HOJA N°

FECHA

Repaso 2023

1) Linea = posible
L = Nuevo texto
Linea = basura

2) en preprocesador: `for (i=10; i>5; i--){ printf("%d ", i);`
(esto es lo que queda }

↳ MOSTRAR está indefinido de acá para abajo

compilación: 10 9 8 7 6

#define

3) `nPares(x,y) (((x%2)==0)+((y%2)==0))`

4) a) F todos los punteros tienen = valor
b) F tendría que ser *(MEO]+4)
c) V $xor\ 0010\ 0101 = 0111 = 1.2^2 + 1.2^1 + 1.2^0 = 7$
d) F "w+" crea siempre un archivo y se para al principio, "a+" si no se crea el archivo lo crea y sino se para al final
e) F "w+" eliminará el contenido existente si ya existe el archivo, pero si el archivo no existe es lo mismo
f) F en este caso "a" creará el archivo, esto solo sucede con "F" y sus derivados (que retorne NULL)
g) V Fputc('a', stdout) escribe a en el archivo que esta como segundo argumento, en este caso stdout es la salida standard
h) F "a" escribe al final, "a+" lee o escribe al final
i) T feof(file) devuelve num!=0 si llego al final, caso contrario 0
j) T fwrite(&dato, sizeof(tipo_dato), cant_a_escribir, archivo) retorna la cantidad de elementos que se escribieron correctamente al archivo

NOTA

5)

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

struct destino{
    int codProv;
    int codLoc;
    char nomLoc[30];
    int nHab;
};
int main(){
    struct destino d;
    FILE * texto;
    FILE * binario;
    struct destino aux;

texto = Fopen("Habitantes.txt", "r");
binario = Fopen("Habitantes.dat", "wb");
    if ( texto == NULL || binario == NULL){
        printf("No se pudo abrir algun archivo\n");
        return -1;
    }
    else {
        while ( fscanf(texto, "%d %d %s %d", &d.codProv,
        &d.codLoc, d.nomLoc, &d.nHab) == 4 ){
            fwrite( &d, sizeof(struct destino), 1, binario);
        }
fclose (texto);
fclose (binario);
```

NOTA

HOJA N°

FECHA

```
binario = Fopen ("Habitantes.dat", "rb");
if ( binario == NULL){
        printf("No se pudo abrir");
        return -1;
}
   aux.nHab = -1;
   while (fread (&d, sizeof (struct datos), 1, binario)==1){
        if ( d.nHab > aux.nHab){
                aux.nHab = d.nHab;
                aux.codLoc = d.codLoc;
                strcpy (aux.nomLoc, d.nomLoc);
        }
   }
   fclose(binario);
   printf("%s(%d)/n" , aux.nomLoc, aux.codLoc);
   return 0;
}
```

6)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc , char * argv[]){
    if( argc <2){
        printf("error/n");
        return -1;
    }
    char * cadena[200] ="";
    for(int i=1; i<argc; i++){
        strcat(cadena, argv[i]);
        strcat(cadena, " ");
    }
printf("%s", cadena); return 0;
```

NOTA

7)
```
# include <stdio.h>
# include <stdlib.h>
# include <time.h>
typedef int ** matriz;

void reservar (matriz * m, int n);
void inicializar (matriz m, int n);
void imprimir (matriz m, int n);
void liberar (matriz * m, int n);

int main() {
        matriz m;
        int n;
        printf ("Dimension: ");
        scanf ("%d", &n);
        reservar (&m, n);
        inicializar (m, n);
        imprimir (m, n);
        liberar (&m, n);
        return 0;
}
void reservar (matriz *m, int n) {
    *m = (matriz) malloc ( n * sizeof (* int));
    for (int i=0; i<n; i++) {
        (*m)[i] = (int *) malloc ((i+1) * sizeof(int));
    }
}
```

NOTA

HOJA N°

FECHA

```
void inicializar (matriz m, int n){
    srand (time (NULL));
    for (int i = 0; i<n; i++){
        for (int j = 0; j<(i+1); j++){
            m[i][j] = (rand() % 21);
        }
    }
}

void imprimir (matriz m, int n){
    for (int i = 0; i<n; i++){
        for (int j=0; j<(i+1); j++){
            printf("%d ", m[i][j]);
        }
        printf("\n");
    }
}

void liberar (matriz *m, int n){
    for (int i = 0; i<n; i++){
        free((*m)[i]);
    }
    free(*m);
    *m = NULL;
}
```