Trabajo Práctico número 2

Programación I – Laboratorio I.
Tecnicatura Superior en Programación.
UTN-FRA

Autores: *Marina Cardozo*

Revisores: *Laura Carelli*

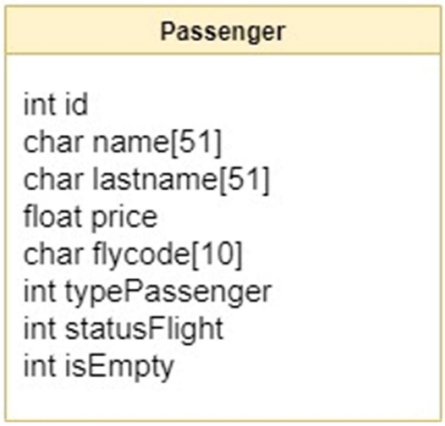
Versión : 1

Índice de contenido

1 Enunciado

Una aerolínea requiere un sistema para administrar los pasajeros de cada vuelo. Se sabe que no puede tener más de 2000 pasajeros.

Datos:



El sistema deberá tener el siguiente menú de opciones:

1. ALTAS: Se debe permitir ingresar un pasajero **calculando automáticamente el númerode Id**. El resto de los campos se le pedirá al usuario.

2. MODIFICAR: Se ingresará el Número de Id, permitiendo modificar: o Nombre o Apellido o Precio o Tipo de pasajero o Código de vuelo

3. BAJA: Se ingresará el Número de Id y se eliminará el empleado del sistema.

4. INFORMAR:

    1. Listado de los pasajeros ordenados alfabéticamente por Apellido y Tipo de pasajero.
    2. Total y promedio de los precios de los pasajes, y cuántos pasajeros superan el precio promedio.
    3. Listado de los pasajeros por Código de vuelo y estados de vuelos 'ACTIVO'

1-NOTA: Se deberá realizar el menú de opciones y las validaciones a través de funciones. Tener en cuenta que no se podrá ingresar a los casos 2, 3 y 4; sin antes haber realizado la carga de algún pasajero.

2-NOTA: Es deseable generar una segunda entidad y utilizarla en el programa.

Para la realización de este programa, se utilizará una biblioteca llamada "ArrayPassenger" que facilitará el manejo de la lista de pasajeros y su modificación. En la sección siguiente se detallan las funciones que esta biblioteca debe tener.

2 Biblioteca "ArrayPassenger"

Agrupa funciones que permiten trabajar con los datos almacenados en la estructura Passenger,la cual representa los datos de un empleado de la empresa.

```
struct
{
    int id;
    char name[51];
    char lastName[51];
    float price;
    char flycode[10];
    int typePassenger;
    int isEmpty;

}typedef Passenger;
```

2.1 Función `initPassengers`

Para indicar que todas las posiciones del array están vacías, esta función pone la bandera (isEmpty) en TRUE en todas las posiciones del array.

```
/** \brief  To indicate that all position in the array are empty,
 *          this function put the flag (isEmpty) in TRUE in all
 *          position of the array
 * \param list Passenger* Pointer to array of passenger
 * \param len int Array length
 * \return int Return (-1) if Error [Invalid length or NULL pointer] - (0) if Ok
 *
 */
int initPassengers(Passenger* list, int len)
{
    return 0;
}
```

Ejemplo uso:

```
r = initPassengers(arrayPassengers, ELEMENTS);
```

2.2 Función addPassengers

Agrega en un array de pasajeros existente los valores recibidos como parámetro en la primer posición libre.

```
/** \brief add in a existing list of passengers the values received as parameters
 *         in the first empty position
 * \param list passenger*
 * \param len int
 * \param id int
 * \param name[] char
 * \param lastName[] char
 * \param price float
 * \param typePassenger int
 * \param flycode[] char
 * \return int Return (-1) if Error [Invalid length or NULL pointer or without
free space] - (0) if Ok
```

```
 */
int addPassenger(Passenger* list, int len, int id, char name[],char
lastName[],float price,int typePassenger, char flycode[])
{
    return -1;
}
```

Ejemplo uso:

```
r = addPassenger(arrayPassenger,ELEMENTS,id,name,lastName,price,typePassenger,flycode);
```

2.3 Función findPassengerById

Busca un pasajero recibiendo como parámetro de búsqueda su Id.

```
/** \brief find a Passenger by Id en returns the index position in array.
 *
 * \param list Passenger*
 * \param len int
 * \param id int
 * \return Return passenger index position or (-1) if [Invalid length or
NULLpointer received or passenger not found]
 *
 */
int findPassengerById(Passenger* list, int len,int id)
{
    return NULL
}
```

Ejemplo uso:

```
int index = findPassengerId(arrayPassengers, ELEMENTS,9);
```

2.4 Función removePassenger

Elimina de manera lógica (isEmpty Flag en 1) un pasajero recibiendo como parámetro su Id.

```
/** \brief Remove a Passenger by Id (put isEmpty Flag in 1)
 *
 * \param list Passenger*
 * \param len int
 * \param id int
 * \return int Return (-1) if Error [Invalid length or NULL pointer or if can't
find a passenger] - (0) if Ok
 *
 */
int removePassenger(Passenger* list, int len, int id)
{
    return -1;
}
```

Ejemplo uso:

```
r = removePassenger(arrayPassengers, ELEMENTS,20);
```

2.5 Función sortPassengers

Ordena el array de pasajeros por apellido y tipo de pasajero de manera ascendente o descendente.

```
/** \brief Sort the elements in the array of passengers, the argument order
indicate UP or DOWN order
 *
 * \param list Passenger*
 * \param len int
 * \param order int  [1] indicate UP - [0] indicate DOWN
 * \return int Return (-1) if Error [Invalid length or NULL pointer] - (0) if Ok
 *
 */
int sortPassengers(Passenger* list, int len, int order)
{
    return 0;
}
```

Ejemplo uso:

```
r = sortPassengersByName(arrayPassengers, ELEMENTS, 1);
```

2.6 Función printPassengers

Imprime el array de pasajeros de forma encolumnada.

```
/** \brief print the content of passengers array
 *
 * \param list Passenger*
 * \param length int
 * \return int
 *
 */
int printPassenger(Passenger* list, int length)
{
    return 0;
}
```

Ejemplo uso:

```
r = printPassenger(arrayPassengers, ELEMENTS);
```

2.7 Función sortPassengersByCode

Ordena el array de pasajeros por código de vuelo y estado de vuelo de manera ascendente o descendente.

```
/** \brief Sort the elements in the array of passengers, the argument order
indicate UP or DOWN order
 *
 * \param list Passenger*
 * \param len int
 * \param order int  [1] indicate UP - [0] indicate DOWN
```

```
 * \return int Return (-1) if Error [Invalid length or NULL pointer] - (0) if Ok
 *
 */
int sortPassengers(Passenger* list, int len, int order)
{
    return 0;
}
```

Ejemplo uso:

```
r = sortPassengersByCode(arrayPassengers, ELEMENTS, 1);
```

3 Cómo realizar y entregar el trabajo práctico

El trabajo práctico deberá ser entregado en el repositorio de GIT correspondiente al TP2 de la materia.

El mismo consistirá en el proyecto de Eclipse con el programa funcionando y comentado, respetando las reglas de estilo de la cátedra. La compilación no deberá arrojar mensajes de error ni de warnings.

El proyecto deberá contar con más de una biblioteca. Se tiene que encontrar la biblioteca pedida en el trabajo práctico y una personal con funciones correspondientes al pedido de dato. Se deberá hacer uso de dichas funciones para resolver la lógica del programa.

No debe haber desarrollo de funciones en el main, solo el uso del menú y arranque del programa.

3.1 Objetivo

El objetivo del siguiente trabajo es que el alumno sea capaz de demostrar que puede integrar los conocimientos aprendidos hasta la clase 12. Los conocimientos necesarios para la realización del TP son los siguientes:

- Manejo de arrays.
- Manejo de pasaje por referencia.
- Manejo de estructuras.