

PROGRAMACIÓN I

EVALUACIÓN DIAGNÓSTICA

# UNIDAD 1

**REPASO E INTRODUCCIÓN A LA MATERIA**

**Autor de contenidos:**

Nicolás Battaglia

# OBJETIVOS

Realizar un repaso general de programación estructurada

# ENUNCIADO

1. Describir con sus palabras, cada uno de los siguientes items
   * Programación: es uno de los pasos en el desarrollo de aplicaciones. Estrictamente hablando, es la parte de codificación en un proyecto de creación de un programa.
   * Algoritmos: Es una secuencia de instrucciones propuesta como solución a un determinado problema de índole informática.
   * Vectores: colección unidimensional de datos del mismo tipo.
   * Funciones: subconjunto de instrucciones llamados por un procedimiento padre que devuelve una respuesta.
   * Procedimientos: subconjunto de instrucciones del que no se espera respuesta y que es ejecutado desde un procedimiento padre.
   * Variables: porciones de memoria etiquetadas en las que se alojan valores a ser recuperados y procesados a través del nombre de esa etiqueta.
2. Realizar el código necesario para resolver el siguiente enunciado. Utilizar C.

Una fábrica de autos tiene los siguientes lotes de datos

**Lote 1** composición del auto

* + Modelo
  + Pieza
  + Cantidad a usar de esa pieza

**Lote 2** stock de modelos terminados

* + Modelo
  + Stock

**Lote 3** stock de piezas

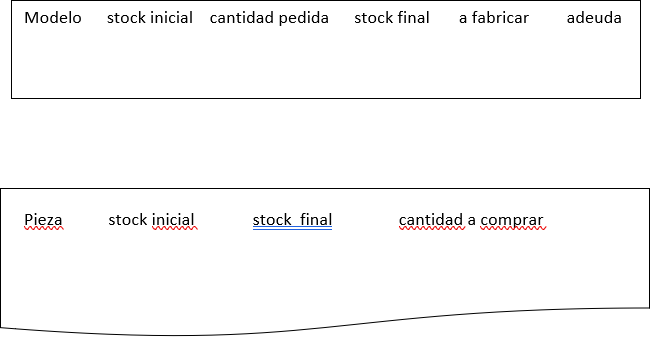
* + Pieza
  + Stock

**Lote 4** pedidos de las sucursales, finaliza con modelo = 0

* + Modelo
  + Cantidad pedida en total

La fábrica construye 20 modelos de auto, que utilizan entre sí 100 piezas distintas

**Se desea saber**



#include <stdio.h>

// bloques de datos

struct MODELO

{

int modelo;

int pieza;

int piezas;

};

struct TERMINADO

{

int modelo;

int stock;

};

struct PIEZA

{

int pieza;

int stock;

};

struct PEDIDO

{

int modelo;

int cantidad;

};

// rutinas

void

cargarModelos (struct MODELO \*apuntador, int N)

{

for (int A = 0; A < 20; A++)

{

for (int B = 0; B < 100; B++)

{

apuntador[B].modelo = A + 1;

apuntador[B].pieza = B + 1;

apuntador[B].piezas = B + 101;

printf ("Modelo: %d \t", apuntador[B].modelo);

printf ("Pieza: %d \t", apuntador[B].pieza);

printf ("Piezas: %d \n", apuntador[B].piezas);

}

}

};

void

cargarTerminados (struct TERMINADO \*apuntador, int N)

{

for (int A = 0; A < 20; A++)

{

apuntador[A].modelo = A + 1;

apuntador[A].stock = (A + 1) \* 2;

printf ("Modelo: %d \t", apuntador[A].modelo);

printf ("Stock: %d \n", apuntador[A].stock);

}

};

void

cargarPiezas (struct PIEZA \*apuntador, int N)

{

for (int A = 0; A < 100; A++)

{

apuntador[A].pieza = A + 1;

apuntador[A].stock = A \* 10 / 2;

printf ("Modelo: %d \t", apuntador[A].pieza);

printf ("Stock: %d \n", apuntador[A].stock);

}

};

void

cargarPedidos (struct PEDIDO \*apuntador, int N)

{

int respuesta;

for (int A = 0; A < 10; A++)

{

printf ("Modelo: \t");

scanf ("%d", &respuesta);

if (respuesta == 0)

break;

apuntador[A].modelo = respuesta;

printf ("Cantidad: \t");

scanf ("%d", &respuesta);

apuntador[A].cantidad = respuesta;

if (A + 1 == 10)

printf ("Talonario de Pedidos agotado");

}

};

// principal

int

main ()

{

struct MODELO modelos[100];

struct TERMINADO terminados[20];

struct PIEZA piezas[100];

struct PEDIDO pedidos[10];

// simular datos

printf ("\nMODELOS (modelo, pieza, cantidad de piezas que usa)\n");

cargarModelos (modelos, 100);

printf ("\nTERMINADOS (modelo, stock)\n");

cargarTerminados (terminados, 20);

printf ("\nPIEZAS (pieza, stock)\n");

cargarPiezas (piezas, 100);

// cargar datos manualmente

printf ("\nPEDIDOS (modelo, cantidad)\n");

cargarPedidos (pedidos, 10);

//reportes

printf ("\nREPORTE DE A FABRICAR\n");

int size = sizeof pedidos / sizeof pedidos[0];

for (int A = 0; A < size; A++)

{

for (int B = 0; B < 20; B++)

{

if (pedidos[A].modelo == terminados[B].modelo)

{

printf ("Modelo: %d ", terminados[B].modelo);

printf ("Stock: %d ", terminados[B].stock);

printf ("Pedidos: %d ", pedidos[A].cantidad);

printf ("Stock Final: %d ",

terminados[B].stock - pedidos[A].cantidad);

if (terminados[B].stock - pedidos[A].cantidad < 0)

{

printf ("A fabricar: %d\n",

-1 \* (terminados[B].stock - pedidos[A].cantidad));

}

else

{

printf ("A fabricar: %d\n", 0);

}

}

}

}

return 0;

};