PRÀCTICA 1

Format i data de lliurament

El lliurament s'ha de fer a l'apartat "Lliurament i registre d'AC" de l'aula de teoria, abans del dia **23 de març de 2020 a les 23:59**.

S'ha de lliurar un fitxer en format ZIP que contingui un directori **'UOC2019_2'** amb el codi resultant dels exercicis.

Els lliuraments han de complir els següents punts per a poder ser avaluats correctament:

- Cal lliurar un fitxer en format ZIP que contingui el directori principal 'UOC2019_2' (i que no contingui altres fitxers .zip a dins).
- El sistema i nom de fitxers del projecte lliurat han de seguir l'estructura del codi adjuntat amb l'enunciat de les pràctiques (no modifiqueu ni el nom dels fitxers, ni els de les carpetes ni en modifiqueu l'estructura de fitxers).
- El codi lliurat **no ha de presentar errors de compilació** per poder ser avaluat. Si una part del codi us funciona correctament però heu fet algun exercici que genera algun error, esborreu o comenteu aquesta part per tal que la resta de codi compili i es pugui avaluar la pràctica.
- **El codi ha de poder-se executar.** Pot ser que alguns tests fallin, però, com a mínim, el programa ha de ser capaç de mostrar el resultat de les proves.
- S'han de lliurar tots els fitxers .c i .h utilitzats en el projecte i han d'estar presents en el lliurament en les carpetes correctes (src, tests, etc.).
- S'han de lliurar els fitxers .workspace i .project que defineixen el projecte complet de CodeLite.
- Es recomana afegir dins la carpeta principal del projecte un fitxer de nom README.txt especificant el sistema operatiu utilitzat.
- Els fitxers de test s'han de lliurar sense modificacions. Si per a alguna prova durant el desenvolupament han estat modificats, en el lliurament s'ha de tornar a la versió original.

El incompliment de qualsevol dels punts indicats pot suposar un suspens de la pràctica.

Presentació

Durant les diferents pràctiques desenvoluparem un únic projecte, del qual anireu dissenyant i implementant funcionalitats.

Treballarem seguint una metodologia basada en proves. Això significa que amb l'enunciat us facilitarem un conjunt de proves funcionals de l'aplicació. L'objectiu és implementar el codi necessari per a passar totes aquestes proves. Els exercicis que es proposen et guiaran per a facilitar la resolució de les diferents funcionalitats provades. És important destacar el fet que un codi que passi tots els tests no significa que sigui un codi 100% correcte. Existeixen altres criteris d'avaluació que es detallen més endavant. A més, els tests no tenen perquè provar que s'han realitzat tots els punts sol·licitats en l'enunciat, és a dir, poden no cobrir el 100% de l'enunciat.

Durant la programació és normal que sorgeixin dubtes relacionats amb el llenguatge, tant amb la sintaxi com en la forma de treballar amb les estructures de dades. Dirigiu els vostres dubtes sobre la programació al fòrum del laboratori de l'assignatura.

Nota: Al conjunt de proves que us adjuntem amb l'enunciat habitualment se les anomena "tests unitaris".

Objectius

Els objectius d'aquesta pràctica són:

- Compilació d'un projecte de codi.
- Comprensió del projecte a desenvolupar.
- Familiarització amb les funcionalitats i l'ús del codi proporcionat pels professors.
- Recordatori de la sintaxi, l'estil i l'entorn de programació en C.
- **Ús de memòria dinàmica.**

Competències

Transversals

- Capacitat de comunicació en llengua estrangera.

Específiques

- Capacitat de dissenyar i construir aplicacions informàtiques mitjançant tècniques de desenvolupament, integració i reutilització.

Recursos

Per a realitzar aquesta pràctica disposeu dels següents recursos:

Recursos Bàsics

Laboratori de Pràctiques de Programació: Disposeu d'un laboratori assignat a l'assignatura. En aquest laboratori podeu resoldre tots els dubtes sobre la programació en llenguatge C, i la instal·lació i utilització de l'entorn de programació.

Material de Laboratori: En el laboratori, a part del col·laborador docent, disposeu de diferents documents que us ajudaran. En concret, hi ha un manual de C i una guia de programació en C.

Transparències de Síntesi: Als materials de l'assignatura (xWiki) teniu les transparències de síntesi. Per a aquesta pràctica us serà d'especial interès els apartats d'«Introducció», concretament les transparències de síntesis TS2, TS3 i TS4.

Recursos Complementaris

Cercador web: La forma més ràpida d'obtenir informació actualitzada sobre els paràmetres, funcionament i exemples d'utilització de qualsevol mètode estàndard de C (i en general de qualsevol llenguatge), és mitjançant un cercador web.

Criteris de valoració

Per a la valoració dels exercicis es tindrà en compte:

- El codi obté el resultat esperat donades unes condicions i dades d'entrada dissenyades per a provar algunes situacions de funcionament normal i altres casos especials.
- El codi lliurat segueix la guia d'estil i les bones pràctiques de programació.
- Es separa correctament la declaració i implementació de les accions i funcions, utilitzant els fitxers correctes.
- El codi està correctament comentat, valorant especialment la utilització de comentaris en anglès.
- El grau d'optimització en temps i recursos utilitzats en la solució lliurada.
- El codi realitzat és modular i estructurat, tenint en compte la reutilització del codi.
- Es realitza una gestió de la memòria adequada, alliberant la memòria quan sigui necessari.

Enunciat

En el context de les epidèmies per virus d'origen animal molt contagioses i amb alta mortalitat, se'ns proposa el desenvolupament d'un sistema de registre dels diferents casos produits en països al voltant del món.

L'objectiu del sistema es facilitar l'anàlisi estadístic de l'impacte, per a poder estudiar la eficiència de diferents mesures de prevenció que evitin la seva propagació.

Les indicacions que ens han donat són les següents:

- S'ha de poder gestionar un conjunt de **reservoris** del virus, emmagatzemant el seu nom i els animals que en formen part.
- S'ha de poder gestionar un conjunt d'**agents infecciosos** potencialment perillosos i contagiosos de recent aparició.
- Es vol registrar l'evolució de les **infeccions** que es produeixen a cada país per aquests agents infecciosos. Per a cadascun, s'emmagatzemarà informació relativa a aquesta.

En aquesta primera pràctica es proposa la implementació dels mètodes per a la gestió bàsica de les estructures de dades associades al model que es proposa: reservoris, agents infecciosos i infeccions.

Exercici 1: Preparació de l'entorn

Juntament amb l'enunciat trobareu un fitxer ZIP amb el codi de la pràctica. L'estructura és:

- **UOC2019infection**
  - **UOCinfectiousAgent**: Llibreria estàtica per a gestionar el sistema de registre de la plataforma.
    - **src**: Fitxers de codi (*.c)
      - Els exercicis que es proposen en la pràctica cal que es completin en els fitxers existents en aquest directori.
    - **include**: Fitxers de capçalera (*.h)
  - **UOCinfection**: Projecte executable que utilitza el sistema de registre implementat en la llibreria estàtica, i executa un conjunt de test automatitzats.
    - **src**: Fitxer main.c, punt d'entrada.
    - **test**: Conté la implementació de les proves que permeten comprovar les funcionalitats de la llibreria.
      - **src**: Fitxers de codi (*.c)
      - **include**: Fitxers de capçalera (*.h)

Caldrà crear un nou espai de treball (Workspace) en CodeLite anomenat UOC2019infection que contingui dos projectes (que també heu de crear). Un de tipus executable **UOCinfection** i l'altre de tipus llibreria estàtica **UOCinfectiousAgent**.

Per tal que el projecte compili correctament, cal configurar:

- **UOCinfectiousAgent**:
  - Cal afegir el directori 'include' a la llista de directoris on el CodeLite anirà a buscar els fitxers de capçalera.

- **UOCinfection**:
  - Cal afegir el directori 'include' i el directori 'test/include' a la llista de directoris on el CodeLite anirà a buscar els fitxers de capçalera.
  - Cal afegir el directori 'include' de la llibreria **UOCinfectiousAgent**.
  - Cal que s'indiqui que aquest projecte depèn de la llibreria **UOCinfectiousAgent**.

Nota: En els materials de l'assignatura trobareu un apartat anomenat 'Modularitat en CodeLite', que conté indicacions per dur a terme aquests passos.

Tan bon punt s'hagi configurat correctament el projecte, a l'executar el projecte **UOCinfection** hauria d'aparèixer una sortida com la que es mostra en la captura de pantalla següent. La sortida mostra el conjunt de proves que s'han executat, i per cada conjunt de proves, si aquest ha passat (OK) o ha fallat (FAIL), amb una petita descripció del què s'està provant.

```
================================================================================
         TEST RESULTS
================================================================================

        ================================================================
        Tests for PR1 exercices
        ================================================================
                [OK]:    [PR1_EX1_1] Initialize the table of reservoirs
                [OK]:    [PR1_EX1_2] Initialize a reservoir
                [OK]:    [PR1_EX1_3] Add a new reservoir
                [OK]:    [PR1_EX1_4] Add more reservoirs
                [OK]:    [PR1_EX1_5] Remove a reservoir
                [OK]:    [PR1_EX1_6] Remove a non existing reservoir
                [FAIL]:  [PR1_EX2_1] Initialize a infectious agent
                [FAIL]:  [PR1_EX2_2] Copy a infectious agent
                [FAIL]:  [PR1_EX2_3] Get the reservoirs list
                [FAIL]:  [PR1_EX3_1] Initialize the table of infectious agents
                [FAIL]:  [PR1_EX3_2] Add a new infectious agent
                [FAIL]:  [PR1_EX3_3] Add more infectious agent
                [FAIL]:  [PR1_EX3_4] Remove a infectious agent
                [FAIL]:  [PR1_EX3_5] Remove a non existing infectious agent
                [FAIL]:  [PR1_EX4_1] Initialize an infection
                [FAIL]:  [PR1_EX4_2] Update an infection
                [FAIL]:  [PR1_EX4_3] Initialize the table of infections
                [FAIL]:  [PR1_EX4_4] Add a new infection
                [FAIL]:  [PR1_EX4_5] Add more infections
                [FAIL]:  [PR1_EX4_6] Copy an infection
                [FAIL]:  [PR1_EX4_7] Add duplicated infection
                [FAIL]:  [PR1_EX4_8] Remove infection
                [FAIL]:  [PR1_EX4_9] Get Max Infection
                [FAIL]:  [PR1_EX4_10] Get mortality rate
        ================================================================
        Total Tests: 24
        Passed Tests: 6 ( 25.00 % )
        Failed Tests: 18 ( 75.00 %)
        ================================================================

================================================================================
Total Tests: 24
Passed Tests: 6 ( 25.00 % )
Failed Tests: 18 ( 75.00 % )
================================================================================
Press enter to end...
```

Comproveu que totes les proves per a l'exercici 1 de la pràctica 1 **[PR1_EX1_XX]** passen correctament. Pot ser que alguna prova d'altres exercicis estigui com a passada, però el seu estat pot canviar a mida que es vagi avançant en el projecte.

Exercici 2: Gestió dels agents infecciosos [30%]

Dins del projecte, en el fitxer reservoir.h hi trobareu la definició d'un tipus de dades per a emmagatzemar la informació d'un reservori dels virus que provoquen la malaltia (tReservoir). Podeu trobar tant la definició de les estructures de dades, com els mètodes que es proposen per accedir-hi (altes, baixes i modificacions).

Ambdós reservoir.h i reservoir.c contenen comentaris explicant punts clau en la gestió de la memòria dinàmica. Utilitzant com a referència el codi existent per als mètodes associats a tReservoir, es proposa la implementació dels mètodes de gestió d'un agent infecciós perillós molt contagiós.

1. infectiousAgent_init(): Inicialitza una dada de tipus tInfectiousAgent, a partir dels valors rebuts com a paràmetre durant la seva crida, emmagatzemant el seu nom, grau de contagi R sub-zero, mitjà de transmissió, data i lloc d'esclat i una referència (punter) a la llista de reservoris.
2. infectiousAgent_free(): Elimina una dada de tipus tInfectiousAgent, alliberant la memòria utilitzada per aquest.
3. infectiousAgent_getReservorios(): Retorna un punter a llista de reservoris.
4. infectiousAgent_equals(): Compara dos dades de tipus tInfectiousAgent i ens diu si són iguals, en base al seu nom i grau de contagi r subzero.
5. InfectiousAgent_cpy(): Copia el contingut d'una dada tInfectiousAgent origen, en una altra dada tInfectiousAgent destí.

Al llarg d'aquest i la resta d'exercicis, **en cas que no es pugui reservar o alliberar memòria mitjançant les funcions de la llibreria estàndard de C, es retornarà un error amb codi ERR_MEMORY_ERROR en el mètode corresponent** - excepte en aquells mètodes que no facin cap crida a funcions de gestió de memòria dinàmica. Veure definició de codis d'error a error.h, dins del projecte de l'enunciat.

En cas d'execució correcta del mètode proposat, retornarem el codi OK - excepte en els mètodes on a l'enunciat es demana explícitament el retorn d'un valor particular o si la funció retorna void.

Després d'haver implementat tots els mètodes de l'exercici, haurien de passar correctament tots els tests referents a l'exercici 2 [PR1_EX2_XX]. Els principals conceptes que han de quedar clars després de l'exercici són:

- Ús de les funcions malloc, realloc i free de la llibreria estàndard de C
- Us bàsic d'apuntadors i significat de l'operador direcció (&)

Exercici 3: Gestió de taula d'agents infecciosos [30%]

Donada la definició de tInfectiousAgentTable en el fitxer InfectiousAgent.h implementa els mètodes següents en el fitxer InfectiousAgent.c:

1. infectiousAgentTable_init(): Inicialitza una taula d'agents infecciosos.
2. infectiousAgent_free(): Allibera la memòria que s'hagi reservat dinàmicament per tal de desar les dades.
3. infectiousAgent_Table_add(): Afegeix un agent infecciós a la taula d'agents infecciosos. En cas que es produeixi algun problema al gestionar la memòria, aquest mètode retornarà un valor d'error ERR_MEMORY_ERROR. Si ja existeix l'agent infecciós a la taula no l'afegirà i retornarà l'error ERR_DUPLICATED. Si tot va bé, afegirà l'agent infecciós i retornarà el valor OK.
4. infectiousAgent_Table_size(): Retorna el nombre d'agents infecciosos existents a la taula.
5. infectiousAgent_Table_find(): Busca a la taula agents infecciosos amb el nom donat. Retorna un punter a l'element corresponent si hi és, o un valor NULL en cas que no trobi cap agent infecciós amb el nom donat.
6. infectiousAgent_Table_remove(): Elimina un agent infecciós de la taula agents infecciosos. En cas que es produeixi algun problema en la gestió de la memòria, aquest mètode retornarà un valor d'error ERR_MEMORY_ERROR. En cas que s'intenti eliminar un agent infecciós que no existeix a la taula, es retornarà el valor d'error ERR_NOT_FOUND. Si tot funciona correctament retornarà el valor OK.

Després d'haver implementat tots els mètodes de l'exercici, haurien de passar correctament tots els tests referents a l'exercici 3 [PR1_EX3_XX]. Els principals conceptes que han de quedar clars després de l'exercici són:

- Definició i gestió de taules en C amb memòria dinàmica

Exercici 4: Gestió d'infeccions i de la taula infeccions [40%]

En el fitxer infection.h hi ha la declaració del tipus tInfection que utilitzarem per emmagatzemar les infeccions produïdes en tots els països.

Per a cada infecció volem emmagatzemar:

- **Agent infecciós**
- País
- Data
- Casos fins a la data
- Morts fins a la data

Donada la definició de tInfection a infection.h es proposa la implementació dels següents mètodes:

1. infection_init(): Inicialitza una dada de tipus tInfection amb la data, element tInfectiousAgent, país, nombre d'infectats i nombre de morts rebuts com a paràmetres.
2. infection_free(): Elimina una dada de tipus tInfection alliberant la memòria utilitzada per aquesta.
3. infection_equals(): Compara dos infeccions i ens retorna true si són iguals, en base al seu nom i país.
4. infection_update(): Afegeix una actualització dels casos i/o morts.

Donada la definició de tInfectionTable en el fitxer infection.h implementa els mètodes següents en el fitxer infection.c:

5. infectionTable_init(): Inicialitza una taula d'infeccions.
6. infectionTable_free(): Allibera els recursos emmagatzemats per una dada de tipus infectionTable existent.
7. infectionTable_add(): Afegeix una nova infecció a la taula, rebuda com a paràmetre.
8. infectionTable_size(): Retorna el número d'infeccions que hi ha a la taula.
9. infectionTable_find(): Busca una infecció dins d'un país a la taula d'infeccions. Retorna un punter a l'element corresponent si es troba en aquell país, o un valor NULL en cas que no es trobi la infecció en el país donat.
10. infectionTable_equals(): Compara dues taules d'infeccions i retorna true si tenen els mateixos elements d'infecció.
11. infection_cpy(): Copia les dades d'una infecció origen a una infecció destí.
12. infectionTable_remove(): Esborra un element d'infecció de la taula.
13. infectionTable_getMaxInfectation(): Donat un agent infecciós i una taula de tipus tInfectionTable, realitza una cerca del país amb major població infectada, oferint-nos un punter a aquest. En cas d'empat, es retornarà el país que primer

es trobi a la llista. En cas de no trobar l'agent infecciós a la lista, retornarà NULL.

14. infectionTable_getMortalityRate(): Donat un agent infecciós i una taula de tipus tInfectionTable, calcula la taxa de mortalitat d'un agent infecciós a tot el món, sumant tots els morts i dividint-los entre el nombre d'afectats.

Després d'haver implementat tots els mètodes de l'exercici, haurien de passar correctament tots els tests referents a l'exercici 4 [PR1_EX4_XX]. Els principals conceptes que han de quedar clars després de l'exercici són:

- Definició i gestió de taules amb memòria dinàmica.