# Informática I

Claudio Paz

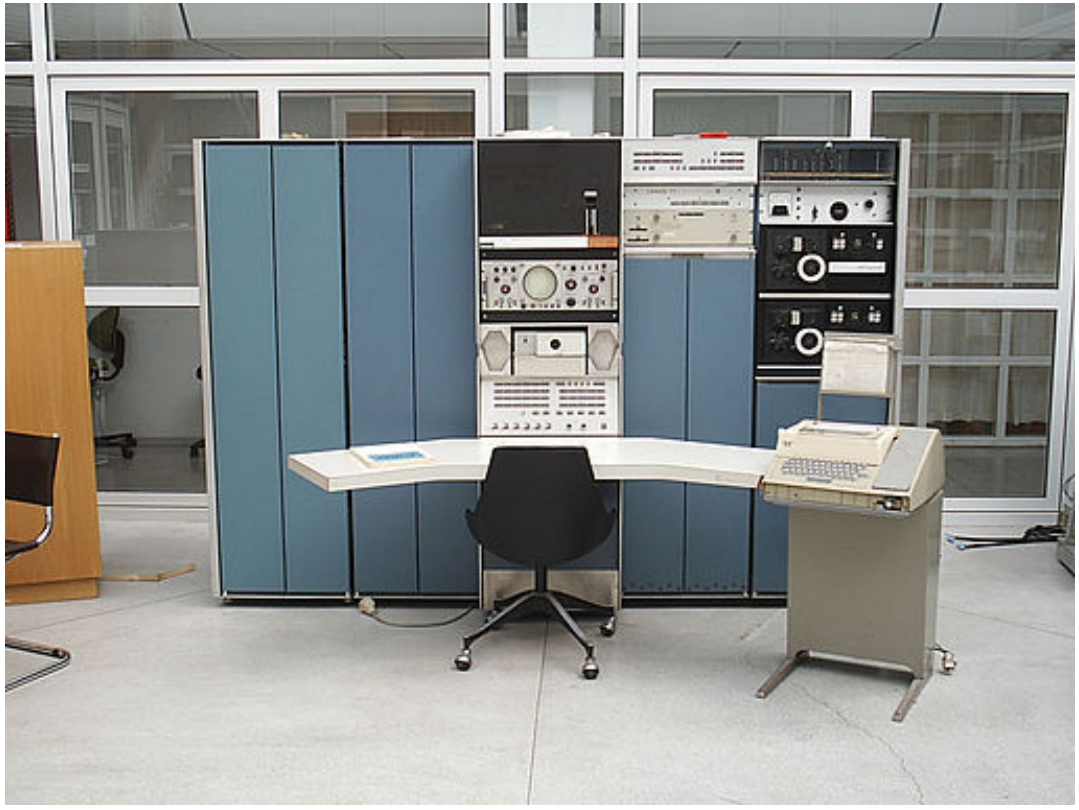
claudiojpaz@gmail.com

Abril 2019

# Unidad 3

## Introducción al lenguaje C

# Reseña histórica



DEC PDP-7. Oslo, Noruega

Toresbe [CC SA 1.0], via Wikimedia Commons

# Reseña histórica

## Fines de los 60's y principios de los 70's

- 1966 - BCPL - Martin Richards.
- 1969 - B - Ken Thompson con Dennis Ritchie.
- 1969-1973 - C - Dennis Ritchie.

# Reseña histórica

## Fines de los 70's hasta fines de los 90's.. y más

- 1978 - Libro: Lenguaje de Programación C
  Kernighan & Ritchie
- 1989 - American National Standards Institute - ANSI C - C89
- 1990 - International Organization for Standardization - ISO C - C90
- 1999 - ANSI adopta el estándar ISO para C - C99
- 2011 - ISO C - C11
- 2018 - ISO C - C18

# Elementos del lenguaje C.

# Elementos del lenguaje C.

**Token:** conjunto de símbolos que tiene un significado coherente en un lenguaje de programación

# Elementos del lenguaje C.

**Token:** conjunto de símbolos que tiene un significado coherente en un lenguaje de programación

Existen seis clases de tokens en el vocabulario del lenguaje C

# Elementos del lenguaje C.

**Token:** conjunto de símbolos que tiene un significado coherente en un lenguaje de programación

Existen seis clases de tokens en el vocabulario del lenguaje C

| | |
|---|---|
| Palabras clave | Identificadores |
| Constantes | Comentarios |
| Operadores | Separadores |

# Elementos del lenguaje C.

# Elementos del lenguaje C.

## Palabras clave

# Elementos del lenguaje C.

## Palabras clave

32 palabras reservadas para el lenguaje

# Elementos del lenguaje C.

## Palabras clave

# Elementos del lenguaje C.

## Palabras clave

| | | | |
|---|---|---|---|
| `auto` | `break` | `case` | `char` |
| `const` | `continue` | `default` | `do` |
| `enum` | `extern` | `float` | `for` |
| `goto` | `if` | `int` | `long` |
| `else` | `return` | `short` | `signed` |
| `sizeof` | `static` | `struct` | `double` |
| `register` | `switch` | `typedef` | `union` |
| `unsigned` | `void` | `volatile` | `while` |

# Elementos del lenguaje C.

# Elementos del lenguaje C.

## Identificadores

# Elementos del lenguaje C.

## Identificadores

Conjunto de caracteres alfanuméricos que asocian a entidades del programa (variables, funciones, etc.).

# Elementos del lenguaje C.

## Identificadores

Conjunto de caracteres alfanuméricos que asocian a entidades del programa (variables, funciones, etc.).

No pueden ser iguales a ninguna palabra reservada.

# Elementos del lenguaje C.

## Identificadores

Conjunto de caracteres alfanuméricos que asocian a entidades del programa (variables, funciones, etc.).

No pueden ser iguales a ninguna palabra reservada.

Pueden estar formados por letras del alfabeto inglés, números o guiones bajos (_).

# Elementos del lenguaje C.

## Identificadores

Conjunto de caracteres alfanuméricos que asocian a entidades del programa (variables, funciones, etc.).

No pueden ser iguales a ninguna palabra reservada.

Pueden estar formados por letras del alfabeto inglés, números o guiones bajos (_).

No pueden comenzar con números.

# Elementos del lenguaje C.

## Identificadores

Conjunto de caracteres alfanuméricos que asocian a entidades del programa (variables, funciones, etc.).

No pueden ser iguales a ninguna palabra reservada.

Pueden estar formados por letras del alfabeto inglés, números o guiones bajos (_).

No pueden comenzar con números.

No pueden tener espacios ni operadores aritméticos.

# Elementos del lenguaje C.

## Identificadores

# Elementos del lenguaje C.

## Identificadores

### Correcto

```
var
n1
_node05
contador
max_temp
i
```

# Elementos del lenguaje C.

## Identificadores

### Correcto

```
var
n1
_node05
contador
max_temp
i
```

### Incorrecto

```
3var
if
5
node-05
max temp
```

# Elementos del lenguaje C.

# Elementos del lenguaje C.

## Constantes

# Elementos del lenguaje C.

## Constantes

Son valores que no pueden cambiar una vez que el programa fue compilado.

# Elementos del lenguaje C.

## Constantes

Son valores que no pueden cambiar una vez que el programa fue compilado.

También se llaman literales.

# Elementos del lenguaje C.

## Constantes

Son valores que no pueden cambiar una vez que el programa fue compilado.

También se llaman literales.

Pueden ser números, caracteres individuales o cadenas de texto.

# Elementos del lenguaje C.

## Constantes

# Elementos del lenguaje C.

## Constantes

### Números

```
3   3.1416   31.416e-1   0.31416e1   .31416e1   .31416e+1
```

# Elementos del lenguaje C.

## Constantes

### Números

```
3   3.1416   31.416e-1   0.31416e1   .31416e1   .31416e+1

3.1416f   3.1416F   31.416E-1   0xfe01   0xFE01   0XFE01
```

# Elementos del lenguaje C.

## Constantes

### Números

```
3  3.1416  31.416e-1  0.31416e1  .31416e1  .31416e+1

3.1416f  3.1416F  31.416E-1  0xfe01  0xFE01  0XFE01
```

### Caracteres

# Elementos del lenguaje C.

## Constantes

### Números

```
3   3.1416   31.416e-1   0.31416e1   .31416e1   .31416e+1

3.1416f   3.1416F   31.416E-1   0xfe01   0xFE01   0XFE01
```

### Caracteres

```
'c'    'F'    '\n'    '7'
```

# Elementos del lenguaje C.

## Constantes

### Números

```
3   3.1416   31.416e-1   0.31416e1   .31416e1   .31416e+1

3.1416f   3.1416F   31.416E-1   0xfe01   0xFE01   0XFE01
```

### Caracteres

```
'c'    'F'    '\n'    '7'
```

### Cadenas de caracteres

# Elementos del lenguaje C.

## Constantes

### Números

```
3   3.1416   31.416e-1   0.31416e1   .31416e1   .31416e+1

3.1416f   3.1416F   31.416E-1   0xfe01   0xFE01   0XFE01
```

### Caracteres

```
'c'    'F'    '\n'    '7'
```

### Cadenas de caracteres

```
"Hola, mundo!"
```

# Elementos del lenguaje C.

# Elementos del lenguaje C.

## Operadores

# Elementos del lenguaje C.

## Operadores

Los operadores en C son conjuntos de caracteres (uno o dos) que indican al programa que debe hacer.

# Elementos del lenguaje C.

## Operadores

Los operadores en C son conjuntos de caracteres (uno o dos) que indican al programa que debe hacer.

Aritméticos: +, -, *, / y %

Asignación: =, +=, -=, *= y /=

Incrementales: ++ y --

Relacionales: <, >, >=, <=, == y !=

Lógicos: && y ||

# Elementos del lenguaje C.

# Elementos del lenguaje C.

## Separadores o delimitadores

# Elementos del lenguaje C.

## Separadores o delimitadores

Se utilizan en distintas construcciones del lenguaje

# Elementos del lenguaje C.

## Separadores o delimitadores

Se utilizan en distintas construcciones del lenguaje

[ ] ( ) {} , ; : ... * = #

# Elementos del lenguaje C.

# Elementos del lenguaje C.

## Comentarios

# Elementos del lenguaje C.

## Comentarios

Cadenas de texto que no se compilan.

# Elementos del lenguaje C.

## Comentarios

Cadenas de texto que no se compilan.

Sirven para que el programador deje aclaraciones.

# Elementos del lenguaje C.

## Comentarios

Cadenas de texto que no se compilan.

Sirven para que el programador deje aclaraciones.

O para anular partes del código.

# Elementos del lenguaje C.

## Comentarios

Cadenas de texto que no se compilan.

Sirven para que el programador deje aclaraciones.

O para anular partes del código.

La doble barra (//) anula todo hasta el final de la línea.

# Elementos del lenguaje C.

## Comentarios

Cadenas de texto que no se compilan.

Sirven para que el programador deje aclaraciones.

O para anular partes del código.

La doble barra (//) anula todo hasta el final de la línea.

/* comenta todo hasta encontrar un */ aunque sea muchas líneas más abajo

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

Todos los programas deben tener una función `main`

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

Las llaves definen *bloques*.

En este caso el bloque es el *cuerpo* de la función `main`

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

A partir del estándar C90 en adelante, la función `main` debe terminar con un `return`.

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

Indica al *proceso* que *lanzó* el programa, que el mismo ya terminó.

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

Las líneas que comienzan con # se llaman *directivas de preprocesador*.

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

Son necesarias para que el compilador trabaje de forma correcta.

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

Otras son:

`#define`
`#ifdef`
`#pragma`

etc.

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

En este caso `printf` y `return 0` son sentencias simples...

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

Sentencias simples deben terminar con punto y coma (;)

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

`printf` es una función definida en `stdio.h`...

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

# Primer ejemplo: Hola, Mundo!

```c
#include <stdio.h>

// programa de prueba

int main (void)
{

  printf("Hola, Mundo!\n");

  return 0;
}
```

...utilizada para imprimir mensajes en pantalla.