CENTRO UNIVERSITARIO
DE TECNOLOGÍA Y ARTE DIGITAL

INTRODUCCIÓN A LA PROGRAMACIÓN II

Ejercicios Propuestos Programa Refuerzo Programación:
Sesión 3 IPR2: Reserva de memoria dinámica

Contenido

1. Ejercicio 1. Estadísticas de una lista de números enteros pedidos al usuario de tamaño determinado

Crea un programa que calcule el máximo, el mínimo y la media de una lista de números racioneales de un tamaño previamente determinado. Al usuario se le pide tanto número de elementos de la lisa como cada uno de los números (printf / scanf).

El programa principal debe encargarse de:

- Pedir al usuario el número de racionales que forman la lista.
- Reservar memoria
- Pedir al usuario los números e irlos almacenando
- Realizar las estadísticas
- Presentar estadísticas y el conjunto de números que ha introducido el usuario
- Liberar memoria

El programa debe incluir las siguientes funciones:

```
//Función que calcula las estadísticas a partir de los valores
// de la lista de números

void calcular_estadisticas(float *lista, int n, float *max,
float *min, float *media);
```

```
Prototipos:

     void* malloc (size_t n)

     void free (void *);

Invocaciones:

     puntero = (tipo *) malloc (tamaño en bytes);

     free (puntero);
```

2. Ejercicio 2. Estadísticas de una lista de números enteros pedidos al usuario de tamaño indeterminado

Crea un programa que calcule el máximo, el mínimo y la media de una lista de números decimales de un tamaño indeterminado y que se le piden al usuario (printf / scanf). La introducción de números finalizará cuando el usuario quiera

El programa principal debe encargarse de:

- Reservar memoria para el primer número
- Bucle mientras que el usuario quiera seguir introduciendo números
    - Pedir al usuario número
    - Almacenar el número
    - Preguntar al usuario si quiere introducir otro númro
- Realizar las estadísticas
- Presentar estadísticas
- Liberar memoria

```
Prototipos:

      void* malloc (size_t n)

      void *realloc(void *ptr, size_t size);

      void free (void *);

Invocaciones:

      puntero = (tipo *) malloc (tamaño en bytes);

      puntero = (tipo *) realloc (puntero, tamaño en bytes);

      free (puntero);
```

3. Ejercicio 3. Igualdad y concatenación de Cadenas de caracteres

Escribe un programa en lenguaje C al que se le solicite al usuario dos cadenas de caracteres de tamaño no determinado, compruebe si son iguales y si no lo son realzar la concatenación de las dos cadenas (tamaño máximo 20 caracteres) y que compruebe que son iguales.

El programa principal debe encargarse de:

- Pedir al usuario las dos cadenas de caracteres.
- Almacenar lo que introduzca el usuario en dos punteros a char. Para ello realizar una función leeLineaDinamica que devuelva una cadena de caracteres y que realice la reserva de memoria dinámica para la cadena.

- Comprobar si son iguales
    - Si lo son imprimir un mensaje por pantalla que lo muestre
    - Si no lo son concatenarlas y mostrar la cadena concatenada
- Liberar memoria

El programa debe incluir las siguientes funciones:

```
//Función que devuelve una cadena de caracteres leída del
//buffer I/O y que realiza la reserva de memoria dinámica por
//bloques para la cadena

char* leeLineaDinamica();

// Función que se le pasa como parámetros dos cadenas y devuelve
// verdadero si son iguales y falso si no lo son

int cadenasIguales(char* cadena1, char* cadena2);

// Función que se le pasa como parámetro una cadena y devuelve
//el número de caracteres que tiene

int tamCadena (char* cadena);

// Función que se le pasa como parámetros dos cadenas y devuelve
// la cadena concatenación de las dos que se pasan

char* concatenarCadenas(char* cadena1, char* cadena2);
```

4. Ejercicio 4. Lista de animales

Escribe un programa en C que almacene una lista de un número indeterminado de aves.

El programa solicitará al usuario el nombre de cada ave. Esta acción se repetirá mientras que el ave que escriba el usuario no esté ya incluida en la lista.

El programa debe incluir las siguientes funciones:

```
//Función que devuelve una cadena de caracteres leída del
//buffer I/O y que realiza la reserva de memoria dinámica por
//bloques para la cadena

char* leeLineaDinamica();

// Función que se le pasa como parámetros dos cadenas y devuelve
// verdadero si son iguales y falso si no lo son

int cadenasIguales(char* cadena1, char* cadena2);

// Función que se le pasa como parámetro una cadena y devuelve
//el número de caracteres que tiene

int tamCadena (char* cadena);

// Función que se le pasa como parámetros dos cadenas y devuelve
// la cadena concatenación de las dos que se pasan
```

5. Ejercicio 5. Gestión zapatería

Escribe un programa en C que realice la gestión de una zapatería.

Una zapatería está formada por un conjunto de zapatos

Los datos de cada uno de los zapatos son:

- Referencia. (Identificador único de zapato entero)
- Color: Por ejemplo: negro
- Tipo. Los tipos de zapatos son: MOCASIN, SANDALIA y ZAPATILLA
- Talla (entero)
- Precio (racional)

El programa debe incluir lo siguiente:

1. Definir un tipo estructura "struct zapato_t", con los datos anteriores.
2. Definir una variable para la zapatería.
3. Implementad un menú, utilizando para ello un enum, cuyas opciones sean:
    1. Introducir zapato
    2. Imprimir zapatos zapatería
    3. Seleccionar los zapatos de talla superior a 45
    4. Salir

El programa debe incluir las siguientes funciones:

```
// Función que devuelve una cadena de caracteres leída del
// buffer I/O y que realiza la reserva de memoria dinámica por
// bloques para la cadena

char* leeLineaDinamica();

// Función que se le pasa como parámetros la dirección de
// memoria donde comienza la biblioteca y el índice que ocupa el
// zapato dentro de la zapateria e incluye en la zapatería los
// valores del nuevo zapato

void introducirZapato (struct zapato_t* zapateria, int NumZap);

// Función que  reciba una copia de un zapato y lo imprima por pantalla

void ImprimeZapato(struct zapato_t zapato);
```