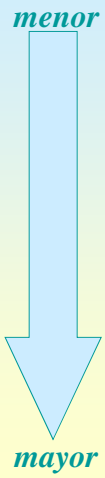
Conceptos generales acerca de los Lenguajes de programación

Tipos de ABSTRACCIÓN
(en los lenguajes de programación)

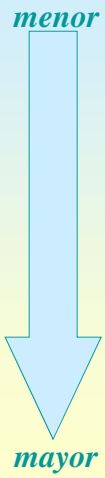
- **DE DATOS**
  - Se refiere a las facilidades que provee el lenguaje para describir y manipular los datos.
- **DE CONTROL**
  - Se refiere a las herramientas que provee el lenguaje para dirigir el flujo de instrucciones.
- **MODULAR**
  - Se refiere a la manera en que el lenguaje permite estructurar el código un programa.

Abstracción de datos



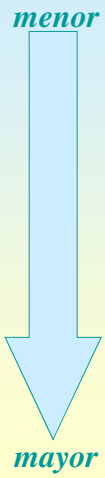
- Memoria: bits y bytes...
- Datos atómicos/escalares: enteros, reales, caracter, booleanos...
- Datos estructurados: arreglos, registros, conjuntos, archivos...
- Datos definidos por el usuario.
- Tipos de datos abstractos...
- Objetos...

Abstracción de control



- Secuencia y saltos condicionales e incondicionales a direcciones de memoria: *jump*, *call*...
- Saltos a líneas de programa: *Goto*, *Gosub*...
- Estructuras de decisión: *if*, *case*...
- Estructuras de repetición: *while*, *do*, *repeat*, *for*…
- Recursividad...

Abstracción modular



- Secuencias de instrucciones con saltos a direcciones de memoria con regreso: *call*, *ret*...
- Secuencias de instrucciones con saltos a subrutina: *Gosub, endsub*...
- Macros…
- Funciones y Procedimientos...
- Unidades…
- Objetos...

Características de un módulo

- FUNCIONALIDAD
  - Tener sólo un propósito.
- INDEPENDENCIA
  - No depender de los que realicen los otros módulos del sistema.
- EXCLUSIVIDAD
  - No realizar las funciones que otros módulos realizan en el sistema.

Beneficios de la modularidad

- Ocultamiento de información.
- Independencia de datos.
- Compilación separada.
- Ejecución concurrente.

Tipos de parámetros

- Por valor
- Por referencia o variables
- Resultantes
  - Sólo de salida.
- Por valor-resultantes
  - Copia de la entrada.
- Por nombre
  - Asociación de símbolos

Ejemplo

```
Módulo Ejemplo (entero x, y)
Begin
    x = x + 1;
    y = y + 1;
End.

a = 1;
Ejemplo(a, a);
```

¿Cuál es el valor de a si x y y son parámetros por valor?

Ejemplo

```
Módulo Ejemplo (entero x, y)
Begin
    x = x + 1;
    y = y + 1;
End.

a = 1;
Ejemplo(a, a);
```

¿Cuál es el valor de a si x y y son parámetros por referencia?

Ejemplo

```
Módulo Ejemplo (entero x, y)
Begin
    x = x + 1;
    y = y + 1;
End.

a = 1;
Ejemplo(a, a);
```

¿Cuál es el valor de a si x y y son parámetros por valor-resultantes?

Traducción de un lenguaje

TRADUCTOR:

- Programa que recibe como entrada el código fuente de un programa (o texto) en un lenguaje (de programación) X y lo transforma en un programa (o texto) equivalente o código objeto en un lenguaje (de programación) Y.

Traducción de un lenguaje

FUNCIONES DE UN TRADUCTOR:

1. Verificar el uso correcto del lenguaje fuente.
   ETAPA DE ANÁLISIS

2. Si no hay errores, convertir la entrada al lenguaje objeto.
   ETAPA DE SÍNTESIS

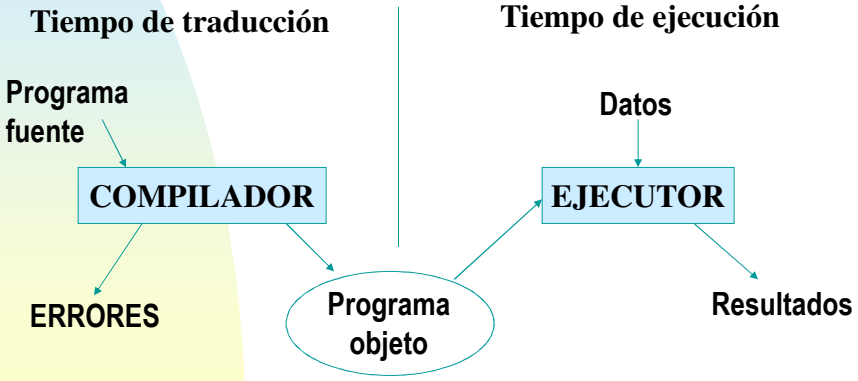
Traducción de un lenguaje

TIPOS DE TRADUCTORES:

- Ensamblador
- Compilador
- Interpretador
- Preprocesador

Compilación

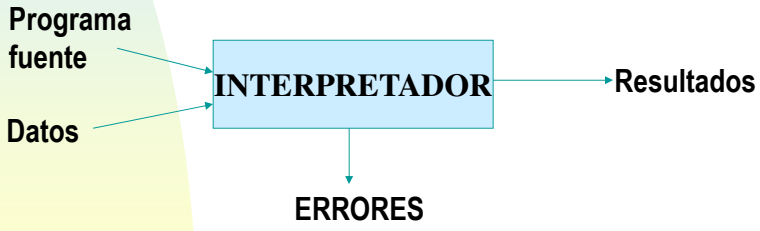
Compilador: Programa que traduce código fuente a código objeto.



** Un ENSAMBLADOR es el "compilador" del lenguaje ensamblador*

Interpretación

Intérprete: Programa que traduce alguna forma de código fuente a una representación intermedia que puede ser evaluada inmediatamente.



Interprete vs Compilador

- Ventajas y desventajas de cada tipo de traducción, en cuanto:
  - Tiempo de traducción.
  - Tiempo de ejecución.
  - Uso de memoria.
  - Ambiente de programación para el programador.

Compilación vs. Interpretación

- Mejor tiempo de traducción.
  - *Compilador*
- Mejor tiempo de ejecución del programa.
  - *Compilador*
- Mayor eficiencia para detectar los errores.
  - *Interpretador*

Compilación vs. Interpretación

- Mayor portabilidad en el código.
  - *Interpretador*
- Mejor uso de los espacios de memoria para datos.
  - *Interpretador*
- Mayor independencia del ambiente de programación.
  - *Compilador*

Preprocesamiento

- Traducción de un código que contiene macroinstrucciones, al lenguaje propio.
- Expansión de macros.
- Posteriormente se compila o interpreta.
- Ejemplo: *#define* , *#include* , etc.

Ejemplos: Preprocesador de C

- **Inclusión de Archivos**

```
#include <stdio.h>
int main (void) {
    printf("Hello, world!\n");
    return 0;
}
```

- **Uso de macros**

```
#define PI 3.14159
#define RADTODEG(x) ((x) *
    57.2)
```

- Compilación condicional

```
#ifdef _WIN32
#include <windows.h>
#else
#include <unistd.h>
#endif
```

- Errores y warnings definidos por el usuario

```
#error “Sistema operativo erroneo”
#warning “Usar XYZ en lugar de ABC”
```

Ligado de códigos

- El proceso de compilación, puede arrojar un código objeto, que es posteriormente tomado por un “Linker” que lo une con otros códigos objetos, para finalmente generar un código ejecutable (normalmente en lenguaje maquinal).

Asociaciones (Bindings) en un programa

- Todo elemento de un programa, tiene que tener asociado:
  - Un nombre
  - Un valor
  - Un tipo
  - Una dirección de memoria
- Las asociaciones tienen un alcance dentro del programa (*Scope*).

Tipos de Asociaciones (Bindings)

- ESTÁTICAS: Aquellas que se hacen durante el tiempo de traducción de un programa, o durante la carga (load) de un código ejecutable a memoria.
- DINÁMICAS: Aquellas que se hacen durante la ejecución de un programa.