**FINAL PROGRAMACIÓN 2 – DICIEMBRE 2018**

1. a)

int \*pa, b, a[10] = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19};

pa = &a[4];

b=\*pa + 1;

¿Qué valor está almacenado en b?

* 11
* 10
* 8

El valor almacenado en 10 ya que: \*pa = a[4] = 9. Si a \*pa = 9 se le suma 1, (9 + 1) el valor termina siendo 10.

b) Encontrar el error y justificar:

void main ()

{

int a = 1, b;

int \*const p = &a;

p = &b;

\*p = 4;

}

Si declaro el puntero como constante entonces no puedo hacer que después apunte a otra dirección de memoria.

1. Definición de estructura. Crear estructura Persona con los siguientes campos: dni, nombre, apellido, fecha de nacimiento (estructura de tipo fecha con campos dia, mes, año). Crear una variable de tipo Persona y escribir el código para llenarla por teclado.

Las estructuras son agrupaciones de datos con organización interna y operaciones propias. Esto es ventajoso para acceder a la información y manejar los datos. Los lenguajes de programación permiten manipular una estructura completa como si se tratara de un dato individual. Pueden clasificarse en:

• Contiguas: son aquellas que se van a representar en la memoria en posiciones adyacentes.

• Enlazadas: aquellas en que no hay obligación de situarse de forma contigua en la memoria. Los datos se relacionan unos con otros mediante punteros.

• Estáticas: su tamaño ocupado en memoria se define con anterioridad a la ejecución del programa y durante este no puede modificarse. Las estructuras estáticas siempre se representan en la memoria de forma contigua.

• Dinámicas: pueden crecer o decrecer durante la ejecución, adquiriendo o liberando posiciones de memoria y la única limitación respecto a tamaño es la memoria de la computadora; son representadas con la ayuda de punteros.

typedef struct Persona

{

int dni;

char nombre[30];

char apellido[30];

fecha nacim;

}Persona;

typedef struct fecha

{

int dia;

int mes;

int ao;

}fecha;

int main()

{

Persona p;

printf(“Ingrese el dni: “);

scanf(“%d”, &(p.dni));

printf(“\nIngrese el nombre: “);

fflush(stdin);

gets(p.nombre);

printf(“\nIngrese apellido: “);

fflush(stdin);

gets(p.apellido);

printf(“\nIngrese fecha de nacimiento (d-m-a): “);

scanf(“%d-%d-%d”, &(p.nacim.dia), &(p.nacim.mes), &(p.nacim.ao));

return 0;

}

1. a) Escribir una instrucción que abra el archivo “Personas.dat” en modo escritura para grabar un registro al final del archivo.

FILE \* arch = fopen(“Personas.dat”, “ab”);

b) Grabar un registro en el archivo con la información guardada en la variable tipo Persona del punto 2.

fwrite(&p, sizeof(Persona), 1, arch);

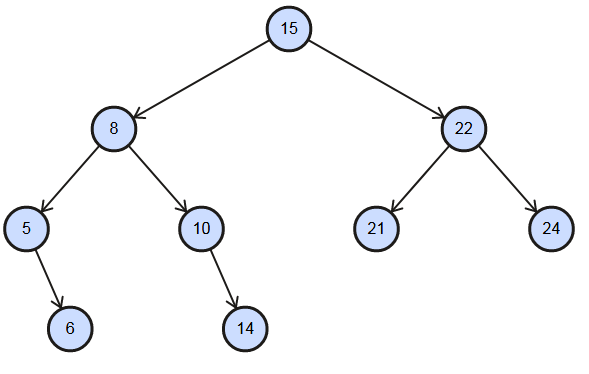
1. Características de árbol de búsqueda binaria. Llenar uno con los siguientes datos partiendo de un árbol vacío:

15 - 22 - 8 – 10 – 24 – 5 – 14 – 22 – 21 – 6

Mostrarlo inorden y plantear el algoritmo.

Árbol binario: Es un tipo especial de árbol en el que de cada nodo solo pueden colgar 2 subárboles. Es una estructura de datos no lineal jerárquica que consiste en un campo para un dato entero y dos punteros que indican la posición de sus hijos. Cumple que el subárbol izquierdo de cada nodo, si no está vacío, contiene un valor menor a dicho nodo, y el subárbol derecho, si no está vacío, contiene uno mayor.

Se tienen varios tipos de nodos: hijo (cualquiera de los nodos apuntado por uno de los nodos del árbol), padre (nodo que apunta a otro nodo), raíz (nodo que no tiene padre), y hoja (nodo que no tienen hijos).



inorden: 5 – 6 – 8 – 10 – 14 – 15 – 21 – 22 – 24

Para imprimir inorden primero se llama a la misma función inorden pero sobre el subárbol izquierdo, luego imprime el dato en la raíz y por último vuelve a llamar a la función inorden recursivamente pero esta vez sobre el subárbol derecho.

Suponiendo una estructura:

typedef struct nodo

{

int dato;

struct nodo \* der;

struct nodo \* izq;

}nodo;

void inorden (nodo\*r)

{

if(r==NULL)

{

printf(“El árbol está vacío”);

} else

{

inorden(r->izq);

printf( “%d – “, r->dato);

inorden(r->der);

}

}

1. Escribir un algoritmo que permita eliminar todos los nodos de una lista simplemente enlazada con estructura:

typedef struct lista

{

int dato;

struct lista \* sig;

}nodo;

void eliminarLista (nodo\*\*c)

{

nodo\* aux;

while ((\*c)!=NULL)

{

aux=\*c;

(\*c) = (\*c) -> sig;

free(aux);

}

}