**Cuestionario Practica 2**

1. ***¿Cuál es la diferencia entre el operador de referencia, &, y el operador de-referencia,\*?***

Ambos operadores son utilizados en el manejo de apuntadores en programación en C, la diferencia es que el operador de referencia “&” o también llamado operador de dirección, devuelve la dirección en donde se almacena el operando. Mientras que el operador “\*”, también llamado operador de indirección o de des-referencia devuelve el valor que se encuentra almacenado en la variable a la que apunta.

1. ***¿Cuál es la diferencia entre el paso de argumentos por copia y por referencia?***

La diferencia es que cuando se hace paso de argumentos por referencia, se pasa la posición de memoria donde esta guardada la variable, por lo que la función sabe el valor que contiene y cualquier cambio realizado en la función, se realiza directamente en la variable.

Cuando se hace un paso de argumentos por copia la función recibe una copia de lo que contiene la variable, por lo que cualquier cambio realizado en ella durante la función, no cambia el contenido de la variable original.

1. ***¿Cómo se pasa un apuntador a una función?***

Uno de los usos de punteros es el paso por referencia de parámetros a una función, lo que se hace es pasar un puntero al objeto que se pasa. Así la función tiene acceso no sólo al valor del parámetro sino también a su situación en memoria, lo que permite su modificación.

En ejemplo de paso de un entero por referencia a una función que lo incremente:

int x=14;

inc(&x);

…

void inc(int \*par){

(\*par)++;

}

El paso por referencia posibilita que una función pueda generar más de un valor. La función devolverá uno de los valores de retorno y almacenará en parámetros pasados por referencia el resto de los valores.

1. ***¿Qué se puede hacer para que una función devuelva varios resultados?***

Se pueden utilizar apuntadores de forma que se haga un paso de parámetros por referencia.

Ejemplo:

Void suma\_y\_resta(\*p,\*q)

{

x = \*p;

y = \*q;

\*p=x+y;

\*q=x-y;

}

main(){

int n,m;

printf(“Dame un numero”);

scanf(“n”);

printf(“Dame otro numero”);

scanf(“m”);

suma\_y\_resta(&n,&m);

printf(“La suma es: %d”, n);

printf(“La resta es: %d”, m);

}

***5. ¿Qué es una estructura?***

Es una colección de uno o más tipos de elementos denominados miembros, cada uno de los cuales puede ser de un tipo diferente.

Declaración:

struct<nombreDeLaEstructura> {

<tipoDeDato> <nombreDelMiembro>

<tipoDeDato> <nombreDelMiembro>

…

<tipoDeDato> <nombreDelMiembro>

};

***6. ¿Cómo se accesa a los elementos de una estructura?***

Cuando se accesa a una estructura, o bien, se almacena información en la estructura o se recupera información de ella, puede acceder a los miembros de una estructura utilizando el operador punto [.]

Sintaxis:

<nombre\_variable\_estructura>.<nombre\_miembro>;

Ejemplo:

strcpy(cd\_1.titulo, “Titulo\_de\_mi\_cd\_1”);

cd\_1.cro\_canciones = 20;

cd\_1.precio = 78:

***7. ¿Cómo se pasa por referencia una variable estructura a una función? ¿Por qué se recomienda pasar variables estructuras por referencia en lugar de por copia?***

Las estructuras se pueden pasar directamente a una función igual que se hace con las variables. En la definición de la función se debe de indicar el tipo de argumento que se usa.

Otra forma de pasar estructuras en mediante punteros, pero se corre el riesgo de poder cambiar los datos de la estructura de la variable que se utiliza en el main.

Otra posibilidad es solo pasar los miembros de la estructura que sean necesarios.

***8. ¿Cómo se accesa a los campos de una estructura que fue pasada a una función por referencia?***

Se accede de la misma forma que si se estuviera manejando la estructura en el main.