***RESUMEN PROGRAMACION 2DO PARCIAL:***

* ***STRINGS:***
* ***CARACTERÍSTICAS:***

1. Es un **ARREGLO** de **LETRAS y/o CARACTERES**.
2. Recibe un **TRATAMIENTO ESPECIAL**: se lo trata como **TEXTO**.
3. Se le agrega un **CARÁCTER ESPECIAL** denominado **TERMINATOR ó NULO (‘\0’).** Indica el **FIN** del string.

**Ejemplo:**

char palabra[10] 🡪 espacio para 9 caracteres + \0

* ***FUNCIONES (includias en la librería <string.h>):***

1. **Void STRCPY** (destino, origen): **COPIA** todos los caracteres de **origen hasta destino** **HASTA** un nulo, y **AGREGA** un nulo al final.
2. **Int STRCMP** (string1, string2): **COMPARA** los strings. Devuelve **0** si son **IGUALES**, Devuelve **> 0** si el **string 1 > string2**, y devuelve **<0**, si el **string1 < string2**.

Compara **HASTA** encontrar uno distinto. **SENSIBLE** a **MAYÚS. y MINÚSC.**

**Ejemplo:**

Str1 = ´María’

Strcmp(str1, str2) = Devuelve <0 (M = M, a = a, r = r, i < t)

Str2 = ‘Marta’

‘María’ < ‘Marta’

1. **Int STRCMPI** (string1, string2): **MISMA** función que **STRCMP**. **NO DISTINGUE** entre **MAYÚSC. de MINÚSC.**
2. **Int STRLEN** (string1): Devuelve la **LONGITUD** del string (**SIN CONTAR** el carácter nulo).
3. **Char \*STRCAT** (destino, origen): **CONCATENA** al string destino el string origen, **UNIENDO** al extremo **DERECHO** del string destino el **INICIO** del string origen.

**Ejemplo:**

Origen = ‘Marta’, Destino = ‘María’. 🡪 strcat (destino, origen) = ‘MaríaMarta’.

1. **Gets** (palabra): **OTORGA** un valor a un string. Reconoce **TODOS** los caracteres del string (incluidos los **espacios**), y **AGREGA** el carácter **nulo** (‘\0’) al final.
2. **Puts** (palabra): **IMPRIME** strings. Misma función que printf(“%s”,palabra).

* ***ARREGLO DE STRINGS. CARACTERÍSTICAS:***

1. Es una **MATRIZ DE CARACTERES.** A cada **FILA** se la interpreta como un **BLOQUE DE TEXTO DISTINTO.**
2. Se recorre **SÓLO POR FILAS,** para cada fila hay un string.
3. **DECLARACIÓN:** char arregloStrings[4][10] 🡪 Se pueden cargar **4 filas** con una **longitud** (para el string) de 9 caracters + \0.

**Ejemplo:**

“Federico” 8 caracteres + nulo (9 caracteres usados)

“Gonzalo” 7 caracteres + nulo (8 caracteres usados)

4 filas “Abigail” 7 caracteres + nulo (8 caracteres usados)

“Agustina” 8 caracteres + nulo (9 caracteres usados)

* ***MATRICES:***
* ***CARACTERÍSTICAS:***

1. Una **MATRIZ** es un **ARREGLO** que se divide en **FILAS y COLUMNAS.**
2. **DECLARACIÓN:** int matriz[5[[5] 5 **filas** y 5 **columnas**.
3. **PARA AÑADIR/IMPRIMIR ELEMENTOS:** Se utilizan **DOS CICLOS FOR.** Uno que recorra las **FILAS** y otro las **COLUMNAS.**

**FORMA DE RECORRER LA MATRIZ:** Se recorren primero todas las **COLUMNAS** de la **FILA 1.** Luego, todas las **COLUMNAS** de la **FILA2,** luego todas las de la 3, y así sucesivamente hasta recorrer toda la matriz.

* ***ESTRUCTURAS:***
* ***CARACTERÍSTICAS:***

1. Son **REGISTROS.**
2. Crean un **NUEVO TIPO DE DATO.**
3. Permiten **AGRUPAR** datos de **DISTINTO TIPO.**
4. Al conjunto de datos internos se lo conoce como **CAMPO.**
5. **DECLARACIÓN:** Se utiliza la palabra reservada **STRUCT +** la sentencia **TYPEDEF** (typedef crea un **nuevo tipo de variable que tiene la forma de la estructura**).

**Ejemplo:**

**typedef Struct** { Variable **alumno** (nueva variable creada por typedef)

con la forma de la estructura (gracias a **struct**).

**int** edad;

**int** año;

**char** genero;

**char** nombre[30];

} **alumno**;

* ***ACCESO A LOS MIEMBROS DE LA ESTRUCTURA:*** Se escribe el **nombre de la variable de tipo estructura,** un **punto ‘.’**, y por último el **nombre del elemento** al que se quiere acceder.

**Ejemplo:**

alumno.genero

* ***ESCRIBIR/IMPRIMIR VALORES EN PANTALLA:*** Utilizando la misma metodología del punto anterior, para escribir/imprimir valores por pantalla, **se utilizan las funciones de escritura ESTÁNDAR**.

**Ejemplo:**

scanf(“%i”, &alumno.edad);

printf(“%i”, alumno.edad);

* ***ESTRUCTURAS ANIDADAS:***

Typedef Struct {

**Char** nombre [30];

**Struct dir** dirección;

**Long** teléfono;

**Float** sueldo;

} empleado;

Struct **dir** {

**Char** calle[25];

**Int** numero;

**Int** piso;

**Char** dpto.[4];

};

Para referenciar elementos de la **estructura anidada** se emplea **SUCESIVAMENTE el OPERADOR de SELECCIÓN ‘.’ (el punto).**

**Ejemplo:**

empleado.direccion.piso = 5;

* ***ARCHIVOS:***
* ***CARACTERÍSTICAS:***

1. Se utilizan para **NO PERDER INFORMACIÓN** y para **NO TENER QUE INGRESAR** los datos **CADA VEZ** que se ejecuta el programa.

* ***TIPOS:***

1. **TEXTO – CON FORMATO – SECUENCIA DE CARACTERES.**
2. **BINARIOS – SIN FORMATO – SECUENCIA DE BYTES.**

* ***MODOS DE ACCESO:***

1. **SECUENCIAL:** Para acceder a un **DATO ESPECÍFICO,** recorro **DATO A DATO** (leo dato a dato).
2. **DIRECTO:** Se accede a una posición determinada del archivo **como si fuese un arreglo.** Se avanza una secuencia de bytes.
3. **FILE\* fp** 🡪 puntero a FILE 🡪 **MANEJADOR del archivo.**

* ***FUNCIONES DE CONTROL:***

1. **FILE\* fopen** (ruta, modo): Si se abre, devuelve el puntero a FILE (FILE\*), sino devuelve NULL.
2. **Int fclose** (FILE\*): Si se cierra, devuelve 0, sino un -1.

* ***POSICIONAMIENTO:***

1. **Int Fseek** (FILE\*, desplazamiento, origen): **POSICIONA**  el puntero desde un **ORIGEN** determinado y un **DESPLAZAMIENTO.**

**\*DESPLAZAMIENTO:** Es la **CANTIDAD DE BYTES** que se desplaza el puntero **HACIA EL FINAL (>0) ó HACIA EL INICIO (<0)** del archivo **DESDE** la posición **ORIGEN.**

* **SEEK\_SET (0):** Indica el **COMIENZO.**
* **SEEK\_CUR (1):** Indica la **POSICIÓN** **ACTUAL.**
* **SEEK\_END (2):** Indica el **FIN DEL ARCHIVO.**
* En caso de **ERROR,** devuelve un **valor distinto de 0x0.**
  1. **DESPLAZAMIENTO** hacia la **DERECHA 🡪 POSITIVO.**
  2. **DESPLAZAMIENTO** hacia la **IZQUIERDA 🡪 NEGATIVO.**

**Ejemplos:**

* Desde **el COMIENZO** hacia la **Derecha:**

**Fseek** (fp, sizeof(int) \* 2, SEEK\_SET);

* Desde la **Posicion Actual** hacia la **Derecha:**

**Fseek** (fp, sizeof(int) \* 5, SEEK\_CUR);

* Desde la **Posición Final** hacia la **Izquierda:**

**Fseek** (fp, sizeof(int) \* -2, SEEK\_END);

1. **Int Ftell** (FILE\*): Retorna **DÓNDE ESTÁ POSICIONADO EL PUNTERO DEL ARCHIVO** en **CANTIDAD DE BYTES.**

* ***OPERACIONES DE LECTURA Y ESCRITURA:***

1. **Int Fwrite** (dirección de la variable a escribir, tamaño del tipo de dato, cantidad de datos que indica de a cuánto va a ir escribiendo, FILE\*).
2. **Int Fread** (dirección de la variable a leer, tamaño del tipo de dato, cantidad de datos que indica de a cuánto va a ir leyendo, FILE\*).

* ***MODOS DE ESCRITURA:***

1. **Modo WB 🡪 “Write Binary” 🡪 ABRE** el archivo. Si no existe, **LO CREA. POSICIONA** al **puntero** al **INICIO** del archivo. **SOBREESCRIBE** el archivo.
2. **Modo AB 🡪 “Append Binary” 🡪 ABRE** el archivo. Si no existe, **LO CREA. POSICIONA** al **puntero** al **FINAL** del archivo. Listo para **AÑADIR información.**

* ***MODOS DE LECTURA:***

1. **Modo RB** 🡪 “**Read Binary”** 🡪 **LEE** el archivo. Si no existe el archivo, **NO LO CREA.**
2. **Modo R+B 🡪** Abre un archivo binario para **operaciones de LECTURA Y ESCRITURA.**