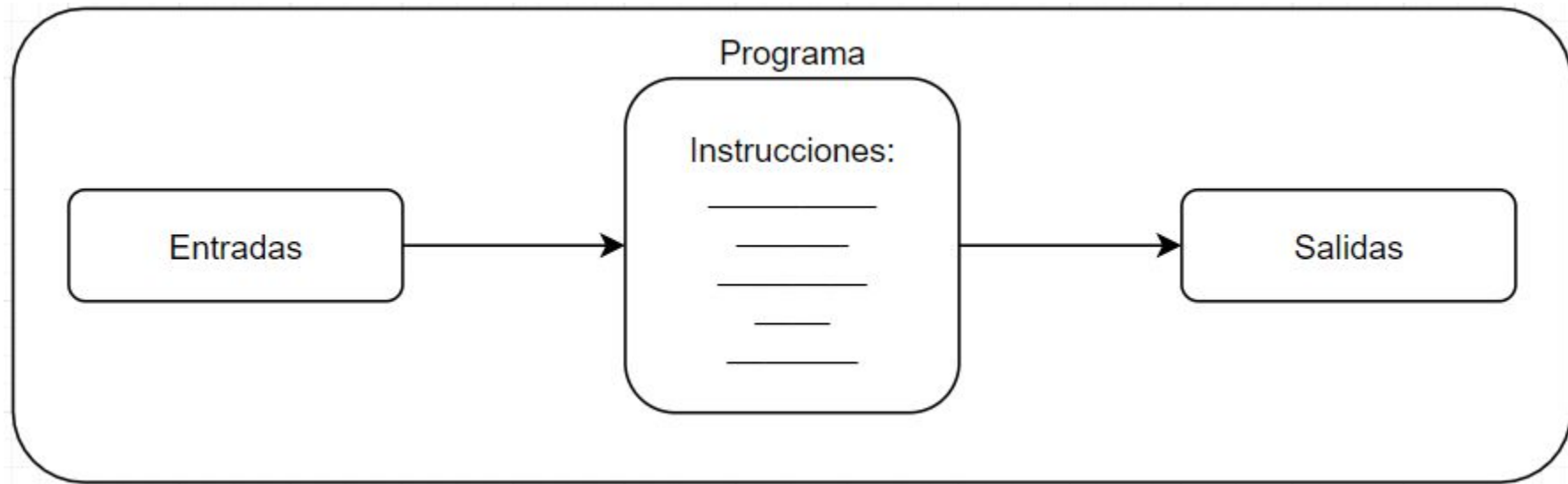


# Introducción al cómputo y al desarrollo de software

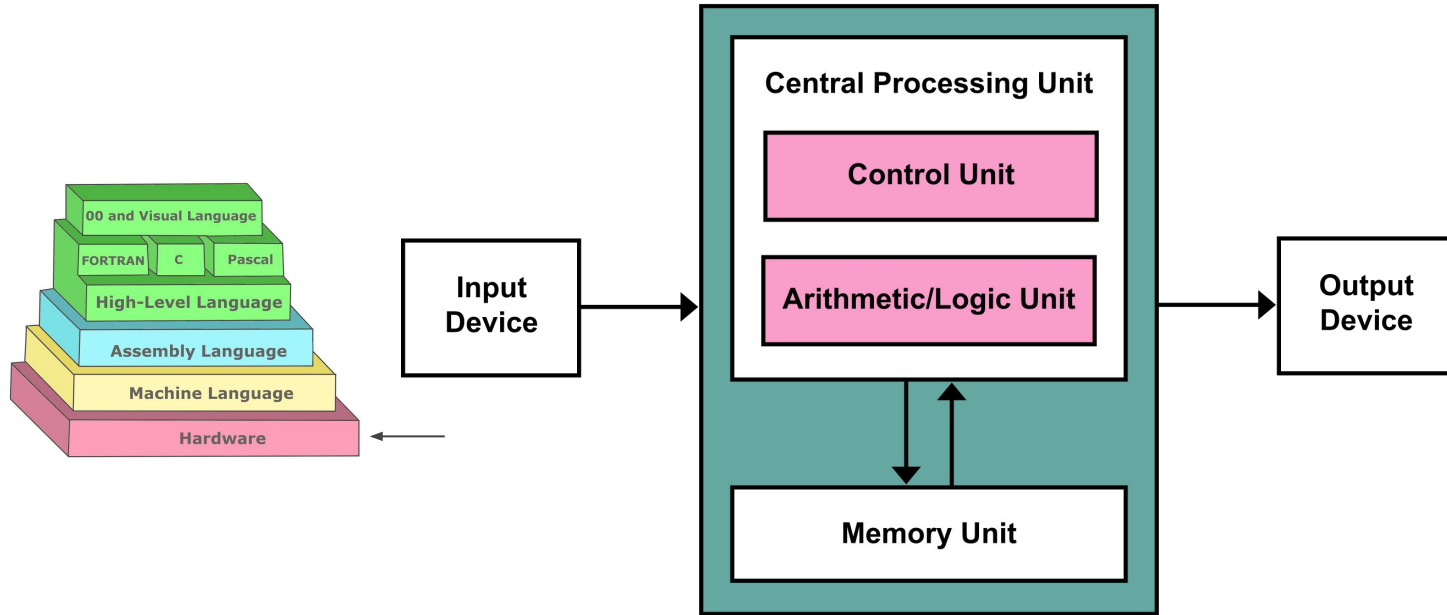
Clase 1: Programas, lenguajes, estructura y operaciones básicas.

Lic. Agustín Bernardo & MSc. Rodrigo Bonazzola

# ¿Qué es un programa?

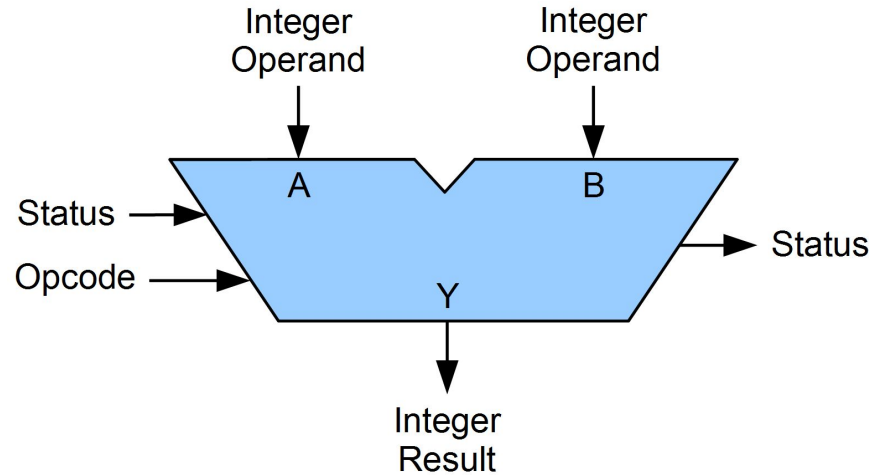
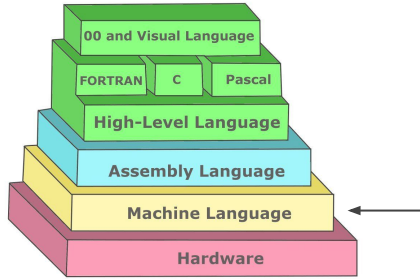


# ¿Cómo se llevan a cabo las instrucciones?



Arquitectura de  
Von Neumann

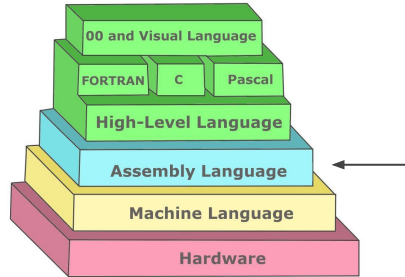
# Operaciones, ALU, y código de máquina



Machine code bytes

```
B8 22 11 00 FF
01 CA
31 F6
53
8B 5C 24 04
8D 34 48
39 C3
72 EB
C3
```

# Asm (assembly)



```

C000                                ORG    ROM+$0000 BEGIN MONITOR
C000 8E 00 70  START  LDS     #STACK

*****
* FUNCTION: INITA - Initialize ACIA
* INPUT: none
* OUTPUT: none
* CALLS: none
* DESTROYS: acc A

0013      RESETA EQU    %00010011
0011      CTLREG EQU    %00010001

C003 86 13  INITA  LDA A  #RESETA  RESET ACIA
C005 B7 80 04      STA A  ACIA
C008 86 11          LDA A  #CTLREG  SET 8 BITS AND 2 STOP
C00A B7 80 04      STA A  ACIA

C00D 7E C0 F1      JMP     SIGNON   GO TO START OF MONITOR

*****
* FUNCTION: INCH - Input character
* INPUT: none
* OUTPUT: char in acc A
* DESTROYS: acc A
* CALLS: none
* DESCRIPTION: Gets 1 character from terminal

C010 B6 80 04  INCH  LDA A  ACIA      GET STATUS
C013 47          ASR A                SHIFT RDRF FLAG INTO CARR
C014 24 FA      BCC  INCH             RECIEVE NOT READY
C016 B6 80 05  LDA A  ACIA+1          GET CHAR
C019 84 7F      AND A  #$7F          MASK PARITY
C01B 7E C0 79  JMP     OUTCH          ECHO & RTS

*****
* FUNCTION: INHEX - INPUT HEX DIGIT
* INPUT: none
* OUTPUT: Digit in acc A
* CALLS: INCH
* DESTROYS: acc A
* Returns to monitor if not HEX input

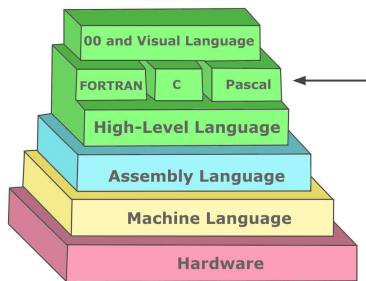
C01E 8D F0  INHEX  BSR     INCH      GET A CHAR
C020 81 30      CMP A  #'0          ZERO
C022 2B 11      BMI     HEXERR      NOT HEX
C024 81 39      CMP A  #'9          NINE
C026 2F 0A      BLE     HEXRTS      GOOD HEX
C028 81 41      CMP A  #'A          NOT HEX
C02A 2B 09      BMI     HEXERR
C02C 81 46      CMP A  #'F
C02E 2E 05      BGT     HEXERR
C030 80 07      SUB A  #7           FIX A-F
C032 84 0F  HEXRTS AND A  #$0F      CONVERT ASCII TO DIGIT
C034 39          RTS

C035 7E C0 AF  HEXERR JMP     CTRL   RETURN TO CONTROL LOOP

```

Hay una relación 1:1  
entre Asm y el código  
de máquina.

# Lenguajes compilados: C



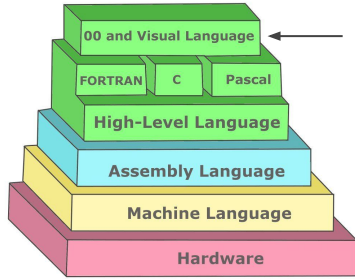
```
1  #include <stdlib.h>
2
3  int main() {
4      printf("Hola, esto sería un programa de ejemplo en C.")
5      return 0;
6  }
```

Nos permiten diagramar una lógica proposicional:

If-else, while, for, etc.

El compilador lo lleva a código de máquina.

# Lenguajes orientados a objetos (python, C++)



```
#include <iostream>

using namespace::std;

class Pato{
private:
    int peso;
    string nombre;
public:
    Pato(string n) {
        nombre = n;
    }

    void cuak () {
        cout<<"Cuac!"<<endl;
    }
};

int main() {
    Pato donald("Donald");
    donald.cuak();

    return 0;
}
```

Abstraemos las entidades que pertenecen al problema a objetos

Los objetos pertenecen a clases

Las clases tienen propiedades y métodos



**1960**  
**MANDE EL HOMBRE**  
**A LA LUNA**  
**CON ASSEMBLER**



**2020**  
**A LA PAGINA WEB**  
**QUE ESCRIBÍ EN PYTHON**  
**NO LE ANDAN LOS BOTONES**



# Entradas y salidas

Por ahora vamos a trabajar con la entrada y salida de la consola.

```
cin >> a;
```

```
cout << "a imprimir"<< endl;
```

```
entradaSalida.cpp X
Clase 2 > entradaSalida.cpp
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int a;
7
8      cout << "Introduzca un número entero." << endl;
9      cin >> a;
10
11     cout << "El número introducido es: " << a << endl;
12     return 0;
13 }
```

# Variables

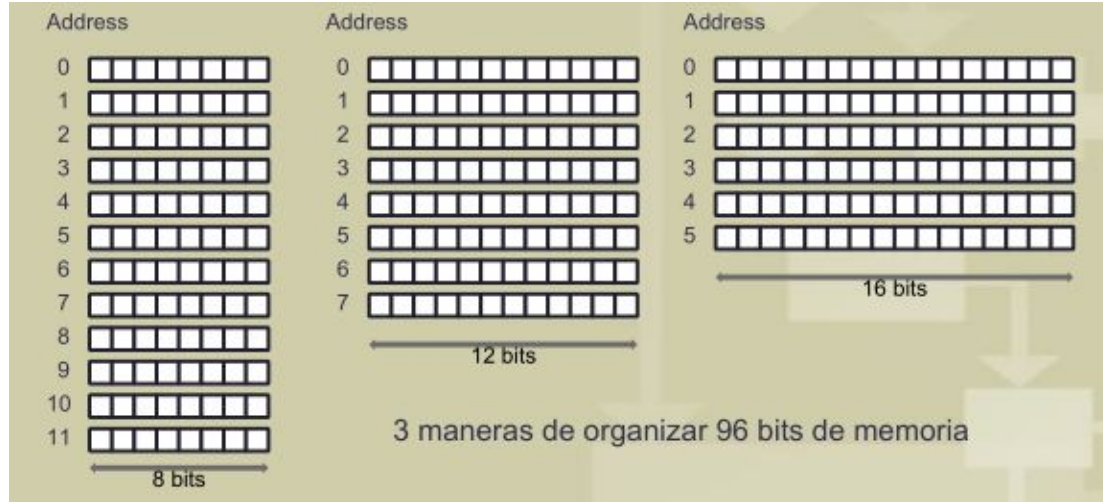
Cada tipo de variable corresponde a una forma de leer el contenido almacenado en ceros y unos.

Hay tipos nativos y tipos definidos por el usuario (UDT).

También hay clases.

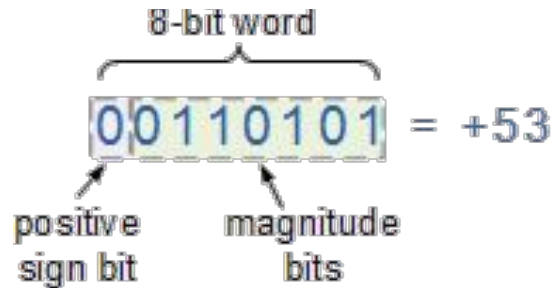
```
int main() {  
    int a = 8;  
    char b = 'b';  
    float c = 3.45e5;  
    bool d = true;  
  
    string e = "Pablito clavó un clavito";  
    Pato f("donald");  
  
    return 0;  
}
```

# Variables, memoria



# Variables, memoria - y números binarios

$$N_D = (-1)^{n_7} \cdot (n_6 \cdot 2^6 + n_5 \cdot 2^5 + \cdots + n_1 \cdot 2^1 + n_0 \cdot 2^0)$$



$$00110101 = (-1)^0 \left( \underbrace{0 \cdot 2^6}_0 + \underbrace{1 \cdot 2^5}_{32} + \underbrace{1 \cdot 2^4}_{16} + \underbrace{0 \cdot 2^3}_0 + \underbrace{1 \cdot 2^2}_4 + \underbrace{0 \cdot 2^1}_0 + \underbrace{1 \cdot 2^0}_1 \right)$$

# Variables, memoria - y números binarios

Para pasar de entero a binario, tomo la parte entera y el módulo de la división por dos.

Es un ejercicio para el lector.

Este programa lo hace directo.

intToBits.cpp X

Clase 2 > intToBits.cpp > main()

```
1  #include <iostream>
2  #include <bitset>
3
4  using namespace std;
5
6  int main() {
7      int a;
8
9      cout << "Introduzca un numero entero." << endl;
10     cin >> a;
11
12     cout << "El numero introducido es: " << a << endl;
13
14     bitset<8> x(a);
15
16     cout << "Su representacion en binario es: " << x << endl;
17
18     return 0;
19 }
```

# Variables, memoria - y caracteres

Los caracteres (char) se almacenan en 7 bits según el código ASCII.

Hay otros códigos con más caracteres.

USASCII code chart

<div> <div> <div> <div>b7</div> <div>b6</div> <div>b5</div> </div> <div> <div>b4</div> <div>b3</div> <div>b2</div> <div>b1</div> </div> </div> <div> <div>Column</div> <div>Row</div> </div> </div>	0	1	2	3	4	5	6	7
0 0 0 0 0	NUL	DLE	SP	0	@	P	\	p
0 0 0 0 1	SOH	DC1	!	1	A	Q	a	q
0 0 0 1 0	STX	DC2	"	2	B	R	b	r
0 0 0 1 1	ETX	DC3	#	3	C	S	c	s
0 0 1 0 0	EOT	DC4	\$	4	D	T	d	t
0 0 1 0 1	ENQ	NAK	%	5	E	U	e	u
0 0 1 1 0	ACK	SYN	&	6	F	V	f	v
0 0 1 1 1	BEL	ETB	'	7	G	W	g	w
0 1 0 0 0	BS	CAN	(	8	H	X	h	x
0 1 0 0 1	HT	EM	)	9	I	Y	i	y
0 1 0 1 0	LF	SUB	*	:	J	Z	j	z
0 1 0 1 1	VT	ESC	+	;	K	[	k	{
0 1 1 0 0	FF	FS	,	<	L	\	l	
0 1 1 0 1	CR	GS	-	=	M	]	m	}
0 1 1 1 0	SO	RS	.	>	N	^	n	~
0 1 1 1 1	SI	US	/	?	O	_	o	DEL

1724 x 1241

# Variables, memoria - y caracteres

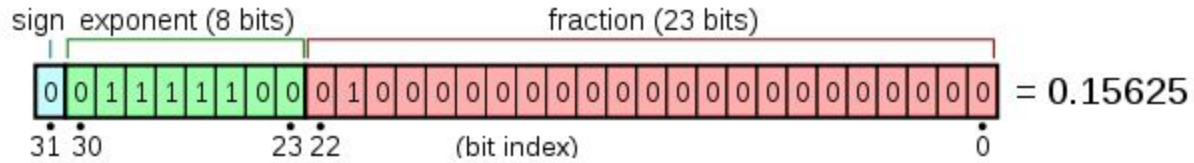
Este programa toma un caracter y nos devuelve su valor entero, y en binario.

Modificar el valor del caracter sumándole números suele ser muy útil.

```
charToBits.cpp X
Clase 2 > charToBits.cpp > main()
1  #include <iostream>
2  #include <bitset>
3
4  using namespace std;
5
6  int main() {
7      char a;
8
9      cout << "Introduzca una letra." << endl;
10     cin >> a;
11
12     cout << "La letra introducida es: " << a << endl;
13     cout << "Su valor entero es: " << int(a) << endl;
14
15     bitset<8> x(a);
16
17     cout << "Su representacion en binario es: " << x << endl;
18
19     return 0;
20 }
```

# Variables, memoria - y floats

Para almacenar un número flotante se usan 32 bits (es 'más pesado').



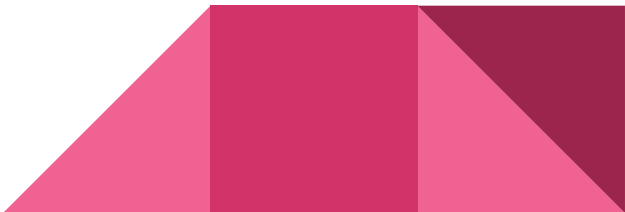
Tiene signo, exponente y fracción.



# Reglas para nombrar variables

- Se pueden usar letras latinas (excepto la ñ y las vocales con acento o diéresis), números y el guión bajo.
- No se puede comenzar con un número.
- No se pueden usar palabras reservadas de C (veremos algunas después).

## Ejemplos:

- Nombre válidos: x, A, Numero, Numero2, \_variable\_.
  - Nombres inválidos: 2x, hola.mundo, int.
- 

# Operaciones

- Suma: `var + var`
- Resta: `var - var`
- División entera: `int / int`
- Módulo: `int % int`
- División flotante: `float / float`
- AND: `&&`
- OR: `||`
- Dirección: `&a`
- Contenido: `*a`

```
#include <iostream>
#include <bitset>

using namespace std;

int main() {
    float f1 = 420.69;
    int n = 12;
    int d = 5;
    char a = 'a';

    cout << "La suma de n y d es: " << n+d << endl;
    cout << "El cociente de n y d es: " << n/d << endl;
    cout << "El resto de n y d es: " << n%d << endl;
    cout << "Si le sumo n a la letra a obtengo: " << char(a+d) << endl;

    cout << "Si divido un entero por un flotante: " << n/f1 << endl;

    cout << "La dirección de memoria donde está f1 es: " << &f1 << endl;
    return 0;
}
```

# Operadores de comparación

- > : mayor que
- < : menor que
- >= : mayor o igual que
- <= : menor o igual
- == : igual a (se usan dos '=' para evitar confundirlo con el operador de asignación)
- != : distinto de

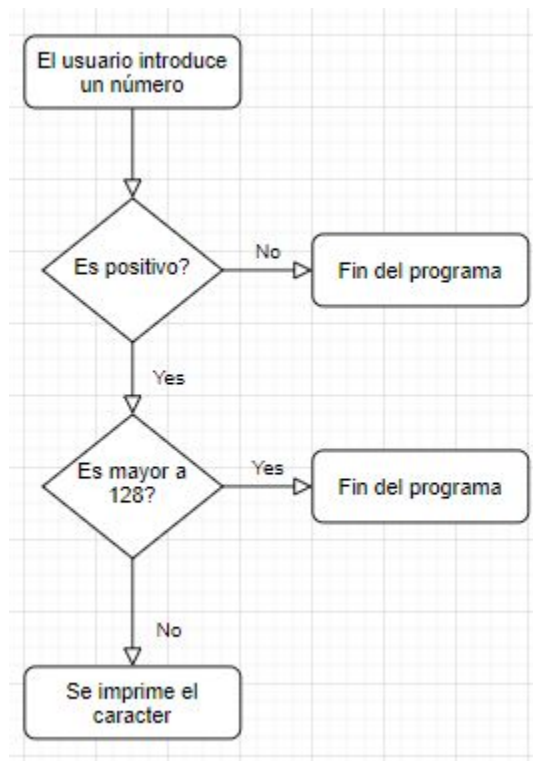


# Ejercicio:

- Escribir un programa que acepte dos números enteros de la consola, e imprima su suma, resta, división y módulo. Además, que intente crear una letra con ellas y que la imprima.



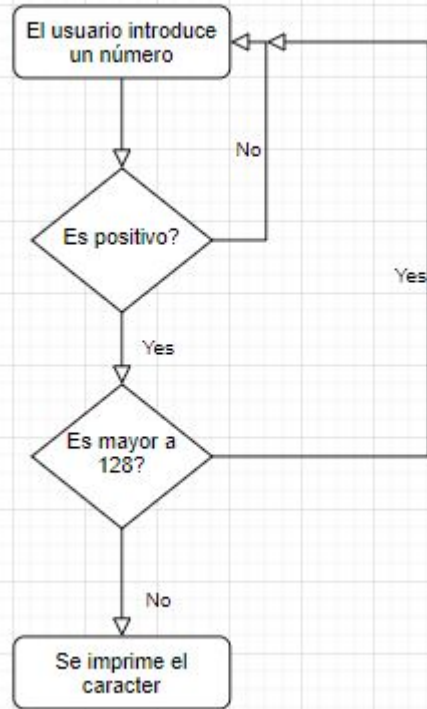
# Flujo secuencial de un programa



# Flujo secuencial de un programa - if

```
firstif.cpp X
Clase 2 > firstif.cpp > ...
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int a;
7      cout << "Introduzca el valor de a" << endl;
8      cin >> a;
9
10     if (a<0){
11         cout<<"La proxima introduzca un valor positivo."<<endl;
12     }
13     else
14     {
15         cout<<"El valor es positivo. " <<endl;
16     }
17
18     return 0;
19 }
```

# Flujo secuencial de un programa - while



firstWhile.cpp X

Clase 2 > firstWhile.cpp > main()

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int a = -1;
7
8      cout<<"Ingrese un número positivo, menor a 128." <<endl;
9      while (a<0 || a>128){
10         cin>>a;
11     }
12     cout<<"Al fin! es: "<< a << endl;
13 }
```

# Flujo secuencial de un programa - for

firstFor.cpp X

Clase 2 > firstFor.cpp > main()

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int a = -1;
7
8      cout<<"Ingrese un número positivo, menor a 128." <<endl;
9
10     for(int i=0; i<a; i++){
11         cout<<"Contando números: "<<i<<endl;
12     }
13     return 0;
14 }
```



# Ejercicios:

- Escribir un programa que devuelva los factores primos de un número entero.
- Escribir un programa que sume todos los números del 1 a N, y devuelva su valor.



Más ejercicios en la guía.

¿Preguntas?

