UNIVERSIDAD NACIONAL EXPERIMENTAL DE GUAYANA
VICE RECTORADO ACÁDEMICO
COORDINACION DE INGENIERIA EN INFORMATICA

Técnicas de Programación I

TEMA 1

Del seudocódigo al lenguaje de programación C

Profesor: Jose Luis Salazar

AGENDA

1. Lenguaje de programación.
2. Sintaxis y semántica del lenguaje de programación (Lenguaje C).
3. Traducción de seudocódigo al lenguaje de programación (equivalencias).

1. Lenguaje de programación.

Lenguaje de Programación

Conjunto de instrucciones, mediante el cual se interactúan con las computadoras. Este, nos permite comunicarnos con las computadoras a través de **algoritmos e instrucciones** escritas en una sintaxis que la computadora entiende e interpreta en lenguaje de máquina.

Jose Luis Salazar → *jlsalazarg@gmail.com*

Tipos de Lenguaje de Programación

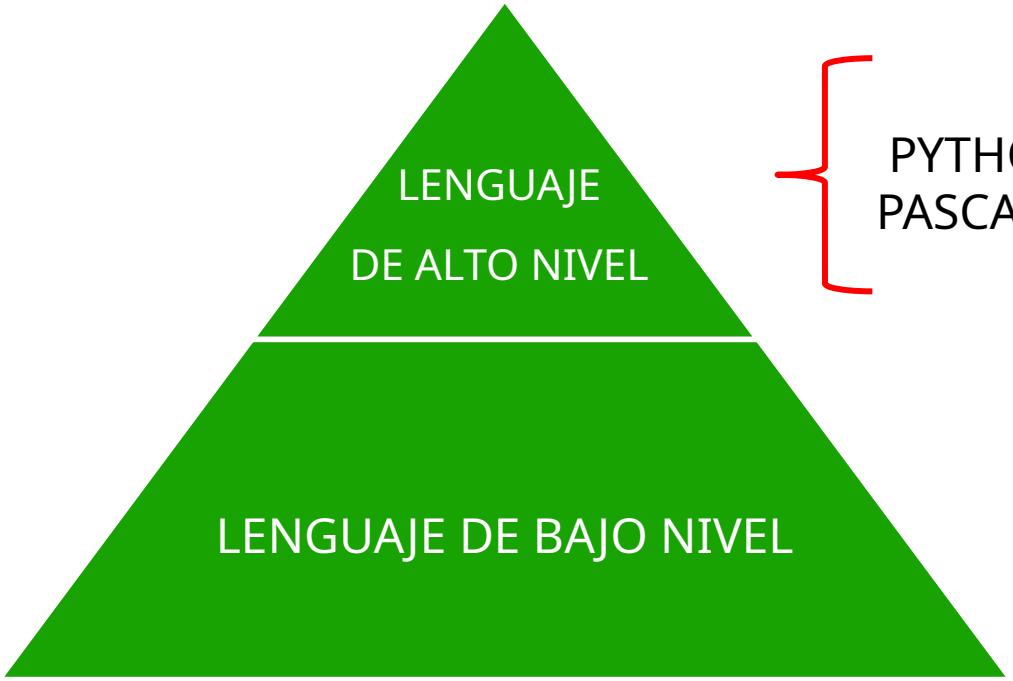
1. **Lenguaje de alto nivel:**
   - **Compilados e Interpretados.**
   - **Cercano al lenguaje humano.**
   - **Mas claros, faciles de aprender y utilizar.**
   - **Portables.**

1. **Lenguaje de bajo nivel:**
   - **Cercano al lenguaje de maquina.**
   - **Requieren que el programador tenga mayor conocimiento tecnico sobre el funcionamiento interno de la maquina.**

Jose Luis Salazar → *jlsalazarg@gmail.com*

Lenguaje de Programación

Tipos de Lenguaje de Programación



PYTHON, JAVA, C, C++, C#,
PASCAL, OTROS.

ENSAMBLADOR,
LENGUAJE DE MAQUINA.

HARDWARE.

Jose Luis Salazar → *jlsalazarg@gmail.com*

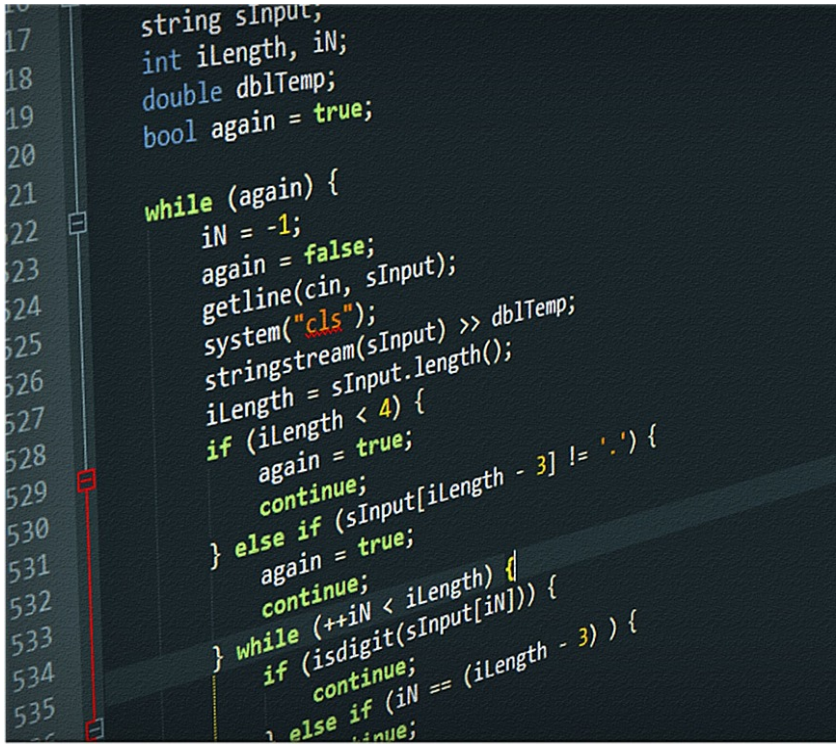
Proceso de compilación

Es el proceso mediante el cual un programa escrito en un lenguaje de alto nivel, como C++, se traduce a un lenguaje de bajo nivel, como lenguaje máquina, que la computadora puede entender y ejecutar. El compilador es el software encargado de realizar esta traducción, y durante el proceso de compilación se realizan diversas etapas como análisis léxico, análisis sintáctico, generación de código intermedio, optimización de código y generación de código objeto. Una vez completado el proceso de compilación, se obtiene un archivo ejecutable que puede ser ejecutado en la computadora.

Lenguaje de Programación

Código fuente

Es el programa escrito por el programador en un lenguaje de programación de alto nivel, como C. Este código es legible para los humanos y contiene las intrucciones y algoritmos necesarios para que el programa realice sus funciones. El código fuente debe ser traducido a lenguaje de maquina mediante un compilador para que la computadora pueda ejecutarlo.



Jose Luis Salazar → *jlsalazarg@gmail.com*

Código objeto

Es el resultado de la compilación del código fuente, pero aún no es un programa ejecutable. El código objeto contiene instrucciones en lenguaje de máquina que representan las operaciones del programa, pero todavía necesita ser enlazado con otras partes del programa para formar un ejecutable completo.

```
00000000: 7f45 4c46 0101 0100 0000 0000 0000 0000  .ELF............
00000010: 0200 0300 0100 0000 8000 0000 3400 0000  ............4...
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
...
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

Código ejecutable

Es el resultado final del proceso de compilación, en el que código objeto se combina con otras partes del programa (bibliotecas, módulos, etc.), para formar un archivo ejecutable. Este archivo puede ser ejecutado directamente por el sistema operativo.

Jose Luis Salazar → *jlsalazarg@gmail.com*

Familia C

Las similitudes entre C y C++ son:

1. **Tienen una sintaxis similar.**
2. **La estructura del código de ambos idiomas es la misma.**
3. **La compilación de ambos idiomas es similar.**
4. **Comparten la misma sintaxis básica. Casi todos los operadores y palabras clave de C también están presentes en C++ y hacen lo mismo.**

Jose Luis Salazar → *jlsalazarg@gmail.com*

Familia C

Las similitudes entre C y C++ son:

1. **C++ tiene una gramática ligeramente más amplia que C, pero la gramática básica es la misma.**
2. **El modelo básico de memoria de ambos está muy cerca del hardware.**
3. **Las mismas nociones de pila, montón, ámbito de archivo y variables estáticas están presentes en ambos lenguajes.**

Jose Luis Salazar → *jlsalazarg@gmail.com*

Familia C

Las diferencias entre C y C++ son:

C++ se puede decir que es un superconjunto de C. Las principales características añadidas en C++ son la programación orientada a objetos, el manejo de excepciones y la rica biblioteca de C++.

Jose Luis Salazar → *jlsalazarg@gmail.com*

Familia C

C#

- C# es programación orientada a objetos.
- C#, también llamado C Sharp, es un lenguaje de programación de alto nivel derivado del lenguaje de programación C de bajo nivel y desarrollado por Anders Hejlsberg, líder de un equipo de Microsoft, en 2002.
- C# no es un lenguaje compilado como C++. Este se pasa a un lenguaje intermedio, como Java. C# trabaja sobre la plataforma.NET, igual que VisualBasic.NET o F#.

Jose Luis Salazar → *jlsalazarg@gmail.com*

2. Sintaxis y semántica del lenguaje de programación (Lenguaje C).

Sintaxis y Semántica (Lenguaje C)

Sintaxis

- Se refiere a las reglas y regulaciones para escribir cualquier declaración en un lenguaje de programación como C / C++ .
- No tiene nada que ver con el significado de la declaración.
- Una declaración es sintácticamente válida si sigue todas las reglas.
- Está relacionado con la gramática y la estructura del lenguaje.

Semantica

- Se refiere al significado asociado con la declaración en un lenguaje de programación.
- Se trata del significado de la declaración que el programa interpreta fácilmente.
- Los errores se gestionan en tiempo de ejecución .

Jose Luis Salazar → *jlsalazarg@gmail.com*

Estructura de un Programa

Un programa incluye secciones determinadas y un orden para las mismas.

En la siguiente forma:

Archivos de cabecera.

Declaración de constantes y macros.

Declaración de Variables Globales.

Declaración de prototipos de Funciones.

Desarrollo de la función principal.

Jose Luis Salazar → *jlsalazarg@gmail.com*

Estructura General de un Programa

PLANTILLA

```
Comentarios
Directivas del preprocesador
Declaración de variables globales y funciones
int main( )   // Función principal main
{
Declaraciones locales de la función principal
Instrucciones de la función principal
}
Otras funciones:
funcion1(.....)
{
Declaraciones locales de la función 1
Instrucciones de la función 1
}
funcion2(.....)
{
Declaraciones locales de la función 2
Instrucciones de la función 2
}
.....
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

Estructura General de un Programa

```
/* Programa que muestra el mensaje          →   Comentarios “Encabezado del programa”
   Hola mundo!!! y mi nombre por
   Pantalla */
#include <stdio.h>                          →   Declaración de librerías en este caso para
                                                utilizar printf o scanf

#define MAX 20                              →   Para def. el valor de una constante

int edad = 45;                              →   Declaracion de Variables Globales y
                                                prototipo de funciones

int main() {                                →   Función Principal
    char nombre[] = “Jose Luis”;            →   Declaracion de variables loales
    printf("Hola mundo!!!\n");
    printf("Mi nombre es: %s\n", nombre);
    printf("Tengo %d años.\n",edad);
    printf("En mi clase hay un maximo de %d estudiantes.\n",MAX);
    return 0;
}

.
.                                           →   Desarrollo o Cuerpo de Funciones
.
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

Estructura General de un Programa

Recomendación

Debe usar comentarios a lo largo del programa, esto para futuro mantenimiento para Ustedes o otros que usen sus programas

Los comentarios en C, pueden abarcar grandes bloques de texto o pequeños fragmentos de código o comentar líneas individuales.

1.- Una línea, usar barras // seguido de comentario.

Ejemplo:

```
// comentario de 1 línea
```

2.- Para escribir comentario de varias líneas, puede usar /* y */

Ejemplo:

```
/* Este programa realizar la suma de dos números ingresados
 por el usuario */
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

Estructura General de un Programa

Recomendación

Al inicio de programa debes colocar el siguiente encabezado y a lo largo del programa debe realizar los comentarios que den a lugar

```
// Autor(es): {Nombres y apellidos}
// {Descripción del Programa}
// Fecha de realización: {99-99-9999}
// Fecha de modificación: {99-99-9999}
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

Estructura General de un Programa

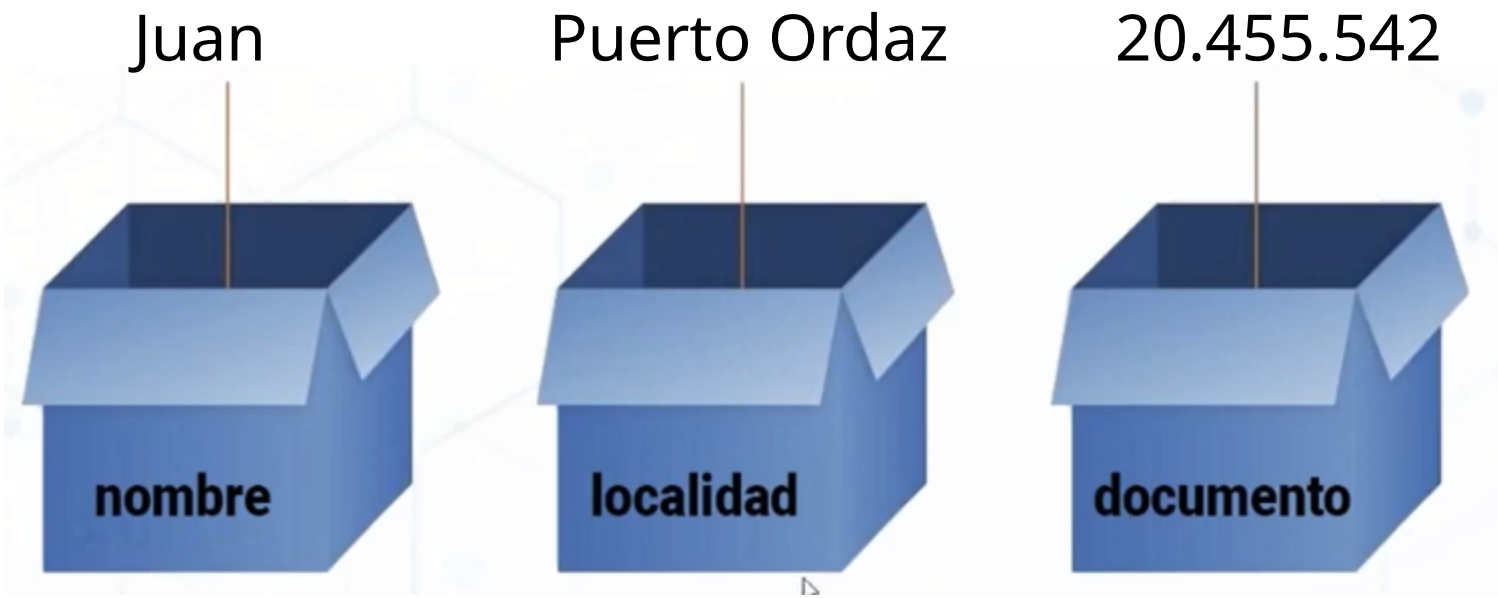
EJEMPLO CON LA PLANTILLA DE PROGRAMA

```
// Autor(es): Jose Luis Salazar
// Calcular el perímetro de Circulo (C) = 2 × π × radio (r)
// Fecha de realización: {01-01-1999}
// Fecha de modificación: {01-01-2000}

// librerías de lenguaje C;
#include <stdio.h>
#include <string.h>
#include <math.h>
// declaraciones de constantes
#define pi 3.1416
// declaraciones de variables Globales

// Declaración de Prototipo de Funciones

// función principal
int main()
{
    int r; // radio
    printf("Introduzca el radio: ");
    scanf("%d", &r);
    // Calcula e imprime Resultado
    printf("El perímetro es: %f\n", 2 * pi * r);

    return 0;
}
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

Variables

Contenedor que nos permite guardar datos.



Jose Luis Salazar → *jlsalazarg@gmail.com*

Variables

Se lee de arriba hacia abajo y de izquierda a derecha

Ejemplo

```
x = 20
y = x -  17
```

¿ Cual es el valor de Y ?

Jose Luis Salazar → *jlsalazarg@gmail.com*

Variables

Se lee de arriba hacia abajo y de izquierda a derecha

Ejemplo

```
x = 20
y = x -  17
y = 6
```

¿ Cual es el valor de Y ?

Jose Luis Salazar → *jlsalazarg@gmail.com*

Variables

Ámbito.

- **Según el lugar donde se declaren las variables tendrán un ámbito**

- **Según el ámbito pueden ser utilizadas desde cualquier parte del programa o únicamente en la función donde han sido declaradas.**

Jose Luis Salazar → *jlsalazarg@gmail.com*

Variables

Ámbito.

- **Locales:** **Cuando están declaradas dentro de la función**
- **Globales:** **Son conocidas a lo largo de todo el programa y se pueden utilizar desde cualquier parte del código**
- **De Registro:** **Son aquellas que se guardan en registros internos del microprocesador el acceso a ellas es más rápido y directo.**
- **Estáticas:** **Son aquellas variables locales donde se requiere mantener el valor entre una llamada y otra de una función. Se añade la palabra static delante del tipo.**
- **Externas:** **Se aplica a las variables globales cuyo valor se requiere cuando la compilación es por separado en pequeños módulos. Se añade la palabra extern delante del tipo.**

Jose Luis Salazar → *jlsalazarg@gmail.com*

Variables.

Características

Declaración: Se deben declarar antes de usarlas, indicando su tipo.

TIPOS	*RANGO*	*TAMAÑO*	*DESCRIPCIÓN*
char	-128 a 127	**1**	Para una letra o un dígito.
unsigned char	0 a 255	**1**	Letra o número positivo.
int	-32.768 a 32.767	**2**	Para números enteros.
unsigned int	0 a 65.535	**2**	Para números enteros.
long int	±2.147.483.647	**4**	Para números enteros
unsigned long int	0 a 4.294.967.295	**4**	Para números enteros
float	3.4E-38 decimales(6)	**6**	Para números con decimales
double	1.7E-308 decimales(10)	**8**	Para números con decimales
long double	3.4E-4932 decimales(10)	**10**	Para números con decimales

Jose Luis Salazar → *jlsalazarg@gmail.com*

Sintaxis y Semántica (Lenguaje C)

Variables.

Sintaxis

El nombre de las variables conocido como *identificadores* debe cumplir con las siguientes normas:

- La longitud puede ir de un (1) carácter a 31.
- El primero de ellos debe ser siempre una letra.
- No puede contener espacios en blanco, ni acentos y caracteres especiales.
- Hay que tener en cuenta que el compilador distingue entre mayúsculas y minúsculas.

```
tipo nombre=valor_numerico;

tipo nombre='letra';

tipo nombre[tamaño]="cadena de letras",

tipo nombre=valor_entero.valor_decimal;
```

Conversion (casting)

Las conversiones automáticas pueden ser controladas por el programador. Bastara con anteponer y encerrar entre parentesis, el tipo al que se desea convertir. Este tipo de conversiones solo es temporal y la variable a convertir mantiene su valor.

```
variable_destino=(tipo)variable_a_convertir;

variable_destino=(tipo) (variable1+variable2+variableN);
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

Sintaxis y Semántica (Lenguaje C)

Variables.

Ejemplo: convertimos 2 variables float para guardar la suma en un entero.

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

void main (void) {
    float num1=25.75, num2=10.15;
    int total=0;
    system("clear");
    total = (int) num1 + (int) num2;
    //total = (int) (num1 + num2) ;
    printf("El total de la suma: %d\n", total);
    getchar();
}
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

Constantes

Las constantes, son valores que no cambian durante la ejecución del programa. Son útiles para almacenar datos que deben permanecer fijos, como pi o tasas de interés.

Características:

Declaración con #define: Se utiliza en el preprocesador para definir una constante simbólica.

Declaración con const: Se declara como una variable, pero su valor no puede modificarse.

Jose Luis Salazar → *jlsalazarg@gmail.com*

Constantes

```
#define PI 3.1416    // Definición de una constante con preprocesador
const int MAX = 100; // Definición de una constante con const
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

Identificadores de formato

Son especificadores, que se usan en funciones como printf y scanf para indicar el tipo de dato que se desea mostrar o leer. Estos especificadores, le dicen al compilador cómo interpretar y manipular los datos.

Jose Luis Salazar → *jlsalazarg@gmail.com*

Identificadores de formato

Identificadores de Formato	
Identificador	**Descripción**
%c	Carácter
%d	Entero
%e ó %E	Notación Cientifica
%f	Coma flotante
%o	Octal
%s	Cadena
%u	Sin signo
%x ó %X	Hexadecimal
%p	Puntero
%ld	Entero Largo
%h	Short
%%	Signo %

Constantes de Carácter	
Constante	**Descripción**
\n	Salto de Linea
\f	Salto de Pagina
\r	Retorno de Carro
\t	Tabulación
\b	Retroceso
\’	Comilla simple
\”	Comillas
****	Barra invertida
\?	Interrogación

Existen especificadores de formato asociados a los identificadores que alteran su significado ligeramente. Se puede especificar la longitud mínima, el numero de decimales y la alineación. Estos modificadores se sitúan entre el signo de porcentaje y el identificador.

%**modificador**Identificador

Jose Luis Salazar → *jlsalazarg@gmail.com*

Identificadores de formato

printf

- El especificador de longitud mínima hace que un dato se rellene con espacios en blanco para asegurar que este alcanza una cierta longitud mínima. Si se quiere rellenar con ceros o espacios hay que añadir un cero delante, antes del especificador de longitud.

```
printf ("%f", numero);      //salida normal.
printf ("%10f", numero);    //salida con 10 espacios.
printf ("%010f", numero);   //salida con los espacios colocando ceros (0).
```

- El especificador de precisión sigue al de longitud mínima(si existe). Consiste en un nulo y un valor entero. Según el dato al que se aplica, su función varia. Si se aplica a datos en coma flotante determina el numero de posiciones decimales. Si es a una cadena determina la longitud máxima del campo. Si se trata de un valor entero determina el número de dígitos.

```
printf ("%10.4f ", numero);     //salida con 10 espacios con 4 decimales.
printf ("%10.15s ", cadena);    //salida con 10 caracteres dejando 15 espacios.
printf ("%4.4d ", numero);      //salida de 4 digitos minimo, completa con ceros
                                hasta 4 digitos.
```

- El especificador de ajuste fuerza la salida para que se ajuste a la izquierda, por defecto siempre lo muestra a la derecha. Se consigue añadiendo despues del porcentaje un signo menos.

```
printf ("%8d", numero);     //salida ajustada a la derecha.
printf ("%-8d",numero);     //salida ajustada a la izquierda.
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

Identificadores de formato

scanf

Es la rutina de entrada por consola. Puede leer todos los tipos de datos incorporados y convierte los números automáticamente al formato incorporado. En caso de leer una cadena lee hasta que encuentra un cáracter de espacio en blanco. El formato general:

```
scanf("identificador", &variable_numerica o char);.

scanf("identificador", variable_cadena);
```

La funcion scanf() tambien utiliza modificadores de formato, se especifica el numero máximo de caracteres de entrada y eliminadores de entrada. Para especificar el numero máximo solo hay que poner un entero despues del signo de porcentaje (%). si se desea eliminar entradas hay que añadir %*c en la posicion donde se desee eliminar la entrada.

```
scanf("%10s ", cadena);

scanf("%d%*c%d",&x, &y);
```

En muchas ocasiones se combinaran varias funciones para pedir datos y eso puede traer problemas para el buffer de teclado, es decir que asigne valores que no queramos a nuestras variables. Para evitar esto hay dos funciones que se utilizan para limpiar el buffer de teclado.

```
fflush(stdin);

fflushall();                    //limpia todos los datos almacenados en el buffer en flujos abiertos (stdin, stdout y stderr)
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

Identificadores de formato

Ejemplo práctico:

```
#include <stdio.h>

int main() {
    int edad = 25;
    float altura = 1.75;
    char inicial = 'A';
    char nombre[] = "Carlos";

    printf("Nombre: %s\n", nombre);      // Imprime la cadena
    printf("Inicial: %c\n", inicial);    // Imprime el carácter
    printf("Edad: %d\n", edad);          // Imprime el entero
    printf("Altura: %.2f metros\n", altura); // Imprime el flotante con dos decimales

    return 0;
}
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

Sintaxis y Semántica (Lenguaje C)

Otras Funciones Entrada/Salida

El archivo de cabecera de todas estas funciones es <stdio.h>. todas estas funciones van a ser utilizadas para leer caracteres o cadenas de caracteres. Todas ellas tienen asociadas una función de salida por consola.

FUNCIONES	*DESCRIPCIÓN*
var_char=getchar();	Lee un carácter de teclado, espera un salto de carro.
var_char=getche();	Lee un carácter con eco, no espera salto de carro.
var_char=getch();	Lee un carácter sin eco, no espera salto de carro.
gets(var_cadena);	Lee una cadena del teclado.
putchar(var_char);	Muestra un carácter en pantalla.
puts(variables);	Muestra una cadena en pantalla.

Jose Luis Salazar → *jlsalazarg@gmail.com*

OPERADORES ARITMÉTICOS

Pueden aplicarse a todo tipo de expresiones. Son utilizados para realizar operaciones matemáticas sencillas, aunque uniéndolos se pueden cualquier tipo de operaciones. En la siguiente tabla se muestran todos los operadores aritméticos.

OPERADOR	*DESCRIPCIÓN*	*ORDEN*
-	Resta.	3
+	Suma	3
*	Multiplica	2
/	Divide	2
%	Resto de una división	2
-	signo (monario).	2
--	Decremento en 1.	1
++	Incrementa en 1.	1

Jose Luis Salazar → *jlsalazarg@gmail.com*

OPERADORES LÓGICOS Y RELACIONALES

Los operadores relacionales hacen referencia a la relación entre unos valores y otros. Los lógicos se refiere a la forma en que esas relaciones pueden conectarse entre si. Los veremos a la par por la estrecha relación en la que trabajan.

OPERADORES RELACIONALES		
OPERADOR	***DESCRIPCIÓN***	***ORDEN***
<	Menor que.	**5**
>	Mayor que.	**5**
<=	Menor o igual.	**5**
>=	Mayor o igual	**5**
==	Igual	**6**
!=	Distinto	**6**

OPERADORES LÓGICOS		
OPERADOR	***DESCRIPCIÓN***	***ORDEN***
&&	Y (AND)	**10**
||	O (OR)	**11**
!	NO (NOT)	**1**

Jose Luis Salazar → *jlsalazarg@gmail.com*

OPERADORES A NIVEL DE BITS

Estos operadores son la herramienta mas potente, pueden manipular internamente las variables, es decir bit a bit. Este tipo de operadores solo se pueden aplicar a variables de tipo char, short, int y long. Para manejar los bits debemos conocer perfectamente el tamaño de las variables.

OPERADOR	*DESCRIPCIÓN*	*ORDEN*
&	Y (AND) bit a bit.	7
|	O (OR) Inclusiva.	9
^	O (OR) Exclusiva.	8
<<	Desplazamiento a la izquierda.	4
>>	Desplazamiento a la derecha.	4
~	Intercambia 0 por 1 y viceversa.	1

Jose Luis Salazar → *jlsalazarg@gmail.com*

Estructuras de Control

Condicionales o de Selección:

if	instrucción de selección simple
switch	instrucción de selección múltiple

Bucles o iteración:

while	instrucción de iteración con condición inicial.
do-while	instrucción de iteración con condición final.
for	instrucción de iteración especial (similar a las de repetición con contador)

Jose Luis Salazar → *jlsalazarg@gmail.com*

Sintaxis y Semántica (Lenguaje C)

Estructuras de Control — Condicionales

IF

La sentencia **if** elige entre varias alternativas en base al valor de una o más expresiones booleanas. Las sintaxis mas comunes son:

```
if (expresion-booleana) {
    bloque-sentencias;
}
```

```
if (expresion-booleana) {
    bloque-sentencias;
}else {
    Bloque-sentencias;
}
```

O la forma anidada, por ejemplo:

```
if (expresion-booleana) {
    bloque-sentencias;
}else if(expresion-booleana) {
    bloque-sentencias;
}else {
    bloque-sentencias;
}
```

Existe un *operador ternario* muy relacionado con la sentencia **if** es "**?:**", que puede sustituirlo asignaciones condicionales, el modo de trabajo es el siguiente, evalúa la condición y si es cierta toma el valor de la primera expresion y se la pasa a la variable; si es falsa, toma el valor de la segunda expresion y la pasa a la variable:

```
variable = condicion ? Expresion1 : expresion2;
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

Estructuras de Control

Condicionales

IF (ejemplo)

```
//Comentario de encabezado
#include <stdio.h>
#include "biblioteca.h"
int main(void){
    int peso;
    clrscr();
    gotoxy(5,3);
    printf("Introducir Peso: ");
    gotoxy(22,3);
    scanf("%d", &peso);
    if (peso < 500) {
        gotoxy(5,5);
        printf("No es ni media Tn.");
    } else {
        gotoxy(5,5);
        printf("Es más de media Tn.");
    }
    getchar();
    return 0;
}
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

Estructuras de Control — Condicionales

SWITCH

Realiza distintas operaciones en base al valor de unica variable o expresión. Es una sentencia muy similar a if-else, pero esta es mucho mas cómoda y facil de compremder. Si los valores con los que se comprara son números se ponen directamente, pero si es un carácter se debe encerrar entre comillas simples:

```
switch (expresion) {
    case valor1:
        bloque-sentencias;
        break;
    case valor2:
        bloque-sentencias;
        break;
    case valor3:
        bloque-sentencias;
        break;
    case valorN:
        bloque-sentencias;
        break;
    default:
        bloque-sentencias;
}
```

El valor de la expresión se compara con cada uno de los literales de la sentencia *case*; si coincide alguno, se ejecuta el código que le sigue, si ninguno coincide, se realizala sentencia *default* (opcional), si no hay sentencia *default*, no se ejecuta nada.

Jose Luis Salazar → *jlsalazarg@gmail.com*

Sintaxis y Semántica (Lenguaje C)

Estructuras de Control — Condicionales

SWITCH (ejemplo)

```
#include <stdio.h>
int main() {
    int day;
    printf("Ingrese un número del 1 al 7 para representar un día de la semana: ");
    scanf("%d", &day);
    switch (day) {
        case 1:
            printf("Lunes\n");
            break;
        case 2:
            printf("Martes\n");
            break;
        case 3:
            printf("Miércoles\n");
            break;
        case 4:
            printf("Jueves\n");
            break;
        case 5:
            printf("Viernes\n");
            break;
        case 6:
            printf("Sábado\n");
            break;
        case 7:
            printf("Domingo\n");
            break;
        default:
            printf("Número inválido. Ingrese un número del 1 al 7.\n");
    }
    return 0;
}
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

Sintaxis y Semántica (Lenguaje C)

Estructuras de Control — Bucles

WHILE

Ejecuta repetidamente el mismo bloque de código hasta que se cumpla una condición de terminación. Hay cuatro partes en cualquier bucle: *inicializacion, cuerpo, iteración y terminación.*

```
[inicialización];
while (terminación) {
    [cuerpo]; //bloque-sentencias;
    [iteración];
}
```

Ejemplo:

```
#include <stdio.h>

int main() {
    int i = 0;
    while (i < 5) {
        printf("El valor de i es: %d\n", i);
        i++;
    }
    printf("El bucle ha terminado.\n");
    return 0;
}
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

Estructuras de Control — Bucles

DO-WHILE

Es lo mismo que el caso anterior pero aquí como minimo siempre se ejecutara el cuerpo una vez, en el caso anterior es posible que no se ejecute ni una sola vez.

```
[inicialización];
do {
    [cuerpo]; //bloque-sentencias;
    [iteración];
} while (terminación);
```

Ejemplo:

```
#include <stdio.h>

int main() {
    int numero;
    do {
        printf("Ingrese un número: ");
        scanf("%d", &numero);
        printf("El número ingresado es: %d\n", numero);
    } while (numero != 0);
    printf("Programa terminado.\n");
    return 0;
}
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

Estructuras de Control — Bucles

FOR

Realiza las mismas operaciones que en los casos anteriores pero la sintaxis es una forma mas compacta. Normalmente la condición para terminar es de tipo numérico. La iteración puede ser cualquier expresion matemática válida.

```
for (inicio; fin; iteración){
    [cuerpo]; //bloque-sentencias;
}
```

Ejemplo:

```
#include <stdio.h>

int main() {
    int i;

    // Bucle for
    for (i = 0; i < 5; i++) {
        printf("El valor de i es: %d\n", i); // Imprime el valor de 'i' en cada iteración
    }

    return 0;
}
```

Nota: en este ejemplo se puede apreciar la similitud con el while pero de forma mas compacta.

Jose Luis Salazar → *jlsalazarg@gmail.com*

3. Traducción de Pseudocódigo al lenguaje de programación (equivalencias)

Traducción de Pseudocódigo al lenguaje de programación (equivalencias)

ENTORNO DE DESARROLLO INTEGRADO (IDE)

Aunque puedes programar en un editor de texto sencillo, un IDE hace el trabajo más cómodo al integrar el editor, el compilador y el depurador. Algunas opciones populares son:

- **Visual Studio Code**:
  - Ligero, personalizable y compatible con extensiones para C/C++.
  - Instala la extensión "C/C++" para mejorar tu experiencia.
  - Linux o windows
- **Dev-C++**:
  - Ligero y diseñado específicamente para C/C++.
  - Solo windows.
- **Coding C:**
  - Android.
  - No requiere internet para funcionar.
- Hacerlo con algun Editor de Lenguaje C en línea.

Jose Luis Salazar → *jlsalazarg@gmail.com*

Traducción de Pseudocódigo al lenguaje de programación (equivalencias)

Para programar, sigue los siguientes pasos :

1. Lectura del enunciado del problema o ejercicio, hasta comprenderlo.
2. Traslada a grafico de partes básicas de un Sistema

ENTRADA	PROCESO	SALIDA

3. Realizar el **Pseudocódigo**, atendiendo a la estructura de entrada, proceso y salida, cuidando la **indentacion.**
4. Traslada a lenguaje de programación, codifica en lenguaje C.

Pseudocódigo, forma de describir instrucciones lógicas que estructuralmente se asemeja a los lenguajes de programación

Jose Luis Salazar → *jlsalazarg@gmail.com*

Traducción de Pseudocódigo al lenguaje de programación (equivalencias)

Para programar, sigue los siguientes pasos :

La **INDENTACIÓN** de un código fuente, **es una práctica esencial en el mundo de la programación** que implica el uso de espacios o tabulaciones para estructurar y organizar el código de manera legible. Esta técnica permite destacar bloques de código, como bucles, condicionales o funciones, de manera que sea más fácil de entender para los programadores y menos propensa a errores.

La **INDENTACIÓN,** tiene sus raíces en la programación temprana, cuando los programadores se dieron cuenta de la importancia de organizar sus códigos de manera más legible. Surgió como una respuesta a la necesidad de hacer que el código fuera más estructurado y fácil de entender, especialmente a medida que los programas se volvían más complejos.

Jose Luis Salazar → *jlsalazarg@gmail.com*

Traducción de Pseudocódigo al lenguaje de programación (equivalencias)

Para programar, sigue los siguientes pasos :

En resumen, la INDENTACIÓN **es una práctica fundamental en la programación que mejora la legibilidad**, facilita la depuración y fomenta la consistencia en el código fuente. Es una herramienta esencial para los programadores de todos los niveles y contribuye a la eficiencia y la calidad en el desarrollo de software.

Jose Luis Salazar → *jlsalazarg@gmail.com*

Traducción de Pseudocódigo al lenguaje de programación (equivalencias)

Para programar, sigue los siguientes pasos :

1) Lectura del enunciado del problema o ejercicio, hasta comprenderlo.

   Realizar un programa que ingrese el nombre de estudiante y 3 notas de diferentes asignaturas y calcule el promedio de las notas. En caso que el promedio sea ≤ 5,49 muestre el mensaje: Reprobado, en caso contrario muestre el mensaje: Aprobado.

   → Luego ir al siguiente paso...

Jose Luis Salazar → *jlsalazarg@gmail.com*

Traducción de Pseudocódigo al lenguaje de programación (equivalencias)

Para programar, sigue los siguientes pasos :

1) Lectura. Realizar un programa que ingrese el nombre de estudiante y 3 notas de diferentes asignaturas y calcule el promedio de las notas. En caso que el promedio sea ≤ 5,49 muestre el mensaje: Reprobado, en caso contrario muestre el mensaje: Aprobado

2) Traslada a grafico de partes básicas de un Sistema

ENTRADA	PROCESO	SALIDA
• Nombre del estudiante • Nota1 • Nota2 • Nota3	Promedio = Nota1 + Nota2 + Nota3 DIVIDIR 3 Si promedio ≤ 5,49 Mensaje = reprobado De lo contrario Mensaje = Aprobado	• Nombre • Promedio • Mensaje

→ Luego ir al siguiente paso...

Jose Luis Salazar → *jlsalazarg@gmail.com*

Traducción de Pseudocódigo al lenguaje de programación (equivalencias)

Para programar, sigue los siguientes pasos :

3) Realizar el **Pseudocódigo**, atendiendo a la estructura de entrada, proceso y salida.

```
Variables:
        Nombre carácter (20)
        Nota1 decimal (2,2)
        Nota2 decimal (2,2)
        Nota3 decimal (2,2)
        Promedio (2,2)
Escribir “Ingrese el nombre del Estudiante“
Lectura Nombre
Escribir “Ingrese Nota 1“
Lectura Nota1
Escribir “Ingrese Nota 2“
Lectura Nota2
Escribir “Ingrese Nota 3“
Lectura Nota3
Promedio = Nota1 + Nota2+Nota3 /3
Si promedio ≤ 5,49
        Mensaje = Reprobado
De lo contrario
        Mensaje = Aprobado
Mostrar
        Nombre, Promedio, Mensaje
```

→ Luego ir al siguiente paso...

Jose Luis Salazar → *jlsalazarg@gmail.com*

Traducción de Pseudocódigo al lenguaje de programación (equivalencias)

Para programar, sigue los siguientes pasos :

4) Prepara el entorno de trabajo (IDE), y traslada a lenguaje de programación, codifica en lenguaje C.

Jose Luis Salazar → *jlsalazarg@gmail.com*

Traducción de Pseudocódigo al lenguaje de programación (equivalencias)

```
// Autor: Jose Luis Salazar
/* Programa que ingrese el nombre de estudiante y 3 notas de diferentes
   asignaturas  y calcula el promedio de las notas.
   En caso que el promedio sea ≤ 5,49 muestre el  mensaje: Reprobado,
   en caso contrario muestre el mensaje: Aprobado */
// Fecha de realización: 01/01/2025
// Fecha de modificación: 02/01/2025

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
   //  Definicion de Variables
   float nota1,nota2,nota3, promedio;
   char nombre[20];

   //  Entrada de Datos
   printf(" Ingrese Nombre del Estudiante.  ");
   gets(nombre);
   printf(" Ingrese la nota1.  ");
   scanf("%f",&nota1);
   printf(" Ingrese la nota2.  ");
   scanf("%f",&nota2);
   printf(" Ingrese la nota3.  ");
   scanf("%f",&nota3);
   system("clear");

   //  Calculo de Promedio de notas
   promedio  = (nota1 + nota2 + nota3) /3;

   //  Muestra resultados
   printf(" \n Estudiante: %s", nombre);
   printf(" \n Nota 1 => %010.2f", nota1);
   printf(" \n Nota 2 => %.2f", nota2);
   printf(" \n Nota 3 => %.2f", nota3);
   printf(" \n Promedio = %.2f", promedio);

   if (promedio >5.49)   {
      printf("\n Estudiante esta APROBADO.");
   }  else  {
      printf("\n Estudiante esta REPROBADO.");
   }

   return 0;
}
```


Jose Luis Salazar → *jlsalazarg@gmail.com*

¿Preguntas?

Ejercicios

Realizar los siguientes ejercicios, usando las técnicas dadas, de lectura, análisis, pseudocódigo y codificación

1. Escribir un programa que solicite un valor entero al usuario y determine si es positivo o negativo.
2. Escribe un programa que lea dos números enteros y muestre su suma.
3. Escribir un programa que solicite un valor entero al usuario y determine si es par o impar.
4. Crea un programa que convierta grados Celsius a Fahrenheit. Fórmula: Fahrenheit = (Celsius × 9/5) + 32
5. Escribe un programa que muestre la tabla de multiplicar de un número.
6. Crear un programa que calcule el mayor de tres números enteros introducidos por teclado.