Intro a C

Flujo

With 💜 by @vichoeq & @KnowYourselves

Bloques de Código

Bloques de código

```c
int main()
{
  printf("Hello World!\n");
  return 0;
}
```

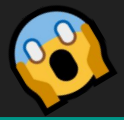
```python
def main():
  print("Hello world!")
```

?

Bloques de código

```c
int main()
{
  printf("Hello World!\n");
  return 0;
}
```

```c
int main(){printf("Hello World!\n");return 0;}
```



```python
def main():
  print("Hello world!")
```

Control de Flujo

if - else

```c
if (x > 7)
{
  printf("x > 7\n");
}
else if (x < 5)
{
  printf("x < 5\n");
}
else
{
  printf("x <= 7 & x >= 5\n");
}
```

```python
if x > 7:
  print("x > 7")
elif x < 5:
  print("x < 5")
else:
  print("x <= 7 & x >= 5")
```

while

```c
int i = 0;
while (i < 5)
{
  printf("Ciclo %i!\n", i);
  i += 1;
}
```

```python
i = 0
while i < 5:
  print(f"Ciclo {i}!")
  i += 1
```

for

```c
for (int i = 0; i < 5; i+=1)
{
  printf("Ciclo %i!\n", i);
}
```

```
for (INIT; COND; STEP)
{
  // Código
}
```

El for de C y Python son muy distintos.

Este consiste en 2 bloques y una condición de término. Es equivalente a un while.

for

```c
for (int i = 0; i < 5; i+=1)
{
  printf("Ciclo %i!\n", i);
}
```

```c
for (INIT; COND; STEP)
{
  // Código
}
```

INIT: Se ejecuta antes de entrar al for

COND: Salimos del for si deja de cumplirse

STEP: Se ejecuta luego de cada iteración

for

```c
for (int i = 0; i < 5; i+=1)
{
  printf("Ciclo %i!\n", i);
}
```

```c
for (INIT; COND; STEP)
{
  // Código
}
```

INIT: Se ejecuta antes de entrar al for

COND: Salimos del for si deja de cumplirse

STEP: Se ejecuta luego de cada iteración

for

```c
for (int i = 0; i < 5; i+=1)
{
  printf("Ciclo %i!\n", i);
}
```

```c
for (INIT; COND; STEP)
{
  // Código
}
```

INIT: Se ejecuta antes de entrar al for

COND: Salimos del for si deja de cumplirse

STEP: Se ejecuta luego de cada iteración

Ejemplo
for - if

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

```
n = 4
count = 0
```

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 0

(for)

i = 1

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 0

(for)

i = 1

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 1

(for)

i = 1

(if)

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 1

(for)

i = 1

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 1

(for)

i = 2

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 1

(for)

i = 2

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 1

(for)

i = 2

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 1

(for)

i = 2

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 2

(for)

i = 2

(if)

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 2

(for)

i = 2

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 2

(for)

i = 3

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 2

(for)

i = 3

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 2

(for)

i = 3

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 0

(for)

i = 3

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 2

(for)

i = 3

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 2

(for)

i = 4

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 2

(for)

i = 4

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 2

(for)

i = 4

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 2

(for)

i = 4

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 3

(for)

i = 4

(if)

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 3

(for)

i = 4

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 3

(for)

i = 5

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 3

(for)

i = 5

for - if

```c
int n = 4;
int count = 0;

for(int i = 1; i <= n; i+=1)
{
  if(n % i == 0) count += 1;
}
```

variables

n = 4

count = 3

Control de Control Flujo

continue - Pasar a la siguiente iteración

```c
for (int i = 0; i < 10; i+=1)
{
  for (int j = 0; j < 10; j+=1)
  {
    if (i == j) continue;
    printf("%d, %d\n", i, j);
  }
}
```

En este caso es más fácil y legible usar continue que hacer otra anidación más.

break - Salirse del bloque

```c
while (alive)
{
  // Codigo

  if (must_kill) break;
}
```

Un caso de uso de break es romper un ciclo de forma anticipada.

Ojo, si se usa break en ciclos anidados solo se va a salir de uno.

switch

```c
char gato = 'X';
switch (gato)
{
  case 'O':
    printf("¡Vas segundo!\n");
    break;
  case 'X':
    printf("¡Vas primero!\n");
    break;
  default:
    printf("Así no se juega :(\n");
    break;
}
```

switch es útil cuando debemos manejar un set pequeño de posibles valores.

Así no es necesario hacer una secuencia innecesariamente larga de if else.

Booleanos

Truthiness & Falseness

```c
if (0)
  printf("0 es False\n");
if (1)
  printf("1 es True\n");
if (8912)
  printf("y todo lo que no es 0 es True\n");
```

```
$ gcc main.c -o main
$ ./main
1 es True
y todo lo que no es 0 es True
```

En **C** no existen los valores booleanos como `true` y `false`.

Todo número tiene un valor de verdad:

`false`: 0
`true`: todo lo demás

bool - Valor de verdad (módulo)

```c
typedef int bool;

#define true 1
#define false 0
```

```c
#include <stdbool.h>

bool a = true;
bool b = false;
```

Si quieres ser ordenado en **C** podrías definir tu propio tipo *bool*, junto con las constantes true y false.

Eso ya está hecho en el módulo <stdbool.h>

bool - Operadores lógicos

```c
#include <stdbool.h>

int p = 17;       // true
bool q = false;
bool A = p || q; // true
bool B = p && q; // false
bool C = !p;      // false
```

```python
p = True
q = False
A = p or q  # true
B = p and q # false
C = not p   # false
```

¡Muchas Gracias!

With 💜 by @vichoeq & @KnowYourselves