Estructuras de Datos y Algoritmos I

Primer examen parcial 2021-1

Nombre:\_\_Sánchez Alvirde Andrés Iván

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

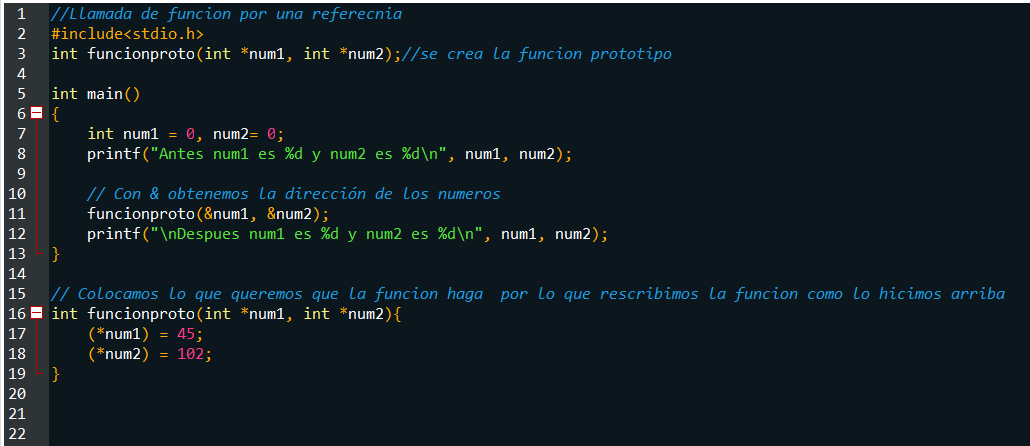
( **G** ) Permite almacenar N elementos de un mismo tipo y acceder a ellos mediante un índice

( **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)

1. Archivo
2. Pila
3. Estructura
4. nodo
5. Apuntador
6. Lista simple
7. Arreglo
8. Lista doble
9. Cola
10. \*
11. &
12. . (punto)
13. ¿Cómo se realizan las llamadas a una función por referencia?

****

1. ¿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.**

1. 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** )

1. ¿Cuántos bytes se reservan?

**Se reservan 16 bits**

1. 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

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

|  |  |
| --- | --- |
|  |  |
| #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:  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);  } |  |

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

|  |  |
| --- | --- |
| **CÓDIGO** | **IMAGEN** |
| #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");  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");  } |  |

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

|  |  |
| --- | --- |
| **CÓDIGO** | **IMAGEN** |
| #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);  }  void insertarNodo(){  nodo \*nuevo=(nodo\*)malloc(sizeof(nodo));  printf("Ingrese el dato:\n");  scanf("%d",&nuevo->dato);  if(primero==NULL){  primero=nuevo;  primero->siguiente=primero; //ligarse  ultimo=primero;  primero->anterior=ultimo;  }  else{  ultimo->siguiente=nuevo;  nuevo->siguiente=primero;  nuevo->anterior=ultimo;  ultimo=nuevo;  primero->anterior=ultimo;  }  printf("\nDato ingresado correctamente\a\n");  }  void buscarNodo(){  int nodoBuscado,encontrado=0;  nodo \*actual=(nodo\*)malloc(sizeof(nodo));  actual=primero;  printf("Ingresa el dato a buscar:\n");  scanf("%d",&nodoBuscado);  if(primero!=NULL){  do{  if(actual->dato==nodoBuscado){  printf("Se encuentra el dato\n");  encontrado=1;  }  actual=actual->siguiente;  }while(encontrado==0 && actual!=primero);  if(encontrado==0){  printf("No se encuentra el dato\n");  }  }  else{  printf("La lista se encuentra vacia\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);  }  }  else{  printf("La lista se encuentra vacia\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 vacia\n");  }  } |  |

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