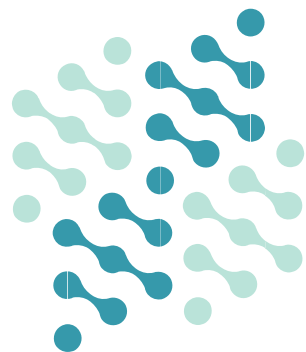


Introducción a la Computación

Primer Cuatrimestre de 2019



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Programa

Un **programa** es una secuencia finita de **instrucciones**.

Ejemplo:

- 1.- Moje el cabello.
- 2.- Coloque shampoo.
- 3.- Masajee suavemente y deje actuar por 2 min.
- 4.- Enjuague.
- 5.- Repita el procedimiento.

Programa

Otro ejemplo:

Ingredientes: 15 huevos, 600 gramos de harina, 600 gramos de azúcar

- 1.- Mientras no estén espumosos, batir los huevos junto con el azúcar,
- 2.- agregar la harina en forma envolvente sin batir,
- 3.- batir suavemente,
- 4.- colocar en el horno a 180 grados,
- 5.- si le clavo un cuchillo y sale húmedo, entonces ir a 4.-
- 6.- retirar del horno,
- 7.- mientras no esté frío, esperar
- 8.- desmoldar y servir

Instrucción

Una **instrucción** es una operación que:

- transforma los datos (el *estado*), o bien
- modifica el flujo de ejecución.

Instrucción

Una **instrucción** es una operación que:

- transforma los datos (*el estado*), o bien
- modifica el flujo de ejecución.

- 1.- Moje el cabello.
- 2.- Coloque shampoo.
- 3.- Masajee suavemente y deje actuar por 2 min.
- 4.- Enjuague.
- 5.- Repita el procedimiento.

Instrucción

Una **instrucción** es una operación que:

- transforma los datos (el *estado*), o bien
- modifica el flujo de ejecución.

- 1.- Mientras no estén espumosos, batir los huevos junto con el azúcar,
- 2.- agregar la harina en forma envolvente sin batir,
- 3.- batir suavemente,
- 4.- colocar en el horno a 180 grados,
- 5.- si le clavo un cuchillo y sale húmedo, entonces ir a 4.-
- 6.- retirar del horno,
- 7.- mientras no esté frío, esperar
- 8.- desmoldar y servir

Datos

Los programas manipulan **valores** de diferentes **tipos**.


Ejemplos:


- 1 es un valor de tipo **entero**.
- 2.5 es un valor de tipo **real**.
- “Hola” es un valor de tipo **string**.
- false es un valor de tipo **bool (lógico)**.

Tipos de datos

Enteros (`int`):

Los enteros para una computadora son parecidos a los enteros matemáticos, con una *pequeña* diferencia: están acotados por encima y por debajo.

- ∞ , ..., -2, -1, 0, 1, 2, .. ∞ 

-2.147.483.648, ..., -2, -1, 0, 1, 2, ..., 2.147.483.647 

¿Por qué esas cotas?

Porque lenguajes como C++ o Python usan una **cantidad finita de bits** para representar enteros. Por ejemplo, 32 bits.

Tipos de datos

Operaciones de enteros:

Operador C++	Operación	Ejemplo
+	Suma	$3 + 4 \rightarrow 7$
-	Resta	$6 - 2 \rightarrow 4$
*	Producto	$2 * 8 \rightarrow 16$
/	División	$5 / 2 \rightarrow 2$
%	Resto	$5 \% 2 \rightarrow 1$
-	Negación (unaria)	-6

Tipos de datos

Comparaciones entre enteros:

Operador C++	Operación
<code>i==k</code>	Igualdad
<code>i!=k</code>	Distinto
<code>i<k</code>	Comparación por menor
<code>i>k</code>	Comparación por mayor
<code>i<=k</code>	Comparación por menor o igual
<code>i>=k</code>	Comparación por mayor o igual

Tipos de datos

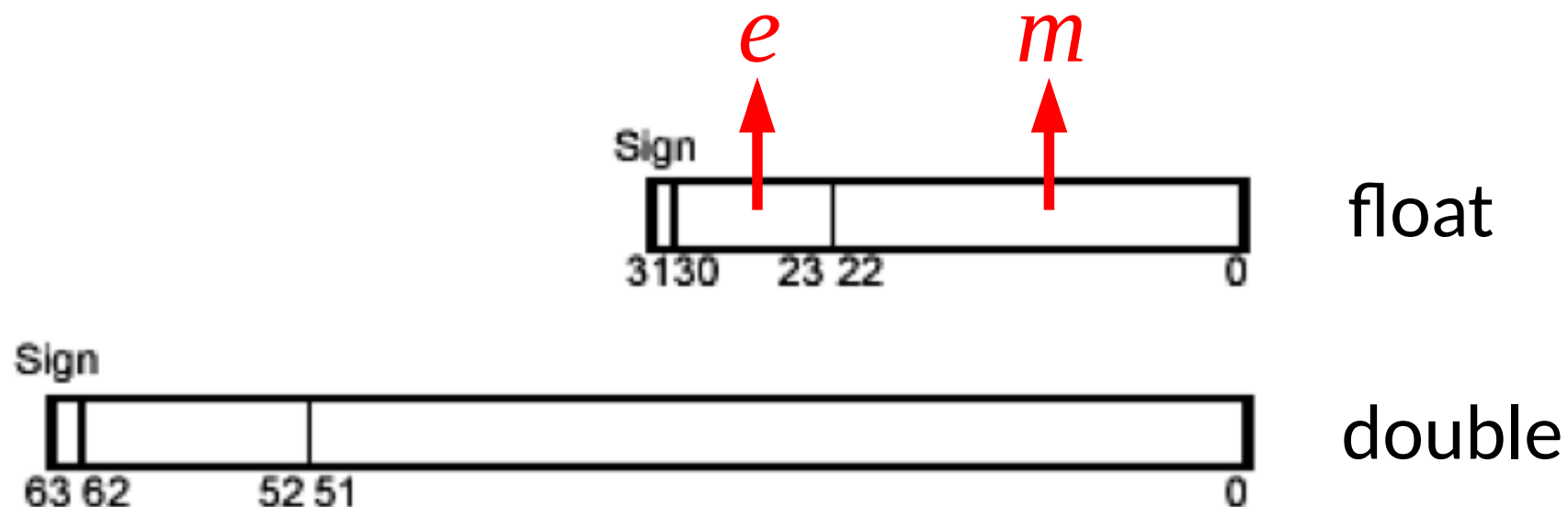
Reales (**float** y **double** en C++):

Un real f representado en punto flotante es un par (m, e) tal que:

$$f \approx \pm m * 10^e \quad \text{donde } 0,1 \leq m < 1$$

(m : mantisa; e : exponente)

Son *bastante* diferentes de los reales matemáticos. Están **acotados por encima y por debajo**, pero también están acotados en la **precisión**.



Tipos de datos

Operaciones de reales:

Operador C++	Operación
+	Suma
-	Resta
*	Producto
/	División
-	Negación (unaria)

Operador C++	Operación
$i==k$	Igualdad
$i!=k$	Distinto
$i<k$	Menor que
$i>k$	Mayor que
$i\leq k$	Menor o igual que
$i\geq k$	Mayor o igual que

(*) No conviene usar $i==k$ entre reales, por los errores de representación. Es probable que querramos que 0.6666667 y 0.6666666 sean considerados *iguales* en la práctica. Conviene usar: $\text{abs}(i - k) < \text{eps}$.

Tipos de datos

Valores de verdad (bool):

Hay dos valores de verdad posibles: “verdadero” (*true*) y “falso” (*false*).

Operaciones de booleanos:

Operador C++	Operación
!	Negación
&&	Conjunción
	Disyunción

Tablas de verdad:

p	!p
true	false
false	true

p	q	p && q	p q
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

Tipos de datos

Arreglo (*array*):

Un arreglo es una **colección de valores** (o *elementos*).

Se accede a cada valor mediante un **índice** (entero ≥ 0).

Todos los valores son de un **mismo tipo**: p.ej., arreglo de enteros.

45	657	-56	4	23	-5	0	113
0	1	2	3	4	5	6	7

Los índices de un arreglo de N elementos
no van de 1 a N, sino **de 0 a N-1**.

Tipos de datos

Operaciones de arreglos:

Operador C++	Operación
<code>array <T, n> a;</code>	Crea un arreglo de tipo <i>T</i> y tamaño <i>n</i> .
<code>a[i]</code>	<i>i</i> -ésimo elemento del arreglo <i>a</i> .
<code>a.size()</code>	Longitud del arreglo <i>a</i> .

Para usar el tipo `array` en C++, incluir al principio:

```
#include <array>
using namespace std;
```

Nota: Hay otras formas de trabajar con arreglos y tipos parecidos en C++. En la materia elegimos `std::array`, que nos parece la más sencilla de aprender.

Tipos de datos

Arreglos en C++:

```
#include <array>
using namespace std;
```

```
array <int, 3> a;
a[0] = 1;
a[1] = 4253;
a[2] = -8;
```

```
array <int, 3> b = {1, 4523, -8};
```


Tipos de datos

Cadena de caracteres (**string**):

Un **character** (**char**) es un símbolo válido en la computadora:

```
abcdefghijklmnopqrstuvwxyz  
ABCDEFGHIJKLMNOPQRSTUVWXYZ  
1234567890  
!@#$%*()-_+=~`' : ; , . "<>?/  
etc.
```

En C++ se escriben entre comillas simples: `'a'`.

Un **string** es una cadena o secuencia de caracteres.

En C++ se escriben entre comillas dobles: `"a"`.

Tipos de datos

Operaciones de strings:

Operador C++	Operación
s.size()	Devuelve la longitud del string s.
s[i]	Devuelve el i-ésimo caracter del string s.
< <= == > >=	Compara dos strings. Ej: s1 <= s2
+	Pega dos cadenas. Ej: s1 + s2
...	...

Para usar el tipo string en C++, incluir al principio:

```
#include <string>
using namespace std;
```

Nota: Hay varias formas de trabajar con strings en C++.
En la materia elegimos **std::string**, que nos parece la más sencilla de aprender.

Tipos de datos - Resumen

Tipo de datos	Ejemplos
bool	true, false
int	3, 0, -5
float, double	3.0, 0.0, -5.0, 3.141592
array	[10, 20, 30], ['a', 'b', 'c']
string	"pepe", "coco"

Memoria

Durante la ejecución de un programa, sus datos se almacenan en la **memoria**.

La memoria de una computadora es una secuencia numerada de **celdas** o **posiciones**, en las cuales podemos almacenar datos.

Unidad elemental: el **bit**, que toma valores **0** ó **1**.

8 bits	= 1 byte → Unidad mínima más usada.
1024 bytes	= 1 KB (kilobyte)
1024 KB	= 1 MB (megabyte)
1024 MB	= 1 GB (gigabyte)
1024 GB	= 1 TB (terabyte)
1024 TB	= 1 PB (petabyte)
...	

Soporte físico: electrónico, magnético, óptico, ...

Variable

Una **variable** es un **nombre** que denota la **dirección** de una celda en la memoria, en la cual se almacena un **valor**.

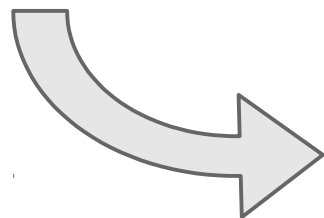
En esa celda de memoria es posible:

- **leer** el valor almacenado, y
- **escribir** un valor nuevo, que reemplace al anterior.

En C++, cada variable tiene asociado un **tipo** (bool, int, float, char, etc.), por lo cual es necesario **declararlas** antes de usarlas.

Ejemplo en C++:

```
int x;           // Declaro la variable x de tipo int.  
x = 10;          // Asigno el valor 10 a la variable x.  
cout << x;       // Imprimo en pantalla el valor de x.
```



Para imprimir en pantalla en C++, incluir al principio:
`#include <iostream>`
`using namespace std;`

Expresión

Una **expresión** es una combinación de literales, variables y operadores.

Un **literal** es un valor particular utilizado directamente en el código.

La **evaluación** de una expresión arroja como resultado un valor.

Ejemplos: ¿Qué valores resultan de evaluar estas expresiones (suponiendo que s es un string con valor “hola”)?

1

`s.size() + 6`

`(1>0) || !('a'<'b')`

`(5.6 > 2.0) && (s.size() < 2)`

Literales en los ejemplos de arriba: 1 6 1 0 'a' 'b' 5.6 2.0 2

Asignación

VARIABLE = EXPRESIÓN ;

Almacena el valor de la *EXPRESIÓN* en la dirección en memoria denotada por *VARIABLE*.

Ejemplos:

x = 1000;	// Bien.
1000 = x ;	// Mal. 1000 no es una variable.
x = y ;	// Bien.
x = x ;	// Bien. Aunque no tiene efecto.
x = x + y * 22;	// Bien.
x + 1 = y ;	// Mal. x + 1 no es una variable.

Estado

Se denomina **estado** al valor de todas las variables de un programa en un punto de su ejecución.

**Es una “foto” de la memoria
en un momento determinado.**

Estado

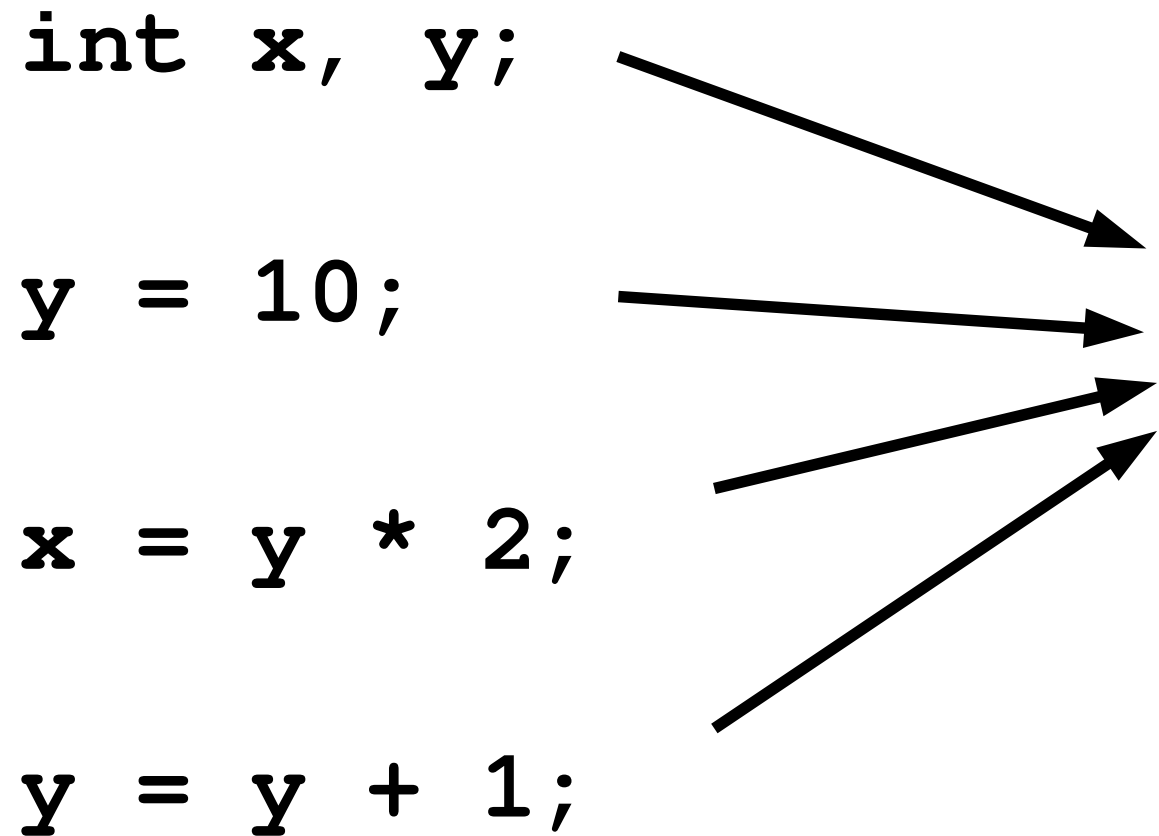
Ejemplo:

```
int x, y;
```

```
y = 10;
```

```
x = y * 2;
```

```
y = y + 1;
```



Instrucciones en el
lenguaje de
programación C++

Estado

Ejemplo:

```
int x, y;
```

```
    {x=↑ ∧ y=↑ }
```

```
y = 10;
```

```
    {x=↑ ∧ y=10 }
```

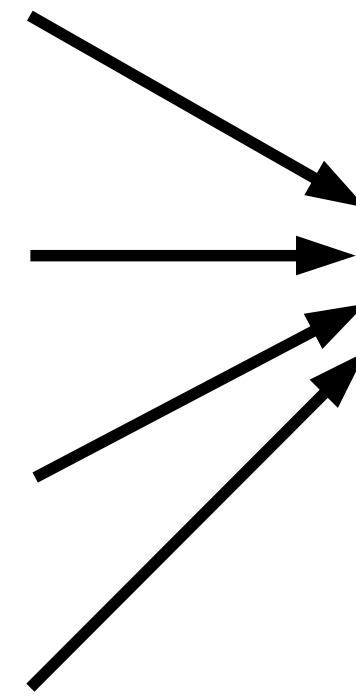
```
x = y * 2;
```

```
    {x=20 ∧ y=10 }
```

```
y = y + 1;
```

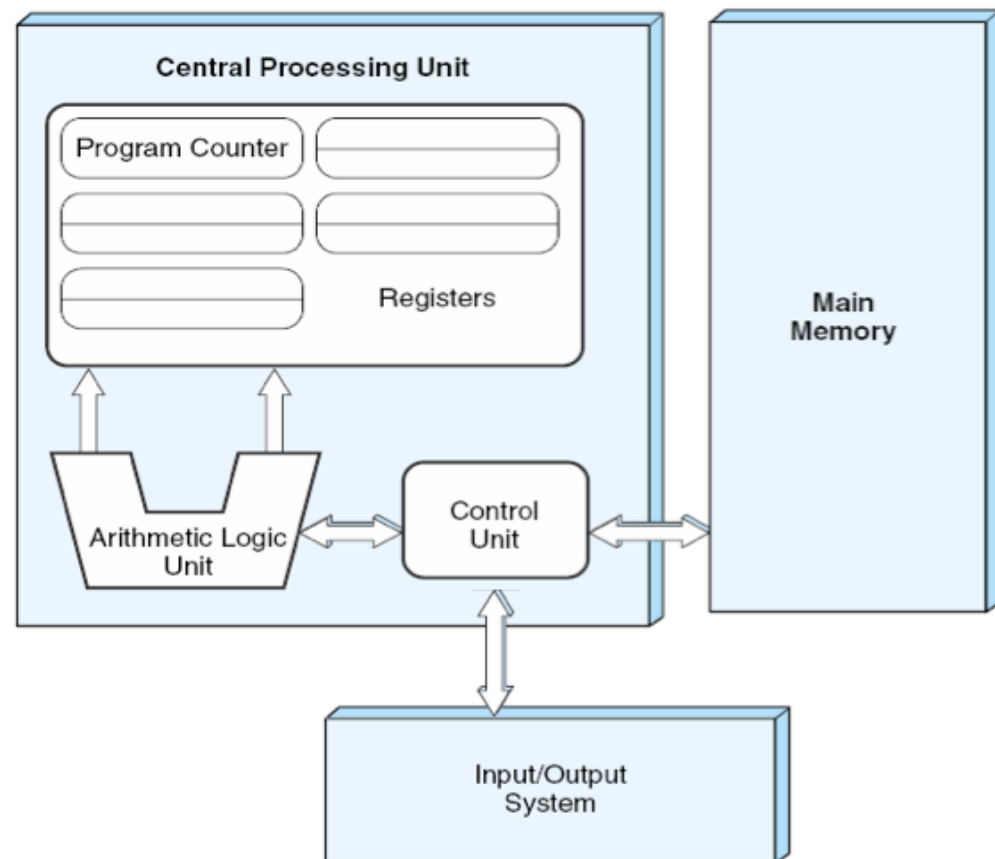
```
    {x=20 ∧ y=11 }
```

Descripción de los
estados del programa
en lógica de predicados



↑ significa “valor indefinido”

Organización de computadoras



- Unidad Central de Procesamiento (CPU):
 - Unidad de control
 - Unidad aritmético-lógica
 - Registros
- Memoria
- Entrada / Salida
 - Disco rígido, pen drive, mouse, teclado, video, audio, red, wifi, etc.

Pasos a seguir por la CPU para ejecutar una instrucción:

1. Leer la siguiente instrucción a ejecutar.
2. Leer de memoria los datos necesarios y guardarlos en los registros.
3. Ejecutar la instrucción sobre los registros.
4. Almacenar el resultado en la posición de memoria que corresponda.

Repaso

- Tipos de datos y sus operaciones:

int	+	-	*	/	%	==	!=	<	>	<=	>=
float, double	+	-	*	/		==	!=	<	>	<=	>=
bool	!	&&									
array	a[i]	a.size()	array<T, n>	a;							
string	s[i]	s.size()	"hola"	<	>	==	+	...			

- Valores, expresiones, variables, literales.

`(x > 2.0) && (s.size() < 2)`

- Programa: secuencia finita de instrucciones.

- Asignación: ***VARIABLE = EXPRESIÓN ;***

- Memoria, estado.

`{ x=20 ∧ y=10 } y = y + 1; { x=20 ∧ y=11 }`

- Próximos temas: condicionales, ciclos, funciones.