Un programa que cuente los pasos para una app rastreador de actividad

Has trabajado muy duro y tu reputación te precede, así que la famosa compañía `BeActive` te está invitando a escribir un software para su nuevo rastreador de actividad: `RunTracker`. A estas alturas, cuentas con la experiencia necesaria trabajando con este tipo de dispositivos e incluso ya tienes algunos fragmentos de código.

El cliente ha elaborado un brief que describe los pasos que tendrás que seguir para poder completar este proyecto satisfactoriamente.

Descripción

Un software para procesar datos para el rastreador de actividad `RunTracker` de la compañía `BeActive`.

Descripción general

Características del módulo:

- Recibe y verifica paquetes de datos entrantes con la siguiente estructura: `package = {<tiempo en segundos>, <pasos>}`. Por ejemplo: `package = {14000, 15000}`.
- Los conteos de segundos es por dia, es decir comienza desde cero hasta finalizar el dia y luego se reinicia.
- Guarda y procesa estos datos en la funcion y en el vector global.
- Genera un resumen para el periodo desde que empieza el día hasta el momento en que se recibió en el paquete de datos.

Formato de salida resumido:

```
Tiempo: <tiempo, en format HH:MM:SS, del paquete de datos recibidos>.
Pasos dados hoy: <suma de los pasos dados desde el principio del día en
curso>.
La distancia fue <suma de pasos desde el inicio del día en curso en km> km.
Quemaste <número de calorías quemadas desde el inicio del día en curso>
calorias.
<Un mensaje motivacional dependiendo de la distancia recorrida>.
```

Lista de mensajes motivacionales dependiendo de la distancia caminada por el usuario:

```
6.5 km o más: ¡Gran entrenamiento! Objetivo cumplido.
3.9 km y más: ¡Nada mal! Hoy ha sido un día productivo.
2 km o más: Menos que el resultado deseado, ¡pero intenta alcanzarlo mañana!
Menos de 2 km: Está bien tomarse el día de descanso. No siempre se puede
ganar.
```

- Devuelve los datos guardados para que sea posible trabajar con ellos más adelante en otras aplicaciones.

Datos entrantes

La funcion recibe paquetes de datos en forma de vectores desde el chip del controlador.

Los paquetes se transfieren al programa cuando un usuario accede al rastreador (a través del botón en la interfaz del usuario).

Orden de los valores en el paquete de datos:
`{<time>, <steps>}`

- `<time>`: tiempo de creación del paquete en segundos desde el inicio del dia; valor del tipo `long`;
- `<steps>`: el número de pasos dados por el usuario desde el último acceso del usuario; un valor del tipo `int`

Podría haber un error al transferirse los paquetes. Debes tomar esto en cuenta en el programa. Cuando llega un paquete, compruébalo y envíalo para su procesamiento únicamente después de la comprobación.

Posibles errores al recibir paquetes:

1. El paquete contiene informacion invalida, por decir no logro marcar el tiempo y nos da un valor de 0 segundos. Otro error podria ser que no marco los numero de pasos y nos da 0 pasos.
2. El valor tiempo en el paquete transmitido es menor que o igual al valor registrado anteriormente (el tiempo solo se registra a lo largo de un día).

Resultado de la ejecución del programa

1. Los paquetes recibidos deben almacenarse en el vector global `storage_data` (almacenamiento de datos). Este vector tiene el tipo de datos `package` y este a su vez tiene los campos, `int steps` y `long seconds`.
2. Se debe mostrar un mensaje en la terminal, por ejemplo:

```
Tiempo: 09:36:02.
Pasos dados hoy: 15302.
La distancia recorrida fue 9.95 km.
Has quemado 1512.00 cal.
¡Qué buen entrenamiento! Objetivo cumplido.
```

3. La funcion debe devolver el vector `storage_data` para que más adelante se pueda trabajar con estos datos en otros programas.

Detalles de la implementación

No tienes que escribir la aplicacion desde cero: tú ya tienes unos headers con constantes y algunos prototipos de funciones.

Necesitas escribir el cuerpo de estas funciones declaradas, determinar el mejor lugar para llamarlas y crear las que creas conveniente.

Recuerda, faltan algunas funciones, y tú debes escribirlas. Una de estas funciónes debe llamarse `show_message()`. Como un argumento, esta función debe recibir todos los valores que se requieren para generar el mensaje: `(tiempo de acceso, número de pasos, distancia, calorías quemadas, felicitaciones)`. La función debe crear un mensaje que use estos valores y se muestre en la terminal.

Punto de entrada del programa

La función `accept_package()` (aceptar paquete) del manejo de paquetes es el punto de entrada al programa, es decir, la función que se llama primero en el main. Esta acepta un paquete de datos como un argumento. La función `accept_package()` debe devolver el vector `storage_data` que contiene los datos añadidos del paquete recibido.

Desde esta función se llaman otras funciones con sus propios conjuntos de instrucciones una por una. Inmediatamente después del inicio, se debe ejecutar la función `check_correct_data()` (comprobar datos correctos) para comprobar si el paquete recibido es correcto y devuelve `True` o `False`, la cual llevará a cabo la ejecución de la función base.

Asegúrate de crear los prototipos de tus funciones en el header correspondiente, `runner.h` y el cuerpo de estas funciones en el source file correspondiente, `runner.cpp`. Recuerda de incluir los headers correspondientes en el main.

Revisa el header `constants.h` para verificar las constantes que tenemos a disposicion.

Detalles útiles

El número de pasos, la distancia en kilómetros, y el número de calorías quemadas se calculan para el periodo desde el inicio del día (comenzando 0:00:00) hasta la hora en que se recibió el nuevo paquete de datos. Cada nuevo día, todos los datos se reinician y los cálculos comienzan de nuevo.

Tus conocimientos sobre ramas y operadores lógicos te serán útiles para implementar las funciones `check_correct_data(data)` y `check_correct_time(seconds)`.

En la función `get_step_day(steps)` (obtener pasos del día), debes iterar sobre el vector en un bucle. La función debe devolver el número total de pasos para el día actual.

Si la función `get_distance(steps)` (obtener distancia), convierte los pasos en kilómetros. La función debe devolver la distancia en kilómetros. Use la siguiente ecuacion:

$$distance = \frac{steps \cdot STEP_M}{1000}$$

La función `get_calories_burned(dist, current_time)` (obtener las calorías quemadas) debe calcular y devolver el número de calorías quemadas durante el día actual. Utiliza la siguiente ecuacion:

$$calBurned = (K_1 \cdot WEIGHT + (\frac{MEANSPEED^2}{HEIGHT}) \cdot K_2 \cdot WEIGHT) \cdot MINUTES$$

En donde la velocidad media se calcula $dist/hour$.

Añade el código para seleccionar el mensaje motivacional a la función el cual depende de la distancia recorrida. `get_achievement(dist)` (obtener logro). La función devuelve el mensaje en lugar de mostrarlo.

Cree una funcion que se llame `get_time_hms`. Esta funcion recibe los segundos del sensor y devuelve una estructura `times` que tiene como campos las `horas`, `minutos` y `segundos`. Por ejemplo, si los segundos obtenidos del sensor es 3600, esto equivale a una hora, cero minutos y cero segundos en la estructura `times`.

Note el formato de tiempo. Por lo que debe crear un funcion que le ayude a obtener este formato dado que la entrada son los segundos transcurridos durante el dia y la salida de esta funcion debe ser una string con el formato `HH:MM:SS`.

Ejemplos de salida:

En la funcion main ya estan definidas algunos valores de entrada a la funcion `accept_package`. Por favor, no modificar estos valores. El resultado esperado para estos

valores son:

```
Tiempo: 02:00:01.
Pasos dados hoy: 505.
La distancia fue 0.33 km.
Has quemado 315.04 cal.
Está bien tomarse un día libre. No siempre se puede ganar.


Tiempo: 09:36:02.
Pasos dados hoy: 15505.
La distancia fue 10.08 km.
Has quemado 1519.89 cal.
¡Qué buen entrenamiento! Objetivo cumplido.
```

Los resultados numericos pueden variar en decimales.

Entregables

Se debe entregar un unico folder en zip o rar. Dentro de este comprimido se debe encontrar lo siguiente:

- Analisis del problema detallado
- Pseudo codigo con pruebas de entradas y salidas. Debe realizar pruebas en papel o cualquier otro software que vea conveniente con los valores presentados en el documento main.cpp
- Los headers y sources files completos
- La funcion main.cpp

Adicional

Sientase libre de agregar cualquier funcionanidad que vea interesante. Esto le dara puntos extras.