Práctica 0

Repaso de programación en lenguaje C

Datos útiles

Para realizar esta práctica puede utilizar la máquina virtual provista por la cátedra (se recomienda hacer un snapshot de la VM antes de empezar a usarla, para poder volver atrás en caso de que algo deje de funcionar).
El link de descarga se encuentra en el moodle junto a las credenciales de acceso. Además, la VM se encuentra instalada en las PCs de la Sala A del Lab. Bacarla donde se realizan las prácticas de la materia.

1) Cuestionario.
    a) Determine cuál es la longitud en bytes de los tipos `char`, `short`, `int`, `long`, `float`, y `double`.
    b) En su compilador existen los tipos `long double` y `long long`? ¿Qué tamaño en bytes tiene cada uno? ¿Cuál es el mínimo y el máximo valor que puede representarse con ellos?
    c) Describa la diferencia entre tipos `signed` y `unsigned`.
    d) Explique que es una cadena de caracteres o un `string`.
    e) ¿Qué funciones conoce para la manipulación de strings?
    f) ¿Cuál es la diferencia entre una cadena y un arreglo?
    g) ¿Qué pasa si utilizo las funciones de manipulación de strings en arreglos de caracteres que NO SON strings?
    h) ¿Existen las cadenas de `int`?
    i) En la declaración de un arreglo, ¿el tamaño del mismo puede ser variable?
    j) Explique que es un puntero y de un ejemplo de utilización de un puntero a entero.
    k) Explique que es un puntero `void` y como se puede cambiar a otro tipo de puntero.

l) Explique que es un archivo. ¿Cuál es la diferencia entre un archivo de texto y uno binario?
m) Explique que es una estructura y en qué se diferencia de un arreglo.
n) ¿Cómo se declara una variable puntero a estructura? ¿Cómo puede accederse a un miembro o campo individual de una estructura utilizando ese puntero?
o) Explique la utilización de la función **`malloc()`**. Escriba un ejemplo en el cual se reserva lugar para un arreglo de 50 **`unsigned long`**.
p) Explique cómo es el mecanismo utilizado para ingresar parámetros por línea de comando.
q) ¿Cuál es la diferencia entre declarar una función y definir una función?
r) Describa qué entiende por entrada estándar (**`stdin`**), salida estándar (**`stdout`**) y salida de error estándar (**`stderr`**).
s) Repase las distintas funciones para ingresar datos por **`stdin`** y presentarlos en **`stdout`** , **`getc()`**, **`gets()`**, **`scanf()`**, **`printf()`**, **`putc()`**, **`puts()`**, etc.
t) Repase las funciones para manejo de archivos: creación, apertura, escritura y lectura en forma binaria y con formato de texto, etc. (**`fgetc()`**, **`fgets()`**, **`fscanf()`**, **`fprintf()`**, **`fputc()`**, **`fputs()`**, **`fread()`**, **`fwrite()`**, **`fseek()`**, **`feof()`**, **`ftell()`**, etc.)
u) Repase el uso del preprocesador de C. Uso de **`#include`**, **`#define`**, **`#ifdef`**, etc.

2) Escriba un programa que permita al usuario ingresar la información relacionada a los usuarios de un sistema estructurada de la siguiente manera, y en el siguiente orden:

- **`Nombre de Usuario:`** hasta `60` caracteres alfanuméricos
- **`ID de Usuario:`** número entero sin signo
- **`e-mail:`** hasta `40` caracteres alfanuméricos

Los datos deben ser almacenados en una estructura definida para tal fin y luego deberán ser almacenados en un archivo binario, cuyo nombre se ingresa en la línea de comandos. Si el archivo ya existe, la información debe ser añadida al final del mismo, en caso contrario, debe crearse un archivo con ese nombre. El programa debe finalizar cuando se ingrese un nombre de usuario vacío. El campo correspondiente al “ID de

Usuario" debe ser generado automáticamente por el programa en forma secuencial, comenzando por el número 1.

Para verificar el funcionamiento utilice el programa leerdatos.exe que se encuentra en la página de la cátedra el cual lee el archivo producido y lista su contenido a razón de una línea por registro, con los campos separados por comas.

3) Escriba un programa que solicite al usuario el ingreso de la siguiente información:

- Un número entero mayor que 100000.
- Un número entero menor que -100000
- Un número entero comprendido entre 128 y 32767
- Un número entero comprendido entre -128 y 127
- Un número entero positivo comprendido entre 40000 y 50000
- Un número entero positivo comprendido entre 10 y 250
- Un número en punto flotante comprendido entre 10^{-24} y 10^{24}
- Un número en punto flotante comprendido entre -10^{-50} y 10^{-50}
- Nombre y apellido del usuario que ingresó los datos, máximo de 80 caracteres.

El tipo de variable donde se almacenen los datos debe ser el tipo más sencillo (que ocupe la menor cantidad de memoria) y que permita el almacenamiento de las variables solicitadas.

Una vez ingresados todos los valores, el programa debe imprimirlos en la consola. Todos los tipos enteros deben imprimir además su valor en hexadecimal.

4) Modifique el programa del ejercicio 3 para que en vez de imprimir los valores en consola, se almacenen en un archivo de texto a razón de un valor por línea.

5) Modifique el programa del ejercicio 3 para que en vez de imprimir los valores en consola, se almacenen en un archivo binario en el orden en que fueron ingresados.

6) Escriba un programa que lea el archivo generado en el ejercicio 5 y luego los imprima en pantalla y en un archivo de texto como el del ejercicio 4.

7) Escriba un programa que lea los archivos generados en el ejercicio 4 y escriba los datos leídos en pantalla.

8) Dibuje un diagrama de la memoria, considerando que almacena datos en porciones de 8 bits, (1 byte) y como estarían ubicados en la misma los datos que se ingresan en el ejercicio 3, considerando que ocupan posiciones de memoria consecutivas.

9) **(Ejercicio Entregable)**

   Una serie de estaciones meteorológicas automáticas, transmiten los datos recogidos a una computadora, donde se agrupan y almacenan como un archivo binario. Los datos están almacenados como estructuras en el archivo binario de la siguiente manera:

```
UINT32 Identificación de estación
UINT16 Presión *10 (milibares)
INT16 Temperatura *10 (grados centígrados)
UINT16 Precipitaciones caidas*10 (milímetros)
UINT8 humedad relativa ambiente en %
UINT32 Fecha y Hora de medición, en segundos contados desde
las 00:00 hs del 01/01/2000.
```

   Se quiere escribir un programa para leer el archivo binario y convertirlo a un archivo de texto tipo CSV con los siguientes campos separados por comas y en el orden especificado:

- Identificación de estación como un número entero.
- Fecha de Medición en formato **dd/mm/yyyy**.
- Hora de medición en formato **hh:mm:ss**

- Temperatura en formato **TT.t** (**TT** grados, **t** décimas de grado)
- Presión en formato **PPPP.p** (**PPPP** milibares, **p** decimas de milibar)
- Precipitaciones caídas en formato **zzz.u** (**zzz** milimetros **u** décimas de milímetro)
- Humedad relativa ambiente en formato **vvv** (porcentaje)
- Existe una restricción de memoria que hace que no pueda almacenarse más que el contenido de una línea a la vez (No se puede leer todo el archivo en memoria y luego procesarlo).
- Para realizar pruebas, pueden utilizar el archivo **datos.bin** disponible en el moodle de la materia.

Pautas de entrega

La resolución debe hacerse en base a las pautas explicadas en clase, es decir:

- Seleccione una plataforma de control de versiones (ej: GitLab, GitHub, Bitbucket, …)
- Dentro de la misma, debe crear un grupo:
  - Nombre: **SOyR 2024 Grupo X** (**X** representa el número de grupo).
  - Descripción: apellido, nombre y número de alumno de cada integrante del grupo.
  - Debe agregar al docente asignado con el rol de `Owner`.
- Dentro del grupo, debe crear un repositorio:
  - Nombre: **Entregable Práctica X** (X representa el número de práctica)
  - El repositorio debe contener lo siguiente:
    - Archivo de programa **eX.Y** (**X** representa el número de ejercicio. **Y** representa la extensión del archivo, ejemplo: **c**, **sh**) correctamente estructurado y con los comentarios pertinentes.
    - Archivo **README.md**. Usando sintaxis **markdown** incorporar lo siguiente:
      - Copia del enunciado
      - Interpretación del problema. (qué debo hacer, qué datos tengo y cómo se organizan, cuál es la interfaz con el usuario)
      - Propuesta de solución (Cómo resuelvo lo pedido) describo los modelos utilizados: algoritmos, matemáticos o conceptuales.

Aproximar la solución con un diagrama o pseudocódigo, modularizar.

- Instrucciones de uso

- Directorio `imagenes` ó `figuras` (opcional: en caso de que se requiera adjuntar imágenes en el `README.md`) con las figuras correspondientes.
- `.gitignore` (opcional: en caso de que haya trabajado con más archivos en local)