Modularización del Software

1. El código en C se organiza en archivos .c y .h. Para cada archivo .c debe existir un .h con el mismo nombre (Person.c, Person.h). A esta pareja de archivos se la denomina **módulo**.

2. Cada módulo tiene una interfaz pública definida en su .h, y una serie de variables y funciones privadas escondidas dentro del .c.

3. Las declaraciones de variables y funciones públicas se ponen en el .h para que otros módulos puedan incluirlas (#include “People.h”) y utilizarlas

4. Las declaraciones de variables y funciones privadas quedan dentro del .c definidas como **static** para no permitir que estas funciones sean utilizadas desde otros módulos.

5. Los módulos contendrán de forma natural conjuntos de funciones relacionadas desde un punto de vista lógico.

6. Los nombres de las funciones públicas contenidas en el módulo comenzarán con el con el nombre del módulo en minúscula seguido por el guión bajo (Ej. **person_**setEdad)

7. Cada archivo .c debe incluir a su propio .h (#include Person.h).

8. Los archivos .h deben contener sólo declaraciones públicas: tipos, constantes, y prototipos de funciones diseñados para ser usados desde fuera del módulo.

9. Todo archivo .h debe evitar inclusiones múltiples.

```
#ifndef PERSON_H
#define PERSON_H
... interfaz pública del módulo…
 #endif /* PERSON_H */
```

10. Cuando se realice la implementación de tipos de datos abstractos (**structs**) definidos por el programador, todas las funciones que acceden al tipo definido deberán estar contenidas en el mismo fichero.

11. Las funciones **set:** deberá existir como mínimo una función de este tipo por cada campo del tipo de dato abstracto (o entidad) administrado por el módulo. Ej: person_setName(Person* this, char* value), recibe como parámetro un puntero a la persona y el String (nombre de la persona) que asigna luego de realizar las validaciones correspondientes al campo name de la persona recibida como parámetro.

12. Las funciones **get:** deberá existir como mínimo una función de este tipo por cada campo del tipo de dato abstracto (o entidad) administrado por el módulo. Ej: person_getName(Person* this), recibe como parámetro un puntero a la persona y retorna un puntero al nombre de la misma.

Archivo: Person.c

```c
#include <stdlib.h>
#include "Person.h"

/* Tipo de dato privado */
typedef struct
{
    int age;
    int something;
}_Person;

/* Funciones Privadas */
static void initialize(Person* this, int age, int something);


/** \brief  Reserva espacio en memoria para una nueva persona y la inicializa
 *
 * \param int age Edad de la persona
 * \param int something Otros datos
 * \return Person* Retorna un puntero a la persona o NULL en caso de error
 *
 */
Person* person_new (int age, int something)
{
    Person* this = malloc(sizeof(_Person));

    if(this != NULL)
        initialize(this,age,something);

    return this;
}

/** \brief  Inicializa a una persona recibida como parámetro
 * \param Person* this Puntero a la persona
 * \param int age Edad de la persona
 * \param int something Otros datos
 * \return void
 *
 */
static void initialize(Person* this, int age, int something)
{
    _Person* _this = (_Person*)this;

    _this->age = age;
    _this->something = something;
}
```

```
/** \brief  Setea la edad de una persona recibida como parámetro
 * \param Person* this Puntero a la persona
 * \param int age Edad de la persona
 * \return void
 *
 */
void person_setAge(Person* this, int age)
{
    _Person* _this = (_Person*)this;

    if(age > 0)
        _this->age = age;
}

/** \brief Obtiene la edad de una persona recibida como parámetro
 * \param Person* this Puntero a la persona
 * \return int age Edad de la persona
 *
 */
int person_getAge(Person* this)
{
    _Person* _this = (_Person*)this;
    return _this->age;
}


/** \brief Libera el espacio ocupado por una persona recibida como parámetro
 * \param Person* this Puntero a la persona
 * \return void
 *
 */
void person_free(Person * this) {
    free(this);
}
```

Archivo: Person.h

```
#ifndef PERSON_H_INCLUDED
#define PERSON_H_INCLUDED
typedef struct
{
    //Tipo de dato público
}Person;

Person* person_new (int age, int something);
void person_setAge(Person* this, int age) ;
int person_getAge(Person* this);
void person_free(Person * this);

#endif //PERSON_H_INCLUDED
```

Archivo: main.c

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


int main()
{
   Person* personArray[50];

   int i;
   for(i = 0; i < 10; i++)
   {
      personArray[i] = person_new(i,i);
   }
   for(i = 0; i < 10; i++)
   {
      person_setAge(personArray[i],i-4);
   }
   for(i = 0; i < 10; i++)
   {
      printf("\nAge: %2d",person_getAge(personArray[i]));
   }
   scanf(" ");
   return 0;
}
```