Preparador Informática

www.preparadorinformatica.com

PRÁCTICA 8

PROGRAMACIÓN EN C

(SOLUCIONES Ejercicios 68 al 72)

68. EJERCICIO "Número Intermedios"

Escribir en lenguaje C un programa que:

1º) Pida por teclado dos números enteros.

2º) En el caso de que el primer número introducido por el usuario, no sea menor o igual que el segundo, muestre por pantalla el mensaje:

"ERROR: <número_superior> no es mayor o igual que <número_inferior>"

3º) Repita los pasos 1º y 2º, mientras que, los números introducidos sean incorrectos.

4º) Muestre por pantalla los números que hay entre los dos números introducidos por el usuario, ambos inclusive.

EJERCICIO 68. SOLUCIÓN PROPUESTA

```c
#include <stdio.h>

int main()
{
    int inferior, superior;

    printf( "\n   Introduzca n%cmero (entero) inferior: ");
    scanf( "%d", &inferior );
    printf( "\n   Introduzca n%cmero (entero) superior: ");
    scanf( "%d", &superior );

    while ( inferior > superior )
    {
        printf( "\n   ERROR: %d no es mayor o igual que %d", superior,
inferior );
        printf( "\n\n   Introduzca n%cmero (entero) inferior: ");
        scanf( "%d", &inferior );
        printf( "\n   Introduzca n%cmero (entero) superior: " );
        scanf( "%d", &superior );
    }

    printf( "\n   " );

      do
    {
        printf( "%d ", inferior);
        inferior++;

    } while ( inferior <= superior );


    return 0;
}
```

69. EJERCICIO "Lectura"

Escribe un programa en C donde a partir de una posición del fichero lea un número de caracteres dados y se escriban por pantalla.

EJERCICIO 69. SOLUCIÓN PROPUESTA

```
# include <stdio.h>

int leer_posicion(void);
int leer_num(void);
int test(int);

int main()
{
	FILE *fp;
	char nombre[36], c;
	int pos, num;
	long desp;
	char continuar;


	printf("Nombre de Fichero ? ");
	scanf("%s",nombre);
	fp=fopen(nombre, "rb");
	if ( ! fp )
		printf("error al abrir fichero de entrada\n");
	else
	{
		do
		{
			pos = leer_posicion();
			printf("Desplazamiento (num. de bytes) ? ");
			scanf("%ld", &desp);
			num = leer_num();
			if ( fseek(fp, desp, pos) != 0 )
				printf("error en posicionamiento\n");
			else
			{
				int i=0;
				c = getc(fp);
				while ( i<=num && c!=EOF )
				{
					putchar(c);
					i=i+1;
					c = getc(fp);
				}
				putchar('\n');
				putchar('\n');
			}

			/* 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');

        }
}

int leer_posicion()
{
        int n;
        printf("Posicion (0:principio,  1:actual,  2:EOF) ? ");
        scanf("%d", &n);
        while ( ! test(n) )
        {
                printf("Posicion (0:principio,  1:actual,  2:EOF) ?");
                scanf("%d", &n);
        }
        return(n);
}

int test ( int n )
{
        if ( n<0 || n>2 )
                return(0);
        else
                return(1);
}

int leer_num()
{
        int n;
        printf("Numero de bytes ( mayor que cero ) ? ");
        scanf ("%d", &n);
        while ( n<0 )
        {
                printf("Numero de bytes ( mayor que cero ) ? ");
                scanf ("%d", &n);
        }
        return(n);
}
```

70. EJERCICIO "ListaEnlazadaRecursivo"

Escribe un programa que construya una lista enlazada con los caracteres de una cadena sin espacios que se lee por pantalla, y se recorre escribiendo los campos de datos de manera recursiva.

EJERCICIO 70. SOLUCIÓN PROPUESTA

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


struct lista
{
	char dato;
	struct lista *siguiente;
};
typedef struct lista ELEMENTO;
typedef ELEMENTO *ENLACE;

ENLACE cadena(char *);
void listar(ENLACE);

int main()
{
	ENLACE principio;
	char s[80];

	printf("Introducir una cadena: \n");
	scanf("%s", s);
	principio=cadena(s);
	listar(principio);
	printf("\n");

	return 0;
}

/*
Creacion de lista encadenada con algoritmo recursivo
*/
ENLACE cadena(char s[ ])
{
	ENLACE principio;
	if ( s[0] == '\0' )
		return(NULL);
	else
	{
		principio=(ENLACE)malloc(sizeof(ELEMENTO));
		principio->dato=s[0];
		principio->siguiente=cadena(s+1);
		return(principio);
	}
}
```

```c
/*
Se recorre una lista con algoritmo recursivo escribiendo los campos de datos.
*/
void listar(ENLACE principio)
{
	if ( principio == NULL )
		printf("NULL");
	else
	{
		printf("%c --> ", principio->dato);
		listar(principio->siguiente);
	}
}
```

71. EJERCICIO "Craps"

Realiza un programa que simule el juego de dados conocido como "craps" (tiro perdedor). Las reglas para los jugadores son:

- **Un jugador tira dos dados. Cada dato tiene seis caras. Las caras contienen 1, 2, 3, 4, 5 y 6 puntos.**
- **Una vez que los dados se hayan detenido, se calcula la suma de los puntos en las dos caras superiores.**
- **Si a la primera tirada, la suma es 7, o bien 11, el jugador gana.**
- **Si a la primera tirada la suma es 2, 3 o 12 (conocido como "craps"), el jugador pierde (es decir la casa "gana").**
- **Si a la primera tirada la suma es 4, 5, 6, 8, 9 ó 10, entonces dicha suma se convierte en el "punto" o en la "tirada".**
- **Para ganar, el jugador deberá continuar tirando los dados hasta que haga su "tirada".**
- **El jugador perderá si antes de hacer su tirada sale una tirada de 7.**

EJERCICIO 71. SOLUCIÓN PROPUESTA

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
//#include <dos.h>
//#include <windows.h>

// varibles utilizadas para comprobar el estado del juego
#define GANA 0
#define PIERDE 1
#define CONTINUA 2

int PrimerTiro=1; // 1 si el lanzamiento actual es el primero
int SumaDados=0;  // suma de los dados
int MiPunto=0;  // tirada o puntos si no gana o pierde en el primer
lanzamiento
int EstadoJuego=CONTINUA;

int LanzaDados();
int Juego();

int main()
{
      int i;
      printf("\n CRAPS \n");
      Juego();
      if (EstadoJuego==GANA)
         printf("\n FELICIDADES! ");
      else
            printf("\n Lo sentimos acaba de perder \n\n");
      return 0;
}

int LanzaDados()
{
      int dado1,dado2, suma;
```

```
	srand(time(0));
	dado1=1+(rand()%6);
	srand(getpid());
	dado2=1+(rand()%6);
	printf("\n dado1 : %d", dado1);
	printf("\n dado2 : %d", dado2);

	suma=dado1+dado2;
	return suma;
}

int Juego()
{
	int puntos,n_lanzamiento;
	n_lanzamiento=1;
	while(EstadoJuego==CONTINUA)
	{
		puntos=LanzaDados();
		if (PrimerTiro==1)
		{
			printf("\n Primer lanzamiento: %d \n", puntos);
			switch(puntos)
			{
				case 7:EstadoJuego=GANA;
				break;
				case 11:EstadoJuego=GANA;
				break;
				case 2:EstadoJuego=PIERDE;
				break;
				case 3:EstadoJuego=PIERDE;
				break;
				case 12:EstadoJuego=PIERDE;
				break;
				default:
				{
					PrimerTiro=0;
					EstadoJuego=CONTINUA;
					MiPunto=puntos;
					system("pause");
					break;
				}
			}
		}
		else
		{
			PrimerTiro=0;
			n_lanzamiento=n_lanzamiento+1;
			printf("\n Lanzamiento numero %d: %d",
n_lanzamiento,puntos);

			if(puntos==MiPunto)
			{
				EstadoJuego=GANA;
			}
			else
			{
				switch(puntos)
				{
					case 7:
```

```
                                        EstadoJuego=PIERDE;
                                        break;
                                default:
                                {
                                        EstadoJuego=CONTINUA;
                                        break;
                                }
                        }
                }
        }
        return EstadoJuego;
}
```

72. EJERCICIO "Ordenar"

Escribir una función que sea capaz de ordenar un array de enteros usando alguna función que contenga manejo de punteros.

EJERCICIO 72. SOLUCIÓN PROPUESTA

```
#include <stdio.h>

int arr[10]={2, 3, 4, 5, 9, 19, 300, 29, 4, 1};

void ordena (int *p, int N);
int compara (int *m, int *n);

int main()
{
	int i=0;
	for (i=0; i < 10; i ++)
	{
		printf("[ %d ] ", arr[i]);
	}

	ordena(arr, 10);
	printf("\n");


	for (i=0; i < 10; i ++)
	{
		printf("[ %d ] ", arr[i]);
	}

	return 0;
}

void ordena (int *p, int N)
{
	int i, j, t;


	for (i=N-1; i >=0; i--)
	{
		for (j = 1; j<=i; j++)
		{
			if (compara(&p[j-1], &p[j]))
			{
				t = p[j-1];
				p[j-1]= p[j];
				p[j] = t;
			}
		}
	}
}

int compara(int *m, int *n)
{
	return (*m >*n);
}
```