Practica primer parcial Laboratorio I.

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

Autores: *Lic. Mauricio Dávila*

Revisores: *Ing. Ernesto Gigliotti*

Versión : 1

Índice de contenido

1 Enunciado

Una empresa requiere un sistema para administrar su nómina de empleados. Se sabe que dicha nómina bajo ninguna circunstancia superara los 10000 empleados.

Datos:

1. ALTAS (Se debe permitir ingresar un empleado calculando automáticamente el numero de Id)

2. MODIFICAR: Se ingresará el Número de Id, permitiendo modificar: o Nombre o Apellido o Salario o Sector

3. BAJA: Se ingresará el Número de Id y se eliminara el empleado mediante una baja lógica.

4. INFORMAR:

    1. Nomina de empleados
    2. Total y promedio de los Salarios, y cuántos empleados superan el salario promedio.

5. LISTAR: Listado de los empleados ordenados alfabéticamente por Apellido y Sector.

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 empleado.

2 Biblioteca Array

Representa a una familia de funciones que permiten trabajar con los datos almacenados en la estructura Employee, la cual representa los datos de un empleado de la empresa.

```
struct
{
    int id;
    char name[51];
    char lastName[51];
    float salary;
    int sector;
    int isEmpty;

}typedef employee;
```

Cada función de la biblioteca cuenta con un Test unitario asociado mediante el cual se podrá verificar el correcto funcionamiento de la misma.

2.1 Función `initEmployees`

Para indicar que todas las posiciones de la matriz 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 pEmployee employee* Pointer to array of employees
 * \param length int Array length
 * \return int Return (-1) if Error [Invalid length or NULL pointer] - (0) if Ok
 *
 */
int initEmployees(employee* pEmployee, int length)
{
    return 0;
}
```

Ejemplo uso:

```
r = initEmployees(arrayEmployees, ELEMENTS);
```

Ejemplo de uso del test:

```
startTesting(1);
```

Casos de Test:

>Case[Return of Array Init]
Verifica el valor de retorno de la función, debe ser [0] si se pudo inicializar el array.

>Case[Return of init Employees array with NULL pointer to array]
Verifica el valor de retorno de la función, debe ser [-1] si se envió un puntero NULL.

>Case[Return of init Employees array with invalid array length]
Verifica el valor de retorno de la función, debe ser [-1] si se envió un largo (**length**) del array inferior a uno.

2.2 Función addEmployees

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

```
/** \brief add in a existing list of employees the values recived as parameters
 *         in the first empty position
 * \param pEmployee employee*
 * \param length int
 * \param id int
 * \param name[] char
 * \param lastName[] char
 * \param salary float
 * \param sector int
 * \return int Return (-1) if Error [Invalid length or NULL pointer or without
free space] - (0) if Ok
 *
 */
int addEmployee(employee* pEmployee, int length, int id, char name[],char
lastName[],float salary,int sector)
{
    return -1;
}
```

Ejemplo uso:

```
r = addEmployee(arrayEmployees, ELEMENTS,id[i],names[i],lastNames[i],salary[i],sector[i]);
```

Ejemplo de uso del test:

```
startTesting(2);
```

Casos de Test:

>Case[Return of add Employees]
Verifica el valor de retorno de la función, debe ser [0] si se pudo agregar un empleado.

>Case[Content of added Employees]
Verifica el contenido del array luego de agregar un conjunto de empleados.

>Case[Return of add Employees without space empty]
Verifica el valor de retorno de la función, debe ser [-1] si no hay lugares libres.

>Case[Return of add Employees with NULL pointer to array]
Verifica el valor de retorno de la función, debe ser [-1] si se envió un puntero NULL.

>Case[Return of add Employees with invalid array length]
Verifica el valor de retorno de la función, debe ser [-1] si se envió un largo (length) del array inferior a uno.

2.3 Función findEmployeeById

Busca un empleado recibiendo como parámetro de búsqueda su Id.

```
/** \brief find a Employee by Id
 *
 * \param pEmployee employee*
 * \param length int
 * \param id int
 * \return employee* Return a (Employee pointer) if ok or (NULL pointer) if
[Invalid length or NULL pointer recived or employeed not found]
 *
 */

employee* findEmployeeById(employee* pEmployee, int length,int id)
{

    return NULL;

}
```

Ejemplo uso:

```
auxEmployee = findEmployeeById(arrayEmployees, ELEMENTS,9);
```

Ejemplo de uso del test:

```
startTesting(3);
```

Casos de Test:

>Case[Return when found a Employee]
Verifica el valor de retorno de la función, debe ser [0] si encuentra un empleado.

>Case[Content of founded Employee]
Verifica el contenido del empleado encontrado.

>Case[Return of find Employee with NULL pointer to array]
Verifica el valor de retorno de la función, debe ser [-1] si se envió un puntero NULL.

>Case[Return of find Employee with invalid array length]
Verifica el valor de retorno de la función, debe ser [-1] si se envió un largo (length) del array inferior a uno.

2.4 Función removeEmployee

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

```
/** \brief Remove a Employee by Id (put isEmpty Flag in 1)
 *
 * \param pEmployee employee*
 * \param length int
 * \param id int
 * \return int Return (-1) if Error [Invalid length or NULL pointer or if can't
find a employee] - (0) if Ok
 *
 */
int removeEmployee(employee* pEmployee, int length, int id)
{

    return -1;
}
```

Ejemplo uso:

```
r = removeEmployee(arrayEmployees, ELEMENTS,20);
```

Ejemplo de uso del test:

startTesting(4);

Casos de Test:

>Case[Return when remove a Employee]
Verifica el valor de retorno de la función, debe ser [0] si elimina un empleado.

>Case[Content of removed Employee]
Verifica el contenido del empleado eliminado.

>Case[Return of remove Employee with NULL pointer to array]
Verifica el valor de retorno de la función, debe ser [-1] si se envió un puntero NULL.

>Case[Return of remove Employee with invalid array length]
Verifica el valor de retorno de la función, debe ser [-1] si se envió un largo (length) del array inferior a uno.

2.5 Función sortEmployeeByName

Ordena el array de empleados por nombre de manera ascendente o descendente.

```
/** \brief Sort the elements in the array of employees by Name, the argument
order indicate UP or DOWN order
 *
 * \param pEmployee employee*
 * \param length int
 * \param order int  [1] indicate UP - [0] indicate DOWN
 * \return int Return (-1) if Error [Invalid length or NULL pointer or if can't
find a employee] - (0) if Ok
 *
 */
int sortEmployeeByName(employee* pEmployee, int length, int order)
{

    return 0;
}
```

Ejemplo uso:

```
r = sortEmployeeByName(arrayEmployees, ELEMENTS, 1);
```

Ejemplo de uso del test:

startTesting(5);

Casos de Test:

>Case[Return when sort a Employees array]
Verifica el valor de retorno de la función, debe ser [0] si se logra ordenar.

>Case[Content of sorted a Employees array UP]
Verifica el contenido del array luego de ordenar de forma ascendente.

>Case[Content of sorted a Employees array DOWN]
Verifica el contenido del array luego de de ordenar de forma descendente.

>Case[Return of sorted a Employees array with NULL pointer to array]
Verifica el valor de retorno de la función, debe ser [-1] si se envió un puntero NULL.

>Case[Return of sorted a Employees array with invalid array length]
Verifica el valor de retorno de la función, debe ser [-1] si se envió un largo (length) del array inferior a uno.

2.6 Función printEmployees

imprime el array de empleados de forma encolumnada.

```
/** \brief print the content of employee array
 *
 * \param pEmployee employee*
 * \param length int
 * \return int
 *
 */
int printEmployees(employee* pEmployee, int length)
{

    return 0;
}
```

Ejemplo uso:

```
r = printEmployees(arrayEmployees, ELEMENTS);
```

Ejemplo de uso del test:

No tiene test.

Casos de Test:

No tiene test.