Lenguaje de programación Typst

Viendo Typst como un lenguaje de programación.

1. Modos

Typst tiene tres modos sintácticos: Etiquetado (Markup), Matemático y Código. En un documento Typst el modo etiquetado(Markup) es el modo por defecto. El modo matemático permite escribir fórmulas y ecuaciones Matemáticas y el modo código permite escribir instrucciones del lenguaje Typst.

1.1. Etiquetado

Typst tiene muchas etiquetas para los elementos de un documento. Por ejemplo para poner en negrilla una palabra o frase, escriba: `*negrilla*` y obtiene: **negrilla**

1.2. Matemático

Las ecuaciones y/o fórmulas matemáticas se deben escribir entre signos $$. Una ecuación en línea, se escribe: `$x^2$` y se obtiene: x^2 y una ecuación centrada en la siguiente línea se escribe: `$ (x+1)^2 $` y se obtiene:

$$(x+1)^2$$

Note que se deja un espacio antes y después de los signos $.

1.3. Código

Inicia con #. Por ejemplo, al escribir: `#(3+4)`, obtiene: 7. También puede escribir bloques de código entre corchetes { }:

```
#{
  let a = [amo ]
  let b = [*Typst*]
  [Hola, ]
  a  + b
}
```

obteniendo:
Hola, amo **Typst**

Note que dentro del bloque ya no se escribe # para `let`.

1.4. Comentarios

Dentro de un documento Typst puede insertar comentarios iniciando con `//`

2. Medidas usadas en Typst

Para seleccionar tamaños se usan puntos(pt), centímetros(cm), milímetros(mm) y pulgadas(in). Por ejemplo para espacios verticales usamos: `\#v(5pt) \#linebreak()`, para tamaños de letra 12pt,... Se tiene

1 in = 72 pt = 2.54 cm= 25.54 mm

2.1. Fracción `fr`

Define como distribuir el espacio en un diseño. En el siguiente ejemplo: toda la línea es 100%=3fr y así 33.3% es 1fr y 66% es 2fr del espacio respectivamente. Es decir `fr` determina la fracción de espacio a usar dentro de una línea u otro espacio.

2.1.1. Ejemplo

```
Negro #h(1fr)  Gris #h(2fr) Blanco
```

Negro Gris Blanco

2.1.2. Otro ejemplo

```
Primero #h(1fr) Segundo \
Primero #h(1fr) Segundo #h(1fr) Tercero \
Primero #h(2fr) Segundo #h(1fr) Tercero \
Uno #h(1fr) Dos #h(1fr) Tres #h(2fr) Cinco
```

Primero Segundo
Primero Segundo Tercero
Primero Segundo Tercero
Uno Dos Tres Cinco

2.2. Ángulos

Usado para rotaciones. Se puede usar radianes o grados, por ejemplo:

```
#rotate(0.5rad)[uno]
#rotate(-16deg)[dos]
```

uno

dos

2.3. Colores

Colores definidos en typst:

Color	Definición
black	luma(0)
gray	luma(170)
silver	luma(221)
white	luma(255)
navy	rgb(«#001f3f»)
blue	rgb(«#0074d9»)
aqua	rgb(«#7fdbff»)
teal	rgb(«#39cccc»)
eastern	rgb(«#239dad»)
purple	rgb(«#b10dc9»)
fuchsia	rgb(«#f012be»)
maroon	rgb(«#85144b»)
red	rgb(«#ff4136»)
orange	rgb(«#ff851b»)
yellow	rgb(«#ffdc00»)
olive	rgb(«#3d9970»)
green	rgb(«#2ecc40»)
lime	rgb(«#01ff70»)

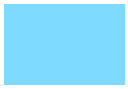
Tabla 1: Colores en Typst

Typst también soporta `rgb`,`cmyk`,`luma`,`oklab`, ... Vea más en:

https://typst.app/docs/reference/visualize/color/

2.3.1. Ejemplos

```
#rect(fill: aqua)
#square(fill: luma(70))
#emph(text(fuchsia)[
  With a function call.
])
```



Texto en color

3. Tipos en Typst

En typst los objetos tienen *tipos*:

```
#type(12) \
#type(14.7) \
#type("hola") \
#type(<glacier>) \
#type([Hola]) \
#type(x => x + 1) \
#type(type) \
#type(true)
```

```
int
float
str
label
content
function
type
bool
```

4. Algunas funciones incluidas en Typst

Una función de typst empieza con #. Algunas funciones son:

```
#emph[Hola] \
#emoji.car \
#strong[Hola] \
#"Murcielago".len() \
#(2+3) \
#false \
#true \
#(1 < 2)
#(1in == 72pt)
```

Hola

🚗

Hola

10

5

false

true

true

true

5. Creación de variables

```
#let editor="Typst"
Documento escrito con #editor.\
#let xy = 8
#xy
```

Documento escrito con Typst.

8

6. Funciones creadas por el usuario

Se pueden crear funciones matemáticas:

```
#let suma(x,y)=x+y
La suma es #suma(4,5)
```

La suma es 9

```
#let f(x,y)= calc.sqrt(x-y)
La función f en 6,4 es #f(6,4)
```

La función f en 6,4 es 1.4142135623730951

7. Funciones que se aplican al contenido

Aquí se crea la función `alerta` que crea un rectángulo rojo, el argumento será el texto debajo de `OJO`

```
  #let alerta(contenido, fill:red)={
  set text(white)
  set align(center)
  rect(fill: fill,
    inset: 8pt,
    radius: 4pt,
    [*OJO : \ #contenido *]
)}
#alerta[Peligro inminente!]
```

OJO :
Peligro inminente!

8. Bloques

Se pueden crear bloques de código.

```
#{ let x = 1; x + 2 }
```

3

9. Arreglos

Creación de vectores:

```
#let vector = (1, 7, 4, -3, 2)
#vector.at(0) \
#(vector.at(0) = 3)
#vector.at(-1) \
#vector.first() \
#vector.last() \
#vector.find(calc.even) \
#vector.filter(calc.odd) \
#vector.map(calc.abs) \
#vector.rev() \
#(1, (2, 3)).flatten() \
#(("A", "B", "C").join(", ", last: "
y ")) \
#(1, 2, 3).contains(2)
```

```
1
2
3
2
4
(3, 7, -3)
(3, 7, 4, 3, 2)
(2, -3, 4, 7, 3)
(1, 2, 3)
A, B y C
true
```

10. Más arreglos

Algunas funciones que se aplican a los arreglos:

```
#let v1 = (1, 2, 3, 4, 5, 6,
7, 8)
#v1.chunks(3) \
#v1.chunks(3, exact: true)
```

```
((1, 2, 3), (4, 5, 6), (7, 8))
((1, 2, 3), (4, 5, 6))
```

11. Más arreglos

Creación de matrices:

```
#let data = ((1, 2, 3), (4, 5, 6))
#let matriz = math.mat(..data)
$ M := matriz $
```

$$M := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

12. Rangos

Funcionan muy al estilo *Python*:

```
#range(5) \
#range(2, 5) \
#range(20, step: 4) \
#range(21, step: 4) \
#range(5, 2, step: -1)
```

```
(0, 1, 2, 3, 4)
(2, 3, 4)
(0, 4, 8, 12, 16)
(0, 4, 8, 12, 16, 20)
(5, 4, 3)
```

13. Intervenir las tablas

El siguiente código colorea las filas impares de todas las tablas del documento:

```
#set table(
  fill: (_, y) => if calc.odd(y) { rgb("EAF2F5") },
)
```

Uno	Dos
1.234	«Color»
$\sqrt{1+x}$	0
$(x^2+y^2)^2$	«No aplica»

14. Calculadora de typst

14.1. Funciones matemáticas

Módulo de cálculos y proceso de valores numéricos:

```
#calc.pi \
#calc.e \
#calc.abs(-5) \
#calc.abs(5pt - 2cm) \
#calc.abs(2fr) \
#calc.abs(decimal("-342.440"))
#calc.pow(2, 3) \
#calc.pow(decimal("2.5"), 2) \
#calc.exp(1) \
#calc.sqrt(16) \
#calc.sqrt(2.5) \
#calc.root(16.0, 4) \
#calc.root(27.0, 3)
```

3.141592653589793
2.718281828459045
5
51.69pt
2fr
342.440 8
6.25
2.7182818284590455
4
1.5811388300841898
2
3

14.2. Funciones trigonométricas y logarítmicas

```
#calc.sin(1.5) \
#calc.sin(90deg) \
#calc.cos(1.5) \
#calc.cos(90deg) \
#calc.tan(1.5) \
#calc.tan(90deg) \
#calc.asin(0) \
#calc.asin(1) \
#calc.acos(0) \
#calc.acos(1) \
#calc.atan(0) \
#calc.atan(1) \
#calc.atan2(1, 1) \
#calc.atan2(-2, -3)
```

```
0.9974949866040544
1
0.0707372016677029
0.00000000000000006123233995736766
14.101419947171719
16331239353195370
0deg
90deg
90deg
0deg
0deg
45deg
45deg
-123.69deg
```

14.3. Funcioneshiperbólicas

```
  #calc.sinh(0) \
#calc.sinh(1.5) \
#calc.cosh(0) \
#calc.cosh(1.5) \
#calc.tanh(0) \
#calc.tanh(1.5)
```

```
0
2.1292794550948173
1
2.352409615243247
0
0.9051482536448664
```

14.4. Ejemplo de cálculo

$$\frac{e + \sqrt{5}}{\pi}$$

```
#((calc.e + calc.sqrt(5))/calc.pi)
```

```
1.577018522849442
```

14.5. Otras funciones matemáticas

```
#calc.log(100) \
#calc.ln(calc.e) \
#calc.fact(5) \
$ "perm"(n, k) &= n!/((n - k)!) \
  "perm"(5, 3) &= #calc.perm(5, 3) $
#calc.binom(10, 5) \
#calc.gcd(7, 42) \
#calc.lcm(96, 13) \
#calc.floor(500.1) \
#calc.ceil(500.1) \
#calc.trunc(15.9) \
#calc.fract(-3.1) \
#calc.round(3.1415, digits: 2) \
#calc.clamp(5, 0, 4) \
#calc.min(1, -3, -5, 20, 3, 6) \
#calc.min("typst", "is", "cool") \
#calc.max(1, -3, -5, 20, 3, 6) \
#calc.max("typst", "is", "cool") \
#calc.even(4) \
#calc.even(5) \
#calc.odd(4) \
#calc.odd(5) \
#calc.rem(7, 3) \
#calc.div-euclid(7, 3) \
#calc.div-euclid(decimal("1.75"),
decimal("0.5")) \
#calc.rem-euclid(7, 3) \
#calc.rem-euclid(1.75, 0.5) \
#calc.rem-euclid(decimal("1.75"),
decimal("0.5")) \
#calc.norm(1, 2, -3, 0.5) \
#calc.norm(p: 3, 1, 2)
```

2
1
120

$$\operatorname{perm}(n, k) = \frac{n!}{(n-k)!}$$

$$\operatorname{perm}(5, 3) = 60$$

252
7
1248
500
501
15
−0.10000000000000009
3.14
4
−5
cool
20
typst
true
false
false
true
1
2
3
1
0.25
3.774917217635375
2.080083823051904

Vea más en https://typst.app/docs/reference/foundations/calc/

15. Ejemplo: Sucesión de Fibonacci

La sucesión de Fibonacci estádefinida por la relación de recurrencia $F_n = F_{n-1} + F_{n-2}$. También se puede expresar:

$$F_n = \left\lfloor \frac{1}{\sqrt{5}} \phi^n \right\rceil, \quad \phi = \frac{1+\sqrt{5}}{2}$$

```
#let count=8
#let nums= range(1, count +1)
#let fib(n)=(
  if n <= 2 {1}
  else { fib(n - 1) + fib(n - 2)}
)

Los primeros #count términos de la sucesión son:
#align(center, table(
  columns: count,
  ..nums.map(n => $F_#n$),
  ..nums.map(n => str(fib(n))),
))
```

Los primeros 8 términos de la sucesión son:

F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
1	1	2	3	5	8	13	21

16. Control de flujo

16.1. Ciclo `if`

Puede calcular diferentes cosas dependiendo de si se cumple o no una condición:

```
#if 1 < 2 [
  Verdad
] else [
  Falso.
]
```

Verdad

16.2. Ciclo `for`

Typst soporta dos tipos de ciclos: `for` y `while`. El primero itera sobre una colección específica y el segundo, itera mientras se cumpla una condición.

```
#for c in "ABC" [
  #c es una letra.
]
```

A es una letra. B es una letra. C es una letra.

16.3. Ciclo `while`

```
#let n = 2
```

```
#while n < 10 {
  n = (n * 2) - 1
  (n,)
}
```

```
(3, 5, 9, 17)
```

17. Operadores

La siguiente tabla presenta los operadores de Typst, su paridad y el nivel de precedencia:

Operador	Efecto	Ario	Precedencia
-	Negación	Unitario	7
+	No tiene efecto (existe por symetría)	Unitario	7
*	Multiplicación	Binario	6
/	División	Binario	6
+	Adición	Binario	5
-	Sustracción	Binario	5
==	Verifica igualdad	Binario	4
!=	Verifica desigualdad	Binario	4
<	Verifica menor que	Binario	4
<=	Verifica menor o igual que	Binario	4
>	Verifica mayor que	Binario	4
>=	Verific mayor o igual que	Binario	4
in	Verifica si está en una colección	Binario	4
not in	Verifica si no está en una colección	Binario	4
not	«no» lógico	Unitario	3
and	Short-circuiting «y» lógico	Binario	3
or	Short-circuiting «o» lógico	Binario	2
=	Asignación	Binario	1
+=	Asignación-más	Binario	1
-=	Asignación-menos	Binario	1
*=	Asignación-multiplicación	Binario	1
/=	Asignación-división	Binario	1

18. Diccionarios

En Typst se pueden construir diccionarios:

```
#let dicc = (
  uno: "1",
  dos: "2",
  tres: "3",
)
#dicc.uno \
#(dicc.cuatro = "4") \
#dicc.len() \
#dicc.keys() \
#dicc.values() \
#dicc.at("dos") \
#dicc.insert("cinco", "5")\
#("uno" in dicc)\
#dicc
```

```
1

4
("uno", "dos", "tres", "cuatro")
("1", "2", "3", "4")
2

true
(
uno: "1",
dos: "2",
tres: "3",
cuatro: "4",
cinco: "5",
)
```

19. Campos

Se puede usar la notación punto (.) para acceder a los campos de un item:

```
#let it = [= Título]
```

```
#it.body \
#it.depth \
#it.fields()
```

```
Título
1
(depth: 1, body: [Título])
```

```
#let dict = (greet: "Hola")
```

```
#dict.greet \
#emoji.face
```

```
Hola
😀
```

20. Métodos

Hay varios métodos para llamar la función `.len`

```
  #str.len("abc") es lo mismo que
#"abc".len()
```

3 es lo mismo que 3

```
#let valores = (1, 2, 3, 4)
```

```
  #valores.pop() \
  #valores.len() \
  #("a, b, c"
      .split(", ")
      .join[ --- ])
  #"abc".len() es lo mismo que
  #str.len("abc")
```

4
3
a — b — c
3 es lo mismo que 3

21. Paquetes

Un paquete de Typst es un bloque de código que se puede reutilizar. Para utilizar bloques de código ya elaborados se debe importar, por ejemplo:

```
  #import "@preview/example:0.1.0":
add
#add(2, 7)
```

9

Se pueden encontrar todos los paquetes de la comunidad Typst en https://typst.app/universe/search/?kind=packages