ALGORITMOS Y ESTRUCTURAS DE DATOS 2023/1

PROFESOR: Manuel Alejandro Moscoso Domínguez
manuel.moscoso.d@gmail.com

Laboratorio Semana 4

En esta oportunidad realizaremos actividades que refuercen el trabajo con estructuras, funciones, listas en C.

Objetivos

- Resolver ejercicios que involucren el uso de funciones.
- Resolver ejercicios que involucren tipos de datos abstractos y operaciones.
- Desarrollar algoritmos que permitan entregar una solución a los problemas entregados.

Ejercicios

Nomenclatura para nombre de archivos fuentes

El nombre del archivo en el cual se almacena el código fuente debe considerar el siguiente formato: labsemanaX_ejercicioY.EXT donde; X corresponde a la semana, la Y al número de ejercicios y EXT a la extensión del lenguaje de programación utilizado.

Ir a la ayuda memoría

Preámbulo

Para todos los ejercicios del presente laboratorio trabajaremos con la siguiente estructura

```
typedef struct {

    char name[50];
    int age;
    int room;
    float temperature;

} Patient;
```

Es importante considerar que podría sufrir modificación en base al enunciado de cada ejercicio.

Ejercicio número 1

Trabajaremos con un número máximo defino dentro del código y con un arreglo para lo cual es necesario realizar además las siguientes funciones:
addPatient: Función que lee por teclado la información solicita hasta alcanzar el límite de pacientes.

printPatients: Función que imprime en pantalla la información de todos los pacientes.

```
int main(){

    Patient patients[MAX];
```

Ejercicio número 2

Trabajaremos con un número variable por cada ejecución del programa que no debe superar el número máximo definido de manera global. Esto involucra la reformulación o refactoring de varias secciones de nuestro programa que debe considerar:

- **numPatients**: Variable presente en el main de nuestro programa que debe ser consultada al usuario.
- Actualizar la función **addPatient** para que pueda; recibir y revisar el valor de **numPatients** e incorporar un nuevo paciente si es factible o informar que ya no es posible. Al momento de agregar un nuevo paciente se debe actualizar el valor de **numPatients**.
- Actualizar la función **printPatients** para que pueda recibir el valor de **numPatients** para poder realizar la correcta impresión de los pacientes.
- Agregar al main() un algoritmo que permita la implementación de opciones al usuario considerando; imprimir lista, agregar paciente y salir.

Ejercicio número 3

Dentro de la implementación necesitamos realizar los ajustes necesarios para trabajar con la posibilidad de almacenar más de una temperatura por cada paciente. Para esto se deben realizar los siguientes ajustes:

- Actualizar estructura **Patient** para trabajar con un arreglo para la temperatura.
- Desarrollar la función **recordTemperature** para registrar una nueva temperatura para un paciente.
- Actualizar la función **printPatients** para trabajar con la nueva información.
- Actualizar el main() la sección que permita la implementación de una opción para agregar la temperatura a un paciente.

Ejercicio número 4

Dentro de la implementación necesitamos realizar los ajustes necesarios para trabajar con una lista en lugar de arreglos debido al dinamismo de la información. Para esto se deben realizar los siguientes ajustes:

- Actualizar estructura **Patient** para trabajar como una lista.
- Actualizar las funciones **addPatient y printPatients** para trabajar con listas.
- Actualizar las funciones **recordTemperature** para trabajar con listas.
- Desarrollar la función **dischargePatient** que da de alta un paciente.

- Desarrollar la función **freePatients** para realizar la liberación de recursos reservados para la problemática.

Ejercicio número 5

La mejor forma de gestionar a los pacientes en un centro de atención es a través de una cola. Como es una unidad de cuidados primarios los pacientes que llegan no son de gravedad y tiene la misma prioridad por lo que solo nos centraremos en la gestión que permita atender al primer que llega. Para esto se deben realizar los siguientes ajustes:

- Actualizar estructura **Patient** para trabajar como una cola en caso de ser necesario.
- Crear las funciones relacionadas con la gestión de colas (**enqueue** y **dequeue**).
- Actualizar el menú de opciones para trabajar con la cola excluyendo el registro de temperatura.
- Realizar la carga inicial de pacientes desde un archivo.
- Implementar las funciones:
    - getNextPatient, para conocer los datos del siguiente paciente sin sacarlo de la cola.
    - getQueueSize, para conocer cuántos pacientes faltan por atender.
    - getLeftPatients, para conocer cuántos pacientes faltan para el turno de un paciente.

Ejercicio	Indicadores	Ponderación
1	Desarrolla las funciones solicitadas con un correcto uso de parámetros	5%
2	Desarrolla la actualización de las funciones según se solicita	5%
	Desarrolla un algoritmo para hacer interactuar la usuario	10%
3	Desarrolla la actualización de la estructura según corresponde	5%
	Desarrolla la implementación de una nuevo función según se solicita	5%
	Desarrolla la actualización del algoritmo de interacción para manejar la temperatura.	15%
4	Desarrolla la implementación de la función para registro de temperatura con Listas	10%
	Desarrolla la implementación de la función para dar de alta un paciente.	10%
5	Desarrolla las funciones de encolar y desencolar	10%
	Desarrolla la implementación de la carga inicial desde un archivo.	5%
	Desarrolla la actualización del algoritmo para trabajar con cola.	10%
	Desarrolla las funciones adicionales e incluye estas opciones en el algoritmo	10%
	Total	100%

Ayuda memoria

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

// Definition
typedef struct patient {
    char name[20];
    int age;
    int room;
    struct patient *next;
} Patient;

Patient* createPatient(char *name, int age, int room);
void addPatient(Patient **head, Patient *newPatient);
void printPatients(Patient *head);
void freePatients(Patient *head);

int main() {
    Patient *head = NULL;
    Patient *newPatient;

    // Create three patient records
    newPatient = createPatient("Manuel", 32, 6);
    addPatient(&head, newPatient);

    newPatient = createPatient("Alejandro", 45, 12);
    addPatient(&head, newPatient);

    newPatient = createPatient("Rocio", 22, 53);
    addPatient(&head, newPatient);

    newPatient = createPatient("Amparo", 22, 53);
    addPatient(&head, newPatient);

    // Print the list of patients
    printPatients(head);

    // Free the memory used by the patients
    freePatients(head);

    system("pause");
    return 0;
}

// Function to create a new patient record
Patient* createPatient(char *name, int age, int room) {
    Patient *newPatient = (Patient*)malloc(sizeof(Patient));
    strcpy(newPatient->name, name);
    newPatient->age = age;
    newPatient->room = room;
    newPatient->next = NULL;
    return newPatient;
}
```

```c
// Function to add a new patient to the list
void addPatient(Patient **head, Patient *newPatient) {
    if (*head == NULL) {
        *head = newPatient;
    } else {
        Patient *current = *head;
        while (current->next != NULL) {
            current = current->next;
        }
        current->next = newPatient;
    }
}

// Function to print the list of patients
void printPatients(Patient *head) {
    printf("List of patients:\n");
    Patient *current = head;
    while (current != NULL) {
        printf("Name: %s, age: %d, Room number: %d.\n", current->name, current->age, current->room);
        current = current->next;
    }
}

// Function to free the list of patients
void freePatients(Patient *head) {
    Patient *current = head;
    while (current != NULL) {
        Patient *next = current->next;
        free(current);
        current = next;
    }
}
```

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

int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Usage: %s <file_name>\n", argv[0]);
        return 1;
    }
    char *file_name = argv[1];
    FILE *file = fopen(file_name, "r");
    if (file == NULL) {
        fprintf(stderr, "Error: Could not open file %s\n", file_name);
        return 1;
    }
```

```
    char line[100];
    while (fgets(line, 100, file) != NULL) {
        char *token;

//https://www.tutorialspoint.com/c_standard_library/c_function_strtok.htm
        token = strtok(line, ",");
        char name[50];
        strcpy(name, token);
        //https://www.tutorialspoint.com/c_standard_library/c_function_atoi.htm
        int age = atoi(strtok(NULL, ","));
        int room = atoi(strtok(NULL, ","));
        float temperature = atof(strtok(NULL, ","));
        printf("%s, age:%d T°:%.1f ROOM:%d\n",name,age,temperature,room);
    }
    fclose(file);

    return 0;
}
```

```
Juan García,28,101,37.2
María López,41,102,36.9
Pablo Martínez,19,103,37.1
Ana Ruiz,55,104,36.8
Carlos González,33,105,37.5
Elena Hernández,47,106,37.0
Diego Sánchez,25,107,37.3
Sofía Gómez,39,108,36.5
Alejandro Pérez,31,109,36.9
Lucía Fernández,63,110,37.2
Miguel Rodríguez,44,111,37.4
Luisa García,22,112,36.7
Javier Torres,51,113,37.0
Raquel Díaz,36,114,37.1
David Moreno,27,115,36.8
```