



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ALUMNO:

JONATHAN SAID GÓMEZ MARBÁN

PROFESOR:

JUAN JESÚS GUTIÉRREZ GARCÍA

FUNDAMENTOS DE PROGRAMACIÓN

PRÁCTICA 2 UNIDAD 3 (P32):

“Matrices”

El propósito de esta práctica es resolver diez distintos problemas, todos ellos relacionados a matrices. En el documento subido a la plataforma moodle, se explica que es lo que se requiere que haga, todo esto se debe hacer apoyándose de los archivos proporcionados por el profesor, los cuales son "matriz.h", "matriz.c" y "usa_mat.c".

Problema 2

Dada una matriz cualquiera realizar una función que extrae en otra matriz cualquier renglón de la matriz original.

Presento el código con el que se ha resuelto este problema:

```
void dos(matriz *m) {
    matriz nueva;
    char salida[100000];
    int renglon = num_aleatorio(m->ren);

    crea_mat(&nueva, 1, m->col);
    for (int i = 0; i < m->col; i++) {
        VAL(&nueva, 0, i) = VAL(m, renglon, i);
    }
    printf("La matriz es:\n%s", acad_mat(salida, &nueva));
}
```

Imagen 1. Función dos.

Se observa que la función dos recibe como parámetros un apuntador a matriz (el tipo matriz es perteneciente al archivo matriz.h); En la primera línea de código se observa que declaramos la variable "nueva" de tipo matriz, posteriormente un arreglo de tipo carácter, al cual, llamamos "salida" y por ultimo a "renglón" el cual es de tipo entero y lo inicializamos, para poder explicar el valor que se le asigna a esta variable mostraré lo que hace la función "num_aleatorio":

```
int num_aleatorio(int num) {
    srand(time(0));
    return (rand() % num);
}
```

Imagen 2. Función num_aleatorio.

Como se observa en la imagen 2, esta función recibe un parámetro de tipo entero, que es nuestro límite superior, ya que, esta función nos regresa un valor de tipo entero de manera aleatoria entre 0 (límite inferior) y el numero que reciba como parámetro.

Una vez explicado esto, retomemos la inicialización de "renglón", nos dice que se le asignara "num_aleatorio(m->ren)", "m->ren" significa que "m" (un apuntador a matriz) esta accediendo al valor de "ren" (el cual es un campo de tipo entero de una estructura que se

llama "matriz"), es decir, que la función num_aleatorio le proporcionará un valor entero al campo "ren", y a su vez, este será asignado a la variable renglon, luego, a partir de la función "crea_mat", la cual nos ayuda a crear una matriz e inicializarla, no sin antes validar que la matriz no esté vacía, que los valores tanto de renglones como de columnas sean positivos y asignar memoria dinámica. Entonces, la función "crea_mat" recibe 3 parámetros, el primero de ellos tiene que ser un apuntador a matriz, y los otros dos son renglones y columnas (de la matriz). Dentro de la función "dos" se usa "crea_mat" y se le pasan los parámetros, el primero de ellos es la primer variable que creamos ("nueva", que es de tipo matriz), el siguiente parámetro es 1 (porque solo queremos extraer un renglón de la matriz original), y por último, se indica que m (nuestro parámetro de la función) va a acceder al campo de "col" (columnas), y este será nuestro tercer parámetro para la función "crea_mat". Luego con un for iteramos desde 0 hasta el valor de las columnas, (recordando que "m->col", quiere decir que "m" está accediendo al campo "col", por tanto, a su valor). En la siguiente línea de código observamos que se usa una macro llamada VAL, dicha macro es perteneciente al archivo "matriz.h" y nos da acceso a la posición i,j de una matriz, al dar acceso a esa posición se puede usar para acceder a su valor pero también para asignarlo; y en esta línea de código asigna el valor de la celda i,j de la matriz original ("m") a la celda 0,i de la nueva matriz ("nueva").

Finalmente vemos que se imprimirá un mensaje y una cadena (debido a la especificación de formato %s), y dentro del printf se encuentra la función "acat_mat"(la cual recibe una cadena, que en nuestro caso es "salida" que nos ayudara a almacenar todos los caracteres y una matriz), esta realiza dos importantes cosas, la primera de ellas, es que convierte la matriz introducida en una cadena y la otra, es que le da tabulaciones a los valores de la matriz (al momento de imprimirla o que aparece en la consola) y le coloca "pipes", es decir, nos ayuda a imprimir la matriz de una forma más estética. Cabe destacar que esta función, "acat_mat", se utiliza al final de todas las funciones que he creado, por lo que ahora solo lo resumiré como: y se usa la función "acat_mat" para imprimir.

Problema 3

Dada una matriz cualquiera realizar una función que extrae en otra matriz cualquier columna de la matriz original.

```
void tres(matriz *m) {
    matriz nueva;
    char salida[100000];
    int columna = num_aleatorio(m->col);

    crea_mat(&nueva, m->ren, 1);
    for (int i = 0; i < m->ren; i++) {
        VAL(&nueva, i, 0) = VAL(m, i, columna);
    }
    printf("La nueva matriz es:\n%s", acat_mat(salida, &nueva));
}
```

Imagen 3. Función tres.

Se observa que la función tres tiene el mismo prototipo que a función dos (ya explicada), una vez ya definiendo la función contemplamos que se crea una variable llamada "nueva" de tipo matriz, y una cadena llamada "salida", estas 2 líneas de código se repiten en todas las funciones que he creado, por lo que, a partir de ahora solo me referiré a ello como: se crea "nueva" y "salida".

En la línea 4 de código de la imagen 3 se observa que de manera análoga al problema 2 se declara y se inicializa la variable "columna", la cual, nos indica que se le asignara "num_aleatorio(m->col)", "m->col" significa que "m" (un apuntador a matriz) está accediendo al valor de "col" (el cual es un campo de tipo entero de una estructura que se llama "matriz"), es decir, que la función num_aleatorio le proporcionará un valor entero al campo "col", y a su vez, este será asignado a la variable "columna". Luego de ello, se crea y se inicializa una matriz (con todo lo explicado anteriormente), pasándole "nueva" (como la matriz que deseo inicializar), "m->ren", la cual me dice que mi matriz original accederá al campo de ren (renglones), y el valor 1 (porque solo quiero 1 columna) con un for iteramos desde 0 hasta el valor de los renglones, (recordando que "m->ren", quiere decir que "m" está accediendo al campo "ren", por tanto, a su valor). Después, con "VAL" asignamos el valor de la celda i,j de la matriz original ("m") a la celda 0,i de la nueva matriz ("nueva") y se usa la función "acat_mat" para imprimir.

Problema 4.

Dada cualquier matriz de cualquier tamaño construir una nueva matriz de tamaño (1×2) que contiene el elemento más pequeño y el más grande de toda la matriz.

```
void cuatro(matriz *m) {
    matriz nueva;
    char salida[100000];
    int minimo = m->datos[0];
    int maximo = m->datos[0];

    for (int i = 0; i < m->ren * m->col; i++) {
        if (m->datos[i] > maximo) {
            maximo = m->datos[i];
        } else if (m->datos[i] < minimo) {
            minimo = m->datos[i];
        }
    }

    crea_mat(&nueva, 1, 2);
    (&nueva)->datos[0] = minimo;
    (&nueva)->datos[1] = maximo;
    printf("La nueva matriz es:\n%s", acad_mat(salida, &nueva));
}
```

Imagen 4. Función cuatro.

Observamos que se crea "nueva" (tipo matriz) y "salida" (tipo cadena). Luego de ello se declaran dos enteros, "minimo" y "maximo" (ambos supondré que es el primer valor que aparece en la matriz), es decir, que mi valor min va a ser el primer valor que aparece en la matriz y a su vez será el máximo. Con un for desde 0 hasta que termine de recorrer toda la matriz ($m \rightarrow ren * m \rightarrow col$) voy comparando tanto la variable "máximo" como la variable "minimo" con el valor de la posición de i ($m \rightarrow datos[i]$), la cual, me dice que la matriz original está accediendo al valor del campo datos en la posición i), si el valor de la posición i resulta ser mayor que la variable "maximo", entonces "máximo" tomara el valor de i en esa posición y si el valor de la posición i resulta ser menor que la variable "minimo", entonces "minimo" tomara el valor de i en esa posición. Una vez que el for concluya, se utilizará la función "crea_mat" y se le pasará "nueva", el valor 1 (referente a un renglón) y 2 (referente a 2 columnas), como nos indica el problema, luego asignamos a la matriz "nueva" en la posición 0 el valor de "minimo" y en la posición 1 el valor de "maximo" y se usa la función "acat_mat" para imprimir.

Problema 5.

Dada cualquier matriz de cualquier tamaño construir una nueva matriz que sea de una sola columna y contiene el elemento mínimo de cada renglón

```
void cinco(matriz *m) {
    matriz nueva;
    char salida[100000];
    crea_mat(&nueva, m->ren, 1);
    for (int i = 0; i < m->ren; i++) {
        int minimo = VAL(m, i, 0);
        for (int j = 0; j < m->col; j++) {
            int valor = VAL(m, i, j);
            if (valor < minimo) {
                minimo = valor;
            }
        }
        VAL(&nueva, i, 0) = minimo;
    }
    printf("La nueva matriz es:\n%s", acad_mat(salida, &nueva));
}
```

Imagen 5. Función cinco.

Observamos que se crea "nueva" (tipo matriz) y "salida" (tipo cadena). Después, se utilizará la función "crea_mat" y se le pasará "nueva", el valor de renglones ($m \rightarrow ren$) y 1 (referente a 1 columna). Posteriormente con un for se iterará desde 0 hasta el último renglón ($m \rightarrow ren$, el valor de renglones), se crea la variable tipo entero "minimo" (el cual vale datos en la posición 0), la cual se le asigna VAL a la cual le pasamos la matriz original (el vector datos en la posición i), i (renglones) y j (columnas) que vale 0. Dentro de este for pongo otro for que

recorrerá las columnas del renglón i y mientras hace eso, pondrá dentro de la variable "valor", los valores que se vayan recorriendo, cada vez que tenga un nuevo valor, lo compara con "minimo" y si "valor" es menor que "minimo", entonces "minimo" se le asignará el valor de la variable "valor". Cuando acabe de recorrer el primer renglón habrá obtenido el valor mínimo que tiene dicho renglón y VAL se encargara de asignárselo a la nueva matriz llamada "nueva" en la posición i del renglón (en este caso 0) y el valor de j en 0 (porque no queremos que haya más de una columna). Este proceso se repite hasta que termine de recorrer todos los renglones y me de todos los valores mínimos obtenidos de cada renglón, con ello se formara mi nueva matriz y con "acad_mat" se convierte a cadena y se acomoda de una forma estética y con printf se imprime.

Problema 6.

Dada una matriz cualquiera y una posición dentro de la matriz crear un vector (matriz de tamaño $(1 \times n)$) que contenga todos los elementos de la diagonal a la que corresponde esa coordenada.

```
void seis(matriz *m, int renglon, int columna) {
    matriz nueva;
    char salida[100000];

    int renglones;
    if (renglon == columna)
    {
        renglones = m->ren;
    }
    else if (renglon > columna)
    {
        renglones = m->ren - renglon + columna;
    }
    else
    {
        renglones = m->col - columna + renglon;
    }

    crea_mat(&nueva, renglones, 1);
    int posicion_nueva_matriz = 0;
    while (columna != 0 && renglon != 0)
    {
        columna--;
        renglon--;
    }

    while (posicion_nueva_matriz < renglones)
    {
        VAL(&nueva, posicion_nueva_matriz, 0) = VAL(m, renglon, columna);
        posicion_nueva_matriz++;
        renglon++;
        columna++;
    }
    printf("La nueva matriz es:\n%s\n", acad_mat(salida, &nueva));
}
```

Imagen 6. Función seis.

En esta función los parametros que se reciben es la matriz con la que se trabajara (original) denominada "m" y dos enteros los cuales son "renglon" y "columna". Se crea una matriz llamada "nueva" y un arreglo de tipo char, llamado "salida". Luego se declara un entero "renglones". Esta variable nos servirá para asignarle un valor dependiendo de cual condición que se presenta en imagen 6 es verdadera. Si sucede el primer caso "renglones" se le va asignar el valor de renglones, es decir, que el valor de los renglones de la nueva matriz será igual al numero de renglones en la matriz origina; Si sucede el segundo caso, es que el renglón sea mayor que la columna, el numero de renglones totales menos el renglon en el que se encuentra más la columna en la que se encuentra; para el else es un caso similar que el segundo, solo que se usa las columnas totales menos la columna en la que se encuentra mas el renglon en el que te encuentras. (Llego a estos tres casos despues de hacer varias pruebas con varias matrices). Se crea una matriz de una sola columna, y del número de renglones que se determinó. El primer while me sirve para restarle tanto a columna como a renglon, y saber la primera posicion de la diagonal, cuando uno de esos dos parametros llegue a cero, significa que la posicion esta en algún borde de la matriz. La variable posicion_nueva_matriz me sirve para saber la posicion del renglon de mi nueva matriz e irla comparando con los la variable "renglones", que fue la que determine con los if, y asi se arma la nueva matriz (con ayuda del while y de VAL), por ultimo se imprime la matriz.

Problema 7.

Dada una matriz cualquiera extraer otra sub matriz que está entre dos coordenadas dadas.

```
void siete(matriz *m, int renglon1, int columna1, int renglon2, int columna2) {
    matriz nueva;
    char salida[100000];

    int renglon_inicial = renglon1;
    int renglon_final = renglon2;
    if (renglon2 < renglon1)
    {
        renglon_inicial = renglon2;
        renglon_final = renglon1;
    }

    int columna_inicial = columna1;
    int columna_final = columna2;
    if (columna2 < columna1)
    {
        columna_inicial = columna2;
        columna_final = columna1;
    }

    crea_mat(&nueva, renglon_final - renglon_inicial + 1, columna_final - columna_inicial + 1);
    for (int i = 0; i < renglon_final - renglon_inicial + 1; i++)
    {
        for (int j = 0; j < columna_final - columna_inicial + 1; j++)
        {
            VAL(&nueva, i, j) = VAL(m, i + renglon_inicial, j + columna_inicial);
        }
    }
    printf("La nueva matriz es:\n%s\n", acad_mat(salida, &nueva));
}
```

Imagen 7. Función siete.

Como se logra observar, siete tiene mas parámetros que otras funciones que hemos visto. Esto es debido a que, además de la matriz "m", se introducen el renglon1 y columna1 (que representan mi primera coordenada), que por lo tanto, el renglon2 y columna2 (representan mi segunda coordenada). Se observa que como en todas las funciones anteriores se crea "nueva" (de tipo matriz) y "salida" (tipo cadena). Las siguientes 15 líneas de código necesitan una explicación acerca de su razonamiento, así que mostrare una imagen mas detallada sobre esto:

```

int renglon_inicial = renglon1;
int renglon_final = renglon2;
if (renglon2 < renglon1)
{
    renglon_inicial = renglon2;
    renglon_final = renglon1;
}

int columna_inicial = columna1;
int columna_final = columna2;
if (columna2 < columna1)
{
    columna_inicial = columna2;
    columna_final = columna1;
}

```

Imagen 7.1. Función siete.

Notamos que se declaran 2 variables tipo entero, llamadas "renglon_inicial" que se inicializa con "renglon1" y "renglon_final" que se inicializa con "renglon2". Después, comparamos si renglon2 es menor que renglon1 entonces intercambiar los valores entre "renglon_inicial" y "renglon_final". Se hace lo mismo con las columnas. Esto es debido a que se pueden introducir las coordenadas (1,3,0,3) y suponiendo que nuestra matriz es la siguiente:

$$= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 0 \\ 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 0 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

Imagen 7.2. Matriz ejemplo.

Con las coordenadas dadas nuestra nueva matriz quedaría de la siguiente forma:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 6 & 7 & 8 & 9 \end{pmatrix}$$

Imagen 7.3. Matriz nueva ejemplo.

Sin embargo, el programa no podría extraer la nueva matriz debido a que el renglon1 es mayor que el renglon 2 y esto no permitiría extraerla, lo mismo pasaría con las columnas, es por eso que primero se comparan entre si y se saca al menor de ellos.

Continuando con la imagen 7 y una vez con la explicación dada utilizamos "crea_mat", (pasamos los valores que se ven en la imagen), y ahora, ¿Por qué se le esta sumando uno al

valor de los renglones y al valor de las columnas?, esto es debido a, y romiendo el ejemplo anterior, si se resta (1-0) para el caso de los renglones, me daría como resultado 1, cuando realmente se necesitan extraer 2 renglones, como se muestra en 7.3 y lo mismo sucede con las columnas. Finalmente, los 2 for anidados y la macro VAL, me ayudaran a "construir" la nueva matriz, que finalmente se imprimirá.

Problema 8.

Dada una matriz cualquiera crear otra que es de dimensión menor en uno, esto es si la original es de $(n_1 \times m_1)$ la respuesta será de $(n_1 - 1 \times m_1 - 1)$ que contenga los mismos datos excepto un renglón y una columna dada.

```
void ocho(matriz *m, int renglon, int columna)
{
    matriz nueva;
    char salida[100000];
    crea_mat(&nueva, m->ren - 1, m->col - 1);
    int nueva_i = 0;
    int nueva_j;
    for (int i = 0; i < m->ren; i++)
    {
        if (renglon == i)
        {
            continue;
        }
        nueva_j = 0;
        for (int j = 0; j < m->col; j++)
        {
            if (columna == j)
            {
                continue;
            }
            VAL(&nueva, nueva_i, nueva_j) = VAL(m, i, j);
            nueva_j++;
        }
        nueva_i++;
    }
    printf("La nueva matriz es:\n%s\n", acad_mat(salida, &nueva));
}
```

Imagen 8. Función 8

Se crea una nueva matriz (van a tener una columna y renglón menos), Se mantien 2 contadores diferentes para la nueva matriz, porque debido a que se borra una columna y una fila, los contadores para la matriz original seran diferentes a los de la nueva, dentro del for hay un si, el cual me indica, si el renglon es igual a i entonces se continua (te regresa al ciclo), hasta que llegas al renglon del que quieres

deshacer, en el ciclo mas anidado se itera sobre las columnas y si se descubre que j es igual a la columna, entonces se salta, ayudado de VAL asigno los valores de mi nueva matriz e imprimo la nueva matriz

Problema 9.

Dada una matriz cualquiera que se calcule la transpuesta.

```
void nueve(matriz *m)
{
    matriz nueva;
    char salida[100000];
    crea_mat(&nueva, m->col, m->ren);
    for (int i = 0; i < m->ren; i++)
    {
        for (int j = 0; j < m->col; j++)
        {
            VAL(&nueva, j, i) = VAL(m, i, j);
        }
    }
    printf("La nueva matriz es:\n%s\n", acad_mat(salida, &nueva));
}
```

Imagen 9. Función nueve.

Se crea la matriz y la cadena, con nombres “nueva” y “salida” respectivamente. Con la función “crea_mat” se inicializan los valores de la matriz nueva (notese que el valor de renglones esta donde debería ir el de columnas y el de columnas donde debería ir el de renglones). Posteriormente con 2 for distintos y anidados uno que recorre renglones y otro que recorre columnas, apoyados de la función VAL se van colocando los valores (ya traspuestos), debido a como se introdujeron los parámetros en “crea_mat”. Posteriormente solo se procede a imprimir la matriz, también apoyándonos de “acad_mat”

Problema 10.

Dada una matriz cualquiera y una coordenada construir un vector con el mismo número de renglones que la matriz original que siga el salto de un caballo de ajedrez (siempre dos hacia abajo uno a la derecha en este caso).

Dado que ya casi es hora de entregar la práctica, lamentablemente no puedo continuar y terminarla como lo deseo, de los 10 problemas que se proponen, resolví 9, solo me falto el numero 1.