Intro a C

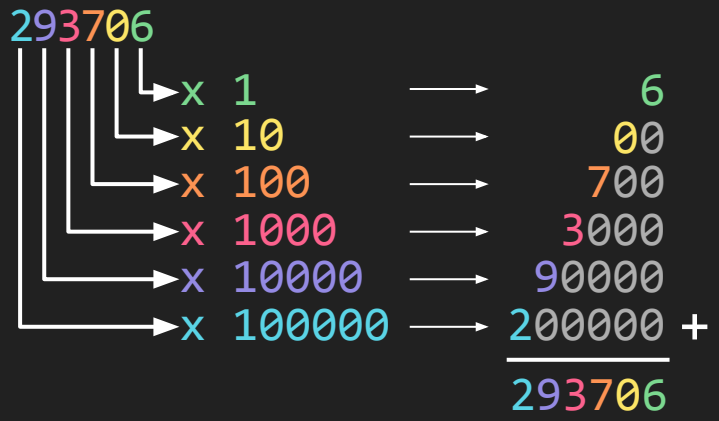
Tipos

With 💜 by @vichoeq & @KnowYourselves

Bases Numéricas #1

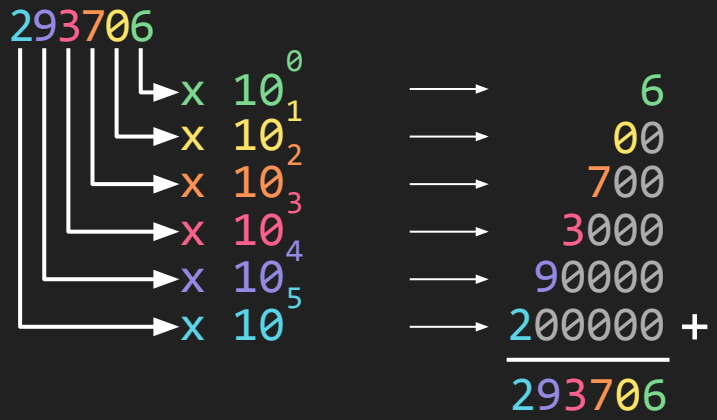
Base Decimal

Los números que usamos normalmente son en base 10



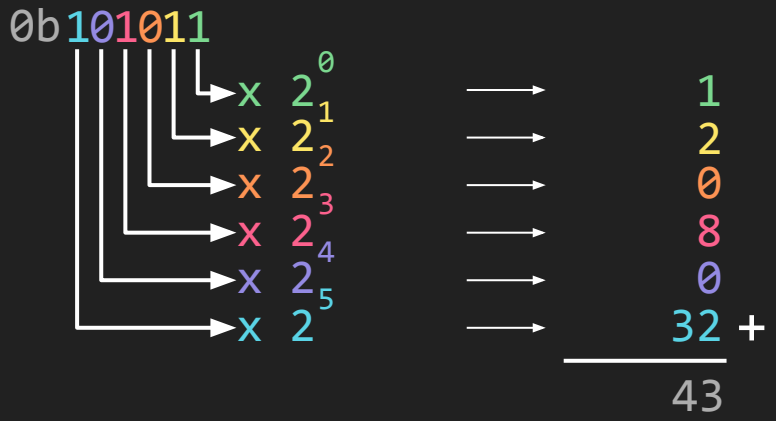
Base Decimal

Los números que usamos normalmente son en base 10



Base Binaria

Un dígito en base 2 se conoce como bit



En un PC la información está guardada en grupos de 8 bits = 1 byte

Variables

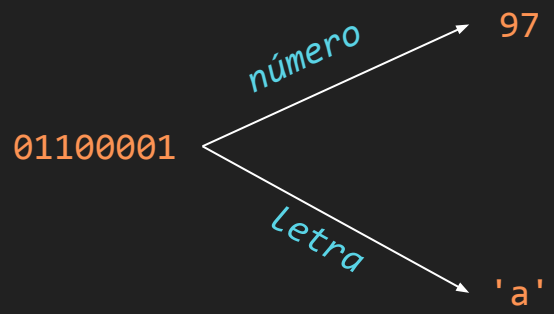
Variables en un programa

Cada variable tiene:

- Una dirección de memoria
- Un *tipo*
- Un valor

El *tipo* indica dos cosas:

- Tamaño en memoria
- Interpretación



tipos

printf

```c
printf("%i %i %f\n", 5, 7, 2.8);

printf("Hello %s\n", "World!");
```

```
$ gcc main.c -o main
$ ./main
5, 7, 2.800000
Hello World!
```

```python
print(f"{5} {7} {2.8}")

print(f"Hola {'World!'}")
```

```
$ python3 main.py
5, 7, 2.8
Hello World!
```

printf - Salto de línea

```c
printf("%i %i %f", 5, 7, 2.8);

printf("Hello %s", "World!");
```

```
$ gcc main.c -o main
$ ./main
5, 7, 2.800000Hello World!
```

A diferencia de `print`, `printf` no agrega un salto de línea automáticamente, por lo cual hay que agregarlo explícitamente.

int - Número entero

```c
int a = 1287;
printf("a = %i\n", a);
printf("size: %zu\n", sizeof(int));
```

```
$ gcc main.c -o main
$ ./main
a = 1287
size: 4
```

Tamaño: 32 bits (4 bytes)

Admite números positivos y negativos

Para imprimir: %i, %d

float - Número decimal

```c
float b = 2.72181;
printf("b = %f\n", b);
printf("size: %zu\n", sizeof(float));
```

```
$ gcc main.c -o main
$ ./main
b = 2.721810
size: 4
```

Tamaño: 32 bits (4 bytes)

Admite números positivos y negativos

Para imprimir: %f

double - Número decimal de doble precisión

```c
double c = 3.141592;
printf("c = %lf\n", c);
printf("size: %zu\n", sizeof(double));
```

```
$ gcc main.c -o main
$ ./main
c = 3.141592
size: 8
```

Tamaño: 64 bits (8 bytes)

Tiene más precisión que un `float`

Para imprimir: `%lf`

char - Caracter / Letra

```c
char d = 'h';
printf("d = %c\n", d);
printf("size: %zu\n", sizeof(char));
```

```
$ gcc main.c -o main
$ ./main
d = h
size: 1
```

Tamaño: 8 bits (1 byte)

Rodeado de comillas simples

Para imprimir: %c

char - ☢ WARNING! ACHTUNG! PELIGRO! ☢

```c
char d = 'h';
printf("d = %c\n", d);
```

```
$ gcc main.c -o main
$ ./main
d = h
```

```c
char d = "h";
printf("d = %c\n", d);
```

```
$ gcc main.c -o main
Warning: initialization from pointer
$ ./main
d = 
```

Modificar *tipos*

casting - Interpretar un *tipo* como otro

```c
double e = 64.7901;
char f = e; // Casting implícito
char g = (char) e; // Casting explícito
printf("f = %c, g = %c\n", f, g);
printf("f = %i, g = %i\n", f, g);
```

```
$ gcc main.c -o main
$ ./main
f = @, g = @
f = 64, g = 64
```

El casting implícito es sólo al asignar o al pasarlo a una función con *tipo* compatible

El casting explícito fuerza el cast.

En este caso *char* se interpreta como un número ASCII.

casting - División decimal

```c
int h = 5;
int i = 2;
double j = h / i;
printf("j = %lf\n", j);
```

```
$ gcc main.c -o main
$ ./main
2.000000
```

```c
int h = 5;
int i = 2;
double j = (double) h / i;
printf("j = %lf\n", j);
```

```
$ gcc main.c -o main
$ ./main
2.500000
```

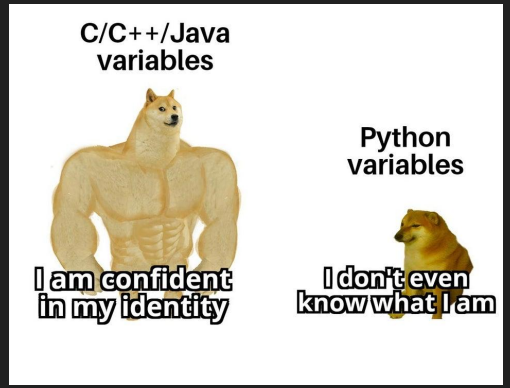
typedef - Alias de *tipos*

```c
typedef int cool_int;

cool_int k = 42;
cool_int l = 1313;
```

```c
typedef TIPO_EXISTENTE_DE_NOMBRE_LARGO ALIAS;

ALIAS k = valor;
```

Con typedef podemos crear un alias para el *tipo*. Va a ser muy útil para referirnos a *tipos* con nombres muy largos.

¡Muchas Gracias!



With 💜 by @vichoeq & @KnowYourselves