

1. Escriba en el paréntesis la letra que corresponde al concepto:

- (**I**) Estructura de datos lineal, en la cual el elemento obtenido a través de la operación ELIMINAR está predefinido y es el que se encuentra al inicio de la estructura
- (**C**) No se referencian bajo un mismo nombre, es un nuevo tipo de dato
- (**E**) Variable que contiene la dirección de otra variable
- (**C**) Conjunto de datos o variables de diferentes tipos
- (**F**) Conjunto de nodos alineados de manera lineal (uno después de otro) y unidos entre sí por una referencia
- (**k**) Concepto lógico que permite almacenar información
- (**D**) Se referencian bajo un mismo nombre
- (**G**) Colección de variables del mismo tipo
- (**F**) Está constituida por un conjunto de nodos alineados de manera lineal (uno después de otro) y unidos entre sí por dos referencias,
- (**E**) Permite acceder a los elementos de una estructura
- (**L**) Para utilizarlo se establece un área de buffer
- (**G**) Permite organizar información en listas o tablas
- (**E**) Se refiere a la dirección de una variable
- (**A**) Se almacenan en memoria secundaria
- (**J**) Permite el acceso al valor de una variable
- (**D**) Unidad básica de una lista
- (**G**) Permite almacenar N elementos de un mismo tipo y acceder a ellos mediante un índice
- (**G**) Se define su tamaño al momento de declararlo
- (**C**) Permite mantener junta y organizada cierta información relacionada
- (**B**) Estructura de datos lineal y dinámica, en la cual el elemento obtenido a través de la operación ELIMINAR está predefinido, debido a que implementa la política Last-In, First-Out (LIFO)

- A. Archivo
- B. Pila
- C. Estructura
- D. nodo
- E. Apuntador
- F. Lista simple
- G. Arreglo
- H. Lista doble
- I. Cola
- J. *
- K. &
- L. . (punto)

2. ¿Cómo se realizan las llamadas a una función por referencia?

```
1 //Llamada de funcion por una referecna
2 #include<stdio.h>
3 int funcionproto(int *num1, int *num2); //se crea la funcion prototipo
4
5 int main()
6 {
7     int num1 = 0, num2= 0;
8     printf("Antes num1 es %d y num2 es %d\n", num1, num2);
9
10    // Con & obtenemos la dirección de los numeros
11    funcionproto(&num1, &num2);
12    printf("\nDespues num1 es %d y num2 es %d\n", num1, num2);
13 }
14
15 // Colocamos lo que queremos que la funcion haga por lo que rescribimos la funcion como lo hicimos arriba
16 int funcionproto(int *num1, int *num2){
17     (*num1) = 45;
18     (*num2) = 102;
19 }
20
21
22
```

3. ¿Qué se muestra en pantalla?

```
#include<stdio.h>
#include<stdlib.h>
main(){
int *p,x=3;
p=&x;
printf("\nel valor de %x %x %x ",x,&x,x);
printf("\nel valor de %x %x %x ",*p,&p,p);
printf("\n");
}
```

Suponiendo que la dirección de x es 22ff10

Lo que se muestra en pantalla en el primer printf es el numero almacenado, la dirección de memoria en la que se encuentra y nuevamente el numero 3 el cual es el valor de x

En el segundo printf se muestra el numero que se almaceno en p, luego la dirección en la que se encuentra guardado dicho numero y luego la dirección en la cual esta almacenado p que es la misma dirección que aparece en el primer printf.

4. Califique falso o verdadero, si es falso justifique su respuesta, esta sección se calificara aciertos menos errores

El nombre de un arreglo es un apuntador (V)

Los archivos solamente se almacenan en memoria secundaria (V)

Los apuntadores no se pueden multiplicar o dividir (F)

No se puede ya que guardan direcciones de los valores almacenados

Los arreglos se pueden recorrer con un ciclo de repetición while (V)

El *, no solamente se aplican a apuntadores enteros (V)

Con & puedo conocer la dirección de un apuntador (V)

El operador -> se utiliza en las estructuras (V)

5. ¿Cuántos bytes se reservan?

Se reservan 16 bits

6. Suponiendo que la dirección de c1[0]=20103000 ¿Cuál sería la dirección de c1[3]

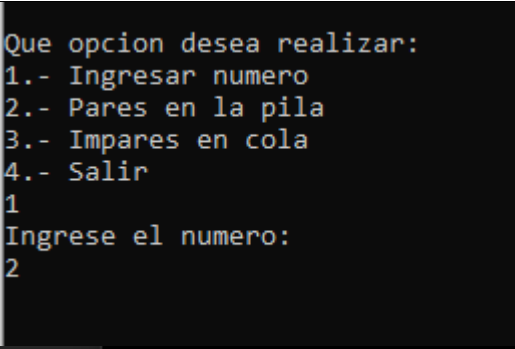
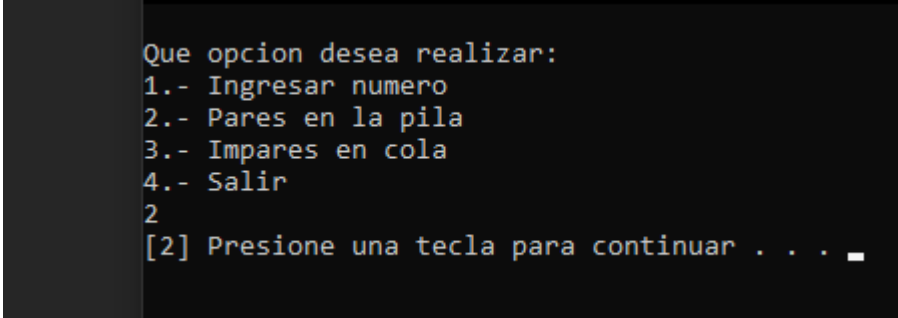
```
#include<stdio.h>
#include<stdlib.h>
struct cuates{
int a,b[4];
};

main(){
struct cuates *c1;
c1=(cuates*)malloc(4*sizeof(cuates));
printf("\n%d %d",&c1[0],&c1[1]);

printf("\n");
system("PAUSE");
}
```

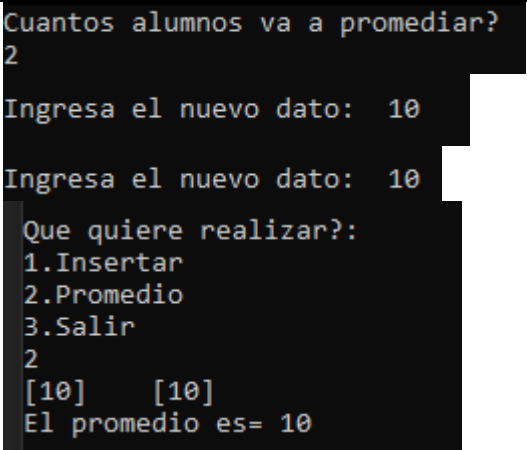
Suponiendo que para un dato int se reservan 4 byte

7. Elaborar un programa donde se pida un numero al usuario si es par almacenarlo en una pila si es impar almacenarlo en una cola

<pre>#include <stdio.h> #include <stdlib.h> int main(){ int *cola,op,i,j,n=50,pila[50],x; cola=(int*)malloc(n*sizeof(int)); do{ printf("\nQue opcion desea realizar:"); printf("\n1.- Ingresar numero"); printf("\n2.- Pares en la pila"); printf("\n3.- Impares en cola"); printf("\n4.- Salir\n"); scanf("%d",&op); switch(op){ case 1: printf("Ingrese el numero: \n"); scanf("%i",&x); break; case 2:</pre>	
	

<pre> if(x%2==0){ pila[i]=x; printf("[%d] ",pila[i]); i++; } system("pause"); break; case 3: if(x%2==1){ cola[j]=x; printf("[%d] ",cola[j]); j++; } system("pause"); break; case 4: printf("Hasta pronto!!!\n"); system("pause"); break; } system("cls"); }while(op<4); } </pre>	
---	--

8. Elaborar un programa que obtenga el promedio de n alumnos almacenados en una lista simple

CÓDIGO	IMAGEN
<pre> #include<stdio.h> #include <stdlib.h> struct listA{ int dato; listA *siguiente; }; listA *lista; void insertar(); void promedioTotal(listA *siguiente); int alum; main(){ printf("Cuantos alumnos va a promediar?\n"); scanf("%d",&alum); int op,b=0; do{ printf("\nQue quiere realizar?: "); printf("\n1.Insertar"); </pre>	 <pre> Cuantos alumnos va a promediar? 2 Ingresa el nuevo dato: 10 Ingresa el nuevo dato: 10 Que quiere realizar?: 1.Insertar 2.Promedio 3.Salir 2 [10] [10] El promedio es= 10 </pre>

```

        printf("\n2.Promedio");
        printf("\n3.Salir\n");
        scanf("%d",&op);
        switch(op){
            case 1:
                insertar();
                break;
            case 2:
                promedioTotal(lista);
                break;
            case 3:
                b=1;
                break;
        }
    }while(b==0);

    return 0;
}

void insertar(){
    listA
    *nuevo=(listA*)malloc(alum*sizeof(listA));
    nuevo->siguiente=lista;
    printf("\nIngresa el nuevo dato: ");
    scanf("%d",&nuevo->dato);
    lista=nuevo;
}

void promedioTotal(listA *siguiente){
    listA *indice=lista;
    int suma=0,n=0,prom=0;
    while(indice!=NULL){
        suma=suma+indice->dato;
        printf("[%d]\t",indice->dato);
        indice=indice->siguiente;
        n++;
    }
    int promedio;
    promedio=suma/n;
    printf("\nEl promedio es=
%d\n",promedio);
    printf("\n");
    if(indice!=NULL)
        printf("\n      No hay datos en
la lista");
}

```

9. Implementar una lista doblemente ligada con las funciones, buscar, insertar, borrar y mostrar

CÓDIGO	IMAGEN
<pre> #include <stdio.h> #include <stdlib.h> struct nodo{ int dato; nodo *siguiente; nodo *anterior; }; nodo *primero=NULL; nodo *ultimo=NULL; void insertarNodo(); void buscarNodo(); void eliminarNodo(); void mostrarLista(); int main(){ int op; do{ printf("Elige una opcion:\n"); printf("1.-Insertar\n"); printf("2.-Buscar\n"); printf("3.-Eliminar\n"); printf("4.-Mostrar\n"); printf("5.-Salir\n"); scanf("%d",&op); switch(op){ case 1: insertarNodo(); break; case 2: buscarNodo(); break; case 3: eliminarNodo(); break; case 4: mostrarLista(); break; case 5: printf("\nVuelva pronto\n"); break; } }while(op<5); } </pre>	 <pre> Ingrese el dato: 3 Dato ingresado correctamente Elige una opcion: 1.-Insertar 2.-Buscar 3.-Eliminar 4.-Mostrar 5.-Salir 1 Ingrese el dato: 4 Dato ingresado correctamente Elige una opcion: 1.-Insertar 2.-Buscar 3.-Eliminar 4.-Mostrar 5.-Salir 4 [3] [4] 1.-Insertar 2.-Buscar 3.-Eliminar 4.-Mostrar 5.-Salir 3 Ingresa el dato a buscar: 4 Se encuentra el dato Nodo eliminado Elige una opcion: 1.-Insertar 2.-Buscar 3.-Eliminar 4.-Mostrar 5.-Salir 4 [3] Elige una opcion: 1.-Insertar 2.-Buscar 3.-Eliminar 4.-Mostrar 5.-Salir </pre>

```

void insertarNodo(){
    nodo
    *nuevo=(nodo*)malloc(sizeof(nodo));
    printf("Ingrese el dato:\n");
    scanf("%d",&nuevo->dato);
    if(primeros==NULL){
        primeros=nuevo;
        primeros->siguiente=primeros;
//ligarse
        ultimo=primeros;
        primeros->anterior=ultimo;
    }
    else{
        ultimo->siguiente=nuevo;
        nuevo->siguiente=primeros;
        nuevo->anterior=ultimo;
        ultimo=nuevo;
        primeros->anterior=ultimo;
    }
    printf("\nDato ingresado
correctamente\n");
}

void buscarNodo(){
    int nodoBuscado,encontrado=0;
    nodo
    *actual=(nodo*)malloc(sizeof(nodo));
    actual=primeros;
    printf("Ingresa el dato a buscar:\n");
    scanf("%d",&nodoBuscado);
    if(primeros!=NULL){
        do{
            if(actual-
>dato==nodoBuscado){
                printf("Se
encuentra el dato\n");
                encontrado=1;
            }
            actual=actual-
>siguiente;
        }while(encontrado==0 &&
actual!=primeros);
        if(encontrado==0){
            printf("No se encuentra
el dato\n");
        }
    }
    else{
        printf("La lista se encuentra
vacía\n");
    }
}

void eliminarNodo(){

```

```

        int nodoBuscado,encontrado=0;
        nodo
        *actual=(nodo*)malloc(sizeof(nodo));
        actual=primero;
        nodo
        *ant=(nodo*)malloc(sizeof(nodo));
        ant=NULL;
        printf("Ingresa el dato a buscar:\n");
        scanf("%d",&nodoBuscado);
        if(primero!=NULL){
            do{
                if(actual->
                >dato==nodoBuscado){
                    printf("Se
                    encuentra el dato\n");

                    if(actual==primero){

                        primero=primero->siguiente;

                        primero->anterior=ultimo;
                        ultimo->
                        >siguiente=primero;
                    }
                    else
                        if(actual==ultimo){

                            ultimo=ant;
                            ultimo->
                            >siguiente=primero;

                            primero->anterior=ultimo;
                        }
                        else{
                            ant->
                            >siguiente=actual->siguiente;
                            actual->
                            >siguiente->anterior=ant;
                        }
                        printf("Nodo
                        eliminado\n");
                        encontrado=1;
                    }
                    ant=actual;
                    actual=actual->
                    >siguiente;
                }while(encontrado==0 &&
                actual!=primero);
                if(encontrado==0){
                    printf("No se encuentra
                    el dato\n");
                }
                else{
                    free(ant);

```


<pre> } } else{ printf("La lista se encuentra vacía\n"); } } void mostrarLista(){ nodo *actual=(nodo*)malloc(sizeof(nodo)); actual=primero; if(actual!=NULL){ do{ printf("[%d]\t",actual-> dato); actual=actual-> siguiente; }while(actual!=primero); printf("\n"); } else{ printf("La lista esta vacía\n"); } } </pre>	
--	--

10. Elaborar un programa que obtenga los coeficientes de la serie de Fibonacci hasta n en una cola circular