Informática-Ingeniería en Mecatrónica-2022
Trabajo Práctico Nº 3 - A

Parte 1 – Estructuras de datos y cadenas

Borquez Juan. Legajo:13567

1. Escriba un programa que:
   a. Genere un vector de tamaño N con valores aleatorios enteros, utilizando la función rand(). Use la función srand() para obtener una mejor aleatoriedad.
   b. Recorra el vector y guarde el promedio de los valores en una variable
   c. Encuentre el menor de los elementos

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

int main(void) {
	short int max = 100, maxval = 10, N;
	int vector[max], minimo = maxval + 1;
	float promedio = 0;

	srand(time(NULL));
	printf("TP3-P1-EJ1:\n\n");
	printf("El tamaño del vector es %d. A continuacion indique el tamaño N a utilizar: ", max);
	scanf("%hd", &N);
	srand(time(NULL));
	for (short int i = 0; (i < N) && (i < max); i++){
		vector[i] = rand()%(maxval + 1);
		promedio += vector[i];
		if (vector[i] < minimo){
			minimo = vector[i];
		}
		printf("VECTOR[%hd]: %d\n", i, vector[i]);
	}
	for (short int i = 0; (i < N) && (i < max); i++){
		if (i == 0){
			minimo = vector[i];
		}
		else{
			if (vector[i] < minimo){
				minimo = vector[i];
			}
		}

	}
	promedio/=(float)N;
	printf("\nEL promedio de los numeros del vector es: %f", promedio);
	printf("\nEl valor minimo es: %d\n", minimo);
 return 0;
```

```
38  }
```

https://replit.com/@JuanBorquez/TP3-P1-EJ1#main.c

2. Implementar una función que genere una matriz con todos los elementos en 0 excepto aquellos para los que i+j sea par; para estos elementos generar un valor aleatorio entre 1 y 10000.

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <time.h>
4   #define size 10
5
6   void cargar(int matriz[0][size], short int f, short int c);
7   void imprimir(int matriz[0][size], short int f, short int c);
8
9   int main(void) {
10          int matriz[size][size];
11          short int f, c;
12          printf("TP3-PARTE1-EJERCICIO 2");
13          printf("\n\nLa matriz es de numeros enteros de %d por %d", size, size);
14          printf("\nA continuacion indique las cantidad de filas f: "); scanf("%hd", &f);
15          printf("A continuacion indique las cantidad de columnas c: "); scanf("%hd", &c);
16          cargar(matriz, f, c);
17          imprimir(matriz, f, c);
18    return 0;
19  }
20  void cargar(int matriz[][size], short int f, short int c){
21          srand(time(NULL));
22          for (short int j = 0; (j < c) && (j < size); j++){
23                  for (short int i = 0; (i < f) && (i < size); i++){
24                          matriz[i][j] = ((i + j)%2)? 0:(1 + rand()%1000);
25                  }
26          }
27  }
28  void imprimir(int matriz[0][size], short int f, short int c){
29          for (short int i = 0; (i < f) && (i < 10); i++){
30                  printf("\n");
31                  for (short int j = 0; (j < c) && (j < 10); j++){
32                          printf("\t\t[%hd][%hd]: %d", i, j, matriz[i][j]);
33                  }
34          }
35  }
```

https://replit.com/@JuanBorquez/TP3-P1-EJ2#main.c

3. Escriba un programa que lea una cadena con una operación matemática básica de dos términos (suma, resta, multiplicación y división) y calcule el resultado. Por ej. el

usuario ingresa “2.4+3”, el resultado a devolver es “5.4”. Considere los datos de entrada números flotantes. Considere una cadena máxima de 100 caracteres.

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

int main(void){
	float op1, op2, resultado;
	char operador;
	printf("TP3-PARTE1-EJERCICIO 3\n");
	printf("\nA continuacion indicar una operacion aritmetica basica (+,-,/,*) con 2 operandos racionales:\n\n");
	printf("\t\tPor ejemplo: 2.3+5 (observar que no se colocan espacios)\n");
	printf("\nLa cantidad maxima de caracteres a ingresar es de 100\n");
	printf("\n\tOPERACION:\t");
	scanf("%f%c%f", &op1, &operador, &op2);
	/*printf("\nop1: %f", op1);
	printf("\nop2: %f", op2);
	printf("\noperador: %c", operador);*/
	switch (operador){
		case '+':
			resultado = op1 + op2;
			printf("\nResultado: %f", resultado);break;
		case '-':
			resultado = op1 - op2;
			printf("\nResultado: %f", resultado);break;
		case '/':
			resultado = op1/op2;
			printf("\nResultado: %f", resultado);break;
		case '*':
			resultado = op1*op2;
			printf("\nResultado: %f", resultado);break;
		default:
			printf("No se ingreso un caracter de operación valido");
	}
}
```

https://replit.com/@JuanBorquez/TP3-P1-EJ3#main.c

4. Escriba un programa que lea una frase y evalúe en forma recursiva si la misma corresponde a un palíndromo. Considere una cadena máxima de 100 caracteres.

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

/*La siguiente es una función recursiva que determina si una cadena de caracteres es palíndromo

-En el primer llamado recibe un puntero a la cadena de caracteres (word) de origen y el tamaño de la misma
-En los llamados recursivos el puntero word es en realidad un puntero a una posición en la cadena de caracteres de origen
De este modo no hay que crear varias cadenas. Es decir que siempre se trabaja con la cadena de origen
- size_of_word es el tamaño de la cadena de origen cuando se llama la función por prrimera vez
- size_of_word se interpreta como la longitud de la porción de la cadena de origen que se toma en
cada llamado recursivo a la función.
- Si la cadena es un palíndromo devuelve 1 y devuelve 0 si no lo es*/
short int is_palindrome(char *word, short int size_of_word);
void no_space(char* word);

int main(void){
	char word[100];
	printf("TP3-PARTE1-EJERCICIO 4\n\n");
	printf("A continuacion indique una palabra de no mas de 100 caracteres:\n");
	printf("\nPALABRA: \t");
	scanf("%[^\n]", word);
	no_space(word);
	if(is_palindrome(word, strlen(word))){
		printf("\nLa frase SI es un palindromo\n");
	}
	else{
		printf("\nLa palabra NO es un palindromo\n");
	}
}
short int is_palindrome(char *word, short int size_of_word){
	if (size_of_word > 1){
		//Si el tamaño de la porción considerada es mayor a 1 hacemos otra determinacion
		if(word[0] == word[size_of_word-1]){//compara el pultimo y primer carácter de la porción considerada
			//si son iguales hace otro llamado a la función
			return(is_palindrome(word + 1, size_of_word - 2));
			//word+1 será un puntero al caracter posterior al primero en la porción de cadena considerada
			//Se restan dos caracteres a la porción en size_of_word - 2;
		}
		else{
			//si en cierto momento el caracter inicial y final no coinciden entonces no es palindromo
			return 0;
		}
	}
	else{
		//si el tamaño de la porción de cadena considerada es 0 o 1 significa que la palabra sí es palindromo
		//Esto incluso cuando la cadena de origen tiene un solo caracter o ninguno
		return 1;
	}
}
void no_space(char* word){
```

```
55          short int i = 0, k = 0;
56          while(word[i] != '\0'){
57                  if (word[i] != ' '){
58                          if (k > 0){
59                                  word[i-k] = word[i];
60                          }
61                  }
62                  else{
63                          k++;
64                  }
65                  i++;
66          }
67          if(k > 0){
                    word[i-k] = '\0';
            }
    }
```

https://replit.com/@JuanBorquez/TP3-P1-EJ4#main.c

5. Modifique el ejercicio 12 del práctico anterior (el de las fechas) utilizando un struct que represente una fecha completa. Realice al menos una función que reciba por parámetro una fecha. Defina la estructura fecha como un tipo definido por el usuario (typedef)

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

typedef struct{
	short int DD;
	short int MM;
	short int AAAA;
} date;
//devuelve la fecha anterior
date previous(date actual);

//devuelve la fecha posterior
date next(date actual);

short int is_leap(date actual);
short int days_month(date actual);
short int is_valid(date actual);

int main(void){
	date actual, prev, nex;
	char c;
	short int daysMM;
	printf("TP3-PARTE1-EJERCICIO 5");
	printf("\n\nA continuacion indicar una fecha con el formato DD/MM/AAAA\n");
	printf("\nSi indica una fecha incorrecta se volvera a pedir la fecha\n");
	printf("Presione una tecla: "); c = getchar();
	do{
		system("clear");
		do{
			system("clear");
			printf("\nDD:\t");scanf("%2hd", &actual.DD);
		}while((actual.DD < 1)||(actual.DD > 31));
		do{
			system("clear");
			printf("\nMM:\t");scanf("%2hd", &actual.MM);
		}while((actual.MM < 1)||(actual.MM > 12));
		system("clear");
		printf("\nAAAA:\t");scanf("%4hd", &actual.AAAA);
	}while(!is_valid(actual));
	system("clear");
	prev = previous(actual);
	nex = next(actual);
	daysMM = days_month(actual);

	printf("\nFecha actual: \t%hd/%hd/%hd", actual.DD, actual.MM, actual.AAAA);
	printf("\nFecha anterior: \t%hd/%hd/%hd", prev.DD, prev.MM, prev.AAAA);
	printf("\nFecha siguiente: \t%hd/%hd/%hd", nex.DD, nex.MM, nex.AAAA);
	printf("\nUltimo dia del mes: \t%hd/%hd/%hd", daysMM, actual.MM, actual.AAAA);
	printf("\nDias para que termine el mes: \t%d\n\n", daysMM-actual.DD);
```

```
50
51  }
52  date previous(date actual){
53          date previous;
54          if((actual.DD - 1) != 0){
55                  previous.DD = actual.DD-1;
56                  previous.MM = actual.MM;
57                  previous.AAAA = actual.AAAA;
58          }
59          else{
60                  previous.MM = actual.MM - 1; previous.AAAA = actual.AAAA;
61                  previous.DD = days_month(previous);
62                  previous.MM = ((actual.MM-1) != 0)? actual.MM - 1:12;
63                  previous.AAAA = ((actual.MM-1) != 0)? actual.AAAA:actual.AAAA-1;
64          }
65          return (previous);
66  }
67  date next(date actual){
68          date next;
69          if(actual.DD < days_month(actual)){
70                  next.DD = actual.DD + 1;
71                  next.MM = actual.MM;
72                  next.AAAA = actual.AAAA;
73          }
74          else{
75                  next.DD = 1;
76                  next.MM = (actual.MM != 12)? actual.MM + 1:1;
77                  next.AAAA = (actual.MM != 12)? actual.AAAA:actual.AAAA+1;
78          }
79          return (next);
80  }
81  short int days_month(date actual){
82          if (actual.MM != 2){
83                  return((((actual.MM%2)&&(actual.MM <= 7))||(!(actual.MM%2)&&(actual.MM >
84  7))||(actual.MM == 0))? 31:30);
85          }
86          else{
87                  return((is_leap(actual))?29:28);
88          }
89  }
90  short int is_leap(date actual){
91          // es bisiesto si es divisible por 4 y no por 100 o si es divisile por 400
92          return(((!(actual.AAAA%4)&&(actual.AAAA%100))||!(actual.AAAA%400))? 1:0);
93  }
94  short int is_valid(date actual){
95          return((actual.DD <= days_month(actual))? 1:0);
    }
```

https://replit.com/@JuanBorquez/TP3-P1-EJ5#main.c

6. Escriba un programa que calcule el producto vectorial de 2 vectores en R3 utilizando structs pare representar los vectores

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

typedef struct{
        float xx;
        float yy;
        float zz;
}vector;

//calcula el producto cruz de dos vectores en R3
vector cruz(vector u, vector v);

int main(void) {
        vector u,v,w;
  printf("TP3-P1-EJERCICIO6");
        printf("\nIndique las componentes del primer vector: ");
        printf("\n\tu_xx = ");scanf("%f", &u.xx);
        printf("\tu_yy = ");scanf("%f", &u.yy);
        printf("\tu_zz = ");scanf("%f", &u.zz);
        printf("\nIndique las componentes del segundo vector: ");
        printf("\n\tv_xx = ");scanf("%f", &v.xx);
        printf("\tv_yy = ");scanf("%f", &v.yy);
        printf("\tv_zz = ");scanf("%f", &v.zz);
        w = cruz(u, v);
        system("clear");
        printf("\n\nu = [%f, %f, %f]", u.xx, u.yy, u.zz);
        printf("\nv = [%f, %f, %f]", v.xx, v.yy, v.zz);
        printf("\n\nW = u X v = [%f, %f, %f]\n", w.xx, w.yy, w.zz);
  return 0;
}
vector cruz(vector u, vector v){
        vector w;
        w.xx = u.yy*v.zz - v.yy*u.zz;
        w.yy = -u.xx*v.zz + v.xx*u.zz;
        w.zz = u.xx*v.yy - v.xx*u.yy;
        return w;
}
```

https://replit.com/@JuanBorquez/TP3-P1-EJ6#main.c

Parte 2 - Apuntadores, memoria dinámica y estructuras dinámicas

7. Implemente un programa que: (a) declare una variable v de tipo int; (b) llame a una función void duplicar(int *v) pasándole como parámetro la dirección de memoria de la variable v (mediante &v); (c) la función debe multiplicar v por 2 y colocar el resultado en la misma dirección de memoria, de manera que el programa main "vea" el cambio de valor; (d) imprima por pantalla el valor de la variable v para verificar que el programa funciona correctamente.

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

void duplicar(int *v);

int main(void) {
	srand(time(NULL));
	int v = rand()%101;;
	printf("v (antes de duplicar) = %d", v);
	duplicar(&v);
	printf("\nv (despues de duplicar) = %d\n", v);
}

void duplicar(int *v){
	*v = 2*(*v);
}
```

https://replit.com/@JuanBorquez/TP3-P2-EJ7#main.c

8. Escriba una función que reciba como argumento un entero N, cree un vector de N elementos de tipo double dinámicamente (utilizando la función malloc), y devuelva un apuntador con la dirección de memoria del arreglo creado.

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

double *crear(short int N);

int main(void) {
        printf("PUNTO 8: TP3-PARTE2");
        srand(time(NULL));
        short int N = rand()%101;
        double *vector = crear(N);
        if (vector != NULL){
                printf("\n\nDireccion apuntada: %p\n", vector);
        }
        else{
                printf("\n\nHubo un problema al asignar la memoria\n");
        }
        free(vector);
  return 0;
}
double *crear(short int N){
        return ((double*)malloc(N*sizeof(double)));
}
```

https://replit.com/@JuanBorquez/TP3-P2-EJ8#main.c

9. Escriba una función que reciba un vector de N elementos double (donde N puede ser variable) y un escalar double s, y escale el mismo arreglo utilizando el factor s (es decir, no debe devolver un puntero ni crear otro arreglo).

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

#define max_val 10
#define max_size 10

/*crea un vector de m elementos*/
double* create_vec(short int m);

/*crea y carga de forma aleatoria un vector con m elementos*/
/*con el máximo valor limitado por la constante max_val*/
double* rand_vec(short int m);

/*funcion para escalar el vector vector*/
/*voy a hacer que el puntero del vector apunte a otra dirección y por eso lo paso por referencia*/
/*m es la cantidad de posiciones original y s es el factor de escala*/
/*también se pasa por referencia porque va a cambiar el tamaño del vector*/
/*si el vector resulta de mayor tamaño lo completa con valores aleatorios*/
void scale_vec(double** vec, short int* m, double s);

/*para mostrar el vector*/
void mostrar(double* vec, short int m);

int main(void) {
	srand(time(NULL));

	/*la siguiente son la cantidad de elementos del vector*/
	short int m = 1 + rand()%max_size;
	/*el siguiente es el puntero para el vector y el segundo es el factor de escala*/
  double* vec = rand_vec(m), s;

	printf("\nVECTOR RANDOM ORIGINAL DE %d ELEMENTOS\n", m);
	mostrar(vec, m);

	printf("\nINDICAR UN FACTOR DE ESCALA s: ");scanf("%lf", &s);

	/*aplica el factor de escala al vector*/
	/*recordando que en la función se cambia el valor de m para indicar el nuevo tamaño del
vector*/
	scale_vec(&vec, &m, s);

	printf("\nVECTOR RANDOM CON FACTOR DE ESCALA, DE %d ELEMENTOS\n", m);
	mostrar(vec, m);

	free(vec);
  return 0;
}
double* create_vec(short int m){
```

```
50          return((double*)malloc(m*sizeof(double)));
51      }
52      double* rand_vec(short int m){
53          double* vec = create_vec(m);
54          for (short int i = 0; i < m; i++){
55              vec[i] = rand()*(double)max_val/RAND_MAX;
56          }
57          return vec;
58      }
59      void scale_vec(double** vec, short int* m, double s){
60          /*un puntero a double auxiliar para almacenar el resultado de realloc*/
61          double* vec_aux;
62          short int m_aux;
63
64          /*solo me quiero quedar con la parte entera dada por el producto entre el tamaño actual
65      y el factor de escala s*/
66          /*de la forma en que se hace a continuación siempre se redondea por defecto porque me
67      quedo con la parte entero y no con el entero mas cercano*/
68          /*el tamaño se reserva en la variable auxiliar para el tamaño*/
69          m_aux = (*m)*s;
70
71          /*aplico el factor de escala para redimensionar el espacio de memoria asociado a vec*/
72          vec_aux = realloc(*vec, m_aux*sizeof(double));
73
74          /*ejecutamos lo siguiente solo cuando siendo m_aux no nulo vec_aux es no nulo. Lo que
75      nos indica que
76      realloc se ejecuto bien. O cuando m_aux es cero, en cuyo caso también es vec_aux nulo*/
77
78          if(((vec_aux != NULL)&&m_aux)||!m_aux){
79              *vec = vec_aux;
80
81              /*en caso de que el vector resulte en una cantidad mayor de elementos lo completo
82      con valores random*/
83              if (m_aux > *m){
84                  for(short int i = *m; i < m_aux; i++){
85                      (*vec)[i] = rand()*(double)max_val/RAND_MAX;
86                  }
87              }
88
89              *m = m_aux;
90          }
91          else{
92              printf("\nNO FUNCIONO REALLOC");
93          }
94      }
95      void mostrar(double* vec, short int m){
96          printf("\n");
97          for(short int i = 0; i < m; i++){
98              printf("\t%3.2lf ", vec[i]);
```

```
        }
        printf("\n");
}
```

https://replit.com/@JuanBorquez/TP3-P2-EJ9#main.c

10. Escriba un programa que, reutilizando las dos funciones anteriores.

a. lea por teclado un entero N,
b. cree 2 vectores de N elementos double dinámicamente con valores aleatorios,
c. lea por teclado un factor de escala double
d. escale 2 de los vectores generados en (b) por el factor
e. muestre todos los resultados
f. Libere la memoria dinámica utilizada

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

#define max_val 10

double* create_vec(short int m);

/*crea y carga de forma aleatoria un vector con m elementos*/
/*con el máximo valor limitado por la constante max_val*/
double* rand_vec(short int m);

/*funcion para escalar el vector vector*/
/*voy a hacer que el puntero del vector apunte a otra dirección y por eso lo paso por referencia*/
/*m es la cantidad de posiciones original y s es el factor de escala*/
/*también se pasa por referencia porque va a cambiar el tamaño del vector*/
/*si el vector resulta de mayor tamaño lo completa con valores aleatorios*/
void scale_vec(double** vec, short int* m, double s);

/*para mostrar el vector*/
void mostrar(double* vec, short int m);

int main(void) {
	srand(time(NULL));
	short int N, aux;
	double *vec1, *vec2, s;
  printf("INDIQUE UN ENTERO N: ");scanf("%hd", &N);
	printf("INDIQUE UN FACTOR DE ESCALA s: "); scanf("%lf", &s);
	system("clear");

	/*creo los vectores con la dimensión indicada de forma aleatoria*/
	vec1 = rand_vec(N);
	vec2 = rand_vec(N);

	/*muestro por pantalla los vectores cargados*/
	printf("\nVECTOR 1 ANTES DE LA ESCALA\n");
	mostrar(vec1, N);

	printf("\nVECTOR 2 ANTES DE LA ESCALA\n");
	mostrar(vec2, N);

	/*como la función de escala va a modificar el valor de N que luego lo tengo que utilizar
para escalar el
segundo vector, lo guardo en una variable auxziliar*/
	aux = N;
	/*escalo el primer vector*/
	scale_vec(&vec1, &N, s);
	/*devuelvo el valor de N*/
	N = aux;
```

```
50          /*escalo el segundo vector*/
51          scale_vec(&vec2, &N, s);
52          /*el valor de N ya esta escalado y es el mismo para los dos vectores*/
53
54          /*muestro por pantalla los vectores cargados luego de la escala*/
55          printf("\nVECTOR 1 DESPUES DE LA ESCALA\n");
56          mostrar(vec1, N);
57
58          printf("\nVECTOR 2 DESPUES DE LA ESCALA\n");
59          mostrar(vec2, N);
60
61          free(vec1);free(vec2);
62    return 0;
63  }
64  double* create_vec(short int m){
65          return((double*)malloc(m*sizeof(double)));
66  }
67  double* rand_vec(short int m){
68          double* vec = create_vec(m);
69          for (short int i = 0; i < m; i++){
70                  vec[i] = rand()*(double)max_val/RAND_MAX;
71          }
72          return vec;
73  }
74  void scale_vec(double** vec, short int* m, double s){
75          /*un puntero a double auxiliar para almacenar el resultado de realloc*/
76          double* vec_aux;
77          short int m_aux;
78
79          /*solo me quiero quedar con la parte entera dada por el producto entre el tamaño actual
80  y el factor de escala s*/
81          /*de la forma en que se hace a continuación siempre se redondea por defecto porque me
82  quedo con la parte entero y no con el entero mas cercano*/
83          /*el tamaño se reserva en la variable auxiliar para el tamaño*/
84          m_aux = (*m)*s;
85
86          /*aplico el factor de escala para redimensionar el espacio de memoria asociado a vec*/
87          vec_aux = realloc(*vec, m_aux*sizeof(double));
88
89          /*ejecutamos lo siguiente solo cuando siendo m_aux no nulo vec_aux es no nulo. Lo que
90  nos indica que
91  realloc se ejecuto bien. O cuando m_aux es cero, en cuyo caso también es vec_aux nulo*/
92
93          if(((vec_aux != NULL)&&m_aux)||!m_aux){
94                  *vec = vec_aux;
95
96                  /*en caso de que el vector resulte en una cantidad mayor de elementos lo completo
97  con valores random*/
98                  if (m_aux > *m){
```

```
99                      for(short int i = *m; i < m_aux; i++){
100                             (*vec)[i] = rand()*(double)max_val/RAND_MAX;
101                     }
102             }
103
104             *m = m_aux;
105     }
106     else{
107             printf("\nNO FUNCIONO REALLOC");
108     }
109 }
110 void mostrar(double* vec, short int m){
111     printf("\n");
112     for(short int i = 0; i < m; i++){
113             printf("\t%3.2lf ", vec[i]);
        }
        printf("\n");
}
```

https://replit.com/@JuanBorquez/TP3-P2-EJ10#main.c

11. Escriba una función que reciba como parámetros 2 matrices A y B de MxN y NxP elementos tipo double, respectivamente, cree una nueva matriz C de MxP elementos dinámicamente, y calcule (almacenando el resultado en C) el producto matricial. La función debe devover el apuntador a C (que fue creada en la misma función). Valide los tamaños de las matrices para que se pueda realizar la operación.

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

/*constantes simbolicas para el tope de rand y para el maximo de filas y columnas respectivamente*/
#define max_val 10
#define max_size 10

double** create_mat(short int m, short int n);

/*Crea y carga de forma aleatoria una matriz de mxn con números desde cero a max_val*/
double** rand_mat(short int m, short int n);

/*hace el producto matricial de dos matrices A y B*/
/*devuelve un puntero nulo si ocurre un problema o no se puede realizar la operación*/
/*Recibe como parametros los punterso a las matrices y las dimensiones de cada una de ellas*/
double** producto(double** A, short int ma, short int na, double** B, short int mb, short int
nb);

/*imprimer el contenido de la matri*/
void mostrar(double** mat, short int m, short int n);

/*libera todo el espacio asociado a la matriz mat*/
/*notar que no hace falta indicar la cantidad de columnas*/
void free_mat(double** mat, short int m);

int main(void) {
	srand(time(NULL));

	/*la cantidad de filas y columnas es mayor a 0 y menor o igual a maxsize*/
	short int m = 1 + rand()%max_size;
	short int n = 1 + rand()%max_size;
	short int p = 1 + rand()%max_size;

	/*Creo y cargo la matriz A*/
	double **A = rand_mat(m, n);

	/*Creo y cargo la matriz B*/
	double **B = rand_mat(n, p);

	/*Calcula el producto matricial de A Y B*/
	double **C = producto(A, m, n, B, n, p);

	printf("Matriz A:\n");
	mostrar(A, m, n);
	printf("\n\nMatriz B:\n");
	mostrar(B, n, p);
	if(C != NULL){
		printf("\n\nMatriz C:\n");
```

```
50                  mostrar(C, m, p);
51                  free_mat(C, m);
52          }
53          else{
54                  printf("\n\nLas dimensiones de las matrices A y B no son compatibles para el
55  producto\n");
56          }
57          free_mat(A, m);
58          free_mat(B, n);
59    return 0;
60  }
61  double** create_mat(short int m, short int n){
62          double** mat = (double**)malloc(m*sizeof(double*));
63          for (short int i = 0; i < m; i++){
64                  mat[i] = (double*)malloc(n*sizeof(double));
65          }
66          return mat;
67  }
68  double **rand_mat(short int m, short int n){
69          /*declaro un puntero a un puntero a double que es el mas externo que recorre las filas*/
70          double** mat = create_mat(m, n);
71          for (short int i = 0; i < m; i++){
72                  for(short int j = 0; j < n; j++){
73                          /*basicamente lo que tenemos aca a continuación es un factor entre 0 y 1 por
74  el valor de max_val con lo cual es como mapear ese conjunto de numeros de 0 a max_val*/
75                          mat[i][j] = rand()*(double)max_val/RAND_MAX;
76                  }
77          }
78          return mat;
79  }
80  void mostrar(double** mat, short int m, short int n){
81          for(short int i = 0; i < m; i++){
82                  printf("\n");
83                  for(short int j = 0; j < n; j++){
84                          printf("\t%3.2lf", mat[i][j]);
85                  }
86          }
87          printf("\n");
88  }
89  double **producto(double** A, short int ma, short int na, double** B, short int mb, short int
90  nb){
91          double** C = NULL;
92          /*verifico que las dimensiones de las matrices permitan realizar la operacion*/
93          if(na == mb){
94                  /*Creo la matriz producto con tantas filas como la primer matriz y con tantas
95  columnas como la segunda matriz*/
96                  C = create_mat(ma, nb);
97                  for (short int i = 0; i < ma; i++){
98                          for (short int j = 0; j < nb; j++){
```

```
99                              C[i][j] = 0;
100                             for (short int k = 0; k < na; k++){
101                                     C[i][j] += (A[i][k]*B[k][j]);
102                             }
103                     }
104             }
105     }
106     return C;
107 }
108 void free_mat(double** mat, short int m){
109     for(short int i = 0; i < m; i++){
110             free(mat[i]);
        }
        free(mat);
    }
```

https://replit.com/@JuanBorquez/TP3-P2-EJ11#main.c

12. Escriba una función que reciba como parámetro una matriz de M x N elementos, y calcule la transpuesta de la matriz. El resultado debe ser almacenado en la misma matriz (es decir que no se debe reservar memoria para el resultado ni devolver ningún apuntador)

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

/*constantes simbolicas para el tope de rand y para el maximo de filas y columnas
respectivamente*/
#define max_val 10
#define max_size 10

/*Para crear una matriz sin nada*/
double** create_mat(short int m, short int n);

/*para crear y cargar la matriz random*/
double** rand_mat(short int m, short int n);

/*para mostrar*/
void mostrar(double** mat, short int m, short int n);

/*Para trasponer*/
/*Como no tiene que devolver un puntero es una función de tipo void*/
/*El puntero de la matriz tiene que apuntar a otra dirección en memoria cuando se cree el espacio
para la traspuesta
y este cambio tiene que verse en el main por lo tanto el puntero de la matriz también se pasa como
parametro*/
void traspuesta(double*** mat, short int m, short int n);

/*para liberar todo el espacio asociado a una matriz*/
/*notar que no hace falta indicar la cantidad columnas*/
void free_mat(double **mat, short int m);

int main(void) {
  srand(time(NULL));

        /*defino aleatoriamente la cantidad de filas(m) y de columnas(n) entre un mínimo de 1
y un máximo de max_size*/
        short int m = 1 + rand()%max_size;
        short int n = 1 + rand()%max_size;

        /*creo y cargo la matriz de forma aleatoria*/
        double** mat = rand_mat(m, n);

        printf("\nMATRIZ\n");
        mostrar(mat, m, n);

        /*traspongo la matriz*/
        traspuesta(&mat, m, n);

        printf("\nMATRIZ TRASPUESTA\n");
        /*hay que tener en cuenta que ahora hay n filas por m columnas*/
```

```
50          mostrar(mat, n, m);
51          /*no hace falta liberar el espacio asociado a ninguna otra matriz
52    dado que siempre usamos la misma*/
53          free_mat(mat, n);
54     return 0;
55    }
56
57    double** create_mat(short int m, short int n){
58          double** mat = (double**)malloc(m*sizeof(double*));
59          for(short int i = 0; i < m; i++){
60                mat[i] = (double*)malloc(n*sizeof(double));
61          }
62          return mat;
63    }
64    double** rand_mat(short int m, short int n){
65          double**mat = create_mat(m, n);
66          for(short int i = 0; i < m; i++){
67                for(short int j = 0;j < n; j++){
68                      /*lo siguiente hace un mapeo del intervalo de 0 a max_val*/
69                      /*se recuerda que max_val es una constante símbolica definia al principio del
70    programa*/
71                      mat[i][j] = rand()*(double)max_val/RAND_MAX;
72                }
73          }
74          return mat;
75    }
76    void mostrar(double** mat, short int m, short int n){
77          for(short int i = 0; i < m; i++){
78                printf("\n");
79                for(short int j = 0; j < n; j++){
80                      printf("\t%3.2lf", mat[i][j]);
81                }
82          }
83          printf("\n");
84    }
85    void traspuesta(double*** mat, short int m, short int n){
86          /*Una matriz auxiliar para trasponer y guardar los datos que hay en mat
87    para que no se pierdan cuando se hagan operaciones con mat*/
88          double** traspuesta = create_mat(n, m);//creo la matriz sin datos
89          //notar que esta definida con las dimesiones de la traspuesta
90
91          for (short int i = 0; i < n; i++){
92                for(short int j = 0; j < m; j++){
93                      /*copio a traspuesta el contenido de mat de modo que sea la traspuesta*/
94                      traspuesta[i][j] = (*mat)[j][i];
95                }
96          }
97
98          /*libero el espacio reservado para mat*/
```

```
 99          free_mat(*mat, m);
100
101         /*apunto con *mat a lo que hay en traspuesta*/
102         *mat = traspuesta;
103     }
104     void free_mat(double **mat, short int m){
105         for(short int i = 0; i < m; i++){
                free(mat[i]);
            }
            free(mat);
        }
```

https://replit.com/@JuanBorquez/TP3-P2-EJ12#main.c

13. Escriba un programa que, utilizando las dos funciones anteriores:

a. Lea por teclado 3 valores M, N y P
b. Cree 2 matrices dinámicamente con valores aleatorios:
   i. La primer matriz (C1)de M x N
   ii. La segunda matriz (C2) de P x N
c. Calcule la transpuesta de la segunda matriz (C2), dando lugar a la matriz C2T
d. Calcule el producto de C1 x C2T
e. Muestre los resultados intermedios por pantalla
f. Libere la memoria dinámica utilizada

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

/*máximo valor aleatoro en las matrices*/
#define max_val 10

/*para crear la matriz*/
double** create_mat(short int m, short int n);

/*para cargar la matriz randomly*/
double** rand_mat(short int m, short int n);

/*para hacer el producto*/
double** producto(double** A, short int ma, short int na, double** B, short int mb, short int nb);

/*par amostrar las matrices*/
void mostrar(double** mat, short int m, short int n);

/*para trasponer*/
void traspuesta(double*** mat, short int m, short int n);

/*para liberar todo el espacio asociado a la matriz*/
/*notar que solaente hace falta indicar la cantidad de filas y no la de columnas*/
void free_mat(double **mat, short int m);

int main(void) {
	srand(time(NULL));
	/*variables para la cantidad de filas y columnas de las matrices*/
	short int m, n, p;

	/*punteros para las matrices. La ultima es del producto*/
	double** C1, **C2, **prod;

	printf("\nA continuación indique 3 numeros enteros M, N y P\n");
	printf("Estas son las dimensiones de dos matrices C1 y C2 con valores aleatorios\n");
	printf("\n\nM: ");scanf("%hd", &m);
	printf("N: ");scanf("%hd", &n);
	printf("P: ");scanf("%hd", &p);

	/*creo y cargo de forma aleatoria las matrices con las dimensiones dadas*/
	C1 = rand_mat(m, n);
	C2 = rand_mat(p, n);

	system("clear");

	/*muestro la primera matriz*/
	printf("\nMATRIZ C1\n");
```

```
50          mostrar(C1, m, n);
51
52          /*muestro la segunda matriz*/
53          printf("\nMATRIZ C2\n");
54          mostrar(C2, p, n);
55
56          /*calculo la traspuesta de la segunda matriz*/
57          traspuesta(&C2, p, n);
58
59          /*ahora hay que tener en cuenta que ahora C2 tiene n filas y p columnas*/
60          printf("\nMATRIZ TRASPUESTA C2T(traspuesta de C2)\n");
61          mostrar(C2, n, p);
62
63          /*calculo el producto de C1 y de C2 que ahora esta traspuesta*/
64          prod = producto(C1, m, n, C2, n, p);
65          printf("\nPODUCTO MATRICIAL DE C1 Y C2T\n");
66          mostrar(prod, m, p);
67
68          /*libero todo el espacio asociado a las matrices*/
69          free_mat(C1, m);
70          free_mat(C2, n);
71          free_mat(prod, m);
72
73      return 0;
74  }
75  double** create_mat(short int m, short int n){
76          double** mat = (double**)malloc(m*sizeof(double*));
77          for (short int i = 0; i < m; i++){
78                  mat[i] = (double*)malloc(n*sizeof(double));
79          }
80          return mat;
81  }
82  double **rand_mat(short int m, short int n){
83          /*declaro un puntero a un puntero a double que es el mas externo que recorre las filas*/
84          double** mat = create_mat(m, n);
85          for (short int i = 0; i < m; i++){
86                  for(short int j = 0; j < n; j++){
87                          /*basicamente lo que tenemos aca a continuación es un factor entre 0 y 1 por
88  el valor de max_val con lo cual es como mapear ese conjunto de numeros de 0 a max_val*/
89                          mat[i][j] = rand()*(double)max_val/RAND_MAX;
90                  }
91          }
92          return mat;
93  }
94  void mostrar(double** mat, short int m, short int n){
95          for(short int i = 0; i < m; i++){
96                  printf("\n");
97                  for(short int j = 0; j < n; j++){
98                          printf("\t%3.2lf", mat[i][j]);
```

```
99                  }
100         }
101         printf("\n");
102 }
103
104 /*Si no son compatibles las dimensiones de las matrices devuelve un punero NULL*/
105 double **producto(double** A, short int ma, short int na, double** B, short int mb, short
106 int nb){
107         double** C = NULL;
108         /*verifico que las dimensiones de las matrices permitan realizar la operacion*/
109         if(na == mb){
110                 /*Creo la matriz producto con tantas filas como la primer matriz y con tantas
111 columnas como la segunda matriz*/
112                 C = create_mat(ma, nb);
113                 for (short int i = 0; i < ma; i++){
114                         for (short int j = 0; j < nb; j++){
115                                 C[i][j] = 0;
116                                 for (short int k = 0; k < na; k++){
117                                         C[i][j] += (A[i][k]*B[k][j]);
118                                 }
119                         }
120                 }
121         }
122         return C;
123 }
124 void traspuesta(double*** mat, short int m, short int n){
125         /*Una matriz auxiliar para trasponer y guardar los datos que hay en mat
126 para que no se pierdan cuando se hagan operaciones con mat*/
127         double** traspuesta = create_mat(n, m);//creo la matriz sin datos
128         //notar que esta definida con las dimesiones de la traspuesta
129
130         for (short int i = 0; i < n; i++){
131                 for(short int j = 0; j < m; j++){
132                         /*copio a traspuesta el contenido de mat de modo que sea la traspuesta*/
133                         traspuesta[i][j] = (*mat)[j][i];
134                 }
135         }
136
137         /*libero el espacio reservado para mat*/
138         free_mat(*mat, m);
139
140         /*apunto con *mat a lo que hay en traspuesta*/
141         *mat = traspuesta;
142 }
143 void free_mat(double** mat, short int m){
144         for(short int i = 0; i < m; i++){
145                 free(mat[i]);
146         }
        free(mat);
```

```
}
```

https://replit.com/@JuanBorquez/TP3-P2-EJ13#main.c