**UNIVERSIDAD SIMÓN BOLÍVAR**

Departamento de Computación y Tecnología de la Información

CI 3825 - Sistemas de Operación I

Trimestre Enero - Marzo 2023

---

Proyecto II

# Pokemon - Implementación de un buscador de información de los personajes en C

---

**Autores:**

Jesus Cuellar 15-10345

Felix Arnos 15-10088

Nestor Gonzalez 16-10455

José Pérez 16-10882

**Profesor:**

Fernando Torre Mora

5 de Abril de 2023

# Introducción

El proyecto se enfoca en desarrollar un buscador de información de los personajes en el lenguaje de programación C, el buscador va a recorrer los directorios del archivo suministrado por el profesor con distintos flags (“banderas”) para ir filtrando la búsqueda y poder contabilizar la cantidad de personajes que han interactuado con el personaje principal del programa de televisión llamado “Pokemon”.

La información de cada personaje que ha interactuado con Ash Ketchum (personaje principal en el que se enfoca la serie de televisión) se encuentra en un documento HTML, la finalidad del proyecto es poder contabilizar cuántas veces estos personajes aparecen en la serie, cada personaje tiene una clasificación, ya sea por región, especie y por número de apariciones.

El objetivo general del programa es poder contabilizar o realizar las cuentas de diferentes consultas sobre los personajes utilizando el lenguaje de programación C que satisfaga los requerimientos del buscador de personajes previamente planteados, teniendo como objetivo específico el poder realizar el debido análisis descendente para poder plantear una solución a este problema que pueda ser implementada de manera eficiente para el usuario y mantener los requerimientos del programa para que el usuario pueda realizar una búsqueda de información cómodamente.

En el informe que se presenta a continuación se puede apreciar cómo se implementan las ideas para el funcionamiento del programa y a su vez el estado actual del proyecto con un análisis de los resultados obtenidos.

# Estructura del programa y decisiones de diseño

Para resolver el problema planteado por el proyecto se realizó el siguiente análisis descendente:

Tomando en cuenta que el programa solicitado estaba planteado en base a una búsqueda por consola de la información de los personajes de la serie de televisión ya antes mencionada y tomando en cuenta las funciones que se debe utilizar en este programa, se chequea las condiciones de búsqueda para poder dar la respuesta esperada por el usuario. Luego de que se tienen las condiciones procedemos a realizar la búsqueda ejecutando las funciones asignadas a cada parte e igualmente dirigir a los directorios correspondientes para ir obteniendo la información solicitada. Para esto se plantearon 3 funciones, una función que abre los directorios correspondientes, una que captura las condiciones de cómo el usuario desea la información y otra para obtener el tamaño de los archivos.

**Estructura de datos utilizadas en el programa:**

En cuanto a la estructura de datos se construyó de la siguiente manera:

**#define PATH_MAX 100:** Este macro se utiliza para representar la longitud máxima de una ruta de archivo en el sistema.

**#include “utils.h”:** Este archivo de encabezado contiene prototipos de funciones, macros u otras declaraciones que se necesitan en el programa.

**#include <string.h>:** Incluye el archivo de encabezado de la biblioteca C estándar `string.h`, que proporciona varias funciones de manipulación de cadenas como `strcpy`, `strcat`, `strlen`, etc. Este archivo de encabezado es necesario en el programa para usar estas funciones de cadena.

**#include <stdio.h>:** Incluye la biblioteca estándar de entrada/salida en el programa. Esta biblioteca proporciona funciones para operaciones de entrada y salida, como `printf` y `scanf`.

**#include <stdlib.h>:** Es una directiva de preprocesador que incluye el archivo de encabezado de biblioteca estándar `stdlib.h` en el programa. Este archivo de encabezado contiene prototipos de funciones para varios estándares. funciones de biblioteca, como las funciones de asignación y desasignación de memoria (`malloc()`, `calloc()`, `realloc()`, `free()`), funciones de conversión (`atoi()`, `atof()`, `strtol()`, `strtod()`), y otras funciones de utilidad (`exit()`, `system()`, `rand()`, `srand()`, etc.). Al incluir este archivo de encabezado, el programa obtiene acceso a estas funciones y puede usarlas en el programa.

**#include <dirent.h>:** Es una directiva de preprocesador que incluye el archivo de encabezado `dirent.h` en el programa. Este archivo de encabezado proporciona funciones y tipos de datos para trabajar con directorios y entradas de directorio (archivos y

subdirectorios) en el sistema de archivos. En este programa, el `opendir()` y las funciones `readdir()` se utilizan para abrir y leer directorios, respectivamente.

**#include <sys/types.h>:** Es una directiva de preprocesador que incluye las declaraciones de varios datos tipos usados en llamadas al sistema y otras funciones en el sistema operativo Unix. Estos tipos de datos incluyen `pid_t`, `uid_t`, `gid_t`, `off_t`, `size_t` y otros. Este archivo de encabezado es necesario para muchas tareas de programación a nivel de sistema en sistemas operativos basados en Unix.

**#include <sys/stat.h>:** Es una directiva de preprocesador que incluye el archivo de encabezado `sys/stat.h` en el programa. Este archivo de encabezado contiene declaraciones para las funciones `stat()` y `fstat()`, que se utilizan para obtener información sobre un archivo o sistema de archivos. También contiene definiciones para el tipo de datos `struct stat` y `struct timespec`, que se utilizan para almacenar información sobre un archivo o sistema de archivos.

**#include <unistd.h>:** Incluye el archivo de encabezado para la API del sistema operativo POSIX, que proporciona acceso a varias funciones y constantes del sistema, incluidas funciones para interactuar con el sistema de archivos, la gestión de procesos y las llamadas al sistema para operaciones de bajo nivel. en este específico código, se usa para incluir la función `getcwd()`, que se usa para obtener el funcionamiento actual directorio.

**contador:** Variable que se usa para contar el número de elementos en los directorios.

**cwd[PATH_MAX]:** Variable que se usa para almacenar la ruta del directorio de trabajo actual.

***filtros[3]:** Arreglo donde se almacena cada filtro de los flags. (Variable global)

no contar: Variable que se usa para indicar si se va a contar la cantidad de archivos encontrados o no. (Variable global)

**listar:** Variable que se usa para indicar si se van a listar los archivos encontrados o no. (Variable global)

**mostrarTamArch:** Variable que se usa para indicar si se va a mostrar el tamaño de los archivos. (Variable global)

***nombreTarget:** Variable que se usa para filtrar las búsquedas. (Variable global)

**Descripción de los subprogramas y funciones implementadas en el programa:**

- **abrirDirectorio:** Función de lectura y evaluación de la entrada. La función abre directorios recursivamente y cuenta la cantidad de archivos regulares, mientras aplica filtros e impresión de nombres y tamaños de archivos basados en flags. Como es del tipo void no retorna nada. Y tiene como parámetro de entrada un string que representa el camino del directorio que se va a abrir y se va a realizar la búsqueda, un entero llamado “nivel”que representa el nivel actual de recursividad en el recorrido del árbol de directorios. Comienza en 0 para el directorio inicial y se incrementa en 1 para cada subdirectorio encontrado y por último tiene la variable contador que es un puntero a

una variable entera que realiza un seguimiento de la cantidad de archivos procesados por la función. La función incrementa esta variable en 1 cada vez que procesa un archivo regular.

- **capturarFlag:** Función que almacena los parámetros obtenidos por los flags y verifica su correctitud. Tiene como parámetro de entrada un arreglo de string llamado “argv” que contiene la línea de comando que se le pasa al programa. El otro parámetro “argc” es el número de argumentos pasados al programa, incluido el nombre del programa en sí.
- **esArchivoRegular:** Función que verifica si un inodo es un archivo regular. Se le pasa como parámetro un string “path” que representa el camino a un archivo. Retorna 1 si el archivo es regular y 0 si no lo es.
- **esDirectorio:** Función que verifica si un inodo es un directorio. Se le pasa como parámetro un string que representa el camino del archivo o directorio que se va a verificar. La función retorna 1 o 0, indicando si el inodo es un directorio o no.
- **tamanoArchivo:** Función que retorna el tamaño de un archivo en “kb”. Se le pasa como parámetro un string llamado “filename” que contiene el nombre del archivo al que se le quiere determinar el tamaño en “kb”.
- **comienzaCon:** La función verifica si una string comienza igual a otro string. Se le pasa un parámetro “a” que es un puntero a un arreglo de caracteres (cadena) que representa la cadena principal que queremos verificar si comienza con otra cadena. El otro parámetro “b” es un puntero a una cadena de caracteres constante que representa el prefijo que queremos buscar en la cadena "a". La función “comienzaCon” devuelve un valor entero de 1 si el string “a” comienza con el string “b”, y 0 en caso contrario.

# Cambios de implementación

Los cambios en la implementación pueden ser desafiantes, pero también pueden ser una oportunidad para mejorar el resultado final. La implementación de cualquier proyecto o programa es un proceso en constante evolución. A medida que se avanza en la ejecución, pueden surgir diversos obstáculos que requieren cambios en la implementación para poder alcanzar los objetivos propuestos. Sin embargo, no se han realizado cambios en la implementación del proyecto. Se sigue trabajando según el plan original y se está avanzando según lo previsto. Se mantienen las mismas estrategias y metodologías que se han venido utilizando hasta ahora, y se continúa trabajando para lograr los objetivos establecidos. Si bien no ha habido cambios en la implementación, se está monitoreando continuamente el progreso y se están evaluando posibles ajustes en caso de que sea necesario en el futuro.

Aunque en esta ocasión no fue posible realizar cambios en la implementación debido a restricciones de tiempo, es importante destacar que en el futuro se buscará realizar una refactorización del código para mejorar su calidad y eficiencia. El código puede ser mejorado constantemente en cualquier proyecto. Esta tarea se llevará a cabo una vez que se hayan completado las actividades prioritarias en curso y se cuente con el tiempo suficiente para dedicarle la atención necesaria.

# Conclusiones y Recomendaciones

Este proyecto de implementación de un buscador de información de personajes en el lenguaje de programación C se han implementado los contenidos correspondientes hasta la publicación del proyecto, tanto en la parte teórica de la materia como en los laboratorios a lo largo del curso. Los resultados obtenidos en el proyecto han sido cumplidos en su totalidad gracias a los objetivos planteados en dicho proyecto. Se notó que a la hora de realizar el proyecto se introdujeron subprogramas que optimizan el funcionamiento y facilitaron el planteamiento y la ejecución del programa. Una de las dificultades presentadas a la hora de realizar el programa fue poder cumplir con la salida del programa y sus diferentes filtros de búsqueda de información sobre los personajes, por otra parte, la otra dificultad que se presentó fue investigar la implementación de otros programas para el correcto funcionamiento de nuestro código.

A la hora de realizar el proyecto se evidencia la manera en los conocimientos obtenidos pueden ser aplicados de manera práctica, desarrollando una forma de pensar y abordar los problemas que se nos presentaron de la mejor manera posible para la realización del mismo, así los objetivos establecidos en éste se cumplieron en todos los campos requeridos para ejecutar el programa.

Teniendo en cuenta lo expuesto en este proyecto. ¿Podríamos decir que la implementación puede ser usada en un problema real? De hecho es muy útil mantener un filtrado de información por lo cuál este proyecto se puede trasladar a aplicaciones de mayor escala que solo una búsqueda de personajes ficticios. Es aplicable a otros problemas.