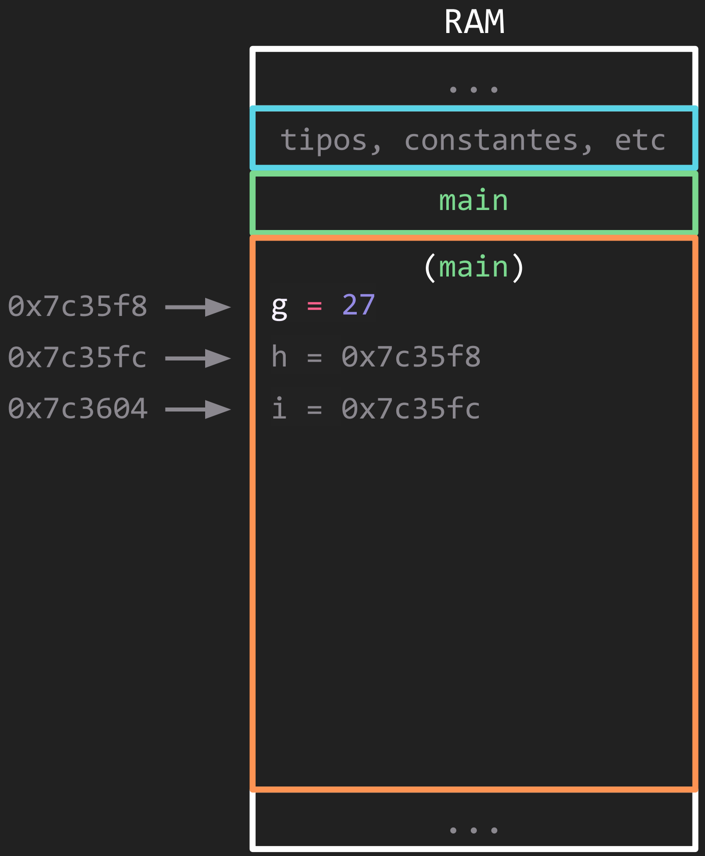
```
$ gcc main.c -o main
$ ./main
27
27
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

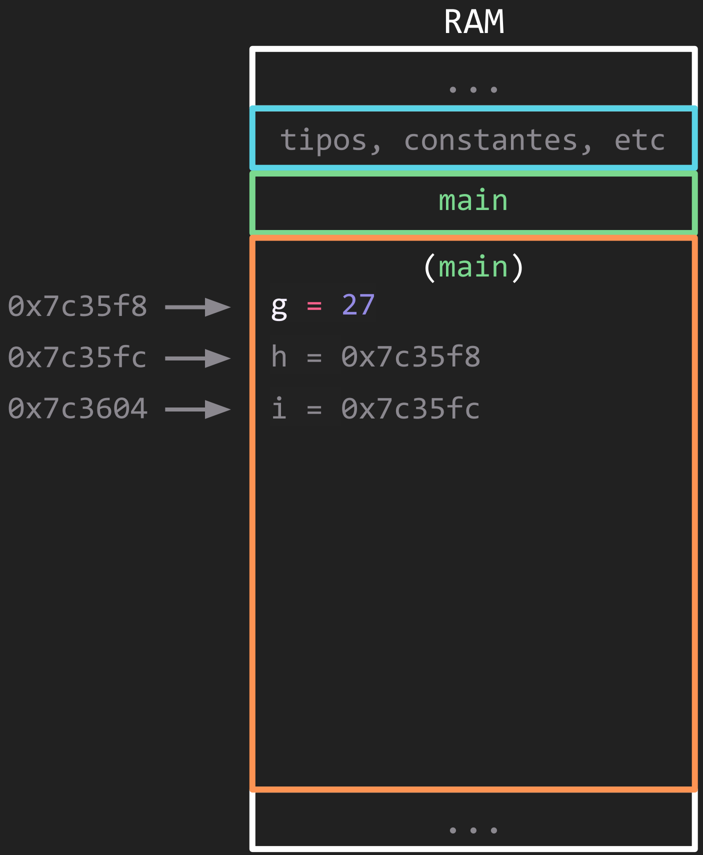
```
$ gcc main.c -o main
$ ./main
27
27
```



Punteros de punteros

```c
int g = 27;
int* h = &g;
int** i = &h;
printf("%d\n", g);
printf("%d\n", *h);
printf("%d\n", **i);
```

```
$ gcc main.c -o main
$ ./main
27
27
27
```



NULL - Puntero vacío

```c
int* a = NULL;
if (!a)
{
  printf("a no tiene un valor asignado\n");
}
```

```
$ gcc main.c -o main
$ ./main
a no tiene un valor asignado
```

NULL es una dirección de memoria especial, no apunta a ningún lado.

Es lógicamente igual a 0 (false).

NULL - ☢ WARNING! ACHTUNG! PELIGRO! ☢

```c
int* d = NULL;

printf("%i\n", *d);
```

```
$ gcc main.c -o main
$ ./main
Segmentation fault (core dumped)
```

La dirección NULL está protegida; acceder a ella causará un error en tu programa.

Aquí hay un ejemplo.

Propiedades de punteros

¿Por qué punteros?

En C todos los parámetros de funciones se pasan por valor.

Para que una función pueda modificar algo, necesitamos referencias.

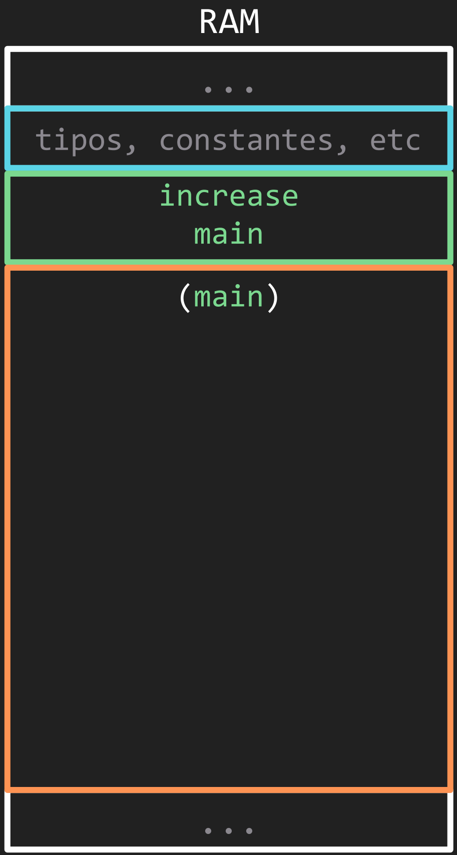
Paso por "referencia"

```c
void increase(int* a)
{
  *a += 1;
}

int main()
{
  int x = 2;
  increase(&x);
  printf("%i\n", x);
  return 0;
}
```

En C no existe el paso por referencia, pero usando punteros podemos simularlo y usar funciones para modificar objetos.

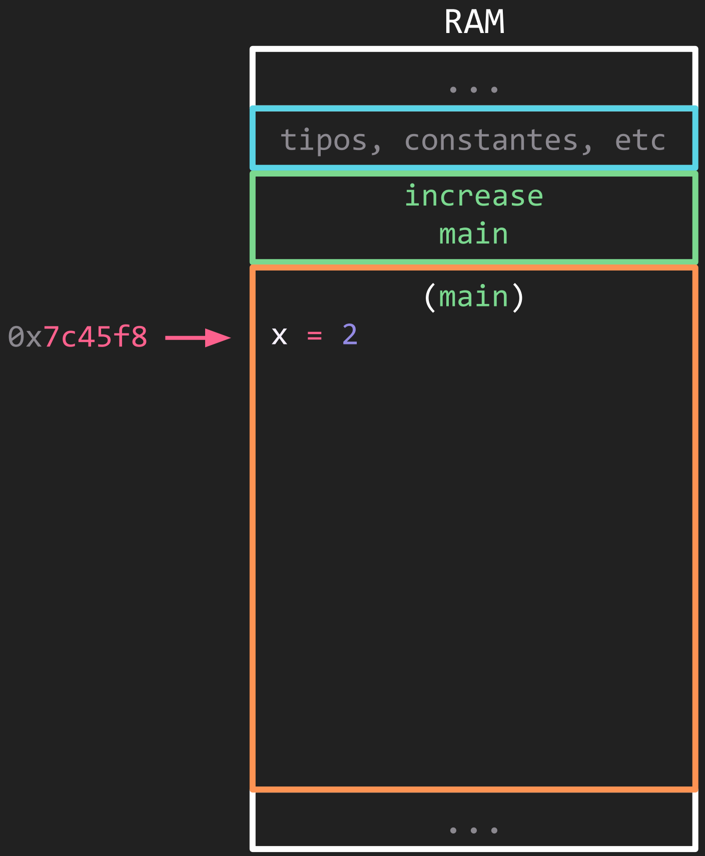
Paso por "referencia"

```
void increase(int* a)
{
  *a += 1;
}

int main()
{
  int x = 2;
  increase(&x);
  printf("%i\n", x);
  return 0;
}
```



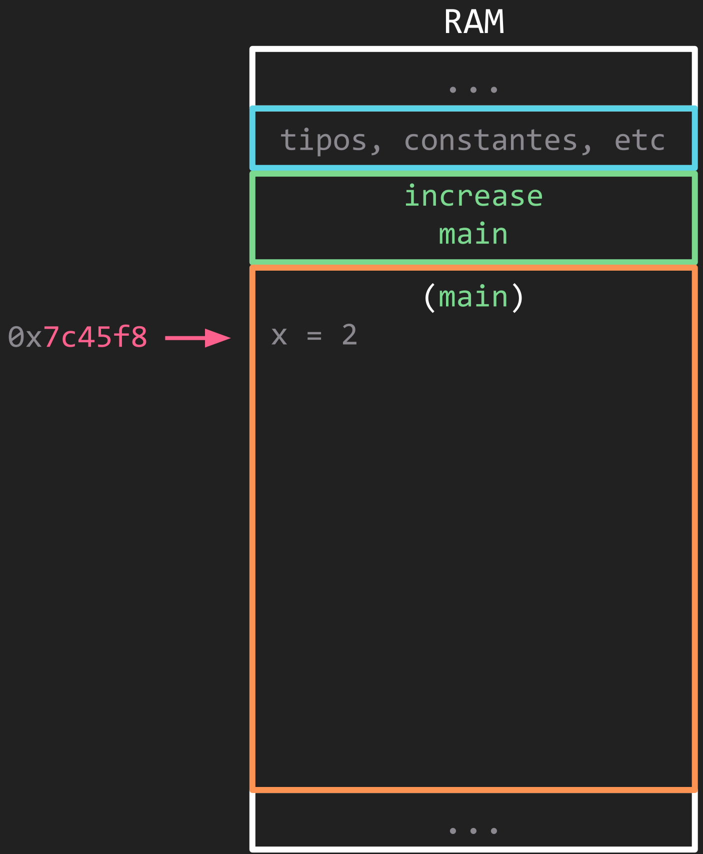
Paso por "referencia"

```c
void increase(int* a)
{
  *a += 1;
}

int main()
{
  int x = 2;
  increase(&x);
  printf("%i\n", x);
  return 0;
}
```



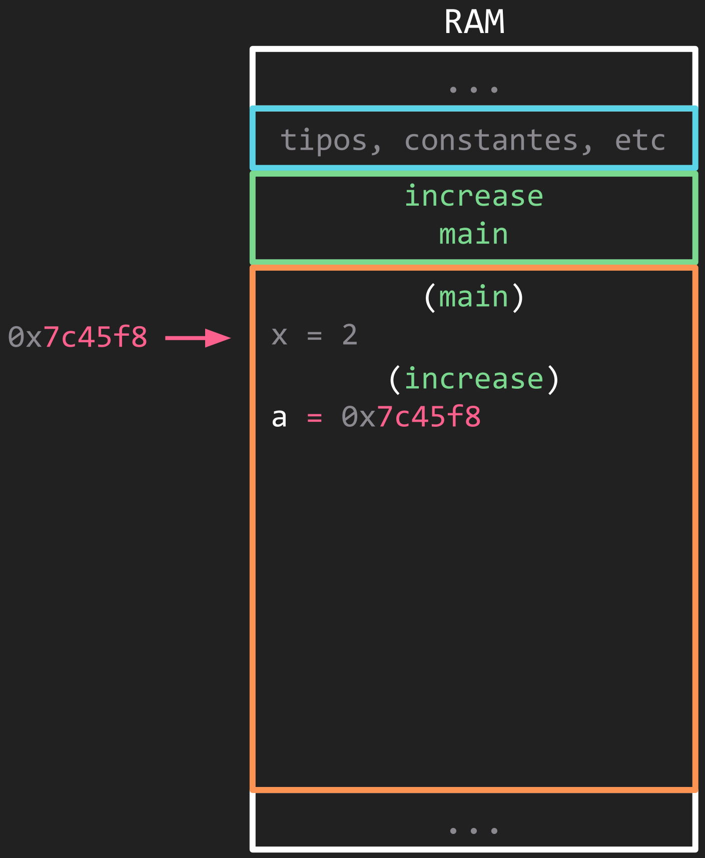
Paso por "referencia"

```c
void increase(int* a)
{
  *a += 1;
}

int main()
{
  int x = 2;
  increase(&x);
  printf("%i\n", x);
  return 0;
}
```



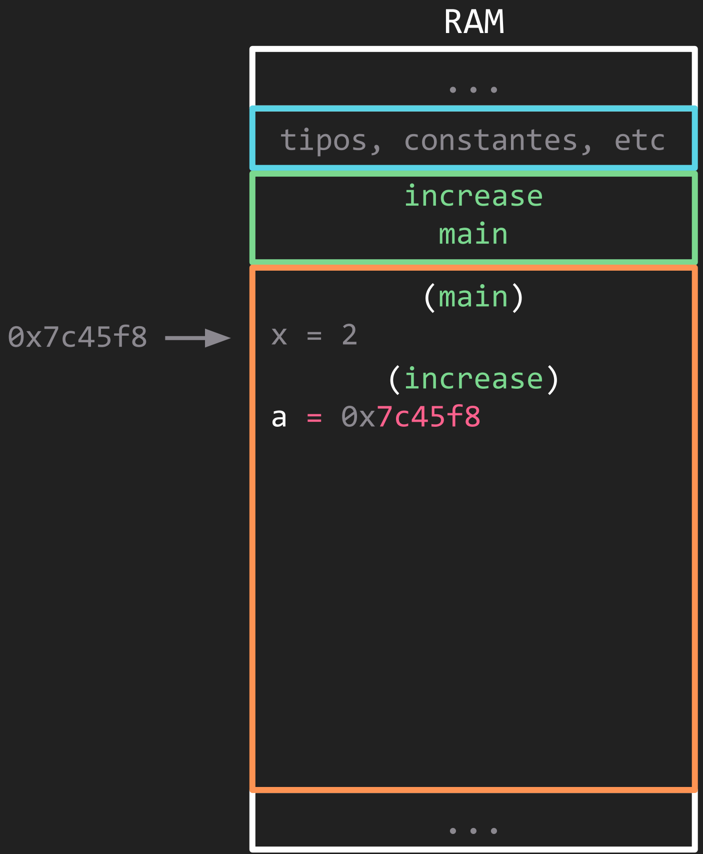
Paso por "referencia"

```c
void increase(int* a)
{
  *a += 1;
}

int main()
{
  int x = 2;
  increase(&x);
  printf("%i\n", x);
  return 0;
}
```



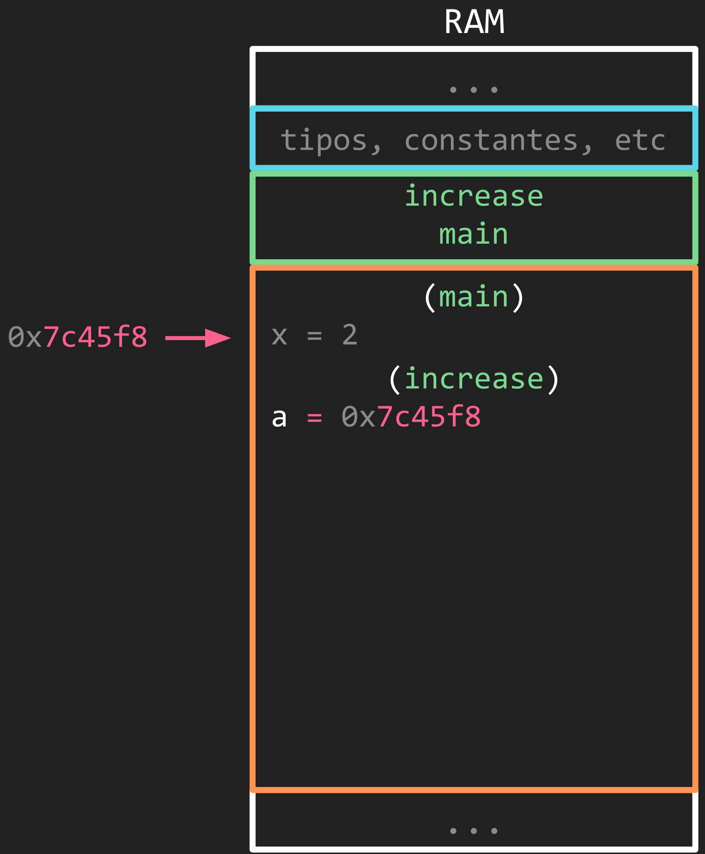
Paso por "referencia"

```c
void increase(int* a)
{
  *a += 1;
}

int main()
{
  int x = 2;
  increase(&x);
  printf("%i\n", x);
  return 0;
}
```



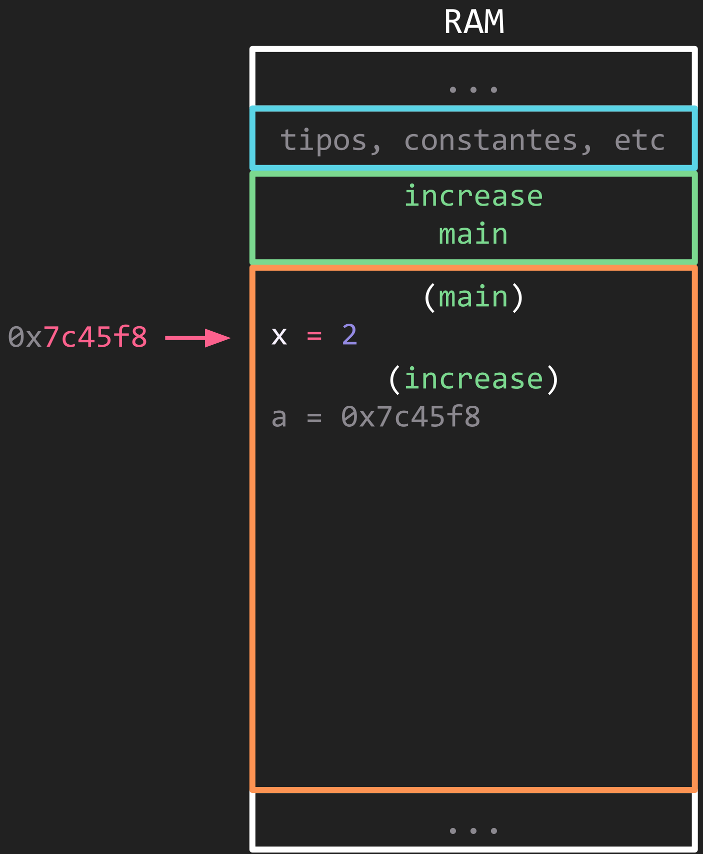
Paso por "referencia"

```c
void increase(int* a)
{
  *a += 1;
}

int main()
{
  int x = 2;
  increase(&x);
  printf("%i\n", x);
  return 0;
}
```



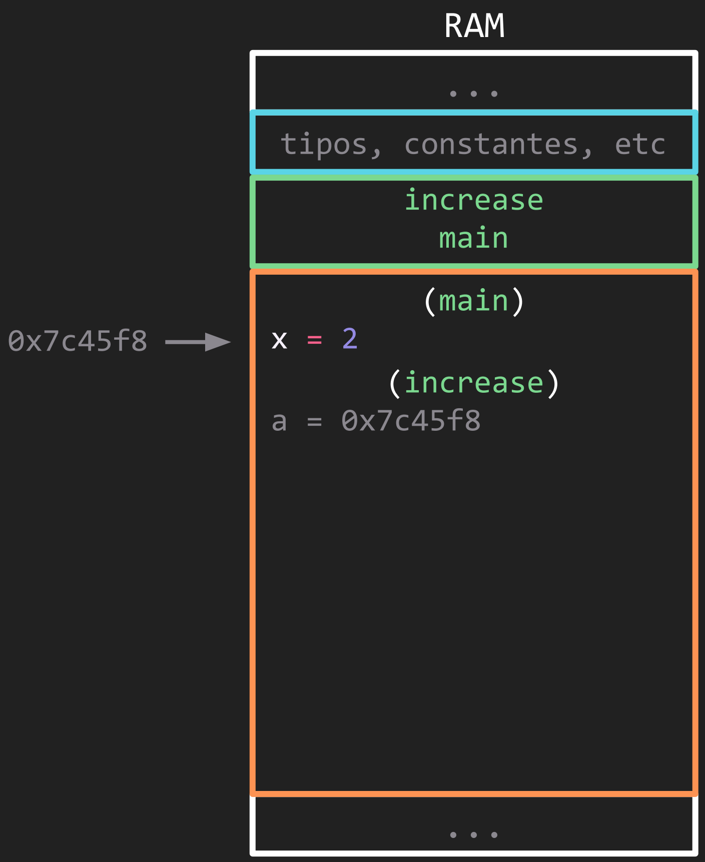
Paso por "referencia"

```c
void increase(int* a)
{
  *a += 1;
}

int main()
{
  int x = 2;
  increase(&x);
  printf("%i\n", x);
  return 0;
}
```



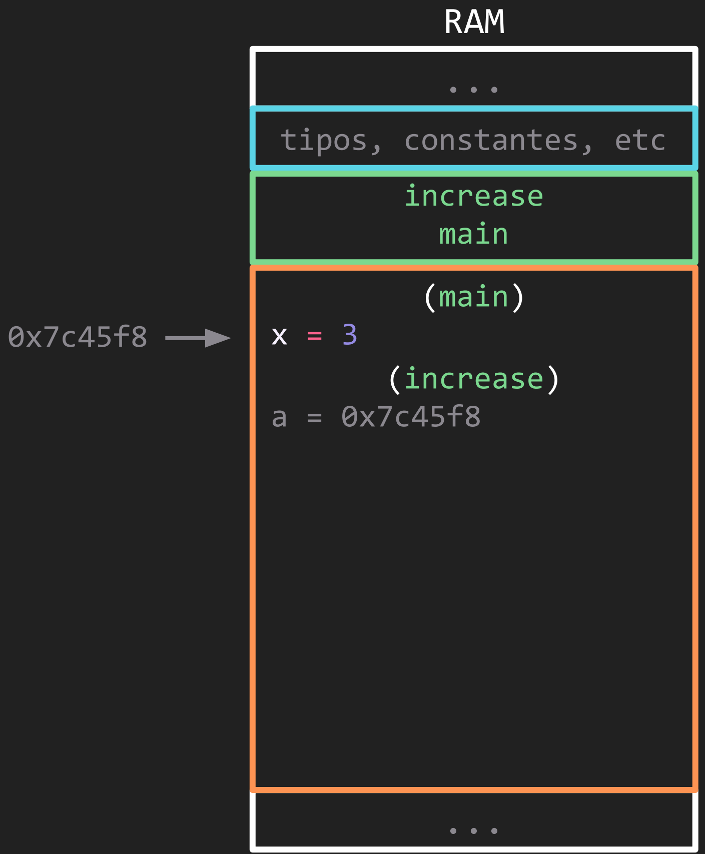
Paso por "referencia"

```c
void increase(int* a)
{
  *a += 1;
}

int main()
{
  int x = 2;
  increase(&x);
  printf("%i\n", x);
  return 0;
}
```



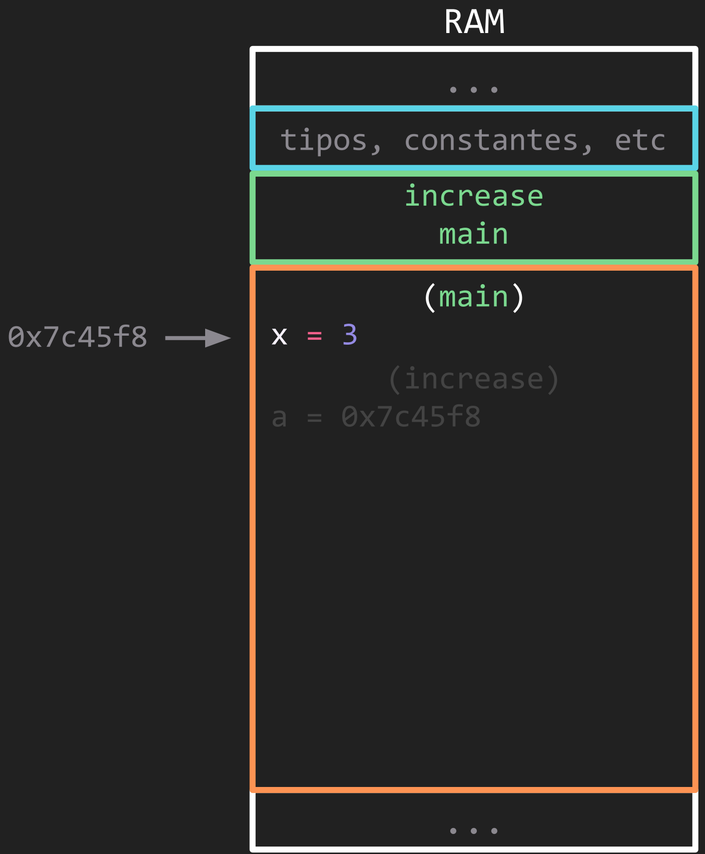
Paso por "referencia"

```c
void increase(int* a)
{
  *a += 1;
}

int main()
{
  int x = 2;
  increase(&x);
  printf("%i\n", x);
  return 0;
}
```



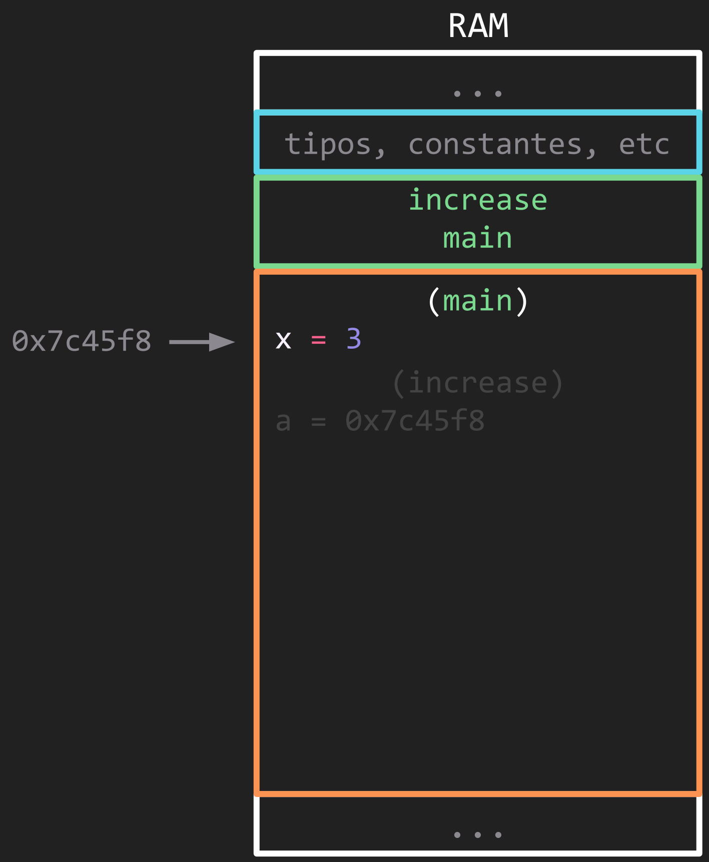
Paso por "referencia"

```c
void increase(int* a)
{
  *a += 1;
}

int main()
{
  int x = 2;
  increase(&x);
  printf("%i\n", x); // Imprime 3
  return 0;
}
```



Paso por "referencia"

```
void increase(int* a)
{
  *a += 1;
}

int main()
{
  int x = 2;
  increase(&x);
  printf("%i\n", x);
  return 0;
}
```



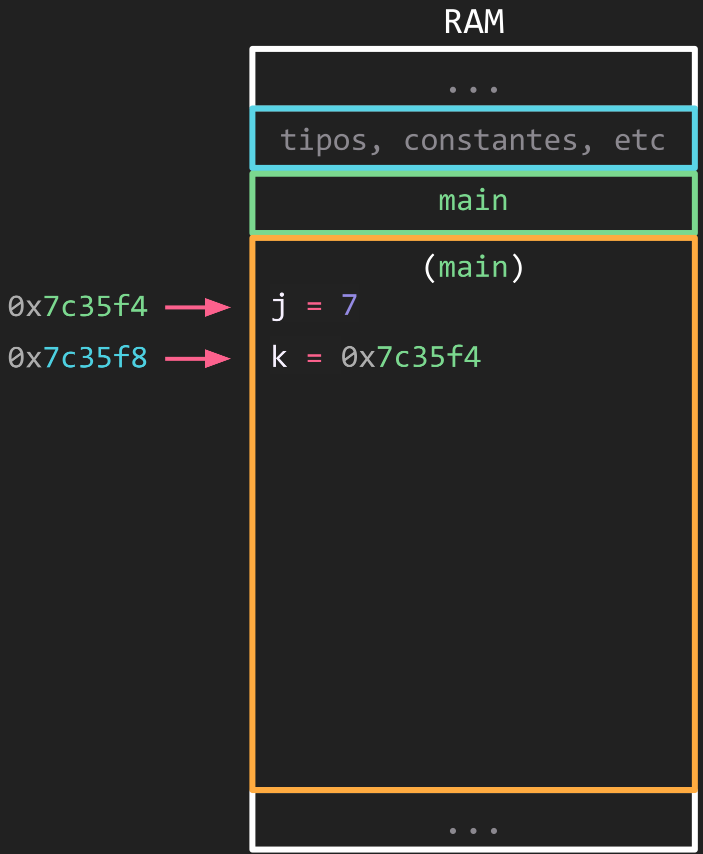
"Alquimia" de punteros

Aritmética de punteros

Aritmética de punteros

```c
int j = 7;
int* k = &j;
printf("%p\n", k);
printf("%p\n", k + 1);
```

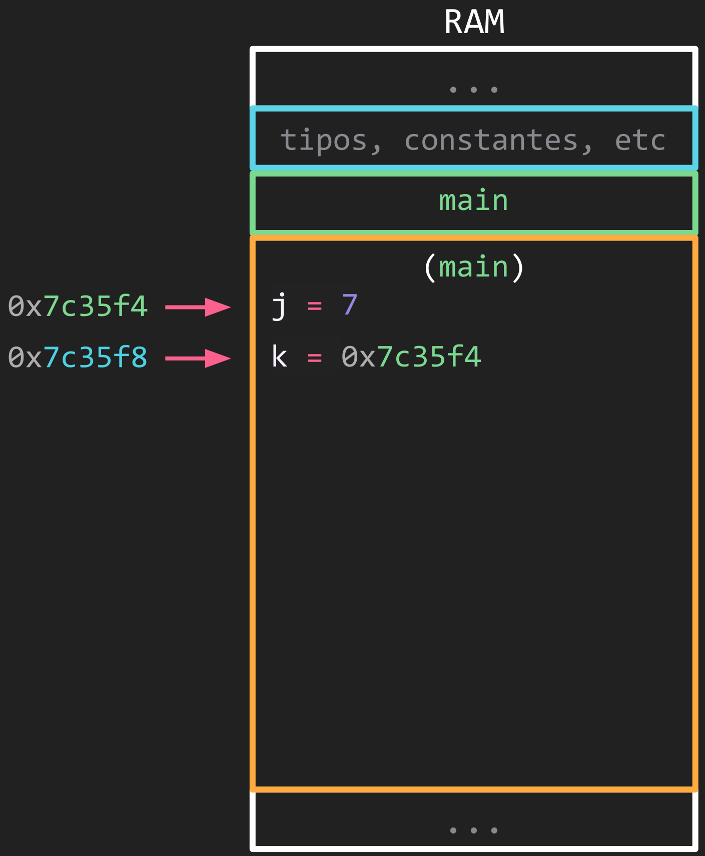
```
?
```



Aritmética de punteros

```c
int j = 7;
int* k = &j;
printf("%p\n", k);
printf("%p\n", k + 1);
```

```
$ gcc main.c -o main
$ ./main
0x7c35f4
0x7c35f8
```



Aritmética de punteros

*type** A

$$\downarrow$$

A + i =

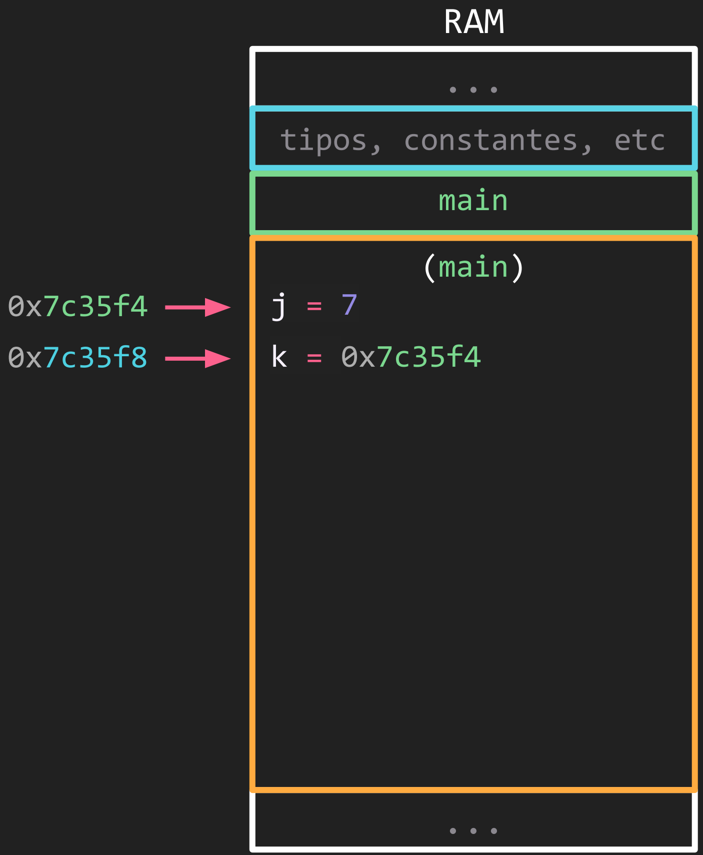
Aritmética de punteros

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

Aritmética de punteros

```c
int j = 7;
int* k = &j;
printf("%p\n", k);
printf("%p\n", k + 1);
```

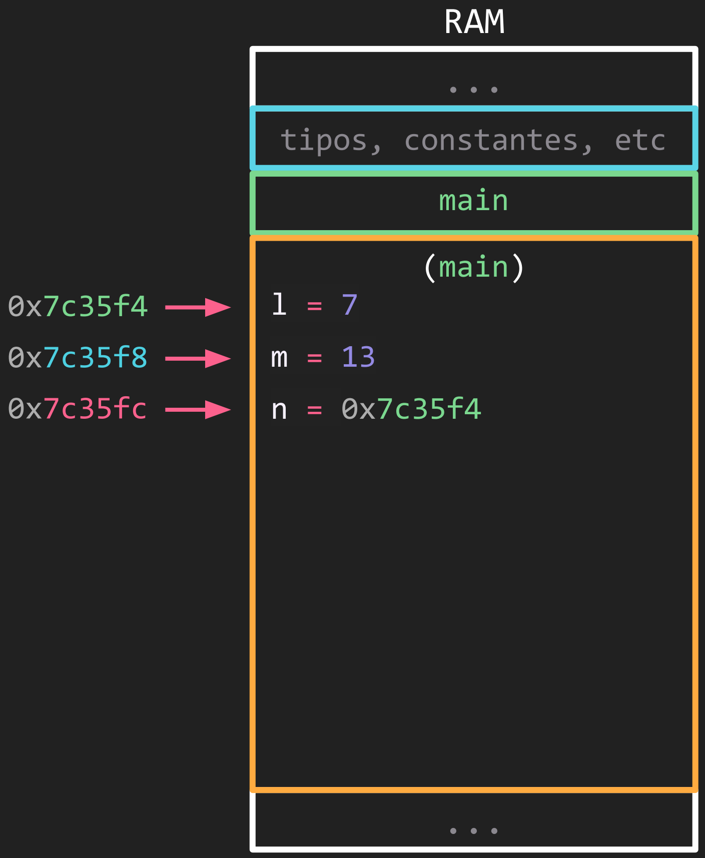
```
$ gcc main.c -o main
$ ./main
0x7c35f4
0x7c35f8
```



Aritmética de punteros

```c
int l = 7;
int m = 13;
int* n = &l;
printf("%p\n", n);
printf("%p\n", n + 1);
```

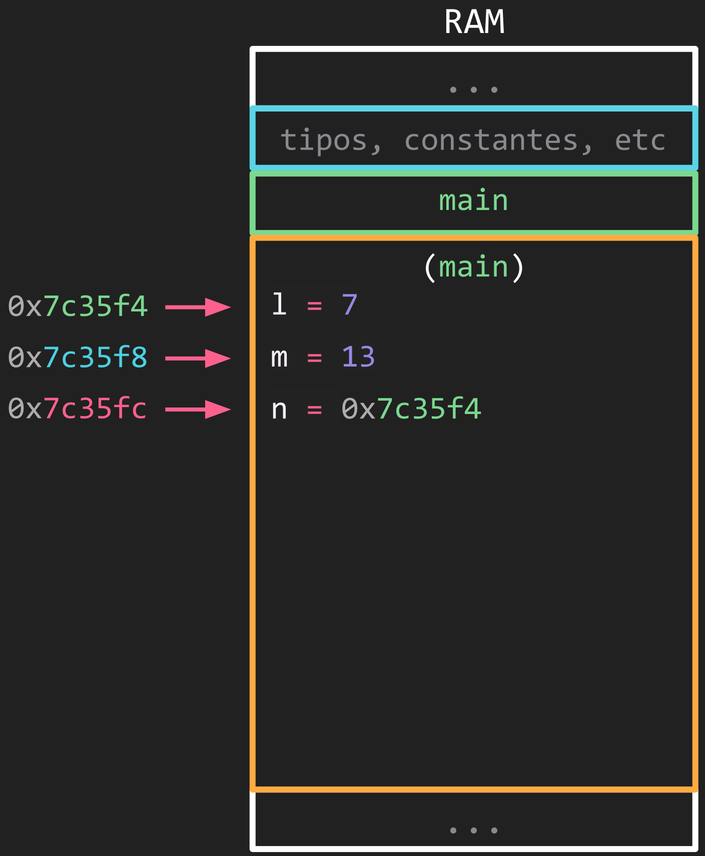
```
$ gcc main.c -o main
$ ./main
0x7c35f4
0x7c35f8
```



Aritmética de punteros

```c
int l = 7;
int m = 13;
int* n = &l;
printf("%d\n", *n);
printf("%d\n", *(n + 1));
```

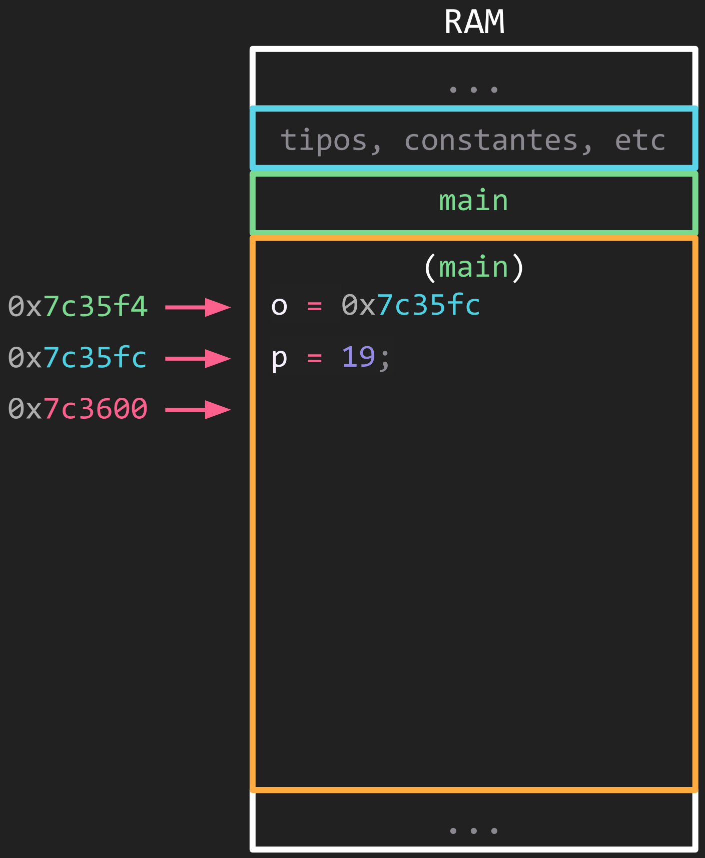
```
$ gcc main.c -o main
$ ./main
7
13
```



Aritmética de punteros

```c
int* o;
int p = 19;
o = &p;
printf("%p\n", o);
printf("%p\n", o + 1);
```

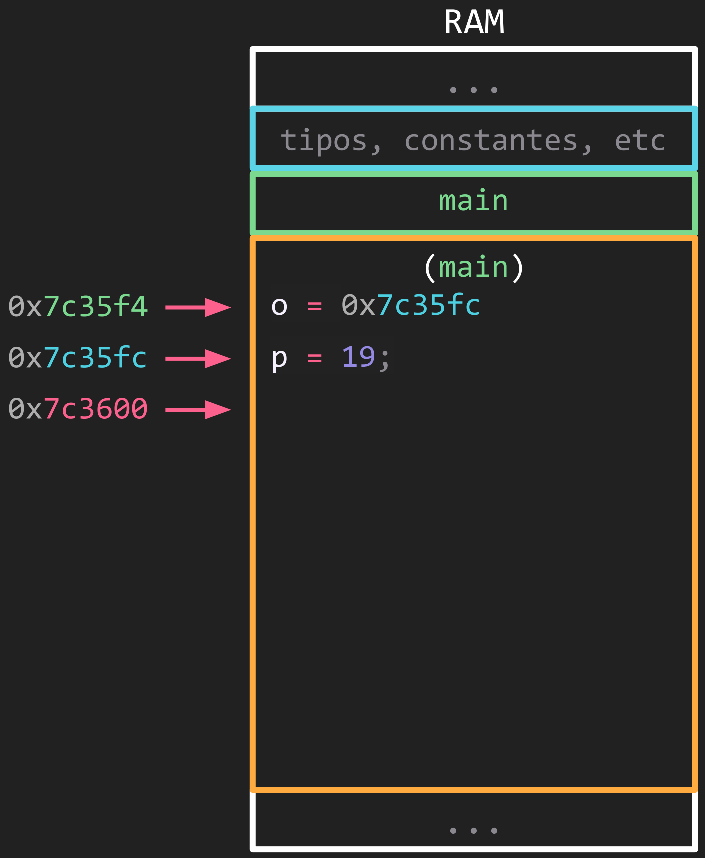
```
$ gcc main.c -o main
$ ./main
0x7c35fc
0x7c3600
```



Aritmética de punteros

```c
int* o;
int p = 19;
o = &p;
printf("%d\n", *o);
printf("%d\n", *(o + 1));
```

```
$ gcc main.c -o main
$ ./main
19
-409884036
```



¡Muchas Gracias!

MAN, I SUCK AT THIS GAME. CAN YOU GIVE ME A FEW POINTERS?
0x3A28213A 0x6339392C, 0x7363682E.
I HATE YOU.

With 💜 by @vichoeq & @KnowYourselves