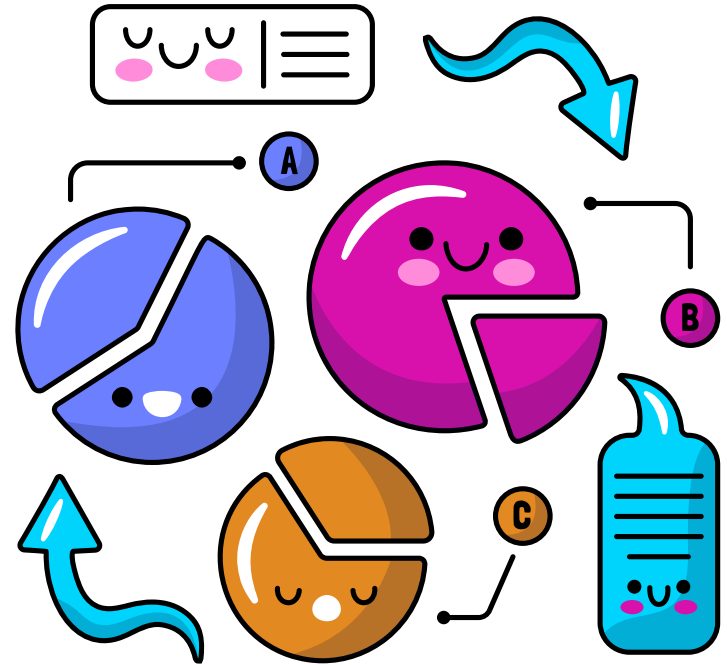


Variables, Constantes y Tipos de Datos

Además de compilación vs interpretación

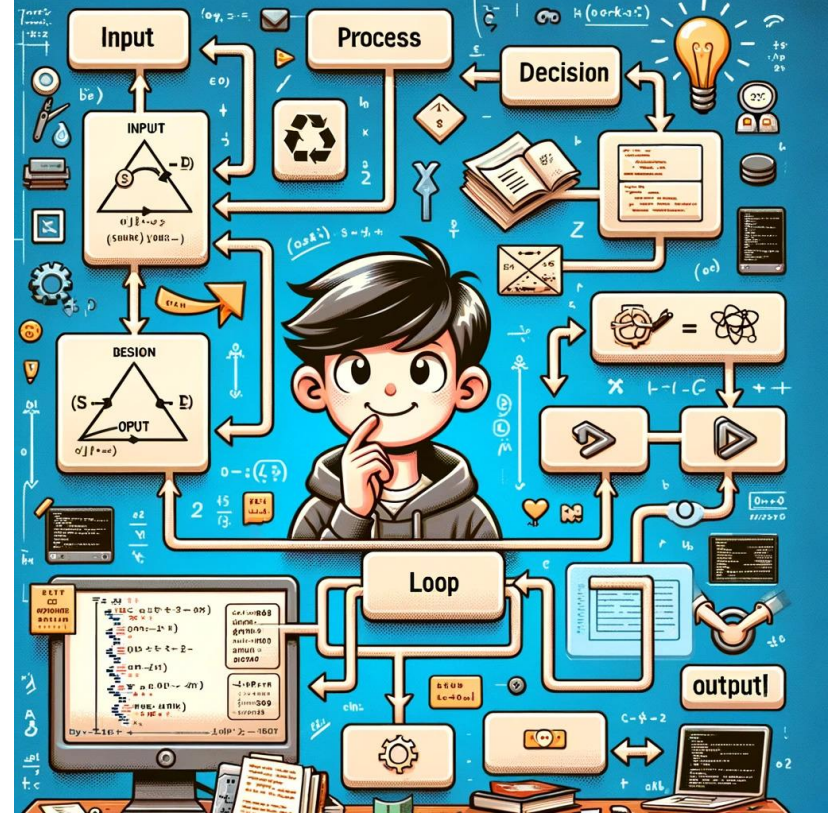


Repaso

- Que es un Algoritmo?
- Que es un Programa?
- Que es una Variable?
- Que es una Estructura de Control?

Algoritmo

Un conjunto de acciones **no-ambiguas, ordenadas y finitas** que permite resolver un **problema**.



Algoritmo

Estas acciones, realizadas bajo **las mismas circunstancias**, con los mismos datos de entrada, dan el mismo resultado.

Ejemplos:

- Cargar un teléfono celular.
- Usar un ascensor.
- Lavarse los dientes.

Sheldon

Ambigüedad

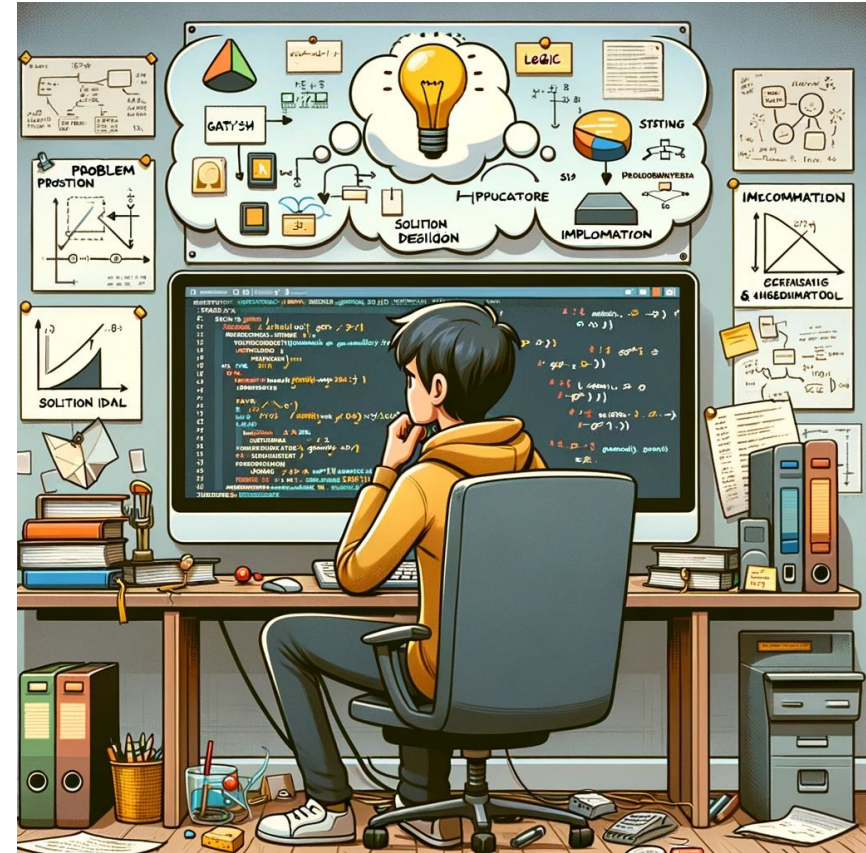
Que es un Problema

Un problema es una situación que presenta una discrepancia entre el estado actual y un estado deseado, requiriendo una solución para superar esta diferencia



Resolución de Problemas Computacionales

Como se resuelven los problemas computacionales?



Resolución de Problemas Computacionales

Especificación de los requerimientos del problema: Se buscan y se describen cuales son los requerimientos del problema. Esta descripción puede ser un dibujo, un párrafo en lenguaje natural, cualquier herramienta que permita comunicar y describir QUE hay que hacer.

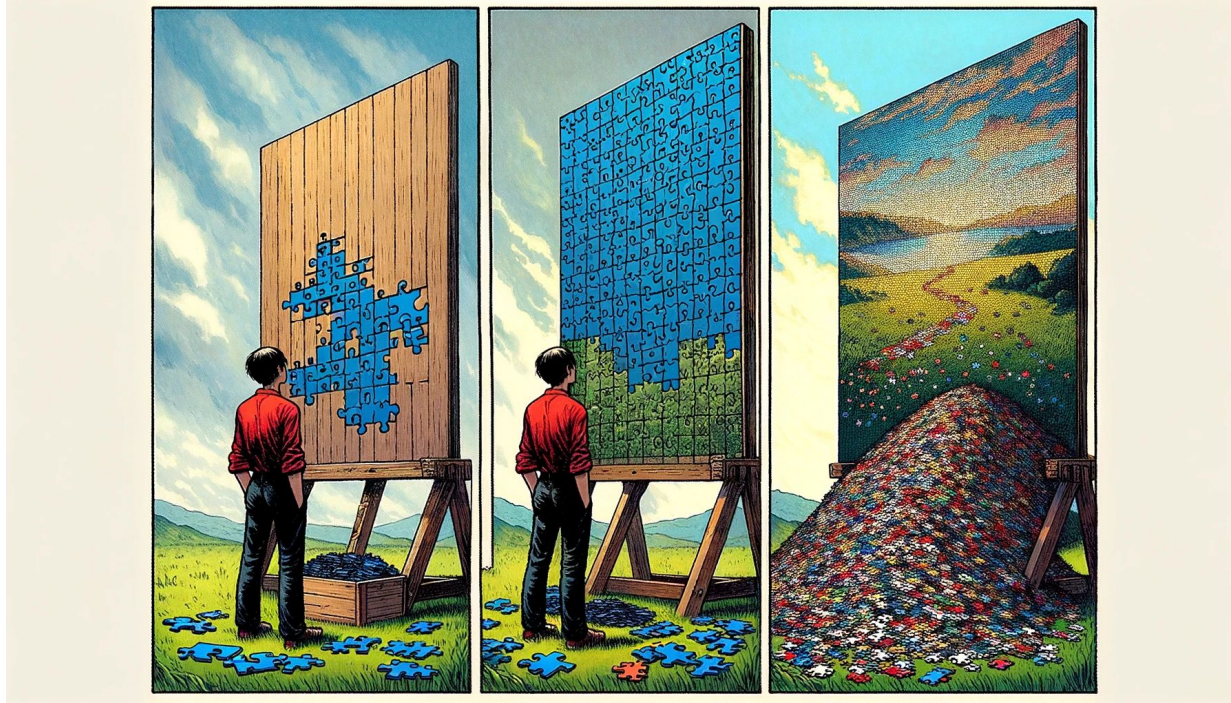
Análisis del problema: El problema se analiza teniendo presente la especificación de los requerimientos, restricciones y datos dados del problema a solucionar.

Resolución de Problemas Computacionales

Diseño del algoritmo para la solución: Una vez analizado el problema, se diseña una o varias soluciones y se elegirá aquella que por su naturaleza se considere la mejor que conducirá a un algoritmo que lo resuelva.

Una técnica de diseño en la resolución de problemas consiste en la identificación de las partes (subproblemas) que lo componen y la manera en que se relacionan. Cada uno de estos subproblemas debe tener un objetivo específico, es decir, debe resolver una parte del problema original.

Resolución de Problemas Computacionales



Resolución de Problemas Computacionales

Además se debe **Especificar el algoritmo**, para solucionar cada subproblema se plantea un algoritmo, este debe ser especificado de alguna forma. Esta etapa busca obtener la secuencia de pasos a seguir para resolver el problema. La elección del algoritmo adecuado es fundamental para garantizar la eficiencia de la solución.

Resolución de Problemas Computacionales

Implementación de un programa: La solución que plantea el algoritmo se transcribe a un lenguaje de programación.

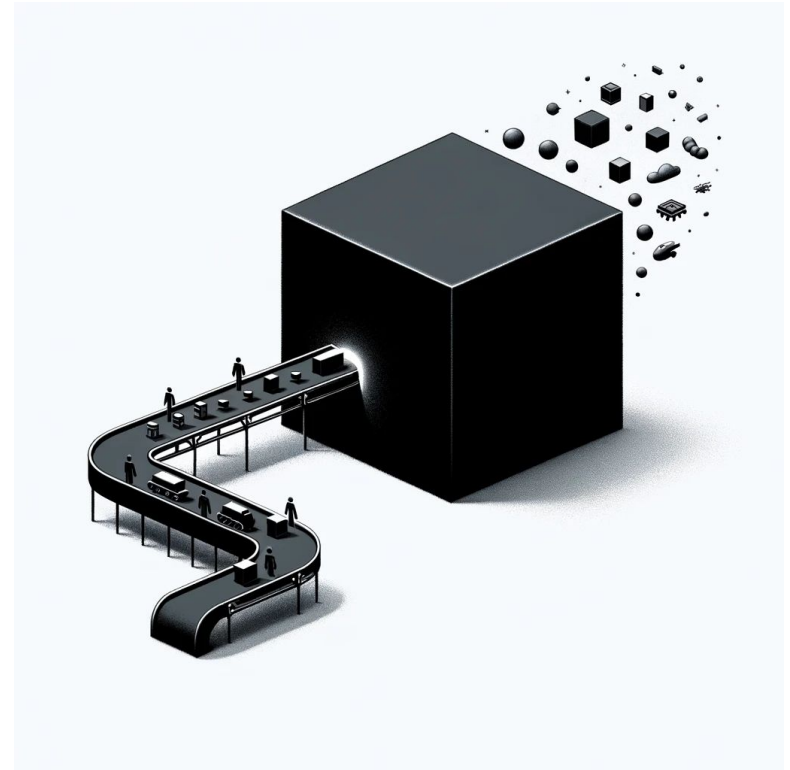
Ejecución, verificación y depuración: El programa se ejecuta, se comprueba rigurosamente y se eliminan todos los errores (denominados “bugs”, en inglés) que puedan aparecer.

Mantenimiento: El programa se actualiza y modifica, cada vez que sea necesario, de modo que se cumplan todas las necesidades de cambio de sus usuarios.

Resolución de problemas Puntos de Vista

Caja Negra:

Un objeto es estudiado desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno



Resolución de problemas Puntos de Vista

Caja Blanca:

Un objeto que es estudiado desde el punto de vista de las interacciones entre los componentes internos que lo integran.



Programa

Un Programa es un
Algoritmo traducido a un
lenguaje de Programación
determinado

Que es programar

“Cualquier tonto puede escribir programas que una computadora puede entender. Los buenos programadores escriben programas que los humanos pueden entender”

Que es programar

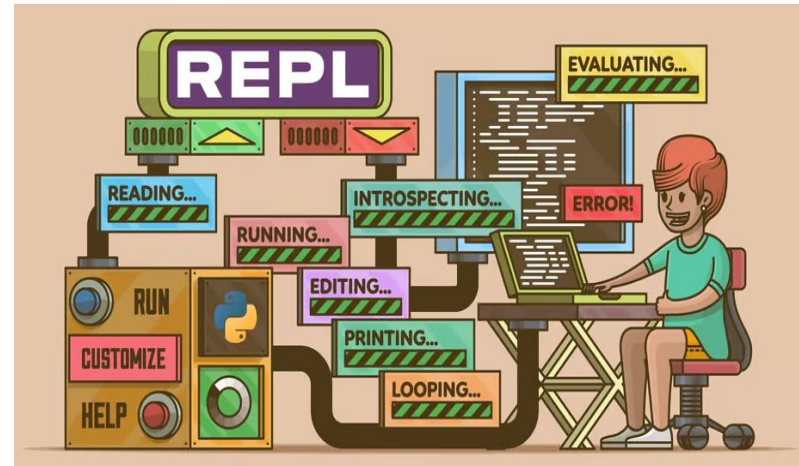
Un Programa es un Algoritmo traducido a un lenguaje de Programación determinado

Tipos de Lenguajes de Programación: Interpretados

Un lenguaje interpretado es aquel que necesita de un programa llamado Intérprete para que sus instrucciones puedan ser ejecutadas. Esta ejecución es una instrucción a la vez en la que el intérprete:

1. Lee la instrucción
2. Ejecuta la instrucción
3. Imprime el resultado
4. Y repite el ciclo

Read Execute Print Loop



Compilados:

El Proceso de Interpretación

Existen lenguajes como Python utilizan a un **intérprete** para poder ejecutar las instrucciones.

Normalmente no se denominan programas sino **Scripts**

El intérprete funciona mediante un ciclo llamado **REPL**

Un programa en Python es una serie de acciones que el intérprete va ejecutando una a una

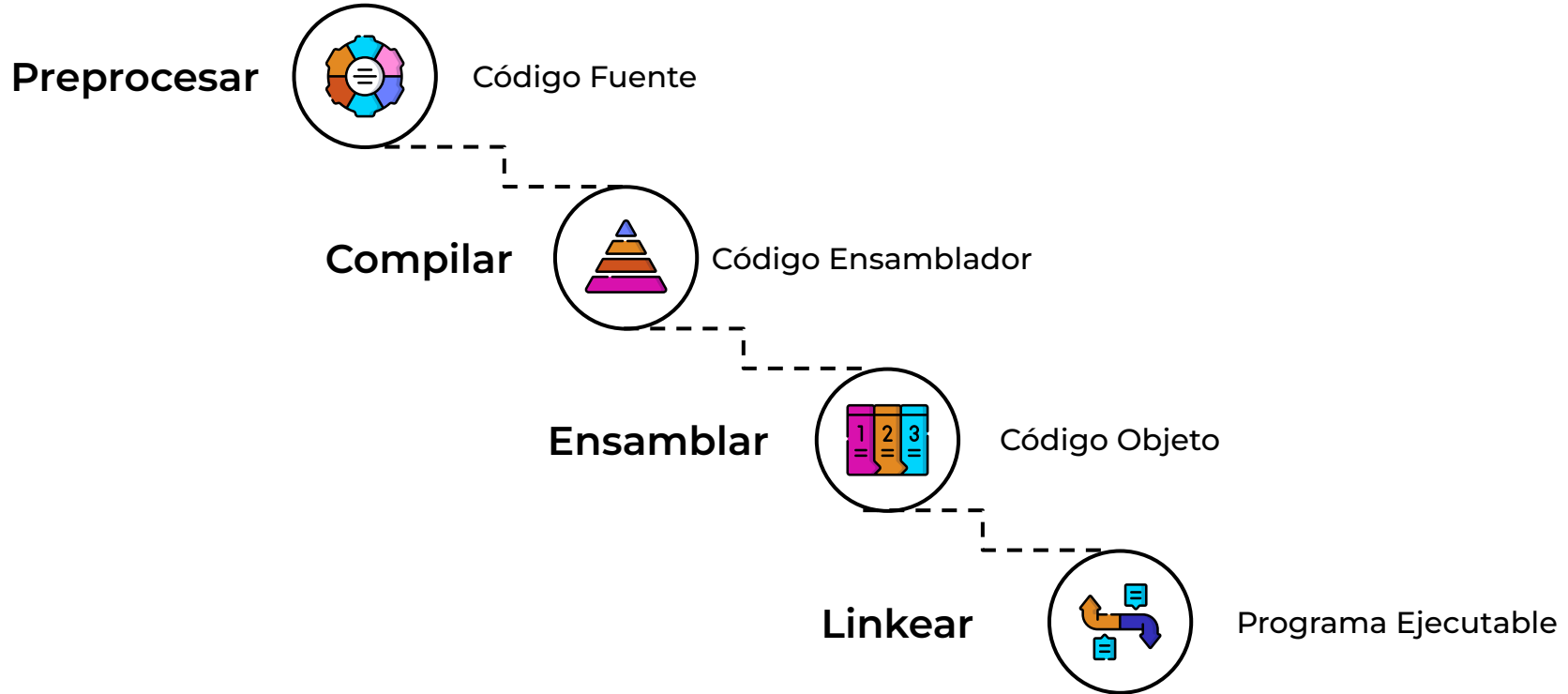
Tipos de Lenguajes de Programación: Compilación

Un lenguaje compilado es aquel que para poder ser ejecutado debe pasar por un proceso llamado compilación en el cual se pasa el programa escrito en texto simple a un formato ejecutable por la computadora:

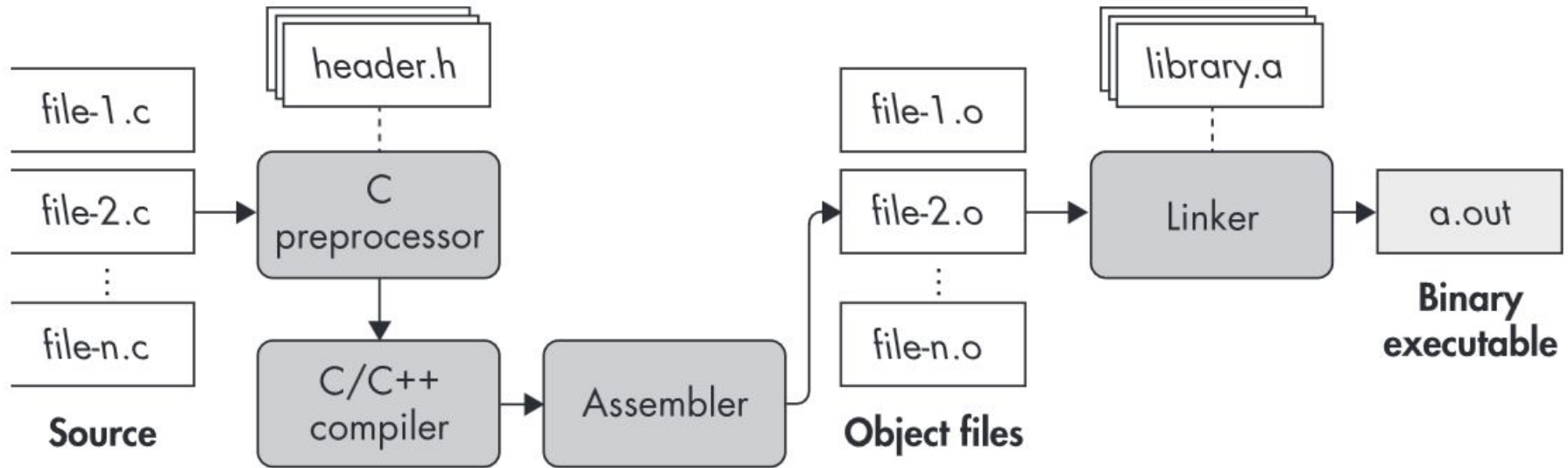
- 1. Preprocesamiento**
- 2. Compilación**
- 3. Ensamblaje**
- 4. Link Edicion**

En cada una de estas etapas se alimenta de la anterior

El Proceso de Compilación



Poniendo Todo en su Lugar



El Lenguaje de Programación C

El lenguaje de programación C, desarrollado en la década de 1970 por Dennis Ritchie en los laboratorios Bell.

Es un lenguaje de programación de propósito general que ha sido fundamental para el desarrollo de sistemas operativos, aplicaciones de software y sistemas embebidos.

C es conocido por su eficiencia, permitiendo a los programadores tener un control preciso sobre los recursos del sistema y la memoria.

El Lenguaje de Programación C

A pesar de su sintaxis concisa y su capacidad para operar a bajo nivel, C es relativamente simple de aprender, lo que lo convierte en una opción popular para la programación de sistemas y la educación en ciencias de la computación.

Su influencia se extiende a muchos otros lenguajes modernos, como C++, C#, y Java, marcando una huella indeleble en el campo de la programación.

El Lenguaje de Programación C



El Lenguaje de Programación C

El Lenguaje C:

El Lenguaje + Compilador + La Biblioteca Estándar

Muy relacionado con el sistema operativo UNIX

Según el índice TIOBE el segundo lenguaje de programación más utilizado.

El lenguaje en sí es muy pequeño.

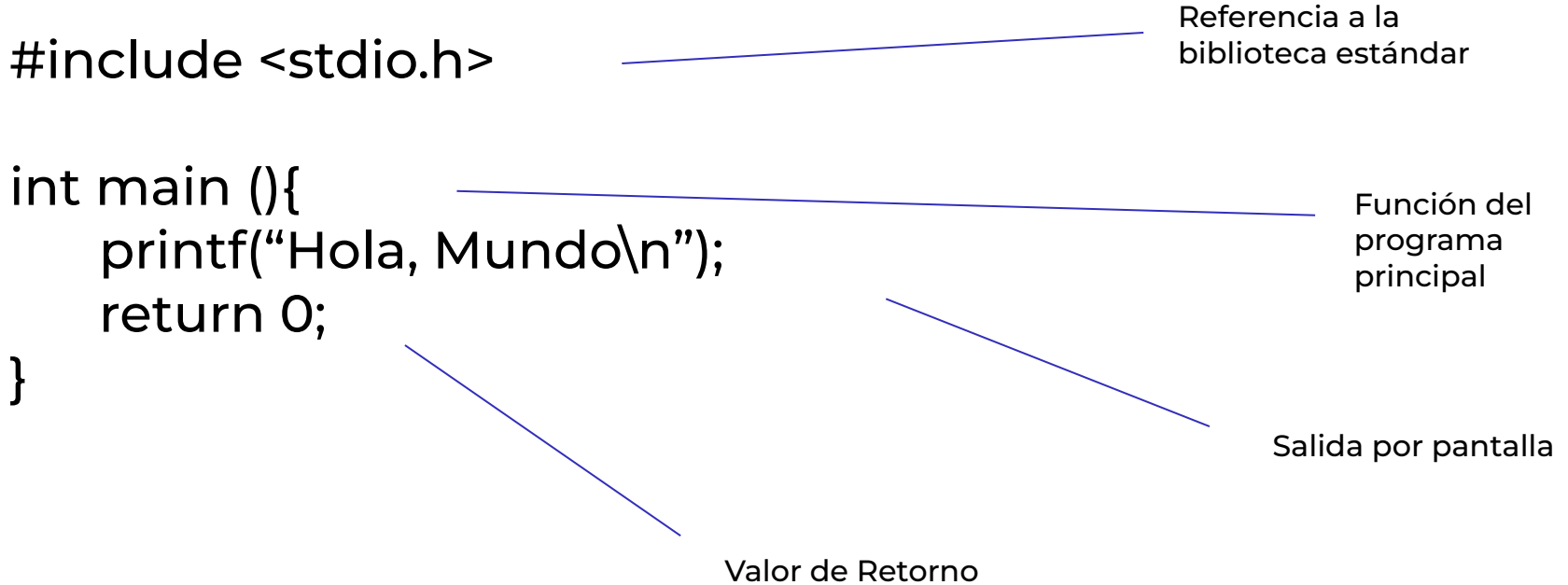
Hola Mundo! en Python

```
print("hola, mundo")
```

Hola Mundo! en C

```
#include <stdio.h>
```

Referencia a la
biblioteca estándar



```
int main (){  
    printf("Hola, Mundo\n");  
    return 0;  
}
```

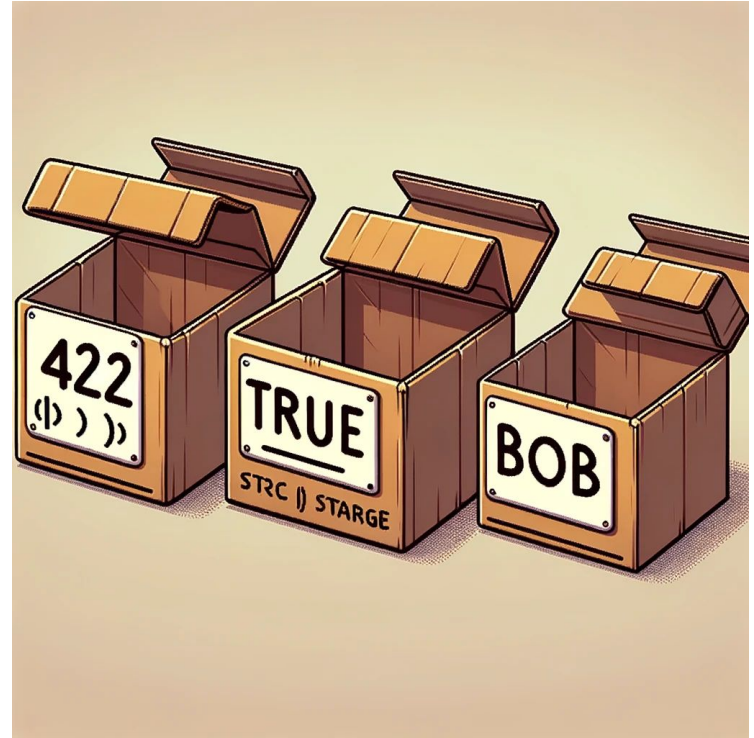
Función del
programa
principal

Salida por pantalla

Valor de Retorno

Variables

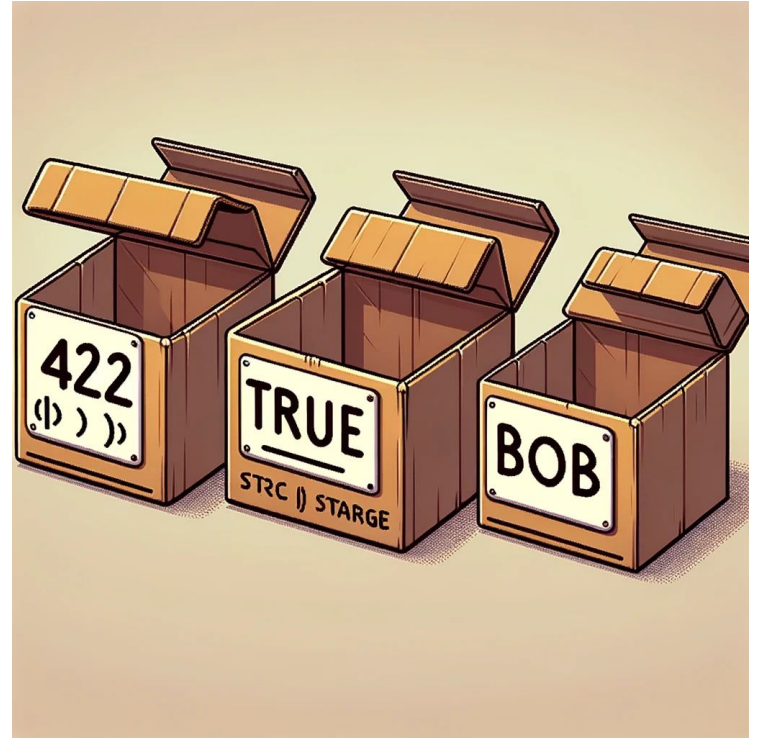
Desde el punto de vista de la algoritmia **una variable es una entidad que puede almacenar un valor de un determinado tipo de dato, y dicho valor puede ser accedido y/o modificado a lo largo de un algoritmo.**



Variables

Esta entidad se caracteriza por tener un ***identificador único***.

Nota: Vale destacar la importancia de tener claro el concepto de que una variable **NO ES UN VALOR en sí mismo sino UN CONTENEDOR DE VALORES.**



Variables

Existen DOS cosas difíciles en Ciencias de la Computación:

- 1- Poner Nombres**
- 2- Concurrencia**
- 3- Errar por uno**



Variables

La elección de nombres no es trivial. A continuación se enumeran algunas reglas empíricas para la elección de nombres:

- Nombres Reveladores
- Evitar la Desinformación
- Nombres Distinguibles
- Nombres Fáciles de Buscar
- No Te Hagas el Canchero
- Una Sola Palabra por Concepto
- Evitar los Nombres Genéricos

Tipos de Datos

Un tipo de dato puede definirse como:

“Todos los valores posibles que una variable de ese tipo de dato puede tomar”

Un tipo de dato es un conjunto de valores acotado por un valor máximo y un valor mínimo. Existen dos tipos de datos primitivos.

Tipos de Datos

“Todos los valores posibles que una variable de ese tipo de dato puede tomar”



Tipos de Datos

Clasificación los tipos de datos se clasifican en ***simples*** y ***compuestos o estructurados***.

Los llamados tipos de datos Simple se dividen en:

- Tipos de datos Ordinales: aquellos cuyos valores poseen sucesor y predecesor.
- Tipos de datos No Ordinales: aquellos en los que no pueden determinarse un sucesor y un predecesor.

Tipos de Datos

Clasificación los tipos de datos se clasifican en ***simples*** y ***compuestos o estructurados***.

Una de las acciones más importantes cuando se especifica un algoritmo es la definición de las variables.

Tipos de Datos en C

char

int

long

float

double

unsigned

signed

Type	Bits	Minimal Range
char	8	−127 to 127
unsigned char	8	0 to 255
signed char	8	−127 to 127
int	16 or 32	−32,767 to 32,767
unsigned int	16 or 32	0 to 65,535
signed int	16 or 32	Same as int
short int	16	−32,767 to 32,767
unsigned short int	16	0 to 65,535
signed short int	16	Same as short int
long int	32	−2,147,483,647 to 2,147,483,647
long long int	64	−(2 ⁶³ − 1) to 2 ⁶³ − 1 (Added by C99)
signed long int	32	Same as long int
unsigned long int	32	0 to 4,294,967,295
unsigned long long int	64	2 ⁶⁴ − 1 (Added by C99)
float	32	1E−37 to 1E+37 with six digits of precision
double	64	1E−37 to 1E+37 with ten digits of precision
long double	80	1E−37 to 1E+37 with ten digits of precision

Asignación de valores a variables

Al igual que python las variables en C pueden ser asignadas con valores. Y el operador encargado de tal cosa es =

int x; <-declaración

x=1; <-asignación

también se puede:

int x=11,

Estructuras de Control

Estructuras de Control

Las estructuras de control en programación son elementos fundamentales que permiten controlar el flujo de ejecución de un programa.

Estas estructuras dictan cómo se deben ejecutar las instrucciones, y permiten a los programadores manejar condiciones y repetir acciones según sea necesario.

Existen principalmente tres tipos de estructuras de control:

- ***Secuenciales***
- ***Condicionales***
- ***Iterativas***

Estructuras de Control: Bloque de Código

En C, un bloque de instrucciones es *una secuencia de instrucciones agrupadas que se ejecutan juntas como una unidad*.

Los bloques de instrucciones son fundamentales para definir el alcance (scope) y la estructura del código, y se utilizan para controlar la ejecución de sentencias en estructuras de control, como if, for, while, y switch, entre otras.

Estructuras de Control: Bloque de Código

```
// Bloque de instrucciones en main
{
    int a = 5; // Declaración e inicialización
    int b = 10;
    printf("La suma de a y b es %d\n", a + b); // Imprime el resultado
}

{
    // Otro bloque de instrucciones
    int c = 20;
    printf("El valor de c es %d\n", c);
}
```

Estructuras de Control: Bloque de Código

Características de un Bloque de Instrucciones

Alcance (Scope):

Las variables declaradas dentro de un bloque solo son accesibles dentro de ese bloque. Este es el concepto de alcance local. Por ejemplo, las variables `a` y `b` en el primer bloque no son accesibles fuera de ese bloque.

```
// Bloque de instrucciones en main
{
    int a = 5; // Declaración e inicialización
    int b = 10;
    printf("La suma de a y b es %d\n", a + b); // Imprime el resultado
}
```

Control de Flujo:

Los bloques se utilizan para agrupar instrucciones dentro de estructuras de control. Por ejemplo, en un `if` o `for`, se puede agrupar varias sentencias para que se ejecuten juntas bajo ciertas condiciones.

Estructuras de Control: Bloque de Código

Anidamiento:

Los bloques pueden estar anidados dentro de otros bloques. Esto permite crear estructuras complejas y manejar el alcance de las variables de manera más controlada.

```
int x = 1;
if (x == 1) {
    int y = 2;
    if (y == 2) {
        printf("x es 1 y y es 2\n");
    }
}
// printf("%d\n", y); // Esto generaría un error porque y no es accesible
```

Estructuras de Control: Secuencial

Descripción: Ejecutan las instrucciones en el orden en que aparecen.

```
print("Hola")  
print("Mundo")
```

Estructuras de Control: Condicionales

Descripción: Permiten tomar decisiones y ejecutar bloques de código diferentes dependiendo de si se cumplen ciertas condiciones.

If simple:

```
x = 10
if x > 5:
    print("x es mayor que 5")
```

```
int x = 10;
if (x > 5) {
    printf("x es mayor que 5\n");
}
```

Estructuras de Control: Condicionales

If else :

```
x = 10
if x > 5:
    print("x es mayor que 5")
else:
    print("x es 5 o menor")
```

```
int x = 10;
if (x > 5) {
    printf("x es mayor que 5\n");
} else {
    printf("x es 5 o menor\n");
}
```

Estructuras de Control: Condicionales

if-elif-else (Python) / if-else if-else (C):

```
x = 10
if x > 10:
    print("x es mayor que 10")
elif x == 10:
    print("x es igual a 10")
else:
    print("x es menor que 10")
```

```
int x = 10;
if (x > 10) {
    printf("x es mayor que 10\n");
} else if (x == 10) {
    printf("x es igual a 10\n");
} else {
    printf("x es menor que 10\n");
}
```

Estructuras de Control: Condicionales

switch - case (solo C)

```
int x = 2;
switch (x) {
    case 1:
        printf("x es 1\n");
        break;
    case 2:
        printf("x es 2\n");
        break;
    default:
        printf("x es diferente de 1 y 2\n");
        break;
}
```


Estructuras de Control: Iterativas

Descripción: Las estructuras de control iterativas, comúnmente conocidas como "bucles" o "loops", permiten repetir un bloque de código varias veces mientras se cumpla una condición.

while:

```
i = 0

while i < 5:
    print(f"El valor de i es: {i}")
    i += 1
```

```
int i = 0;

while (i < 5) {
    printf("El valor de i es: %d\n", i);
    i++;
}
```

Estructuras de Control: Iterativas

for:

```
for i in range(5):  
    print(f"El valor de i es: {i}")
```

```
int i;  
  
for (i = 0; i < 5; i++) {  
    printf("El valor de i es: %d\n", i);  
}
```

Estructuras de Control: Iterativas

do while (solo en C): el bucle do-while garantiza que el bloque de código se ejecute al menos una vez, ya que la condición se evalúa después de la primera ejecución del cuerpo del bucle.

```
int i = 0;

do {
    printf("El valor de i es: %d\n", i);
    i++;
} while (i < 5);
```