

# Programación y Análisis de Algoritmos

Dr. Norberto Alejandro Hernández Leandro

CIMAT-Unidad Monterrey

Septiembre 2020

# Contenido

- Historia
- ¿Qué es una computadora?
- Introducción al lenguaje C/C++
- Introducción a Programación Orientada a Objetos
- Introducción al análisis de algoritmos

# Historia



# Historia



Blaise Pascal (1623–1622)



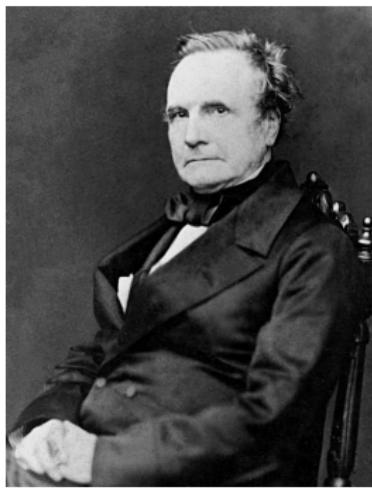
Pascalina (1645)

# Historia

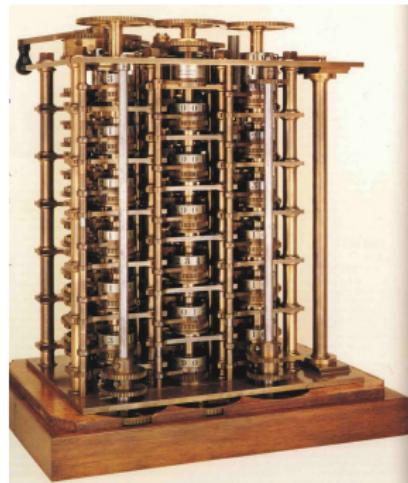
## Padres de la computación

- Charles Babbage
- Alan Turing
- John Atanasoff
- John Von Neumann

# Historia



Charles Babbage  
(1791–1871)

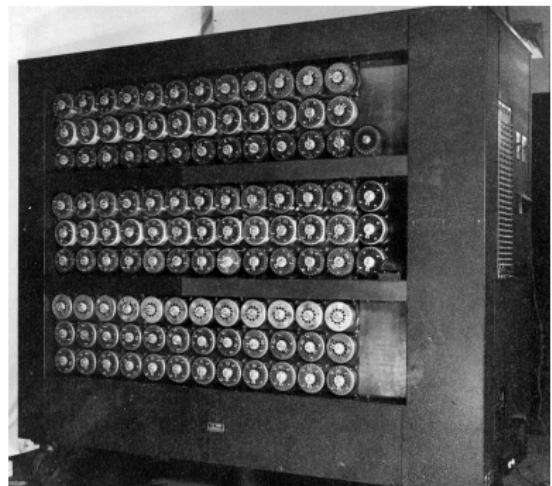


Máquina diferencial (1822)

# Historia

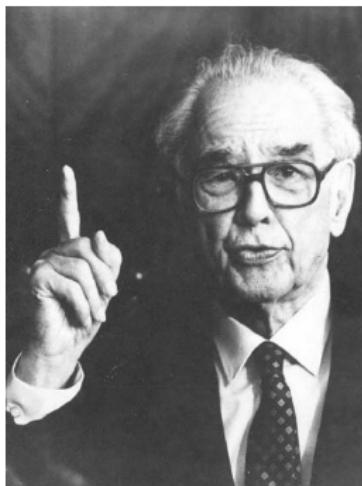


Alan Turing (1912–1954)

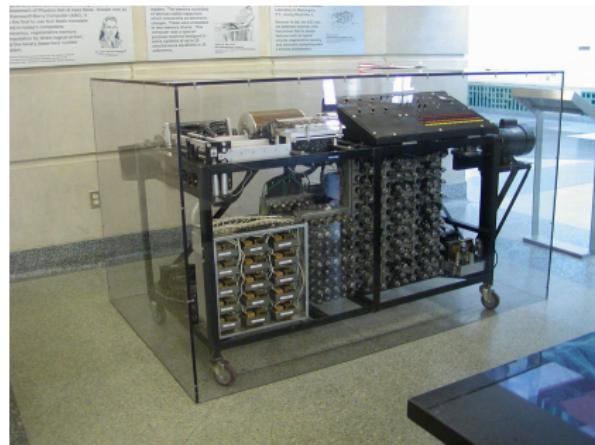


Máquina Bombe (1936)

# Historia



John Atanasoff  
(1903–1995)

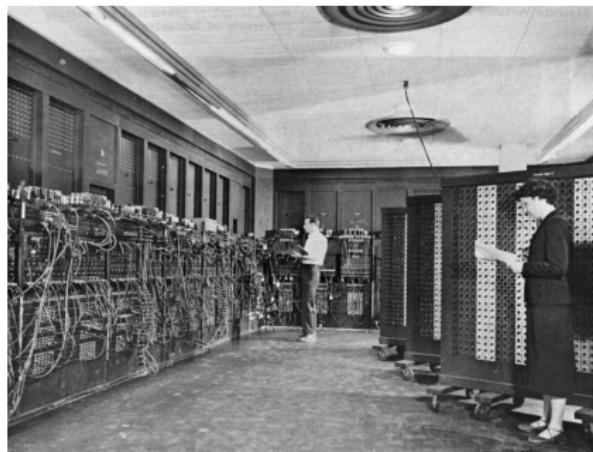


Computadora ABC (1942)

# Historia



John Von Neumann  
(1903–1957)



Computadora ENIAC  
(1946)

# ¿Qué es una computadora?

Es un dispositivo eléctrico/mecánico programable capaz de ejecutar un conjunto de secuencias aritméticas o lógicas para procesar información.

# Componentes de una computadora



- Dispositivos de entrada (Teclado, Mouse, Micrófono, etc.)
- Dispositivos de salida (Monitor, Auriculares, impresora, etc.)
- Unidades de procesamiento
- Memoria

# Componentes de una computadora

## Unidades de procesamiento



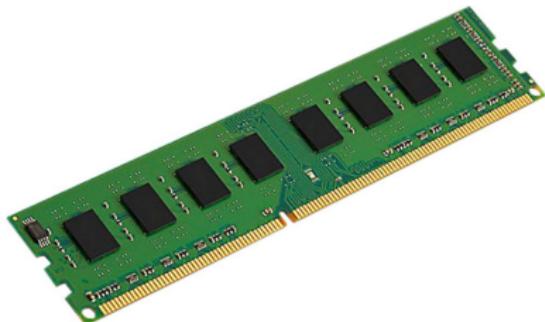
Unidad de procesamiento central (CPU)



Unidad de procesamiento gráfico (GPU)

# Componentes de una computadora

## Unidades de procesamiento



Memoria RAM

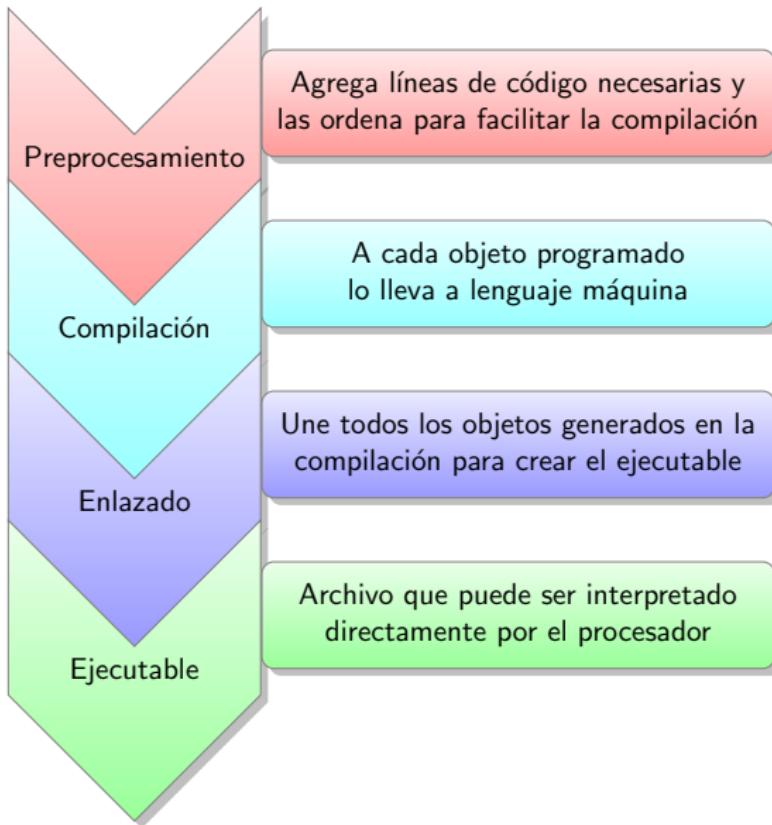


Memoria no volátil

# Introducción a programación en C/C++

- El lenguaje C fue desarrollado a partir de 1969 en los laboratorios Bell de AT&T
- El principal creador fue Dennis Ritchie
- En 1979, Bjarne Stroustrup comenzó el desarrollo de C++ en la Universidad A&M de Texas
- C++ es una extensión del lenguaje de programación C
- C++ permite trabajar con la codificación orientada a objetos
- C++ es capaz de encapsular funciones, constantes y objetos para evitar ambigüedades en el código
- Son considerados lenguajes de nivel medio porque permiten la escritura de código de alto nivel, y cuenta con herramientas que permiten un control a bajo nivel

# Compilación



# IDEs y Compiladores

- **Compiladores**
  - **Windows**: MinGW, Cygwin y Visual Studio
  - **Mac OS**: gcc y g++
  - **Linux**: gcc y g++
- **IDEs**
  - **Windows**: Visual Studio, Dev-C++, Code::Blocks, Eclipse, Netbeans, entre otros.
  - **Mac OS**: Visual Studio, Xcode, Dev-C++, Code::Blocks, Eclipse, Netbeans, entre otros.
  - **Linux**: VS code, Dev-C++, Code::Blocks, Eclipse, Netbeans, entre otros.

# Elementos del lenguaje C/C++

- **Identificadores:** nombre de variables, estructuras, funciones, objetos, etc.
- **Palabras clave:** nombres de tipos, estructuras de control, calificadores de tipos y almacenamiento.
- **Constantes y macros:** definidas por DEFINE.
- **Cadenas de caracteres.**
- **Operadores:** +, \*, ...
- **Comentarios:** /\* Comentario \*/ o //Comentario.

# Palabras reservadas

char	int	float	double
if	else	do	while
for	switch	short	long
extern	static	default	continue
break	register	sizeof	typedef

# Funciones de entrada y salida

- stdio.h: printf y scanf (para C)
- iostream: cout y cin (para C++)

Ejemplo:

---

```
1 #include <iostream>
2 int main()
3 {
4     int n;
5     //Entrada de información vía consola
6     //Imprime en consola
7     std::cout << "Dame un número entero: " << std::endl;
8     //Guarda información tecleada en consola
9     std::cin >> n;
10    return 0;
11 }
```

---

# Operadores

- **Aritméticos:** + (suma), - (resta), \* (multiplicación), / (división) y % (módulo)
- **Asignación:** =
- **Asignación compuesta:** +=, -=, \*=, /=, %= y operador ternario (asignación condicional)

Ejemplo:

---

```
1      c += a; //Equivalente a c=c+a
2      c -= a; //Equivalente a c=c-a
3      c *= a; //Equivalente a c=c*a
4      c /= a; //Equivalente a c=c/a
5      c %= a; //Equivalente a c=c%a;
6      //Operador ternario
7      //Si se cumple la condición c = valor1
8      //si no, c = valor2
9      c = condición ? valor1:valor2;
```

---

# Operadores

- **Lógicos:**

&&        y  
||           o  
!        negación

- **Incrementales:**

++    incremento  
--    decremento

- **Relacionales:**

<           menor que  
>           mayor que  
<=          menor o igual que  
>=          mayor o igual que  
==          igual que  
!=          distinto que

- **Conversión:** (tipo) dato

## ¡Cuidado!

- El operador división permite tambien la división entera.
- Diferenciar los operadores = y ==.
- Los operadores relacionales se utilizan para comparar valores.
- Al utilizar los operadores relacionales tener cuidado con los valores flotantes muy cercanos.
- Los operadores lógicos y relacionales regresan el valor true o 1 si la expresión es verdadera, y false o 0 de otra manera.
- Las expresiones lógicas se evalúan de izquierda a derecha, siguiendo las reglas de agrupación por paréntesis y la jerarquía de las operaciones lógicas.
- Los operadores incrementales pueden usarse antes o después de la variable.

# ¡Cuidado!

- Error:

---

```
1     int a=1, b=2;
2     double c = a/b;
```

---

- Solución:

---

```
1     int a=1, b=2;
2     double c = (double)a/b;
3     double c = a/(double)b;
4     double c = (double)a/(double)b;
```

---

# ¡Cuidado!

- ¿Qué hace?

---

```
1 int main() {
2     int a=1, b=2;
3     if (a==b){
4         cout << "a y b son iguales" << endl;
5     } else{
6         cout << "a y b no son iguales" << endl;
7     }
8     return 0;
9 }
```

---

# ¡Cuidado!

- ¿Qué hace?

---

```
1     int main() {
2         int a=3, b, c;
3         b = ++a;
4         c = a++;
5         return 0;
6     }
```

---

# Operador de conversión

Este operador convierte objetos de un tipo a otro:

---

```
1     tipo1 a;  
2     tipo2 b = (tipo1) a;
```

---

Como ya se mencionó, el operador de conversión puede ser necesario para realizar algunas operaciones que, en otro caso, tendrían un sentido distinto:

---

```
1     int a=1, b=2;  
2     double c = (double) a/b;
```

---

# Operador de conversión

- Los operadores pueden estar aplicados con operandos de diferentes tipos: las variables de tipo más pequeño se convierten al tipo más grande entre los operandos.
- En caso de la asignación, el valor de la derecha se convierte en el tipo del operando de la izquierda.

---

```
1     int i = 2;
2     float x = 2.3;
3     double c = i*x;
4     int j = i*x;
```

---