Preparador Informática

www.preparadorinformatica.com

PRÁCTICA 7

PROGRAMACIÓN EN C

(SOLUCIONES Ejercicios 61 al 67)

61. EJERCICIO "Traspuesta"

Realiza un programa que rellene una matriz de 3x3 con los números del 1 al 9 y haga su traspuesta (la traspuesta se consigue intercambiando filas por columnas y viceversa).

EJERCICIO 61. SOLUCIÓN PROPUESTA

```
#include <stdio.h>


int main(int argc, char *argv[])
{

    int x,y,num=1, numeros[3][3];

    for (x=0;x<3;x++)
    {
        for (y=0;y<3;y++)
        {
            numeros[x][y]=num;
            num++;
        }
    }

    printf("El array original es: \n\n\n");

    for(x = 0;x < 3;x++)
    {
      for(y = 0;y < 3;y++)
      {
          printf("    %d     ", numeros[x][y]);
      }
      printf("\n\n\n");
    }

    printf("La traspuesta es: \n\n\n");

    for(x = 0;x < 3;x++)
    {
      for(y = 0;y < 3;y++)
      {
          printf("    %d     ", numeros[y][x]);
      }
      printf("\n\n\n");
    }


    return 0;
}
```

62. EJERCICIO "MediaPonderada"

La nota del examen de una asignatura corresponde en un 20% al primer parcial, un 20% al segundo parcial y un 60% el examen final. La nota final es la mejor entre la nota del examen y la nota obtenida con la evaluación continua (teniendo en cuenta los dos parciales).

Escriba un programa que pida estas tres notas y calcule la nota final de un alumno. A continuación, el programa debe preguntar al usuario si quiere calcular una nueva nota o finalizar (pulsando la tecla 's').

EJERCICIO 62. SOLUCIÓN PROPUESTA

```
#include <stdio.h>

float calcularMax(float x, float y);
float calcularNota(float p1, float p2, float e);

int main(void)
{
	char c;
	do
	{
		float parcial1, parcial2, examen, final;
		/* Leer valores */
		printf("Introduzca nota parcial 1 [0, 10]: ");
		scanf("%f", &parcial1);
		printf("Introduzca nota parcial 2 [0, 10]: ");
		scanf("%f", &parcial2);
		printf("Introduzca nota examen final [0, 10]: ");
		scanf("%f", &examen);

		/* Calcular nota final*/
		final = calcularNota (parcial1, parcial2, examen);

		/* Imprimir resultados */
		printf("Nota final: %.2f", final);

		/* Continuar? */
		printf("\n\nSalir? (s/n): ");
		/* Para evitar conflictos con el salto de linea (\n) y
		evitar un bucle infinito se recurre al siguiente "truco" */
		do{
			scanf("%c", &c);
		} while (c=='\n');

		printf("\n");

	} while(c != 's');

	return 0;
}
float calcularNota (float p1, float p2, float notaExamen)
{
/* Función calcularNota: Calcula la nota final a partir de:
- primer parcial 20% (p1)
- segundo parcial 20% (p2)
- examen final 60% (notaExamen)
Si la nota final con la evaluación continua es menor a la del
```

```
examen, la nota final es la del examen
*/
	float notaEC, final;
	notaEC = p1*0.2+p2*0.2+notaExamen*0.6;
	final = calcularMax(notaEC, notaExamen);
	return final;
}

float calcularMax(float x, float y)
{
	/* Funcion calcularMax: Devuelve el mayor de los valores x, y pasados
como parámetro */
	float max;
	if(x >= y)
		max = x;
	else
		max = y;
	return max;
}
```

63. EJERCICIO "Expendedora"

La empresa que fabrica un modelo de máquinas expendedoras de refrescos necesita un programa para estas máquinas que realice el cálculo de cuántas monedas de cada tipo debe devolver. Para ello, en primer lugar, se introducirá por teclado la cantidad a devolver en euros. Para devolver el dinero se dispone de monedas de 50, 20, 10, 5, 2 y 1 céntimo. Siempre se dispone de monedas necesarias de todos los tipos y es necesario devolver el menor número de monedas posible.

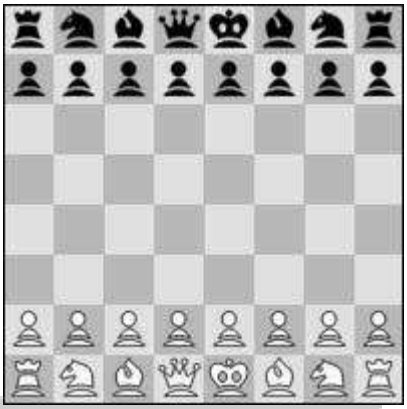
EJERCICIO 63. SOLUCIÓN PROPUESTA

```c
#include <stdio.h>
int main()
{
	double euros_a_devolver;
	int monedas_50c, monedas_20c, monedas_10c,
monedas_5c,monedas_2c,monedas_1c;
	double falta_por_devolver;

	/* Lectura de datos: euros_a_devolver */
	printf ("Euros a devolver: ");
	scanf ("%lf", &euros_a_devolver);

	falta_por_devolver=euros_a_devolver*100.0;

	monedas_50c=falta_por_devolver/50;
	falta_por_devolver=falta_por_devolver-50*monedas_50c;

	monedas_20c=falta_por_devolver/20;
	falta_por_devolver=falta_por_devolver-20*monedas_20c;

	monedas_10c=falta_por_devolver/10;
	falta_por_devolver=falta_por_devolver-10*monedas_10c;

	monedas_5c=falta_por_devolver/5;
	falta_por_devolver=falta_por_devolver-5*monedas_5c;

	monedas_2c=falta_por_devolver/2;
	falta_por_devolver=falta_por_devolver-2*monedas_2c;

	monedas_1c=falta_por_devolver;

	printf ("La cantidad de %f euros se devolvera asi:\n", euros_a_devolver);
	printf ("- %d monedas de 50 centimos\n", monedas_50c);
	printf ("- %d monedas de 20 centimos\n", monedas_20c);
	printf ("- %d monedas de 10 centimos\n", monedas_10c);
	printf ("- %d monedas de 5 centimos\n", monedas_5c);
	printf ("- %d monedas de 2 centimos\n", monedas_2c);
	printf ("- %d monedas de 1 centimos\n", monedas_1c);

	return 0;
}
```

64. EJERCICIO "Ajedrez"

Crea un programa que pinte un tablero de ajedrez, los peones con la letra P, las torres con T, los caballos con C, los alfiles con A, el rey con R y la reina con M.

EJERCICIO 64. SOLUCIÓN PROPUESTA



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

int main(int argc, char *argv[])
{

    int x,y;

    for (x=0;x<8;x++)
    {
        for (y=0;y<8;y++)
        {
            //peones
            if (x==1 || x==6)
            {
             printf("P");
            }
            //torres
            else if ((x==0 && y==0) ||
                    (x==7 && y==0) ||
                    (x==0 && y==7) ||
                    (x==7 && y==7)
                    )
            {
             printf("T");
            }
            //caballos
            else if ((x==0 && y==1) ||
                    (x==7 && y==1) ||
                    (x==0 && y==6) ||
                    (x==7 && y==6)
                    )
            {
             printf("C");
            }
            //alfiles
            else if ((x==0 && y==2) ||
                    (x==7 && y==2) ||
```

```
                    (x==0 && y==5) ||
                    (x==7 && y==5)
                    )
            {
             printf("A");
            }
            //reina
            else if ((x==0 && y==3) ||
                    (x==7 && y==3)
                    )
            {
             printf("M");
            }
            //rey
            else if ((x==0 && y==4) ||
                    (x==7 && y==4)
                    )
            {
             printf("R");
            }
            else
            {
                printf(" ");
            }
        }
        printf("\n");
    }

    return 0;

}
```

65. EJERCICIO "Vectores"

Realiza un programa, que lea dos vectores de 3 componentes, y luego los sume, reste y multiplique, tanto escalar como vectorialmente, imprimiendo después todos estos resultados.

EJERCICIO 65. SOLUCIÓN PROPUESTA

```
#include <stdio.h>
void suma( double*, double*, double*);
void resta( double*, double*, double*);
void productoVectorial(double*, double*, double*);
double productoEscalar(double*, double*);

int main(void)
{
	double vector1[3], vector2[3], vecsum[3], vecres[3];
	double prodvect[3];
	int i;
	printf("Introduce los valores del primer vector:\n");
	for (i=0; i<3; i++)
	{
		printf("\nElmento %d: ", i+1);
		scanf("%lf", &vector1[i]);
	}
	printf("\n");
	printf("Introduce los valores del segundo vector:\n");

	for (i=0; i<3; i++)
	{
		printf("\nElemento %d: ", i+1);
		scanf("%lf", &vector2[i]);
	}
	printf("\n");
	suma(vector1, vector2, vecsum);
	resta(vector1, vector2, vecres);
	productoVectorial(vector1, vector2, prodvect);
	printf("\nImprimo los resultados: \n\n");
	printf("%10s %10s %20s\n", "Suma", "Resta", "Producto Vectorial");
	for (i=1; i<=50; i++)
		printf("-");
	printf("\n");
	for (i=0; i<3; i++)
	{
		printf("%10.2lf  %10.2lf  %10.2lf\n", vecsum[i], vecres[i], prodvect[i]);
		printf("\n");
	}

	printf("Producto escalar = %5.2lf\n",
productoEscalar(vector1,vector2));
	printf("----------------\n");

	return 0;
}
```

```c
void suma(double* vec1, double* vec2, double* sum)
{
	int i;
	for (i=0; i<3; i++)
		sum[i]=vec1[i]+vec2[i];
	return;
}

void resta(double* vec1, double* vec2, double* res)
{
	int i;
	for (i=0; i<3; i++)
		res[i]=vec1[i]-vec2[i];
	return;
}

void productoVectorial(double* vec1, double* vec2, double* vectorial)
{
	vectorial[0] = vec1[1]*vec2[2]-vec2[1]*vec1[2];
	vectorial[1] = vec1[2]*vec2[0]-vec2[2]*vec1[0];
	vectorial[2] = vec1[0]*vec2[1]-vec2[0]*vec1[1];
	return;
}

double productoEscalar(double* vec1, double* vec2)
{
	double escalar = 0.0;
	int i;
	for (i=0; i<3; i++)
		escalar+=vec1[i]*vec2[i];
	return (escalar);
}
```

66. EJERCICIO "PilaCalculadora"

Escribe un programa que simule una calculadora que realiza las opeaciones +, -, * y /, empleando la notación polaca inversa: pimero se introducen los operandos (números) y después el operador que indica la operación a realizar. Utiliza una pila como estructura del ejercicio.

EJERCICIO 66. SOLUCIÓN PROPUESTA

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

typedef struct datos nodo;

struct datos /* estructura de un elemento de la pila */
{
	double dato;
	nodo *siguiente;
};

/* Prototipos de funciones */
void push(nodo **p, double x); // añadir un dato a la pila
double pop(nodo **p); // sacar un dato de la pila
void error(void);
nodo *nuevo_elemento(void); //reserva dinámica de memoria

int main()
{
	nodo *q,*cima = NULL;
	double a, b;
	char op[81];
	printf("Calculadora con las operaciones: + - * /\n");
	printf("Los datos serán introducidos de la forma:\n");
	printf(">operando 1\n");
	printf(">operando 2\n");
	printf("operador\n\n");
	printf("Para salir pulse q\n\n");
	do
	{
		printf("> ");
		gets(op); /* leer un operando o un operador */
		switch (*op)
		{
			case '+' :
				b = pop(&cima);
				a = pop(&cima);
				printf("%g\n", a + b);
				push(&cima, a+b);
				break;
			case '-' :
				b = pop(&cima);
				a = pop(&cima);
				printf("%g\n", a - b);
				push( &cima, a-b);
				break;
			case '*':
				b = pop(&cima);
				a = pop(&cima);
```

```
                        printf("%g\n", a * b);
                        push( &cima, a*b);
                        break;
                case '/':
                        b = pop(&cima);
                        a = pop(&cima);
                        if(b==0)
                        {
                                printf("Division por CERO");
                                break;
                        }
                        printf("%g\n", a / b);
                        push( &cima, a/b);
                        break;
                default:
                        push(&cima, atof(op));
        }
    }

    while(*op != 'q');
    //liberamos memoria

    q = cima;
    while(q != NULL)
    {
        cima = cima->siguiente;
        free (q);
        q = cima;
    }
} //fin de main()


/*Añadir un dato a la pila*/
void push(nodo **p, double x)
{
    nodo *q, *cima;
    cima = *p;
    q = nuevo_elemento();
    q->dato = x;
    q->siguiente = cima;
    cima = q;
    *p=cima;
}
//recuperar de la cima
double pop(nodo **p)

{
    nodo *cima;
    double x;
    cima= *p;
    if(cima ==NULL)
    {
        printf("ERROR");
        return 0;
    }
    else
    {
        x=cima->dato;
        *p=cima->siguiente;
```

```
                free(cima);
                return x;
        }
}
void error(void)

{
        perror("Memoria no reservada");
        exit(0);
}

nodo *nuevo_elemento(void)

{
        nodo *q = (nodo *)malloc(sizeof(nodo));
        if(!q)
                error();
        return (q);
}
```

67. EJERCICIO "ListaNumeros"

Escribir un programa completo que tome desde teclado sucesivos números enteros y los almacene en una lista enlazada simple, gobernada por un puntero *lista, insertando cada nuevo número por el final de la lista (después del elemento que contiene un valor NULL como puntero al siguiente elemento). La introducción de datos terminará cuando se pulse CTRL+D en Linux y CTRL+Z en Windows (marca de fin de fichero del teclado). Los elementos de la lista contendrán estructuras del siguiente tipo:

struct elemento

{

int dato;

struct elemento *psig;

}

Una vez terminada la introducción de datos, el programa los mostrará todos en pantalla y finalmente eliminará de la memoria toda la lista.

EJERCICIO 67. SOLUCIÓN PROPUESTA

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

struct elemento
{
	int dato;
	struct elemento *psig;
};

int main()
{
	struct elemento *lista=NULL, *q, *p;
	int num;

	while (1)
	{
		printf("Dame número (CTRL+D=Fin en Linux ó CTRL+Z=Fin en
Windows): ");
		if (scanf("%d", &num)==EOF)
			break;
		q = (struct elemento *)malloc(sizeof(struct elemento));

		if (q==NULL)
		{
			printf("ERROR: Falta memoria.\n");
			getchar();
		}
		else
		{
			q->dato=num;
			q->psig=NULL;

			if (lista==NULL)
				lista=q;
```

```
                else
                {
                        p=lista;

                        while (p->psig!=NULL)
                                p=p->psig;
                                p->psig=q;
                }
        }
    }

    printf("\nLos números introducidos son:\n");
    q=lista;

    while (q!=NULL)
    {
        printf("%d ",q->dato);
        q=q->psig;
    }
    printf("\nEliminando la lista...\n");
    while (lista!=NULL)
        {
            q=lista;
            lista=lista->psig;
            free(q);
        }
    return 0;
}
```