Tema 1 - To pit or not to pit... this is the strategy

- Deadline: 19.04.2023
- Data publicării: 01.04.2023
- Ultima actualizare a enuntului:
    - 01.04.2023: publicarea enuntului
    - 07.04.2023: clarificare Memory Management
- Responsabili:
    - Timotei Daraban
    - Mircea Preoteasa
    - Rares Constantin
    - Robert Adam
    - Serban Sorohan

Poveste

A fost odata ca niciodata, in lumea Formula 1, o echipa legendara numita Ferrari. Aceasta echipa italiana si-a inceput calatoria in lumea curselelor in anul 1950 si de atunci a devenit una dintre cele mai recunoscute si respectate echipe din istoria sportului cu motor. Povestea Ferrari in lumea Formula 1 este una plina de momente de sacrificiu si durere. Cu toate acestea, pasiunea lor pentru motorsport ramane la fel de puternica ca intotdeauna si continuă sa inspire si sa impresioneze fanii din intreaga lume. Inginerii de la Ferrari au nevoie de ajutorul tau pentu a reusi sa castige curse in acest sezon. Au realizat ca una dintre probelmele masinii lor este cauzata de niste senzori defecti (cumparati de pe AliExpress) care transmit date gresite catre echipa de ingineri. Ajutati echipa sa identifice si sa elimine datele eronate pentru a castiga curse.

Implementare

Implementarea va consta crearea unui vector de tip *Sensor** ce va contine doua tipuri de senzori: *Tire Sensor* si *Power Management Unit Sensor* (vezi mai jos structurile). O structura de tip Sensor va contine pe langa datele sensorului asociat si un vector cu indicii operatiilor ce vor trebui efectuate pe date sensorului. Sunt un numar total de 8 operatii sub forma de functii care trebuie apelate pe datele sensorului. Implementarea acestora se gaseste in fisierul *operations.c* din schelet, avand urmatoarele antete:

```
static void tire_pressure_status(void* data);

static void tire_temperature_status(void* data);

static void tire_wear_level_status(void* data);

static void tire_performance_score(void* data);

static void pmu_compute_power(void* data);

static void pmu_regenerate_energy(void* data);
```

```
static void pmu_get_energy_usage(void* data);

static void pmu_is_battery_healthy(void* data);
```

Pentru a usura utilizarea acestora, va punem la dispozitie urmatoarea functie (tot in *operations.c*) ce va alcatui un vector de operatii:

```
void get_operations(void** operations)
```

Fiecare senzor va primi ca input un vector de indecsi ai operatiilor ce vor trebui apelate din vectorul obtinut dupa apelarea functiei de mai sus. **Functiile trebuie apelate prin intermediul vectorului, NU aveti voie sa apelati functiile explicit.**
Exemplu:

Pentru un senzor de tipul *Tire Sensor* se primeste ca input urmatorul vector:

```
3 1 0 2
```

Se vor apela, in ordine, urmatoarele functii:

```
tire_performance_score();

tire_temperature_status();

tire_pressure_status();

tire_wear_level_status();
```

Prioritati

Valorile primite de la senzorii de tip *Power Management Unit* sunt mai importante decat cele primite de la senzorii de tip *Tire Sensor*. Astfel, vom dorii care senzorii *Power Management Unit* sa se afle primii in vectorul de senzori.
Exemplu:

Primim ca input urmatorii senzori:

```
Tire_1 Tire_2 PMU_1 PMU_2 Tire_3 PMU_3 Tire_4 Tire_5 PMU_4
```

Vectorul va contine senzorii in urmatoarea ordine:

```
PMU_1 PMU_2 PMU_3 PMU_4 Tire_1 Tire_2 Tire_3 Tire_4 Tire_5
```

Structuri

Mai jos aveti implementarea structurilor:

```
enum sensor_type {

        TIRE,
```

```
        PMU

};



typedef struct {

        enum sensor_type sensor_type;  // 0/1

        void *sensor_data;             // TireSensor/PowerManagementUnit

        int nr_operations;

        int *operations_idxs;

} Sensor;



typedef struct __attribute__((__packed__)) {

    float voltage;                     // voltage level of the battery; 10-20
volts

    float current;                     // current draw from the battery; -100
to 100 amps; negative values indicate energy regeneration during braking

    float power_consumption;           // power consumption of the car; 0 to
1000 kilowatts

    int energy_regen;                          //  energy  regenerated  during
braking; between 0-100%

    int energy_storage;                //  amount  of  energy  stored  in  the
battery; between 0-100%

} PowerManagementUnit;



typedef struct __attribute__((__packed__)) {

        float pressure;                // 19-26 psi

        float temperature;             // between 0-120C

        int wear_level;                //  interval  between  0-100%;.  0%  wear
means new tire.

        int performace_score;          // between 1-10; 1 low performance; 10
high performance

} TireSensor;
```

Schelet

In schelet se aflat 4 fisiere:

- main.c - contine implementarea scrisa de voi
- operations.c - contine implementarea operatiilor
- structs.h - contine structurile ce vor fi folosite in cadrul implementarii
- Makefile - un Makefile clasic. Se poate folosii in urmatoarele moduri: **make**, **make run**, **make clean**

Recomandam citirea punctului *6. Memory management* inainte de a va apuca de scris cod.

Programul va citi comenzi de la tastatura pana la primirea comenzii exit, in urma careia programul va elibera memoria si va iesi.

Comenzile primite vor veni in urmatorul format:

- print <index> - se va afisa senzorul de la pozitia data, avand formatul prezentat in urmatoarea sectiune. Daca indexul primit de la tastatura este negativ sau este mai mare decat dimensiunea vectorului se va afisa mesajul: "Index not in range!".
- analyze <index> - se vor efectua toate operatii senzorului de pe pozitia data. Daca indexul primit de la tastatura este negativ sau este mai mare decat dimensiunea vectorului se va afisa mesajul: "Index not in range!".
- clear - se vor sterge toti senzorii care contin valori eronate
- exit - se va elibera memoria si se va iesi din program

Detaliile despre afisarea si pasarea argumentelor de la input va fi prezentata in cele ce urmeaza

1. Print (simple) - 10p

Se va face printarea vectorului de senzori, aplicandu-se urmatorul format:

Pentru *Tire Sensor*:

```
Tire Sensor

Pressure: <pressure>

Temperature: <temperature>

Wear Level: <wear_level>

Performance Score: <computed score>/Performance Score: Not Calculated
```

Pentru *Power Management Unit Sensor*:

```
Power Management Unit
```

```
Voltage: <voltage>

Current: <current>

Power Consumption: <power_consumption>

Energy Regen: <energy_regen>

Energy Storage: <energy_storage>
```

Printarea variabilelor de tip float se va face cu o precizie de 2 zecimale.

2. Print (complex) - 20p

Se va face printarea vectorului de senzori in acelasi format prezentat mai sus, tinandu-se cont de prioritatile acestora.

3. Analyze - 20p

Se vor efectua toate operatiile senzorului dat ca argument, in ordine in care au fost date.

4. Clear - 20p

Se vor sterge din vector, senzorii care contin valori eronate. Un senzor este considerat invalid daca NU respecta cel putin una din urmatoarele conditii:

```
Tire Sensor:

pressure: between 19 and 28 psi

temperature: between 0°C and 120°C

wear_level: between 0% and 100%

Power Management Unit Sensor:

voltage: between 10V and 20V

current: between -100A and 100A

power_consumption: between 0kW and 1000kW

energy_regen: between 0% and 100%

energy_storage: between 0% and 100%
```

5. Exit - 0p

La primirea acestei comenzi memoria este dezalocata si programul se opreste.

6. Memory management - 20p

O alta parte foarte importanta a temei este intelegerea si lucrul cu memoria. Pentru asta, acest task va consta din doua parti:

1) Alocarea corecta de memorie: memoria va fi alocata dinamic pentru toate structurile folosite. Alocare corecta insemna ca in urma operatiei de clear nu va mai fi alocata memorie si pentru senzori eliminati si ca vectorul va fi redimensionat.

2) Dezalocarea corecta a memoriei: memoria va fi dezalocata corect si complet. Pentru testare, vom folosi valgrind si ca punct de referinta, programul nu trebuie sa afiseze niciun read invalid sau orice leak de memorie.(erori de REDIR nu o sa fie depunctate).

`valgrind –leak-check=full –show-leak-kinds=all –track-origins=yes ./main`. Ca sa nu va chinuiti aveti deja la dispozitie o astfel de rulare prin comanda `make check`.
Punctajul pe aceasta cerinta va fi oferit doar daca cel putin 50p au fost obtinute din alte cerinte.

Punctajul pe aceasta cerinta se acorda doar daca aveti memoria dezalocata corect si complet la finalul programului

7. Coding style - 10p

Se acorda 10p pentru coding style si comentarii. README-ul este optional. Daca doriti sa adaugati un README, este de preferat sa fie de tipul Markdown.

Format fisiere input

Fisierele de input sunt sub forma binara.

Exemplu fisier input:

```
2 1 12.3 -50 500 30 70 4 4 5 6 7 0 23.5 80 20 0 4 0 1 2 3
```

Explicatia fisierului:

```
2                       // numar de senzori

1                       // senzor de tip PMU

12.3 -50 500 30 70      // datele senzorului PMU

4                         // numarul de operatii ce vor fi aplicate pe datelor senzorului PMU
```

```
4 5 6 7                 // operatiile ce vor fi aplicate

0                       // senzor de tip Tire

23.5 80 20 0            // datele senzorului Tire

4                         // numarul de operatii ce vor fi aplicate pe datelor
senzorului Tire

0 1 2 3                 // operatiile ce vor fi aplicate
```

Fisierul binar este primit ca argument.

Trimitere și notare

Temă valorează un punct (1p) din nota finală.

Cele 100 de puncte ale temei sunt împărțite astfel:

- 10p printare simplă
- 20p oriuntarea cu priorități
- 20p analyze
- 20p clear
- 0p exit
- 20p memory management
- 10p coding style, comentarii

Temele vor trebui încărcate pe platforma Moodle.
Trebuie încărcată o arhivă care să conțină în rădăcină fișierele sursă ale temei.

După încărcare, vă rugăm să așteptați câteva minute pentru a se afișa scorul și feedback-ul. Altfel, dacă se produce o eroare și arhivă nu este reîncărcată, tema nu se va corecta.

Precizări suplimentare

- Pentru rularea checker-ului consultati README-ul din cadrul skel-ului.
- README-ul este optional.
- Recomandam parcurgerea regulamentului daca nu ati facut-o deja.

Resurse

- Scheletul de cod, Makefile-ul se gasesc pe repository-ul public IOCLA in folderul `teme/tema-1/`.