Correccion Bruno

- No hace falta poner el nombre de la función en las pre/post condiciones.

- La siguiente función :

```
bool cumple_minimo_edad(unsigned int edad){
  if (MINIMO_EDAD<=edad){
    return CUMPLE_EDAD;
  }
  else{
    return (NO_CUMPLE_EDAD);
  }
}
```

Podrias poner directamente:

```
bool cumple_minimo_edad(unsigned int edad){
  return MINIMO_EDAD<=edad;
```

Lo mismo con el resto de las funciones booleanas, podrias devolver directamente la comparación, y eso ya es `true` o `false` de por si.

- Los nombres de las funciones booleanas deberian ser nombres

interrogativos. Por ejemplo `controlar_dato_entero` podria llamarse `es_dato_entero` o `es_dato_entero_valido`. De esta manera uno puede deducir leyendo la firma de la función en que casos devuelve `true` y en cuales devuelve `false` y es más intuitivo. Lo mismo con el resto de las funciones booleanas.

- Las contantes DATO_CORRECTO, DATO_INCORRECTO, ES_MEJOR_AL_CAMPEON, etc. No son necesarios. Se puede usar directamente `false` o `true` según sea necesario.

- Por ejemplo, siguiendo los consejos anteriores, la siguiente comparación:

```
      if(controlar_dato_caracter(*capacidad_magica)==DATO_
INCORRECTO){
      }
```

Podria ser :

```
if(! es_caracter_valido (*capacidad_magica))
```

De esta manera, leyendo el if uno puede deducir que lo que devuelve la función es un valor booleano, y que devuelve true cuando el usuario ingresa un caracter valido, y false en caso contrario. Del otro modo, a simple vista uno no sabe que valor devuelve `controlar_dato_caracter` ni mucho menos que tipo de dato es `DATO_INCORRECTO`. Ahi estás enmascarando la función booleana y

haciendola menos natural de percibir. ¿Me explico?

- En las precondiciones no es necesario especificar el tipo de dato de la función. Eso se puede deducir directamente de la firma de la función:

Por ejemplo la función :

```
bool controlar_dato_caracter(char dato)
```

Se deduce que dato es un caracter por el tipo de dato `char`.

- La postcondicion :

```
    POSTCONDICIONES:
      El valor que quede va a estar entre el 0 y el 10
```

Quedaría mejor:

```
    Le pide un valor al usuario hasta que ingrese un valor
 entre 0 y 10 y lo guarda dentro de la variable fuerza
```

Lo mismo con :

```
     POSTCONDICION: Si es mayor de edad, devuelve que la c
umple
```

Quedaría mejor:

```
   Si la edad recibida como parámetro es mayor de edad, de
vuelve verdadero, en caso contrario devuelve false
```

El trabajo práctico quedó muy bien. Modularizaste bien, usaste bien el tema de pasaje por valor y referencia, pusiste pre y post condiciones en todas las funciones. Sin embargo habria que reveer algunas cosas relacionadas con el tema de las pre y post condiciones, hay pre condiciones que no son necesarias, y post condiciones que son poco claras.
El tema de las funciones booleanas, si son interrogativas se autodescriben mucho mejor.
Recordá además no enmascarar los valores `true` y `false` con nombres de constantes.

Tu tp es merecedor de un increible 8. ¡Felicitaciones!