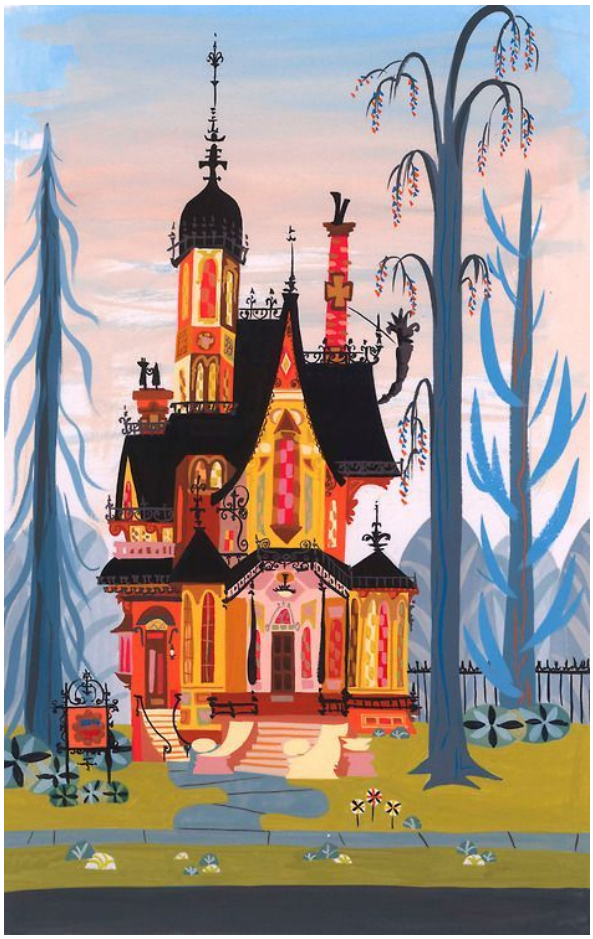
Algoritmos y Programación I
Curso Mendez

.UBAfiuba
FACULTAD DE INGENIERÍA

Eligiendo amigo

TP3



Fecha de Presentación: 11/02/2021

Fecha de Vencimiento: 04/03/2021

1. Introducción

Muchos niños tienen amigos imaginarios pero los abandonan al crecer, por eso existe el hogar de Foster una mansión donde se hospedan los amigos imaginarios de toda la infancia que son abandonados por sus creadores al crecer, pueden vivir hasta ser adoptados por otro niño.

2. Objetivo

El presente trabajo práctico tiene como objetivo que el alumno:

- Se familiaricen con el manejo de archivos en C.
- Utilicen las funciones acordes al tipo de archivo y/o forma de acceso.
- Apliquen apropiadamente las operaciones con archivos si se dan las condiciones para ello.
- Pongan en práctica el uso de argumentos por línea de comando.

Por supuesto, se requiere que el trabajo cumpla con las buenas prácticas de programación profesadas por la cátedra.

Se considerarán críticos la modularización, reutilización de código y la claridad del código.

3. Enunciado

El Sr. Conejo considera que la elección de amigos imaginarios acorde a las solicitudes se está volviendo muy tediosa por lo que solicita que creemos un programa que, según cierta información brindada por un niño, determine cual es el amigo imaginario que mejor se adapta a sus pretensiones.

Se pide, entonces, un sistema que cumpla los siguientes requerimientos:

- **Determinar amigo imaginario:** Dada cierta información, brindada por el niño, determinará el amigo imaginario que mejor le cuadre.
- **Listar amigos imaginarios de la mansión:** Listará todos los amigos imaginarios presentes en la mansión.
- **Devolver amigo imaginario a la mansión:** Devolverá un amigo imaginario a la mansión.

4. Especificaciones

4.1. Comandos

4.1.1. Elegir amigo

Recibirá como parámetro el nombre del archivo donde se encuentran registrados los amigos imaginarios que ha tenido el niño. El archivo recibido tendrá, como primera línea, los requisitos que debe cumplir el amigo imaginario, solicitados por el niño.

Un ejemplo de este comando es:

```
.\mansion_foster elegir_amigo archivo.txt
```

El comando deberá chequear la existencia del archivo enviado, en caso de no existir no deberán realizarse modificaciones.

Como resultado de esta operación se espera que:

- Se actualice el archivo de amigos imaginarios que actualmente están en la mansión.
- Se actualice el archivo de amigos imaginarios adoptados por el niño.

Cabe destacar que:

- Si no se puede seleccionar ningún amigo imaginario para el niño, no deberá modificarse ningún archivo.
- El amigo imaginario elegido no debe estar entre los ex amigos imaginarios del niño.
- En caso de haber más de un amigo imaginario que concuerde con los requisitos del niño, debe elegirse el de mayor nombre.

4.1.2. Listar amigos imaginarios de la mansión

Se mostrarán por pantalla los amigos imaginarios presentes en la mansión.

Un ejemplo de este comando es:

```
./mansion_foster listar_amigos
```

4.1.3. Devolver amigo imaginario a la mansión

Devolverá un amigo imaginario, dejándolo disponible para ser adoptado por otro niño.

Un ejemplo de este comando es:

```
./mansion_foster devolver_amigo nombre_amigo
```

Solo podrán devolverse amigos que ya existían en el archivo de amigos imaginarios.

4.2. Estructuras

```c
typedef struct aspecto {
    char descripcion[MAX_DESCRIPCION];
} aspecto_t;

typedef struct amigo{
    char nombre[MAX_NOMBRE];
    char color[MAX_COLOR];
    int altura;
    aspecto_t virtudes[MAX_VIRTUDES];
    int tope_virtudes;
    aspecto_t defectos[MAX_DEFECTOS];
    int tope_defectos;
    bool esta_en_mansion;
} amigo_t;
```

Los tamaños de los vectores de dicha estructura son:

```
  MAX_NOMBRE 50
  MAX_COLOR 50
  MAX_VIRTUDES 10
  MAX_DEFECTOS 10
  MAX_DESCRIPCION 200
```

4.3. Archivos

4.3.1. amigos_mansion.dat

Este archivo contendrá información de todos los amigos imaginarios de la mansión, actuales o pasados.

Es un archivo binario y está ordenado ascendentemente por nombre del amigo.

4.3.2. amigos_del_ninio.txt

Este archivo contendrá información de todos los amigos imaginarios que tuvo un niño, además, como primera línea contiene los requisitos del niño a la hora de adoptar un amigo imaginario.

Un ejemplo de este archivo se muestra a continuación:

```
ROJO;<150;DEPORTISTA;PEREZOSO
Charly
Coco
Red
...
```

La primer línea tiene las siguientes características:

- El primer elemento es el color, debe coincidir con el del amigo imaginario.
- El segundo elemento corresponde a la altura, la cual viene acompañada de una operación, puede ser <, >o = y la altura debe adecuarse a esa operación.
- El tercer elemento es una virtud, el amigo imaginario debe tener esa virtud entre las suyas.
- El cuarto elemento es un defecto, el amigo imaginario NO debe tener ese defecto entre las suyos.

5. Compilación y Entrega

El trabajo debe poder ser compilado correctamente con la siguiente línea:

```
gcc *.c -o mansion_foster -std=c99 -Wall -Werror -Wconversion
```

Por último debe ser entregado en la plataforma de corrección de trabajos prácticos **Chanutron2021** (patente pendiente), en la cual deberá tener la etiqueta ¡Exito! significando que ha pasado las pruebas a las que la cátedra someterá al trabajo.

Para la entrega en **Chanutron2021** (patente pendiente), recuerde que deberá subir un archivo zip conteniendo únicamente los archivos antes mencionados, sin carpetas internas ni otros archivos. De lo contrario, la entrega no será validada por la plataforma.

IMPORTANTE! Esto no implica necesariamente haber aprobado el trabajo ya que además será corregido por un colaborador que verificará que se cumplan las buenas prácticas de programación.

Al poner *.c el compilador toma todos los .c de la carpeta y los compila. Por ende, no importa el nombre que le pongan a la biblioteca ni al programa.

6. Anexo

6.1. Argumentos por línea de comando

Para poder pasar parámetros a un programa a través de la línea de comandos nos valemos de la siguiente declaración de la función main:

```
int main(int argc, char *argv[]){..
```

Argc contiene la cantidad de argumentos recibidos por el programa, debemos considerar que siempre será el número de argumentos pasados más 1, ya que el primer argumento se reserva para contener el nombre del programa. **Argv** es un vector de strings que contiene los parámetros pasados en el mismo orden en que fueron escritos.

El siguiente programa muestra un ejemplo de cómo hacer uso de estos parámetros:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

```

```c
int main(int argc, char *argv[]){

        printf ("El programa recibió %i argumentos\n", argc);
        for(int i = 0; i < argc; i++) {
                printf("Parámetro %d: %s\n", i, argv[i]);
        }

        if (strcmp (argv[1], "saludar") == 0){
                printf ("Hola!\n");
        } else if (strcmp (argv[1], "sumar") == 0 && argc == 4){
                int sumando_1 =  atoi (argv [2]);
                int sumando_2 =  atoi (argv [3]);
                printf ("%i + %i = %i\n", sumando_1, sumando_2, sumando_1 + sumando_2);
        }
        return 0;
}
```

Una vez compilado, sólo nos resta ingresar la línea de comando según lo que se quiera hacer, si ingresamos:

```
./ejemplo sumar 20 54
```

Siendo ejemplo el nombre del programa, lo que se muestra es:

```
El programa recibió 4 argumentos
Parámetro 0: ./ejemplo
Parámetro 1: sumar
Parámetro 2: 20
Parámetro 3: 54
20 + 54 = 74
```

6.2. Atoi: Array to integer

```c
int atoi(const char *nptr);
```

Esta función sirve para convertir la porción inicial de una cadena de caracteres apuntada por **nptr** en un entero.

Para usarla debemos incluir la biblioteca **stdlib.h**.