INF-253 Lenguajes de Programación
Tarea 2: C

9 de septiembre de 2022

1. Certamen

Después de haber programado exitosamente un traductor para los pixelarts en Minecraft usted se tomó un merecido tiempo de descanso, comenzando una partida del famoso juego de bloques con sus amigos, en donde construyeron y tuvieron aventuras. Antes de que lo pudieran notar pasaron semanas, y ahora están a tan solo días de la fecha de un certamen muy importante para el cual no han estudiado. En un último intento por salvar este, usted decide hacer un programa que le permita simular un certamen para poder practicar tantas veces como sea necesario en el poco tiempo que queda.

2. Definición

Para esta tarea usted deberá crear un programa en C capaz de crear un certamen en base a las especificaciones en un archivo de texto y responderlo en base al input por consola de un usuario. Para lograr esto deberán crear una estructura de datos que almacene el certamen y sus preguntas haciendo uso de memoria dinámica, punteros a void y punteros a función.

2.1. Tipos de preguntas

Los exámenes podrán tener 4 tipos de preguntas:

- `AlternativaSimple`: Estas preguntas tienen un enunciado, múltiples alternativas y se responden seleccionando una de las alternativas.
- `AlternativaMultiple`: Estas preguntas tiene un enunciado, múltiples alternativas y se responden seleccionando múltiples de las alternativas.
- `VerdaderoFalso`: Estas preguntas tienen un enunciado y se responden con un booleano Verdadero o Falso.
- `Completar`: Estas preguntas tienen múltiples secciones de texto que deben ser respondidas completadas con una palabra que une las distintas secciones de texto. Ej:
  - Enunciado: “Este”, “debe ser”, “.”
  - Respuesta: “enunciado”, “completado”
  - Frase completa: “Este enunciado debe ser completado”

2.2. Archivo `certamen.txt`

Su programa leerá el certamen desde el archivo `certamen.txt`. Este archivo comienza con una línea con un entero **N**, seguido de **N** preguntas. Cada pregunta comienza con una linea con un string **tipo** que indica el tipo de pregunta a la que corresponder, seguido del enunciado de la pregunta, la forma del cual dependerá del tipo de pregunta.

- Si la pregunta es de tipo `AlternativaSimple` entonces contendrá una línea con el string **enunciado**, una línea con el entero **n_alternativas** que indica el número de alternativas de la pregunta, **n_alternativas** lineas cada una con un string que corresponde a una alternativa y una línea con un entero correspondiente a la alternativa correcta.
- Si la pregunta es de tipo `AlternativaMultiple` entonces contendrá una línea con el string **enunciado**, una línea con el entero **n_alternativas** que indica el número de alternativas de la pregunta, **n_alternativas** lineas cada un con un string que corresponde a una alternativa, una línea con el entero **n_correctas** que indica el número de alternativas correctas y **n_correctas** lineas cada una con un entero que indica una alternativa correcta.
- Si la pregunta es de tipo `VerdaderoFalso` entonces contendrá una línea con el string **enunciado** y una línea con V o F.
- Si la pregunta es de tipo `Completar` entonces contendrá una línea con el entero **n_textos**, seguido de **n_textos** lineas cada una con un string correspondiente a una parte del texto que debe ser completado, seguido de **n_textos - 1** lineas cada una con un string correspondiente a las respuestas correctas.

Puede asumir que este archivo siempre sera correcto, por lo que no hay que comprobar que cumpla este formato.

2.3. Estructura de datos

La información extraída de `certamen.txt` debe ser almacenada en una estructura de la siguiente forma:

```
typedef struct {
    char enunciado[128];
    int n_alternativas;
    char** alternativas;
    int alternativa_correcta;
} tEnunciadoAlternativa;

typedef struct {
    char enunciado[128];
    int n_alternativas;
    char** alternativas;
    int n_correctas;
    int* alternativa_correcta;
} tEnunciadoAlternativaMultiple;

typedef struct {
    char enunciado[128];
    bool respuesta;
} tEnunciadoVerdaderoFalso;

typedef struct {
```

```
    int n_textos;
    char** textos;
    char** respuestas;
} tEnunciadoCompletar;

typedef struct {
    char tipo[64];
    void* enunciado;
    void* respuesta;
    bool (*revisar)(void*, void*);
} tPregunta;

typedef struct {
    int n_preguntas;
    tPregunta* preguntas;
} tCertamen;
```

2.4. Funciones

Además, se deben implementar las siguientes funciones para manipular la estructura de datos anteriormente explicada:

```
// Crea un nuevo certamen vacio
tCertamen* crearCertamen(int n_preguntas);

// Crea una pregunta con el enunciado y funcion de revision dados
tPregunta* crearPregunta(
    tCertamen* certamen,
    char* tipo,
    void* enunciado
    bool revisar(void*, void*));

// Asigna la pregunta a la posicion n_pregunta del certamen
void asignarPregunta(
    tCertamen* certamen,
    int n_pregunta,
    tPregunta* pregunta);

// Retorna la pregunta en la posicion n_pregunta del certamen
tPregunta leerPregunta(tCertamen* certamen, int n_pregunta);

// Retorna el numero de respuestas correctas que tiene el certamen
int nCorrectasCertamen(tCertamen certamen);

// Retorna el numero de preguntas en un certamen
int largoCertamen(tCertamen certamen);

// Revisa si la respuesta a la pregunta es correcta
bool revisarAlternativaSimple(tPregunta pregunta);
bool revisarAlternativaMultiple(tPregunta pregunta);
bool revisarVerdaderoFalso(tPregunta pregunta);
bool revisarCompletar(tPregunta pregunta);
```

2.5. Responder

Luego de leer el archivo `certamen.txt` y cargar la información en la estructura de datos, se le debe permitir al usuario responder el certamen a través de la consola y una vez que termine de responder se le debe mostrar su calificación. **Para esto deberán hacer uso de la estructura y funciones implementadas.** El formato en el que el usuario responderá el cuestionario queda a su criterio, pero para cada pregunta se debe indicar el numero de pregunta, el tipo de pregunta, el enunciado, las opciones que tiene para responder (si es que aplica) y el que debe ingresar para responder (un string, un entero, múltiples enteros separados por espacios, etc)

3. Ejemplos

3.1. Ejemplo 1

```
3
AlternativaSimple
¿Cual es la respuesta correcta a esta pregunta?
3                    n alternativas
Esta alternativa
La tercera alternativa
La segunda alternativa
1                          alternativa correcta
VerdaderoFalso
A monad is a monoid in the category of endofunctors
V
AlternativaMultiple
¿Cuales de estos juegos usan principalmente cubos?
5                n alternativas
Minecraft
Terraria
Beat Saber
Genshin Impact
Apex Legends
2                  n correctas
1
3
```

3.2. Ejemplo 2

```
3
Completar
2
I don't want to deny who I've been. Because even my
are a part of who I am today.
failures
Completar
3
Un puntero es una variable que almacena la
```

```
de un
.
dirección de memoria
objeto
Completar
4
C es un
de propósito general desarrollado como evolución al lenguaje
. Se trata de un lenguaje
tipado.
lenguaje de programación
B
débilmente
```

4. Sobre la Entrega

- Se deberá entregar un programa con los siguientes archivos:
  - `certamen.c`: Contiene la implementación de las funciones de la estructura de datos.
  - `certamen.h`: Contiene la definición de los struct y funciones de la estructura de datos.
  - `main.c`: Contiene el código encargado de leer `certamen.txt`, generar la estructura de datos con el certamen, preguntarle al usuario por terminal y evaluar el certamen.
  - `makefile`: Contiene el código para compilar su programa utilizando `make`.
- Los ayudantes correctores pueden realizar descuentos en caso de que el código se encuentre muy desordenado.
- Las funciones implementadas deben ser comentadas de la siguiente forma. **SE HARÁN DESCUENTOS POR FUNCIÓN NO COMENTADA**

```
/*
Descripcion de la funcion

    Parametros:
            a (int): Descripcion del parametro a
            b (int): Descripcion del parametro b

    Retorno:
            c (str): Descripcion del parametro c
*/
```

- Debe estar presente el archivo MAKEFILE para que se efectué la revisión, este debe compilar **TODOS** los archivos.
- Si el makefile no está bien realizado, la tarea no se revisará
- Se debe trabajar de forma individual obligatoriamente.
- **La entrega debe entregarse en .tar.gz y debe llevar el nombre: Tarea1LP_RolAlumno.tar.gz**

- **El archivo README.txt debe contener nombre y rol del alumno e instrucciones detalladas para la correcta utilización de su programa. De no incluir README se realizara un descuento.**
- La entrega será vía aula y el plazo máximo de entrega es hasta **05 de Octubre**.
- **Por cada hora de atraso se descontaran 20 pts.**
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.
- **Solo se contestaran dudas realizadas en AULA y que se realicen al menos 48 horas antes de la fecha de entrega original.**
- Se utilizará Valgrind para detectar los leak de memoria.

5. Calificación

5.1. Entrega

- Funciones (**60 pts**)
  - crearCertamen (**10 pts**)
  - crearPregunta (**10 pts**)
  - asignarPregunta (**5 pts**)
  - leerPregunta (**2.5 pts**)
  - nCorrectasCertamen (**10 pts**)
  - largoCertamen (**2.5 pts**)
  - revisarAlternativaSimple (**5 pts**)
  - revisarAlternativaMultiple (**5 pts**)
  - revisarVerdaderoFalso (**5 pts**)
  - revisarCompletar (**5 pts**)
- Carga de datos (**15 pts**)
- Input por consola (**25 pts**)
  - No permite responder el certamen. (**0 pts**)
  - Permite responder el certamen, pero otorga toda la información necesaria o no entrega la calificación correcta al finalizar. (**Max 20 pts**)
  - Permite responder el certamen, otorgando toda la información necesaria y entregando la calificación correcta al finalizar. (**25 pts**)

Se asignara puntaje parcial por funcionamiento parcialmente correcto.

5.2. Descuentos

- Falta de comentarios (-10 pts c/u Max 30 pts)
- Código no compila (-100 pts)
- Warning (c/u -5 pts)
- Memory leaks (entre -5 y -20 pts dependiendo de cuanta memoria se pierda)

- Falta de README (-20 pts)
- Falta de alguna información obligatoria en el README (-5 pts c/u)
- Falta de orden (entre -5 y -20 pts dependiendo de que tan desordenado)
- Día de atraso (-1 pt por cada hora de atraso)
- Mal nombre en algún archivo entregado (-5 pts c/u)