

## (Comisión 7) Programación y Laboratorio 2

Página Principal / Mis cursos / (Comisión 7) Programación y Laboratorio 2 / Exámenes / 1er Parcial Programacion 2021

Comenzado el Wednesday, 29 de September de 2021, 15:40

Estado Finalizado

Finalizado en Wednesday, 29 de September de 2021, 16:51

Tiempo empleado 1 hora 11 minutos

Calificación Sin calificar aún

Pregunta 1

Finalizado

Puntúa como

11,00

Marcar pregunta

Dada la siguiente estructura...

```
3
4  typedef struct{
5      char apellido[30];
6      char nombre[30];
7      char dni[10];
8  } stPersona;
9
```

- a) Codificar la estructura de lista doble  
 b) Codificar la función crearNodo() trabajada con puntero doble.

a)

```
typedef struct{
    stPersona dato;
    struct nodoDoble* anterior;
    struct nodoDoble* siguiente;
}nodoDoble;
```

b)

```
void crearNodo(nodoDoble** nuevoNodo, stPersona datoPersono){
    *nuevoNodo= (nodoDoble*) malloc(sizeof(nodoDoble));
    (*nuevoNodo)-> dato = datoPersono;
    (*nuevoNodo)-> anterior = NULL;
    (*nuevoNodo)-> siguiente = NULL;
}
```

```
typedef struct _nodo2{
    stPersona dato;
    struct _nodo2* anterior;
    struct _nodo2* siguiente;
}nodo2;

void crearNodo(stPersona dato, nodo2** nuevo{

    (*nuevo) = (nodo2*) malloc(sizeof(nodo2));

    (*nuevo)->dato = dato;
    (*nuevo)->anterior = NULL;
    (*nuevo)->siguiente = NULL;
})
```

Pregunta 2

Finalizado

Puntúa como

11,00

Marcar pregunta

Dada la siguiente estructura, hacer una función que cuente de manera iterativa en una lista simplemente enlazada cuántas personas existen en la misma cuyos Nros de dni sean múltiplos de 10.

```
3
4  typedef struct{
5      char apellido[30];
6      char nombre[30];
7      char dni[10];
8  } stPersona;
9
```

```
int contarMultiplos(nodo* lista){
    int rta = 0;
    if(lista){
        while(lista){
            if(lista-> dato.dni%10 == 0){
                rta++;
            }
            lista = lista-> siguiente;
        }
    }
    return rta;
}
```

```
int cuentaDniMultiplos10(nodo* lista)
{
```

Navegación por el cuestionario

1	2	3	4	5	6	7	8
9	10	11	12				

Mostrar una página cada vez

Finalizar revisión

```

nodo* seg = lista;
int cont = 0;

while(seg)
{
    if (seg->dato.dni%10==0){
        cont++;
    }
    seg = seg->siguiente;
}

return cont;
}

```

Pregunta 3  
Parcialmente correcta  
Puntuúa 4.80 sobre 6.00  
F Marcar pregunta

#### Vincular cada estructura con su correspondiente característica

Se necesitan dos vínculos en su estructura, uno para enlazar nuevos elementos y otro para quitarlos.

Los datos se deben insertar en el medio para quitarlos más fácilmente.

Invierte el orden de los elementos que en ella se almacenan.

El elemento que se almacenó primero es el primero en removese.

En esta estructura cada nodo almacena la dirección del nodo siguiente y del nodo anterior.

No coincide con ninguna estructura aprendida



No coincide con ninguna estructura aprendida



Pila



Fila



Lista doblemente enlazada



Respuesta parcialmente correcta.

Ha seleccionado correctamente 4.

La respuesta correcta es: Se necesitan dos vínculos en su estructura, uno para enlazar nuevos elementos y otro para quitarlos. → Fila, Los datos se deben insertar en el medio para quitarlos más fácilmente. → No coincide con ninguna estructura aprendida, Invierte el orden de los elementos que en ella se almacenan. → Pila, El elemento que se almacenó primero es el primero en removese. → Fila, En esta estructura cada nodo almacena la dirección del nodo siguiente y del nodo anterior. → Lista doblemente enlazada

Pregunta 4  
Correcta  
Puntuúa 6.00 sobre 6.00  
F Marcar pregunta

Una la prototipo de las funciones con una sinopsis de su funcionamiento

void agregar(Fila\* f, int dato);

Añade un nuevo dato a la fila y actualiza las referencias.



int filaVacia(Fila\* f);

Retorna 1 si la fila no contiene elementos y 0 si los tiene.



void mostrarFila(Fila\* f);

Invoca a la función mostrarLista del TDA Lista y le pasa como referencia el inicio de la fila



int extraer(Fila\* f);

Retorna y elimina el primer elemento de la fila



void inicFila (Fila\* f);

Recibe un puntero a una fila e inicializa sus atributos internos.



Respuesta correcta

La respuesta correcta es:

void agregar(Fila\* f, int dato); → Añade un nuevo dato a la fila y actualiza las referencias.,

int filaVacia(Fila\* f);

→ Retorna 1 si la fila no contiene elementos y 0 si los tiene.,

void mostrarFila(Fila\* f); → Invoca a la función mostrarLista del TDA Lista y le pasa como referencia el inicio de la fila, int extraer(Fila\* f); → Retorna y elimina el primer elemento de la fila,

void inicFila (Fila\* f); → Recibe un puntero a una fila e inicializa sus atributos internos.

Pregunta 5  
Finalizado  
Puntuúa como 10.00  
F Marcar pregunta

Explique las diferencias entre el comportamiento de una PILA y una FILA

La diferencia que hay es que en una fila el primer elemento en ser ingresado es el primero en salir (FIFO), en cambio la pila funciona al revés. También en la fila tenemos acceso al final y al principio de la misma.

La pila es una estructura de tipo LIFO (Last In - First Out) el último en entrar es el primero en salir en cambio la fila es de tipo FIFO (First In - First Out) el primero en entrar es el primero en salir

Pregunta 6  
Parcialmente correcta  
Puntuía 4,67 sobre 7,00  
 Marcar pregunta

Seleccione las opciones que considera correctas para las características de la función `malloc(..)`

(las opciones incorrectas restan puntos)

Seleccione una o más de una:

- a. Devuelve un puntero a una variable anónima
- b. Se usa solamente para crear un nodo
- c. Devuelve un nodo
- d. Crea y devuelve un puntero a una variable previamente declarada.
- e. Devuelve la dirección de memoria de un dato
- f. Necesita conocer cuántos bytes de memoria se quieren reservar
- g. Garantiza que la zona de memoria concedida no está ocupada por ninguna otra variable



Respuesta parcialmente correcta.

Ha seleccionado correctamente 2.

Las respuestas correctas son: Devuelve un puntero a una variable anónima, Garantiza que la zona de memoria concedida no está ocupada por ninguna otra variable, Necesita conocer cuántos bytes de memoria se quieren reservar

Pregunta 7  
Correcta  
Puntuía 6,00 sobre 6,00  
 Marcar pregunta

### Relación y diferencias entre Recursión e Iteración

Finaliza la ejecución: La iteración termina cuando

falla la condición de continuación del ciclo ✓

La recursión utiliza una

estructura de selección ✓

La iteración utiliza una

estructura de repetición ✓

Para repetir: la recursión consigue la repetición

mediante llamadas de función repetidas. ✓

Para repetir: La iteración utiliza una estructura

de repetición de forma explícita ✓

Respuesta correcta

La respuesta correcta es: Finaliza la ejecución: La iteración termina cuando → falla la condición de continuación del ciclo, La recursión utiliza una → estructura de selección, La iteración utiliza una → estructura de repetición, Para repetir: la recursión consigue la repetición → mediante llamadas de función repetidas., Para repetir: La iteración utiliza una estructura → de repetición de forma explícita

Pregunta 8  
Parcialmente correcta  
Puntuía 2,63 sobre 7,00  
 Marcar pregunta

Dada la siguiente Lista...

```
typedef struct{
    int dato,
    struct nodo* sig;
}nodo;
```

Píense en la función que borra un nodo de la lista buscándolo por un dato específico, cuyo prototipado sería el siguiente:

`nodo* borrarNodo (nodo* lista, int valor);`

Ordene los pasos que la función debería considerar para borrar el nodo y no perder los enlaces generados en la lista.

Primer

Que la lista no sea nula



Segundo

Se usa un puntero auxiliar que apunte a lista.



Tercero

Se asigna a lista el siguiente.



Cuarto

Se borra el nodo auxiliar.



Quinto

Que el nodo a borrar sea uno intermedio o el nodo final, para lo cual hay que trabajar con dos punteros auxiliares para recorrer la lista



Sexto

Se asigna a lista el siguiente.



Séptimo

Una vez encontrado el nodo, se enlaza el anterior con el siguiente del actual y se procede al borrado del actual



Octavo

Se reforma la lista



Respuesta parcialmente correcta.

Ha seleccionado correctamente 3.

La respuesta correcta es: Primero → Que la lista no sea nula, Segundo → Que el nodo a borrar sea el primero, por lo cual hay que modificar el valor de la variable lista, Tercero → Se usa un puntero auxiliar que apunte a lista., Cuarto → Se asigna a lista el siguiente., Quinto → Se borra el nodo auxiliar, Sexto → Que el nodo a borrar sea uno intermedio o el nodo final, para lo cual hay que trabajar con dos punteros auxiliares para recorrer la lista, Séptimo → Una vez encontrado el nodo, se enlaza el anterior con el siguiente del actual y se procede al borrado del actual,

Octavo → Se reforma la lista

Pregunta 9  
Parcialmente correcta  
Puntuía 4,80 sobre 6,00  
 Marcar pregunta

Elija LA o LAS respuestas correctas. Las respuestas incorrectas restan puntos.

Las reglas de la buena recursividad son:

Seleccione una o más de una:

- 1. Que exista una condición de corte explícita
- 2. Que exista una solución trivial explícita
- 3. Que exista un acercamiento a la condición de corte
- 4. Que exista una llamada directa o indirecta de la función a sí misma
- 5. Que exista al menos una llamada a la función recursiva en el main
- 6. Al llegar a la Solución Trivial queda expresada la solución total
- 7. Que exista una condición de corte
- 8. Que la función recursiva utilice al menos un if y un else dentro de su código

\* Puede trabajarse solamente con un if, no es necesario utilizar si o si un if y un else (recordar caso de la condición de corte y solución trivial)

Respuesta parcialmente correcta.

Ha seleccionado demasiadas opciones.

Las respuestas correctas son: Que existe una condición de corte, Que existe un acercamiento a la condición de corte, Al llegar a la Solución Trivial queda expresada la solución total, Que existe una llamada directa o indirecta de la función a sí misma

#### Pregunta 10

Finalizado

Puntúa como  
10.00

Marcar pregunta

Explique las diferencias entre una lista simple y una lista doble, y codifique las estructuras que utiliza cada una de ellas.

La diferencia que hay entre un nodo simple y uno doble es que en el simple la estructura del nodo incluye la dirección de memoria del nodo que le sigue, en cambio el nodoDoble contiene la dirección de memoria del nodo que le sigue y ademas el puntero del nodo anterior al mismo.

Las listas simples pueden recorrerse solo hacia adelante, y las listas dobles pueden recorrerse desde el final (utilizando la dirección de memoria del último nodo en la lista)

```
typedef struct{
    int dato;
    struct nodo* siguiente;
}nodo;

typedef struct{
    int dato;
    struct nodoDoble* anterior;
    struct nodoDoble* siguiente;
}nodoDoble;
```

#### Pregunta 11

Finalizado

Puntúa como  
9.00

Marcar pregunta

Explique porqué es importante inicializar una lista en NULL

Es importante inicializar una lista en NULL ya que si no lo hacemos la lista estaría apuntando a datos basura.

La lista debe inicializarse en NULL porque es necesario saber si está vacía o no al momento de cargarla, y luego también nos servirá para saber si está o no vacía cuando utilicemos otras funciones (borrado de nodos, inserción en orden, etc)

#### Pregunta 12

Finalizado

Puntúa como  
11.00

Marcar pregunta

Complete los 5 datos que faltan para poder realizar el pasaje de datos de un archivo a una lista:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "fila.h"
4
5 int main()
6 {
7     nodo* lista;
8     lista=inicLista();
9
10    archivo2lista( 1 );
11
12    return 0;
13 }
14
15 void archivo2lista( 2 ){
16     FILE* archi = fopen("personas.dat","rb");
17     stPersona p;
18
19     if(archi){
20         while(fread(&p, sizeof(stPersona), 1, archi)>0){
21             3 = agregarEnOrdenApellido( 4 , 5 );
22         }
23     }
24 }
```

1 => &lista  
2 => nodo\*\* lista  
3 => \*lista  
4 => p.apellido(supongo que hay un campo apellido, ya que no estoy seguro de como esta hecha la estructura stPersona)  
5 => \*lista  
el 4 y el 5 podrían ser al revés porque no se en que orden pide los parametros

1. &lista  
2. nodo\*\* lista  
3.(\*lista)  
4. (\*lista)  
5. crearNodo(p)

Finalizar revisión

◀ PPoint Añadir al final en listas con recursividad

Ir a...



Campus Virtual  
UTN Mar del Plata

### Info

[UTN Mar del Plata](#)  
[SICU Mar del Plata](#)  
[DEPORTES Mar del Plata](#)  
[UTN Rectorado](#)

### Contacto

Calle Buque Pesquero Dorrego N° 281  
Teléfono : (0223) 480 - 5049 / 1220  
Correo electrónico : [campus@mdo.utn.edu.ar](mailto:campus@mdo.utn.edu.ar)

Redes sociales



UTN Mar del Plata

Resumen de retención de datos  
Descargar la app para dispositivos móviles