Programación

Librería 1: #include <stdio.h>

Para comenzar: int main (), y para terminar: return 0.

Para usar dobles comillas: \”

Para usar una comilla: \’

Para usar 1 barra invertida: \\

Para cambiar de linea: \n

Para tabular: \t

Para cambiar la letra:

* Rojo: \033[31m
* Verde: \033[32m
* Amarillo: \033[33m
* Azul: \033[34m
* Morado: \033[35m
* Turquesa: \033[36m
* Blanco: \033[37m
* **Negrita**: \033[1m
* *Cursiva*: \033[3m
* Subrayado: \033[4m
* ~~Tachado~~: \033[9m

Para escribir algo: printf(“Hola”);

Para declarar una variable numérica con decimales: double variable

Para declarar una variable numérica con decimales, pero menos.: float variable

Para declarar una variable numérica sin decimales: int variable

Para declarar una variable de un caracter: char variable

Para declarar cadena de caracteres. char\* variable

Para leer una información se usa: scanf (“%lf”, &variable). %lf para double, %d para int, %c para char, %f para float y %s para char\*(no hace falta poner el &). Para elegir número de decimales: %2.lf.

Para leer una variable declarada: printf(“%lf”, variable);

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamentePara saber el tamaño en bits: sizeof(char), sizeof(int)…

Para saber el limite : #include <limits.h>

Una captura de pantalla de un celular con letras

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Codigo ASCII:

Imagen que contiene Interfaz de usuario gráfica, Texto

Descripción generada automáticamenteEscribir Hola en ASCII:

Si no deja escribir un segundo scanf, poner /n:

Texto, Carta

Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamenteOperadores:

Imagen que contiene Tabla

Descripción generada automáticamente

Imagen que contiene Aplicación

Descripción generada automáticamente

Imagen que contiene Aplicación

Descripción generada automáticamente

Instrucción If: **if (indice < 18.5){**  
**printf("Tienes bajo peso\n");  
        printf("El límite inferior de índice de masa corporal es 18.5n”);**

**Instr**ucción else y else if: **if (indice < 18.5)**  
**printf("Tienes bajo peso\n");**  
**else if (indice >= 25)**  
**printf("Tienes sobrepeso\n");**  
**else**  
**printf("Tienes peso normal\n");**

Instrucción switch:   
    **switch (opcion) {**  
**case 1:**            
            printf("Has elegido comenzar la partida.\n");  
            **break;**  
**case 2:**           
            printf("Aquí estará el código para mostrar los ajustes del juego.\n");  
            **break;**  
**case 3:**            
            printf("Aquí estará el código para mostrar la ayuda.\n");  
            **break;**  
**case 4:**           
            printf("¡Hasta luego!\n");  
            break;  
        **default:**  
            printf("Opción incorrecta.\n");  
    }

Enumeraciones: **enum {jugar=1, ajustes, ayuda, salir};**

   switch (opcion) {  
        case **jugar**:             
            printf("Has elegido comenzar la partida.\n");  
            break;  
        case **ajustes**:             
            printf("Aquí estará el código para mostrar los ajustes del juego.\n");  
            break;  
        case **ayuda**:             
            printf("Aquí estará el código para mostrar la ayuda.\n");  
            break;  
        case **salir**:             
            printf("¡Hasta luego!\n");  
            break;  
        default:  
            printf("Opción incorrecta.\n");  
    }

Para asignar un valor constante Macros: #define DIM 34 Para asignar un valor constante: **const double variable = 0.0328; Macro con parámetros:**

#include <stdio.h>  
  
**#define MULT(A, B) A\*B**  
  
int main() {  
    printf("MULT(5,5): %d\n", **MULT(5,5)**);  
    printf("MULT(2+3,3+2): %d\n", **MULT(2+3,3+2)**);  
    return 0;  
}

Tablas: int datos[3]: datos[0]=2 datos[1]=6 datos[2]: 8 Ejemplo de Macro y tablas:

#include <stdio.h>  
  
**#define DIM 3**  
  
int main () {  
    char datos\_c[**DIM**];  
    short datos\_s[**DIM**+1];  
    int datos\_i[2+**DIM**\*2];  
    long datos\_l[3\***DIM**];  
    float datos\_f[**DIM**\*4];  
    double datos\_d[**DIM**%5];  
      
    return 0;  
}

Inicializadores:   int datos[] = {8,7,9};

|  |  |  |
| --- | --- | --- |
| num[0][0] | num[0][1] | num[0][2] |
| num[1][0] | num[1][1] | num[1][2] |

Ejemplo: #define DIM 3  
  
int main () {  
    **int num[DIM][DIM]**

Ejemplo: #include <stdio.h>

#define FIL 3 #define COL 2

int main() {

int fila, columna; char barcos[FIL][COL] = {{'X','O'},{'O','X'},{'X','X'}};

printf("Introduce una fila: ");

scanf("%d", &fila);

printf("Introduce una columna: ");

scanf("%d", &columna) printf("En la fila %d, columna %d, encontramos: %c\n", fila, columna, barcos[fila-1][columna-1]);

printf("Introduce una fila: ");

scanf("%d", &fila);

printf("Introduce una columna: ");

scanf("%d", &columna);

printf("En la fila %d, columna %d, encontramos: %c\n", fila, columna, barcos[fila-1][columna-1]);

return 0;

}

Ejemplo cadena: char saludo[16] = "¡Hola, mundo!"; // ENTRE DOBLES COMILLAS

Para leer la longitud d caracteres se utiliza: strlen (0)

Incluir biblioteca: #include <string.h>

#include <stdio.h>  
#include <string.h>  
  
#define DIM 20   
  
int main() {  
    char nombre1[DIM] = "Ana"; // ENTRE DOBLES COMILLAS  
    printf("La longitud de nombre1 es %d\n", **strlen(nombre1)**); //%d  
    return 0;  
}

Utilizar %s para leer cadenas:

#include <stdio.h>  
  
#define DIM 8  
  
int main() {  
    char nombre[DIM];  
  
    printf("Introduce tu nombre: ");  
    **scanf("%s", nombre);**  
  
    **printf("Hola, %s.", nombre);**  
  
    return 0;  
}

Para leer cadenas también se puede usar gets(cadena), parta usarlo después de un scanf hay que usar getchar();

#include <stdio.h>  
  
#define TAM 16  
  
int main() {  
    char nombre[TAM], ciudad[TAM];  
  
    printf("Introduce tu nombre: ");  
    **scanf("%s", nombre);**  
  
    printf("Introduce tu ciudad de residencia: ");  
    getchar();  
    **gets(ciudad);**  
  
    printf("Hola, %s. Vives en %s.\n", nombre, ciudad);  
  
    return 0;  
}

Para comparar cadenas se usa strcmp(cadena, cadena1): así dira cual es de ls dos la más larga: resultado 1: positivo, resultado2: 0, resultado 3: negativo.

#include <stdio.h>  
#include <string.h>  
  
#define DIM 8  
  
int main() {  
    char nombre1[DIM] = "Ana";  
    char nombre2[DIM] = "Anabel";  
  
    printf("Ana vs. Anabel: %d\n", strcmp(nombre1, nombre2)); //%d  
    printf("Ana vs. Ana: %d\n", strcmp(nombre1, nombre1));  
    printf("Anabel vs. Ana: %d\n", strcmp(nombre2, nombre1));  
  
    return 0;  
}

La función **strcpy()**, que pertenece a la biblioteca **string.h**, permite copiar una cadena de caracteres a otra.

#include <stdio.h>  
#include <string.h>  
  
#define DIM 8  
  
int main() {  
    char nombre1[DIM]="Ana", nombre2[DIM];  
  
    **strcpy(nombre2, nombre1);**  
  
    printf("nombre1=%s, nombre2=%s\n", nombre1, nombre2);  
  
    return 0;  
}

Para crear estructuras se usa: typedef struct{estructura}nombre;

#include <stdio.h>  
  
#define MAX\_STR 128  
  
**typedef struct {**  
**char autor[MAX\_STR];**  
**char titulo[MAX\_STR];**  
**int anyo;**  
**} Libro;**  
  
int main () {  
    **Libro** novela = {"Mario Vargas Llosa", "El héroe discreto", 2013}**;**  
     
    printf("%s fue escrito por %s en %d.\n", novela.titulo, novela.autor, novela.anyo);    
  
    return 0;  
}

Tablas y cadenas de caracteres:

#include <stdio.h>  
  
#define MAX\_STR 128  
#define MAX\_EDICIONES 10  
  
typedef struct {  
    char autor[MAX\_STR];  
    char titulo[MAX\_STR];  
**int fechas\_ediciones[MAX\_EDICIONES];**  
} Libro;  
  
int main () {  
    Libro novela = {"Mario Vargas Llosa", "El héroe discreto", **{2013, 2014, 2016}**};  
  
    **novela.fechas\_ediciones[1]** = 2015;  
     
    printf("%s fue escrito por %s y publicado en %d, %d y %d.\n", novela.titulo, novela.autor, **novela.fechas\_ediciones[0], novela.fechas\_ediciones[1], novela.fechas\_ediciones[2]**);  
      
    return 0;  
}

Ejemplo de estructura anidada:

#include <stdio.h>  
  
#define MAX\_STR 128  
  
**typedef struct {**  
**char nombre[MAX\_STR];**  
**char primer\_apellido[MAX\_STR];**  
**char segundo\_apellido[MAX\_STR];**  
**} Escritor;**  
  
typedef struct {  
    **Escritor autor;**  
    char titulo[MAX\_STR];  
    int anyo;  
} Libro;  
  
int main() {  
    Libro novela;  
  
    printf("Introduce el nombre del autor: ");  
    gets(**novela.autor.nombre**);  
    printf("Introduce el primer apellido del autor: ");  
    gets(**novela.autor.primer\_apellido**);  
    printf("Introduce el segundo apellido del autor: ");  
    gets(**novela.autor.segundo\_apellido**);  
    
    printf("Introduce el título del libro: ");  
    gets(novela.titulo);  
    printf("Introduce el año de publicación del libro: ");  
    scanf("%d", &novela.anyo);  
  
    printf("%s fue escrito en %d por %s %s %s.\n", novela.titulo, novela.anyo, novela.autor.nombre, novela.autor.primer\_apellido, novela.autor.segundo\_apellido);  
    
    return 0;  
}

Ejemplo de tablas y estructura con cadenas:

#include <stdio.h>  
  
#define MAX\_STR 128  
**#define DIM\_NOVELAS 4**  
  
typedef struct {  
    char nombre[MAX\_STR];  
    char primer\_apellido[MAX\_STR];  
    char segundo\_apellido[MAX\_STR];  
} Escritor;  
  
typedef struct {  
    Escritor autor;  
    char titulo[MAX\_STR];  
    int fecha;  
} Libro;  
  
int main () {  
    **Libro novelas[DIM\_NOVELAS];**  
  
    /\* Primer libro \*/  
    printf("Introduce el nombre del autor del primer libro: ");  
    gets(**novelas[0]**.autor.nombre);  
    printf("Introduce el título del primer libro: ");  
    gets(**novelas[0]**.titulo);  
  
    /\* Segundo libro \*/  
    printf("Introduce el nombre del autor del segundo libro: ");  
    gets(**novelas[1]**.autor.nombre);  
    printf("Introduce el título del segundo libro: ");  
    gets(**novelas[1]**.titulo);  
  
    printf("%s fue escrito por %s.\n", **novelas[0]**.titulo, **novelas[0]**.autor.nombre);  
    printf("%s fue escrito por %s.\n", **novelas[1]**.titulo, **novelas[1]**.autor.nombre);  
     
    return 0;  
}

OTRO EJEMPLO

#include <stdio.h>  
  
#define MAX\_STR 128  
#define DIM\_NOVELAS 4  
**#define DIM\_AUTORES 3**  
  
typedef struct {  
    char nombre[MAX\_STR];  
    char primer\_apellido[MAX\_STR];  
    char segundo\_apellido[MAX\_STR];  
} Escritor;  
  
typedef struct {  
    **Escritor autores[DIM\_AUTORES];**  
    char titulo[MAX\_STR];  
    int anyo;  
} Libro;  
  
int main () {  
    Libro novelas[DIM\_NOVELAS];  
  
    /\* Primer libro \*/  
    printf("Introduce el nombre del primer autor del primer libro: ");  
    gets(novelas[0].**autores[0]**.nombre);  
    printf("Introduce el nombre del segundo autor del primer libro: ");  
    gets(novelas[0].**autores[1]**.nombre);  
    printf("Introduce el título del primer libro: ");  
    gets(novelas[0].titulo);  
  
    /\* Segundo libro \*/  
    printf("Introduce el nombre del autor del segundo libro: ");  
    gets(novelas[1].**autores[0]**.nombre);  
    printf("Introduce el título del segundo libro: ");  
    gets(novelas[1].titulo);  
  
    printf("%s fue escrito por %s y %s.\n", novelas[0].titulo, novelas[0].**autores[0]**.nombre, novelas[0].**autores[1]**.nombre);  
    printf("%s fue escrito por %s.\n", novelas[1].titulo, novelas[1].**autores[0]**.nombre);  
     
    return 0;  
}

El**bucle while**nos permite **repetir un conjunto de instrucciones hasta que cierta condición deje de cumplirse:**

#include <stdio.h>  
  
int main() {  
    int opcion;  
  
    printf("Menú:\n  1. Empezar partida\n  2. Ajustes\n  3. Ayuda\n  4. Salir\n");  
    printf("Selecciona una opción: ");   
    scanf("%d", &opcion);  
  
    **while (opcion<1 || opcion>4) {**  
        printf("Opción incorrecta. Inténtalo de nuevo: ");    
        scanf("%d", &opcion);  
    **}**  
  
    switch (opcion) {  
        case 1:            
            printf("Has elegido comenzar la partida.\n");  
            break;  
        case 2:            
            printf("Aquí estará el código para mostrar los ajustes del juego.\n");  
            break;  
        case 3:            
            printf("Aquí estará el código para mostrar la ayuda.\n");  
            break;  
        case 4:            
            printf("¡Hasta luego!\n");  
            break;  
    }  
  
    return 0;  
}

Para crear un bucle se debe inicializar a 0 la variable y luego ir sumando de uno en uno:

#include <stdio.h>  
  
#define DIM 3  
  
int main() {  
    int i=0;  
  
    **while(i<DIM) {**  
        printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");    
        i++;  
    **}**  
  
    return 0;  
}

Ejemplo:

#include <stdio.h>  
  
#define DIM 10  
  
int main() {  
    int **i=0**, suma=0, total, **tabla[DIM]**;  
  
    printf("¿Cuántos números quieres sumar (entre 0 y %d)? ", DIM);  
    scanf("%d", &**total**);  
  
    while(**i<total**) {  
        printf("Introduce un número: ");  
        scanf("%d", &**tabla[i]**);  
        suma += tabla[i];  
        **i++**;  
    }  
  
    printf("La suma de los %d números: ", total);  
  
    **i=0**;  
    while(**i<total**) {  
        printf("%d ", **tabla[i]**);  
        **i++**;  
    }  
  
    printf("es %d.", suma);  
  
    return 0;  
}

Bucle do while:

#include <stdio.h> int main (){

int num, clave; printf ("Teclea la clave: "); scanf ("%d", &clave);

do{

printf ("Introduzca un entero: "); scanf ("%d", &num); if (num < clave) printf ("%d es menor que la clave\n", num); else if (num > clave) printf ("%d es mayor que la clave\n", num);

} while (num != clave);

printf ("Numero correcto.\n");

return 0;

}

Otro ejemplo:

#include <stdio.h>

int main () {

int i, veces, tabla[10] = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};

do{

printf("Â¿CuÃ¡ntos elementos quieres mostrar por pantalla? Elige un nÃºmero entre 1 y 10: ");

scanf("%d", &veces);

if (veces<1 || veces>10)

printf("Has de introducir un nÃºmero entre 1 y 10. Vuelve a intentarlo.\n");

}while(veces<1 || veces>10);

i=0;

while (i<veces){

printf("%d ", tabla[i]);

i++;

}

return 0;

}

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

**Ejemplo tabla puntero:**

#include <stdio.h>

#define TAM 3

int main ()

{

int uno [TAM] = {1, 2, 3};

double dos [TAM] = {1.2, 3.1, 2.5};

int \* p;

double \* q;

int i;

for(i=0, p = uno, q = dos; i<TAM; i++, p++, q++)

{

printf("Lista uno: %d\n", \*p);

printf("Lista dos: %lf\n", \*q);

}

return 0;

}

**Calcular media:**

#include <stdio.h>  
#define DIM 3  
  
double media(**int \*p, int num**) {  
    int i;  
    double m;  
  
    for (i=0, m=0.0; i<num; i++)  
        m += p[i];  
  
    return m/num;  
}  
  
int main() {  
    int datos[DIM] = {2,3,5};  
  
    printf("La media es: %.2lf", **media(datos, DIM));**  
  
    return 0;  
}

**Para incrementar:**

#include <stdio.h>

#define TAM 3

#define INC 5

void incrementa(int \* tabla, int tamano, int incremento)

{

int i;

for(i=0; i<tamano; i++)

tabla[i] += incremento;

}

int main()

{

int tabla[TAM] = {2, 4, 7};

int i;

printf("El contenido de la tabla es:\n");

for(i=0; i<TAM; i++)

printf("%d\n", tabla[i]);

incrementa(tabla, TAM, INC);

printf("El nuevo contenido de la tabla es:\n");

for(i=0; i<TAM; i++)

printf("%d\n", tabla[i]);

return 0;

}

**Incluir texto:**

#include <stdio.h>  
#include <string.h>  
  
#define DIM 10  
  
void agregar\_txt(char \*cadena) {  
    int i = strlen(cadena);  
  
    cadena[i] = '.';  
    i++;  
    cadena[i] = 't';  
    i++;  
    cadena[i] = 'x';  
    i++;  
    cadena[i] = 't';  
    i++;  
    cadena[i] = 0;  
}  
  
int main() {  
    char nombre[DIM] = "data";  
  
    agregar\_txt(nombre);  
    printf("%s", nombre);  
  
    return 0;  
}

**Minusculas a mayusculas:**

#include <stdio.h>  
#define DIM 10  
  
void minu\_a\_mayu(char \*p) {  
    for (; \*p!= 0; p++)  
        \*p += 'A'-'a';  
}  
  
int main() {  
    char nombre[DIM] = "data";  
  
    minu\_a\_mayu(nombre);  
    printf("%s", nombre);  
  
    return 0;  
}

**Para contra palabras:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define TAM 1000

void contarPalabras(char \*s, long \*nPalabras);

int main(){

char cadena[TAM];

long nPalabras;

printf("Escribe una cadea de caracteres:");

gets(cadena);

contarPalabras(cadena,&nPalabras);

printf("El numero de palabras es:%ld\n",nPalabras);

}

void contarPalabras(char \*s, long \*nPalabras)

{

long i,k;

for (i=0;s[i]==' ';i++); //elimina los blancos al principio de la cadena

// cuenta las palabras

(\*nPalabras)=0;

if (strlen(&s[i])) {

for (;i<strlen(s);i++)

if (s[i]==' ') {

while (i<strlen(s) && s[i]==' ')

i++;

if (s[i]!='\0')

(\*nPalabras)++;

}

(\*nPalabras)++;

}

}

**FUNCION malloc(),** ejemplo media de edades**,** biblioteca  **stdlib.h** para free() y malloc()

#include <stdio.h>  
#include <stdlib.h>  
  
int main() {  
    int i, n, \*edades;  
    double total=0;  
  
    printf("¿Cuántas personas han acudido a la fiesta? ");  
    scanf("%d", &n);  
  
    **edades = (int\*) malloc(n\*sizeof(int));**  
    if (edades == NULL) {  
        printf("La reserva de memoria ha fallado.");  
        return 1;  
    }  
  
    for (i=0; i<n; i++) {  
        printf("Dime la edad de la persona %d: ", i+1);  
        scanf("%d", &edades[i]);    
        total += edades[i];     
    }  
  
    printf("La media de edad de la fiesta es: %.2lf", total/n);  
**free(edades);**  
  
    return 0;  
}

Para C, "**NULL**" se define tradicionalmente como (void \*) 0; en otras palabras, es un alias de puntero a la dirección 0. Para C ++, "NULL" normalmente se define como "0".

Biblioteca **stdio.h**

Además de la función malloc(), también podemos utilizar la función calloc() para reservar memoria. Ambas funciones son muy similares. La diferencia está en que la función calloc() inicializa todos los miembros de la memoria reservada a cero. La función calloc() está definida en la biblioteca **stdlib.h**, por eso es necesario *incluirla* en nuestro programa a través de la instrucción #include.

#include <stdio.h>  
#include <stdlib.h>  
  
int main() {  
    int i, n, \*edades;  
    double total=0;  
  
    printf("¿Cuántas personas han acudido a la fiesta? ");  
    scanf("%d", &n);  
  
  **edades = (int\*) calloc(n, sizeof(int));**  
    if (edades == NULL) {  
        printf("La reserva de memoria ha fallado.");  
        return 1;  
    }  
  
    for (i=0; i<n; i++) {  
        printf("Dime la edad de la persona %d: ", i+1);  
        scanf("%d", &edades[i]);    
        total += edades[i];     
    }  
  
    printf("La media de edad de la fiesta es: %.2lf", total/n);  
    free(edades);  
  
    return 0;  
}

**Ejemplo añadiendo numero a la tabla:**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int tamano, \* tabla, i, \*datos\_nuevo;

double media;

printf("Introduce cuantos enteros se van a leer: ");

scanf("%d", &tamano);

tabla = (int \*) malloc (tamano \* sizeof(int));

if (tabla == NULL)

{

printf("ERROR");

return 1;

}

for (i=0, media=0; i<tamano; i++)

{

printf("Introduce el numero %d: ", i+1);

scanf("%d", &tabla[i]);

media+=tabla[i];

}

media /= i;

printf("La media de los numeros leidos es: %lf", media);

/\* Se reserva memoria para otra tabla con un espacio más \*/

datos\_nuevo = (int \*) malloc ((tamano+1) \* sizeof(int));

if (datos\_nuevo == NULL)

{

printf ("ERROR: No se pudo reservar memoria\n");

free(tabla); /\* Antes de terminar, hay que liberar toda memoria que se haya reservado anteriormente \*/

return 1;

}

/\* Se copia en la tabla nueva, uno a uno, todos los datos de la tabla antigua \*/

for (i=0; i<tamano; i++)

datos\_nuevo[i] = tabla[i];

/\* Ya se puede liberar la memoria de la tabla antigua. Los datos de la tabla antigua desaparecen \*/

free(tabla);

/\* Se asigna al puntero de la tabla antigua (tabla) el puntero de la nueva (datos\_nuevo) \*/

/\* A partir de este momento se puede utilizar tanto tabla como datos\_nuevos de forma equivalente pues ambos punteros apuntan a la tabla ampliada \*/

tabla = datos\_nuevo;

/\* Se lee el nuevo dato y se guarda en la última posición de la tabla ampliada \*/

printf("\nIntroduce un nÃºmero extra: ");

scanf("%d", &tabla[tamano]); /\* Recuerda que scanf("%d", &datos\_nuevo[tamano]); sería equivalente y exactamente igual de válido \*/

/\* Se aumenta el valor del entero tamano para indicar que hay un nuevo dato guardado en la tabla \*/

tamano++;

/\* Se vuelve a calcular la media \*/

/\* También se podría haber calculado utilizando el valor previo de media y así ahorrarse el bucle\*/

for (i=0, media=0; i<tamano; i++)

media+=tabla[i];

media /= i;

printf("La nueva media de los numeros leidos es: %lf", media);

/\* Se libera la memoria de la tabla ampliada \*/

free(tabla); /\* La memoria deja de ser accesible, tanto tabla como datos\_nuevo apuntan a una zona de memoria que ya no contiene datos por lo que hacer cosas como tabla[0]=2 o free(datos\_nuevo) serían errores, pues ya no apuntan a ninguna tabla \*/

return 0;

}

**Calculo de fechas con puntero tabla, fecha más antigua :**

#include <stdio.h>

#include <stdlib.h>

typedef struct

{

int dia;

int mes;

int agno;

} Fecha;

int main ()

{

int i, dim, mas\_antigua;

Fecha \* fechas;

printf ("Introduzca dimension de la tabla: ");

scanf ("%d", &dim);

fechas = (Fecha \*) malloc (dim \* sizeof (Fecha));

if (fechas == NULL)

{

printf ("No se pudo reservar memoria\n");

return 1;

}

for (i=0; i<dim; i++)

{

printf ("Introduzca dia de la fecha %d: ", i);

scanf ("%d", &fechas[i].dia);

printf ("Introduzca mes de la fecha %d: ", i);

scanf ("%d", &fechas[i].mes);

printf ("Introduzca agno de la fecha %d: ", i);

scanf ("%d", &fechas[i].agno);

}

/\* CÃ¡lculo de la fecha mas antigua \*/

for (i=1, mas antigua=0; i<dim; i++)

{

if (fechas[i].agno < fechas[mas\_antigua].agno)

mas\_antigua = i;

else if (fechas[i].agno == fechas[mas\_antigua].agno)

{

if (fechas[i].mes < fechas[mas\_antigua].mes)

mas\_antigua = i;

else if ( (fechas[i].mes == fechas[mas\_antigua].mes) && (fechas[i].dia < fechas[mas\_antigua].dia) )

mas\_antigua = i;

}

}

printf ("La fecha mas antigua es %d/%d/%d\n", fechas[mas\_antigua].dia, fechas[mas\_antigua].mes, fechas[mas\_antigua].agno);

free (fechas);

return 0;

}

**Reservar tablas y cadenas bidimensionales:**

#include <stdio.h>  
#include <stdlib.h>  
  
int main() {  
    int i, j, estudiantes, examenes;  
    double \*\*notas;  
  
    printf("¿Cuántos estudiantes hay en tu clase? ");  
    scanf("%d", &estudiantes);  
    printf("¿Cuántos exámenes habéis hecho? ");  
    scanf("%d", &examenes);  
  
    notas = (double\*\*) malloc(examenes\*sizeof(double\*));  
    if (notas == NULL) {  
        printf("La reserva de memoria ha fallado.");  
        return 1;  
    }  
  
    for (i=0; i<examenes; i++) {  
        notas[i] = (double\*) malloc(estudiantes\*sizeof(double));  
          
        if (notas[i] == NULL) {  
            for (j=0; j<i; j++)  
                free(notas[j]);      
            free(notas);          
            printf("La reserva de memoria ha fallado.");  
            return 1;  
        }    
    }  
  
    // Código para almacenar las notas de los estudiantes para cada examen.  
  
    for (i=0; i<examenes; i++)  
        free(notas[i]);  
    free(notas);  
  
    return 0;  
}

**Ejemplo:**

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

double\*\* malloc\_matriz\_double(int m, int n) {  
    int i, j;  
    double \*\*matriz;  
  
    if ((matriz = (double\*\*) malloc(m\*sizeof(double\*))) == NULL)  
        return NULL;  
  
    for (i=0; i<m; i++) {  
        if ((matriz[i] = (double\*) malloc(n\*sizeof(double))) == NULL) {  
            free\_matriz\_double(matriz, i);  
            return NULL;  
        }  
    }  
  
    return matriz;  
}  
  
void free\_matriz\_double(double \*\*matriz, int m) {  
    int i;  
  
    for (i=0; i<m; i++)  
        free(matriz[i]);  
  
    free(matriz);  
}

int main() {  
    int i, j, estudiantes, examenes;  
    double \*\*notas;  
  
    printf("¿Cuántos estudiantes hay en tu clase? ");  
    scanf("%d", &estudiantes);  
    printf("¿Cuántos exámenes habéis hecho? ");  
    scanf("%d", &examenes);  
  
    if ((notas = malloc\_matriz\_double(examenes, estudiantes)) == NULL) {  
        printf("malloc\_matriz\_double() ha fallado.");  
        return 1;  
    }  
  
    // Código para almacenar las notas de los estudiantes para cada examen.  
     
    free\_matriz\_double(notas, examenes);  
  
    return 0;  
}

Como acabamos de ver, estos son los **pasos para leer un fichero**:

1. Declarar una variable de tipo puntero a FILE (*FILE \**).
2. Abrir el fichero con la función *fopen()* en modo lectura ("r").
3. Comprobar que el fichero se ha abierto correctamente.
4. Leer la información del fichero con funciones como *fscanf()*.
5. Cerrar el fichero con la función *fclose()*.

Ejemplo : Contenido del fichero notas.txt: 96545675Z 1 4.34  
96345645D 1 7.2  
94236532G 2 6.56

#include <stdio.h>  
  
int main() {  
    **FILE \*f;**  // Paso 1  
    char dni[10];  
    int modelo, i;  
    double nota, media;  
  
    **f = fopen("notas.txt", "r");** // Paso 2  
    **if (f == NULL)** {  // Paso 3  
        printf("Error abriendo el archivo\n");  
        return 1;  
    }  
  
    for (i=0, media=0.0; **fscanf(f, "%s %d %lf", dni, &modelo, &nota)** == 3; i++)  // Paso 4  
        media += nota;  
  
    printf("La nota media es = %.2lf\n", media/i);  
    **fclose(f);**  // Paso 5  
  
    return 0;  
}

**Ejemplo 2:**

#include <stdio.h>

int main() {

FILE \*f;

char Id[9];

int modelo, i, aciertos, fallos, Rblanco, alumnos1, alumnos2;

double nota, media1, media2;

f = fopen("notas.txt", "r");

if (f == NULL) {

printf("Error abriendo el archivo\n");

return 1;

}

for (i=0, media1=0.0, media2=0.0, alumnos1=0, alumnos2=0; fscanf(f, "%s", Id) == 1; i++){

fscanf (f, "%d", &modelo);

fscanf (f, "%d", &aciertos);

fscanf (f, "%d", &fallos);

fscanf (f, "%d", &Rblanco);

fscanf (f, "%lf", &nota);

if(modelo==1 ){

media1 += nota;

alumnos1++;

}

else{

media2 += nota;

alumnos2++;

}

}

printf("La nota media del modelo 1 es = %.2lf\n", media1/alumnos1);

printf("La nota media del modelo 2 es = %.2lf\n", media2/alumnos2);

fclose(f);

return 0;

}

**Escribir en un fichero:**

#include <stdio.h>  
  
#define NESTUDIANTES 3  
#define MAX\_DNI 10  
  
typedef struct {  
    char dni[MAX\_DNI];  
    int modelo;  
    double nota;  
} notaEstudiante;  
  
int main() {  
    **FILE \*f;**  // Paso 1  
    int i;  
    notaEstudiante notas[NESTUDIANTES]={{"96545675Z",1,4.34},{"96345645D",1,7.2},{"94236532G",2,6.56}};  
  
    **f = fopen("notas.txt", "w");**  // Paso 2  
    if (f == NULL) {  **// Paso 3**  
        printf("Error abriendo el archivo\n");  
        return 1;  
    }  
  
    for (i=0; i<NESTUDIANTES; i++)  
        **fprintf(f, "%s %d %.2lf\n", notas[i].dni, notas[i].modelo, notas[i].nota);**  // Paso 4  
  
    **fclose(f);**  // Paso 5  
  
    return 0;  
}

**Crear ficheros a partir de otro:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente#include <stdio.h>  
  
#define TAM 16  
  
int main() {  
    FILE \*fEntrada, \*fModelo1, \*fModelo2;  
    char entrada[TAM]="notas.txt", modelo1[TAM]="modelo1.csv", modelo2[TAM]="modelo2.csv", dni[TAM];  
    int modelo;  
    double nota;  
     
    if ((fEntrada = fopen(entrada, "r")) == NULL) {  
        printf("Error abriendo el archivo %s.\n", entrada);  
        return 1;  
    }  
  
    if ((fModelo1 = fopen(modelo1, "w")) == NULL) {  
        printf("Error abriendo el archivo %s.\n", modelo1);  
        fclose(fEntrada);  
        return 1;  
    }  
  
    if ((fModelo2 = fopen(modelo2, "w")) == NULL) {  
        printf("Error abriendo el archivo %s.\n", modelo2);  
        fclose(fEntrada);  
        fclose(fModelo1);  
        return 1;  
    }  
  
    while (fscanf(fEntrada, "%s %d %lf", dni, &modelo, &nota) == 3) {  
        if (modelo == 1)  
            fprintf(fModelo1, "%s,%.2lf\n", dni, nota);  
        else  
            fprintf(fModelo2, "%s,%.2lf\n", dni, nota);  
    }  
  
    fclose(fEntrada);  
    fclose(fModelo1);  
    fclose(fModelo2);  
  
    return 0;  
}