UNIVERSIDAD NACIONAL DE RÍO CUARTO

FACULTAD DE CS. EXACTAS, FCO-QCAS Y NATURALES - DEPTO DE COMPUTACIÓN

ASIGNATURA:INTRODUCCIÓN A LA ALGORÍTMICA Y PROGRAMACIÓN(CÓD. 3300)

Año: 2021

PROYECTO FINAL

ALUMNOS:

BUCHIERI GIOVANNI
GIACHERO GABRIEL
PENNONE GENARO

Análisis

Analisis de la Accion Menu

Datos: selec //Pedido como entrada para poder seleccionar una opcion del menu

Resultados: La opcion que el usuario elija del menú

Relaciones y Subproblemas:

term ← falso

Usaremos la estructura iterativa “Mientras” con la condición de continuación “term = falso”, esto quiere decir que hasta que la variable lógica “term” no sea verdadera se continuará ciclando.

Dentro de esta estructura:

Primero se mostrará un menú (mediante la acción opciones() la cual solo muestra un mensaje con las opciones y los correspondientes números). Luego se pedirá la entrada de la variable “selec” en la cual se va poder ingresar un valor entre el 0 al 7, ya que son 7 opciones las cuales el usuario puede elegir dentro del programa, estas son:

- 1 .Insertar al final.
- 2. Suprimir el primero.
- 3 .Mostrar todos.
- 4 .Mostrar menores.
- 5. Buscar por DNI.
- 6. Mayores al primero.
- 7. Edad mayor
- 0. Guardar y salir

Al Seleccionar la Opción que se quiera:

Usaremos la estructura “según” con las siguientes condiciones:

Cuando selección sea 1 (selección = 1): se invocará “InsertarAlFinal(personas)”

Análisis de la Acción InsetarAlFinal

Dato: per //pasado como tipo dato-resultado es un arreglo de Tipo TData

Resultado: per

Relaciones y Subproblemas:

Primero se fija que el arreglo no esté lleno mediante la función “Llena”. En caso que lo este se informará mediante un mensaje que no se podrá cargar la información de la persona.

En caso que no lo esté, se pedirá como entrada el nombre, DNI, edad de la persona y se la cargará al final del arreglo.

Cuando selección sea 2 (selección = 2): se invocará “SuprimirPersona(personas)”

Análisis de la Acción SuprimirPersona

Dato: per //arreglo de Tipo TData pasado como parámetro Dato-Resultado.

Resultados: per

Relaciones y Subproblemas:

Esta acción lo que nos permite es suprimir la información de la persona que se encuentra en el primer lugar del arreglo. Para hacerlo, pasa la información que existe de la última persona al principio y decrementa el per.cant en 1 (per.cant-1)

Si el arreglo está vacío no se podrá eliminar nada, asique se informará mediante un mensaje que no existe ningún elemento cargado. (para saber si está vacío utiliza la función “vacía” la cual devuelve verdadero si el arreglo no tiene ningún nombre cargado)

Cuando selección sea 3 (selección = 3): se invocará “Mostrar(personas)”

Análisis De la Acción Mostrar

Dato: per

Resultado: Mostrará la información de todas las personas que se encuentren en el arreglo.

Relaciones y Subproblemas:

Primero se verificará si el arreglo se encuentra vacío (mediante la función vacía):

- En caso de que lo esté: se informará mediante un mensaje que no existen personas cargadas.
- En caso de que no esté vacío: se mostrará en pantalla la información de todas las personas que se encuentran ingresadas en el arreglo “per”. Usando una estructura iterativa que vaya incrementando una variable de control hasta que se llegue al final del mismo.

Cuando selección sea 4 (selección = 4): Primero se verá si el arreglo “Personas” no está vacío, en caso de estarlo se indicará con un mensaje que no se puede ejecutar la opción ya que no existe información de personas cargados. En caso que no lo esté:

A la variable tipo puntero a TNodo (que contiene 2 campos, info que será tipo TPers y next que será un campo puntero a TNodo) se lo inicializará apuntando a “NIL”.

listaM ← nil

Luego se invocará a la función ListaMenores que el puntero que devuelva sera guardado en la variable listaM

listaM ← ListaMenores(personas, personas.cant, listaM)

Por último se ejecutará una acción que mostrará toda la lista previamente conseguida gracias a la función invocada.

MostrarMenores(listaM)

//MostrarMenores es una acción que da como salida la información almacenada en la lista creada anteriormente mediante la función recursiva ListaMenores

Análisis de la Función recursiva ListaMenores

Dato: per //parámetro tipo dato donde será el arreglo TData

cant //parámetro tipo dato, el cual tendrá la cantidad total de personas cargadas en el arreglo

lis //parámetro de tipo dato que apunta a la cabeza de la lista creada mediante la recursión de la acción

Resultado: puntero a TNodo //la cabeza de la lista creada en la función

Relaciones y Subproblemas:

Esta función es recursiva:

- Como caso base, se tiene: (cant = 0) con lo cual devolverá "lis" que sera la variable tipo puntero que apuntará a la cabeza de la lista creada mediante la recursión.

- Como etapa inductiva se tiene (cant > 0)

Aquí existen 2 caminos, en caso que la edad de la persona que se encuentra en la posición "cant" del arreglo sea menor a 18 años: se ejecutara la acción "CargarLista" la cual recibe como parámetros tipo dato el arreglo "per" y el valor que tenga "cant", también recibe como dato-resultado el puntero "lis", que al terminar con la ejecución de la acción apuntara a la cabeza de la lista donde se almaceno la info de la persona menor de edad.

Al terminar la ejecución de la acción "CargarLista", se volverá a invocar la función ListaMenores pero esta vez decrementándole el cant en 1 (cant-1)

En caso que la persona en la posición cant del arreglo tenga 18 años o sea mayor a 18, se invocará la función recursiva, pero decrementando el cant en 1 (cant-1)

Si selecciona la opción 5 (selección = 5), primero se comprueba si el arreglo "personas" está vacío, en caso de no estarlo, se realiza la llamada a la acción "OrdenarDNI" para ordenarlo, luego se solicita la entrada del dni a buscar, que será usado en la llamada de la función "BusquedaDNI".

En caso de que el arreglo estuviera vacío, se informa

Análisis De la Acción OrdenarDNI

Dato-resultado: per

Resultado: Arreglo per ordenado

Relaciones y Subproblemas: Ordena mediante Bubblesort el arreglo, usando el campo dni

Análisis De la Acción BusquedaDNI

Dato: per y dni

Resultado: Tpersona

Relaciones y Subproblemas: Esquema de búsqueda dicotómica, el arreglo debe estar previamente ordenado.

Si el elemento NO existe lo informa, en caso de que el elemento exista en el arreglo, muestra todos los campos del elemento cuyo dni es igual al dni buscado.

Cuando inserte cualquier número de 1 a 7, se volverá a ejecutar el menú. Cuando inserte 0 a la variable “term” se le asignará verdadero (term ← verdadero). Guardará el arreglo en una archivo y Terminará el programa

Al seleccionar la opción 6 (seleccion = 6), se revisa que el arreglo no esté vacío y entonces es invocada la función “mayoresQueElPrimero” con el arreglo como parámetro.

Se informa si hay 3 personas mayores al primer individuo o si no hay personas mayores al primero. Además se informa si el arreglo personas está vacío.

Análisis De la Acción mayoresQueElPrimero

Dato: per y dni

Resultado: Verdadero / Falso

Relaciones y Subproblemas: Se utiliza el esquema de Recorrido Parcial con marca final. recorre el arreglo de elementos incrementando un contador cuando el campo “edad” del elemento corriente es mayor que el primero. Si se encontraron 3 elementos con esta propiedad devuelve verdadero y sino devuelve falso

Si selecciona la opción 7 (seleccion = 7), primero se comprueba que el arreglo no esté vacío y luego se asigna en el elemento pri, los campos del primer elemento del arreglo personas, esto es necesario para invocar a la función edadMayor con los parámetros per, cant y aux.

Análisis De la Acción edadMayor

Dato: per, cant, aux

Resultado: TPers

Relaciones y Subproblemas: recursivamente compara el campo edad de aux con los elementos de la lista, hasta devolver el de mayor edad.

Por último Si selecciona la opción 0 (selección = 0):

Primero se comprueba que el arreglo no esté vacío, en caso de no estarlo se Invocara la acción Guardar:

Análisis De la Acción Guardar

Dato: per, g //donde per es el arreglo y g la variable con la que le daremos nombre interno al archivo.

Resultado: Almacenará en el archivo Personas.dat la información de las personas cargadas.

Relaciones y Subproblemas:

Primero se abrirá el archivo en modo escritura, osea se reescribirá todo el archivo “Personas.dat” borrando los elementos almacenados anteriormente. Dándole como nombre interno la variable “g” (Abrir(“Personas.dat”, g, e)).

Se inicializa una variable de control “i” en 1 (i ← 1) utilizada en un ciclo. Este irá ciclando hasta que la condición sea falsa, es decir hasta que (i <= per.cant) sea falso. Dentro de esta estructura se irá almacenando en el archivo (mediante la primitiva “Escribir(g,aux)) la información de cada persona del arreglo. La variable que nos ayudará para esto será “aux” la cual es de tipo TPersona al igual que el archivo y en cada ciclo se le dará una nueva información de cada persona correspondiente, agregando en el campo borrado un 0 siempre (aux.borrado ← 0).

En caso que el arreglo esté vacío no se ejecutará esta acción.

y por último independientemente de que si el arreglo esta vacío o no a la variable term se le asignará verdadero (term ← verdadero) con lo cual nos permitirá terminar con la estructura iterativa inicial culminando con el programa.

Diseño

Algoritmo TPrFinal

Léxico

Max = 1000

TPersona = <nombre ∈ Cadena, dni ∈ Z, edad ∈ (1..80), borrado ∈ Z>

TPers = <nombre ∈ Cadena, dni ∈ Z, edad ∈ (1..80)>

TArreglo = arreglo [1..Max] de TPers

TData = < info ∈ TArreglo , cant ∈ (0..Max) > // per.info[n].edad y per.cant

TNodo = < info ∈ TPers, next puntero a TNodo>

// VARIABLES UTILIZADAS!

per ∈ TData //arreglo que tendrá la info

//Función Arreglo Vacío

Función Vacía (dato per ∈ TData) ⟶ Lógico

Inicio

Si (per.cant = 0) entonces

⟵ verdadero

Sino

⟵ falso

fsi

ffunción

//Función Arreglo Lleno

Función Llena(dato per ∈ TData)⟶ Lógico

Inicio

Si (per.cant = 1000) entonces

⟵ verdadero

Sino

```
                ← falso
        fsi
ffuncion

//Acción que Inserta la información de una Persona en el arreglo
Acción InsetarAlFinal(dato-resultado per ∈ TData)
Léxico Local
persona ∈ TPersona
msg ∈ Cadena
Inicio
        Si (Llena(per)) entonces
                msg ← "No es posible insertar la información de la persona ya que el arreglo está lleno"
                Salida: msg
        sino
                //Ingresa la información de la persona que quieres agregar:
                Entrada: persona.nombre
                Entrada: persona.dni
                Entrada: persona.edad
                per.cant ← per.cant + 1
                per.info[per.cant].nombre ←  persona.nombre
                per.info[per.cant].dni ←  persona.dni
                per.info[per.cant].edad ←  persona.edad
                per.info[per.cant].borrado ← 0
                msg ← "Informacion Cargada con éxito"
                Salida: msg
        fsi
facción

//Acción que Elimina la primera Persona del arreglo. last to first
Acción SuprimirPersona (dato-resultado per ∈ TData)
Inicio
Si (Vacía(arreglo)) entonces
```

```
        msg ← "No se pudo suprimir ningún nombre, porque el arreglo está vacío"
Sino
        per.info[1].nombre ← per.info[nom.cant].nombre
        per.info[1].dni ← per.info[nom.cant].dni
        per.info[1].edad ← per.info[nom.cant].edad
        //ver si agregarle el campo borrado, definir el tipo de TPersona
        per.cant ← per.cant – 1
        msg ← "Información suprimida con éxito!"
        Salida: msg
fsi
facción

//Acción que muestra la información de TODAS las personas del arreglo
Acción Mostrar (dato per ∈ TData)
Léxico Local
  msg ∈ Cadena
Inicio
        //ver si poner el campo borrado
        Para (i ← 1, i <= per.cant, i ← i + 1) hacer
                Mostrar: per.info[i].nombre
                Mostrar: per.info[i].edad
                Mostrar: per.info[i].dni
        fpara
faccion
```

INCISO g)

```
Acción Cargar (dato-resultado per ∈ TData, dato f ∈ archivo de TPersona)
Léxico Local
  aux ∈ TPersona
Inicio
  Abrir("Personas.dat", f, l)        //Abre archivo "Personas.dat" en modo lectura dándole el nombre interno f.
```

```
  i ← 0
  Mientras (no EOF(f)) hacer
        Leer(f, aux)
//verdadero si la info de la persona está borrada, falso no lo esta
        Si (aux.borrado = 0 ) entonces
              i ← i + 1
              per.info[i].nombre ← aux.nombre
              per.info[i].edad ← aux.edad
              per.info[i].dni ← aux.dni
        fsi
  fmientras
  Cerrar(f)
facción
```

INCISO h)

```
Acción Guardar (dato per ∈ TData, dato g ∈ archivo de TPers)
Léxico Local
  aux ∈ TPersona
  i ∈ Z
Inicio
  Abrir("Personas.dat", g, e)   //Reescribiendo el archivo Personas.dat
  i ← 1
  Mientras (i <= per.cant) hacer
        aux.nombre ← per.info.[i].nombre
        aux.edad ← per.info.[i].edad
        aux.dni ← per.info.[i].dni
        aux.borrado ← 0 // hardcoded
        Escribir(g, aux)
        i ← i + 1
  fmientras
  Cerrar(g)        //Cierra el archivo
facción
```

INCISO i)

//Esta acción la utilizara la función recursiva "ListaMenores" para poder cargar a la lista las personas mayores a 18 años

Acción CargaALista (dato per ∈ TData, i ∈ (0..Max), dato-resultado aux ∈ puntero a TNodo)

Léxico Local

x ∈ Puntero a TNodo

Inicio

Obtener(x)

(^x).info.nombre ← per.info[i].nombre

(^x).info.edad ← per.info[i].edad

(^x).info.dni ← per.info[i].dni

(^x).next ← aux

aux ← x

faccion

//Antes de invocar a la función, aux debe apuntar a nil.

Función ListaMenores (dato per ∈ TData, cant ∈ (0..Max), lis ∈ puntero a TNodo) -> puntero a TNodo

Inicio

Según

(cant = 0): ← lis *//Caso base*

(cant > 0): *//Etapa inductiva*

Según

(per.info[cant].edad < 18): CargarLista(per, cant, lis)

← ListaMenores (per, cant-1, lis)

(per.info[cant].edad >= 18):

← ListaMenores (per, cant-1, lis)

fsegún

fsegún

ffunción

//Esta acción permitirá mostrar la información de la lista con nombre, edad y dni de los menores.

Accion MostrarMenores(dato lis ∈ puntero a TNodo)

Léxico Local

Inicio

Según

(lis = nil): msg ← “La lista creada de menores se encuentra vacía”

Salida: msg

(lis <> nil)

Mientras (lis <> nil) hacer

Salida: lis.info.nombre

Salida: lis.info.edad

Salida: lis.info.dni

lis ← (^lis).next

fmientras

fsegun

faccion

INCISO j)

//Acción utilizada en “bubblesort” para intercambiar los valores del arreglo al ordenarlo

Acción Swap (dato-resultado x,y ∈ TPers)

Léxico local

temp ∈ TPers *//variable utilizada para poder hacer el intercambio*

Inicio

temp ← x

x ← y

y ← temp

faccion

// Acción que ordena el arreglo mediante “bubblesort”

Acción OrdenarDNI (dato-resultado per ∈ TData)

Léxico Local

i, j ∈ (1..per.cant+1)

```
        term ∈ Z        //esta variable será utilizada para poder terminar el ciclo si ya no se necesita realizar ningún intercambio

Inicio

        i <- per.cant

        term ← 1

        Mientras (i > 1 y term <> 0) hacer

                j ← 1

                term ← 0

                Mientras (j < i) hacer

                        si (per.info[j].dni > per.info[j+1].dni) entonces

                                Swap(per.info[j], per.info[j+1])

                                term ← term + 1

                        fsi

                        j ← j+1

                fmientras

                i ← i-1

        fmientras

facción


//Acción de Búsqueda Dicotómica

Acción BusquedaDNI(dato per ∈ TData,dato dni ∈ Z)

Léxico local

        k, inf, sup ∈ (1..Max)

        msg ∈ Cadena

Inicio

según

_____//Si el dni es menor al 1er elemento o mayor al último, no se encuentra en el arreglo

        ((dni<per.info[1].dni) o (dni>per.info[per.cant].dni)):

                        msg ← “El Elemento no existe en el arreglo”

                        Salida: msg

         //Si el dni es mayor o igual al primer elemento y

        (per.info[1].dni <= dni y dni <= per.info[per.cant].dni):
```

```
                inf ← 1
                sup ← per.cant
                mientras inf<sup hacer
                        k ← (inf+sup) div 2   //toma la mitad
                        según
                            (dni>per.info[k].dni): inf ← k+1 // derecha
                            (dni<=per.info[k].dni): sup ← k //izquierda
                        fsegún
                fmientras
                según
                    (per.info[inf].dni = dni):    Salida:  per.info[inf].nombre //elem encontrado
                                                  Salida:  per.info[inf].edad
                                                  Salida:  per.info[inf].dni
                    (per.info[inf].dni<>dni):     msg ← "No se ha encontrado ninguna Persona con ese DNI"
                                                  Salida:  msg // elemento no encontrado
                fsegún
fsegún
ffaccion
```

INCISO K)

```
//Antes de invocar a la función chequear que el arreglo no esté vacío.
Funcion mayoresQueElPrimero (dato per ∈ TData) → Lógico
Léxico Local
i ∈ Z
cantMayor, edad ∈ Z
Inicio
    i ← 1
    cantMayor ← 0
    pri ← per.info[i].edad
    i ← i + 1
```

```
Mientras (i < per.cant+1 y cantMayor < 3 ) hacer
    Si (per.info[i].edad > pri) entonces
        cantMayor ← cantMayor + 1
    fsi
    i ← i + 1
fmientras
    Según
        (i < per.cant+1): ← verdadero // cantMayor = 3
        (i = per.cant+1):                    // fin sec
                Si (cantMayores < 3) entonces
                    ← falso
                sino
                    ← verdadero
                fsi
    fsegun
ffuncion
```

INCISO L)

```
{pre-cond: aux = per.info[1]}
{pre-cond: per.cant = cant}
Función edadMayor (dato per ∈ TData, cant ∈ (0..Max) ,dato  aux ∈ TPers) → TPers
Inicio
  Según
    (cant = 1): ← aux                          //Caso base
    (cant > 1):  _____// Etapa Inductiva
            si (aux.edad < per.info[cant].edad) entonces
                aux.nombre ← per.info[cant].nombre
                aux.dni ← per.info[cant].dni
                aux.edad ← per.info[cant].edad
            fsi
            ← edadMayor(per, cant-1 ,aux)
```

fsegun

ffuncion

// Muestra las opciones del menú, estática

Acción Opciones ()

Léxico local

msg ∈ cadena

Inicio

msg ←

" MENÚ DE OPCIONES

1 .Insertar al final.

2. Suprimir el primero.

3 .Mostrar todos.

4 .Mostrar menores.

5. Buscar por DNI.

6. Mayores al primero.

7. Edad mayor

0. Guardar y salir "

Salida: msg // Mostrar

facción

//Accion del Menu de Opciones que permitirá al usuario elegir qué operación hacer

Acción Menu (personas ∈ TData)

Léxico Local

term ∈ Lógico //variable utilizada para salir del ciclo

lista ∈ puntero a TNodo //puntero a la cabeza de lista

f ∈ archivo de TPersona //nombre interno que se le dará al archivo Personas.dat

g ∈ archivo de TPers //nombre interno que se le dará al archivo SacaChispas.dat

msg ∈ Cadena //variable que utilizaremos para informar mensajes

listaM ∈ Puntero a TNodo //variable que utilizada para lista de menores de edad

dni ∈ Z //variable utilizada para la entrada del DNI a buscar

```
mayor ∈ Lógico                  //variable utilizada para guardar lo que devolverá la función mayoresAlPrimero
pri ∈ TPers    //var utilizada para guardar el primer campo del arreglo y comparar las edades
selec ∈ (0..7)
dni ∈ Z
Inicio
term ← falso
 opciones()
Mientras(no(term)) hacer
    Entrada : selec____
  Según
      (seleccion = 0):    //Opción De Guardar y Salir
                          Guardar(personas, g)
                          term ← verdadero
      (seleccion = 1):    //Insertar Nombre Al Final del arreglo
                           InsertarAlFinal(personas)
      (seleccion = 2):    //Eliminar primera persona del arreglo
                          Si (no Vacía(personas)) entonces
      _________________SuprimirPersona(personas)
                          sino
                             msg ← "El Arreglo Está Vacío"
                             Salida: msg
                          fsi
      (seleccion = 3):    //Mostrar todo el arreglo
                          Si (no Vacía(personas)) entonces
      _________________Mostrar(personas)
                          sino
                             msg ← "El Arreglo Está Vacío"
                             Salida: msg
                          fsi
      (seleccion = 4):    //Mostrar info de personas menores de edad
                          Si (no Vacía(personas)) entonces
                             listaM ← nil
```

listaM ← ListaMenores(personas, personas.cant, listaM)

MostrarMenores(listaM)

sino

msg ← “El Arreglo Está Vacío”

Salida: msg

fsi

(seleccion = 5): //*Buscar por DNI*

Si (no Vacía(personas)) entonces

OrdenarDNI(personas)

Entrada: dni

BusquedaDNI(personas, dni)

sino

msg ← “El Arreglo Está Vacío”

Salida: msg

fsi

(seleccion = 6): //*Informar si hay por lo menos 3 personas mayores al primero*

Si (no Vacía(personas)) entonces

si (mayoresQueElPrimero(personas)) entonces

msg ← “hay 3 personas mayores al primer individuo del arreglo”

sino

msg ← “no hay 3 personas mayores al primer individuo del arreglo”

fsi

sino

msg ← “El Arreglo Está Vacío”

fsi

Salida: msg

(seleccion = 7): //*Mostrar la información de la persona con mayor edad del arreglo*

Si (no Vacía(personas)) entonces

pri.nombre ← personas.info[1].nombre

pri.edad ← personas.info[1].edad

pri.dni ← personas.info[1].dni

Mostrar(edadMayor(personas,personas.cant,pri))

sino

msg ← “El Arreglo Está Vacío”

Salida: msg

fsi

fsegún

fmientras

facción

INICIO

Menú(per)

FIN

Cosas a Realizar:

- ☑ ~~Análisis~~

PseudoCódigo Parte del TP Integrador 1er Cuatrimestre:

- ☑ ~~Funcion Vacia~~
- ☑ ~~Funcion Llena~~
- ☑ ~~Accion InsetarAlFinal~~
- ☑ ~~Accion SuprimirPrimerNombre~~
- ☑ ~~Accion Mostrar~~
  - ☑ ~~Chequeado?~~
  - ☑ ~~Funcionan?~~

PseudoCódigo TP FINAL

- **Rediseñar los anteriores Incisos**
  - ☑ ~~Hecho~~
  - ☑ ~~Chequeado~~
  - ☐ **Funciona**
  - ☐ **Prueba**

- **Acción para cargar datos de un archivo al arreglo**
  - ☑ ~~Hecho~~
  - ☑ ~~Chequeado~~
  - ☐ **Funciona**
  - ☐ **Prueba**

- **Acción para guardar los datos del arreglo en un archivo**
  - ☑ ~~Hecho~~
  - ☑ ~~Chequeado~~
  - ☐ **Funciona**
  - ☐ **Prueba**

- **Función recursiva que cree una lista almacenando a las personas menores de 18 años**
  - ☑ ~~Hecho~~

- ☑ ~~Chequeado~~
- ☐ Funciona
- ☐ Prueba

- **Accion Búsqueda Dicotómica de un DNI**
  - ☑ ~~Hecho~~
  - ☐ Chequeado
  - ☐ Funciona
  - ☐ Prueba
  - **Acción Ordenar el arreglo (BubbleSort)**
    - ☐ Hecho
    - ☐ Chequeado
    - ☐ Funciona
    - ☐ Prueba

- **Función que indique si hay por lo menos tres personas que sean mayores a la persona que está en primer lugar del arreglo.**
  - ☑ ~~Hecho~~
  - ☑ ~~Chequeado~~
  - ☐ Funciona
  - ☐ Prueba

- **Función recursiva que diga devuelva la información de la persona con mayor edad.**
  - ☑ ~~Hecho~~
  - ☐ Chequeado
  - ☐ Funciona
  - ☐ Prueba