# HOJA DE TRABAJO #2

**Ejercicio # 1:**  
Se dispone de un archivo STOCK correspondiente a la existencia de artículos de un almacén y se desea señalar aquellos artículos cuyo nivel está por debajo del mínimo y que visualicen un mensaje “hacer pedido”. Cada artículo contiene un registro con los siguientes campos: número del código del artículo, nivel mínimo, nivel actual, proveedor, precio.  
  
El programa debe tener un menu que se especifiquen las siguientes opciones:  
1) Agregar productos (Grabar datos a un archivo)  
2) Leer archivo y mostrar en pantalla que productos tiene nivel minimo.

#include <fstream>

#include <iostream>

#include <string>

#include <cstdlib>

#include <ctime>

#include <stdio.h>

#include <vector>

using namespace std;

//Declaración de la estructura correspondiente a productos

struct Productos{

string codArticulo;

int nivMinimo;

int nivActual;

string proveedor;

float precio;

};

//Creación de vector

std::vector<Productos> producto;

//Función para agregar productos

void agregarArticulo(){

Productos \_producto;

//Ingreso de código

cout << "Ingrese el código del producto: " << "\n";

cin >> \_producto.codArticulo;

//Ingreso del nivel mínimo

cout << "Ingrese el mínimo del producto: " << "\n";

cin >> \_producto.nivMinimo;

//Ingreso del nivel máximo

cout << "Ingrese la cantidad acutal del producto: " << "\n";

cin >> \_producto.nivActual;

//Ingreso del proveedor

cout << "Ingrese el proveedor del producto: " << "\n";

cin >> \_producto.proveedor;

//Ingreso del precio

cout << "Ingrese el precio del producto: " << "\n";

cin >> \_producto.precio;

//Grabar producto en archivo

ofstream archivoStock("STOCK.txt", std::ios\_base::app);

archivoStock << "\n" <<

\_producto.codArticulo << "\t" << "\t" << \_producto.nivMinimo << "\t"<< "\t" <<

\_producto.nivActual << "\t"<< "\t" << \_producto.proveedor << "\t" <<"\t" <<

\_producto.precio << endl;

archivoStock.close();

}

//Función que muestra la estructura de datos

void estructuraDatos(){

cout << "Se necesita realizar pedidos de los siguientes productos: " << endl;

cout << "---------------------------------------------" << endl;

cout << "CODIGO"<< "\t" << "MINIMO" << "\t" << "EXISTENCIA" << "\t" << "PROVEEDOR" << "\t" << "PRECIO" << endl;

cout << "---------------------------------------------" << endl;

}

//Función encargada de mostrar los productos de los que se deben realizar pedidos

void pedidoProductos(){

for (int i; i <= producto.size(); i++){

/\*

Validación que se realiza para saber cuál producto está por debajo del nivel mínimo para

realizar el pedido

\*/

if (producto[i].nivMinimo >= producto[i].nivActual){

//Se muestra la información del producto correspondiente

cout <<

producto[i].codArticulo << "\t" <<

producto[i].nivMinimo << "\t" <<

producto[i].nivActual << "\t" << "\t" <<

producto[i].proveedor << "\t" << "\t" <<

producto[i].precio << "\t" <<

endl;

}

}

string empty;

cin >> empty;

return;

}

//Función para mostrar datos de los productos

void mostrarProductos(){

//Muestra la estructura de los datos

estructuraDatos();

//Invocación de la función correspondiente para mostrar los datos del producto

pedidoProductos();

}

//Función encargada de leer archivos

void leerArchivo(){

//Declaración de variable de tipo archivo.

ifstream archivoStock;

//Declaración de variable de tipo estructura de Productos

Productos stProductos;

archivoStock.open("STOCK.txt", ios::in);

//Obtener los datos desde el archivo.

while(archivoStock >> stProductos.codArticulo >> stProductos.nivMinimo >> stProductos.nivActual >> stProductos.proveedor >> stProductos.precio){

//Guardar los datos obtenidos en un vector

producto.push\_back(stProductos);

}

//Cerrar archivo.

archivoStock.close();

//Invocación de la función para mostrar los productos

mostrarProductos();

return;

}

// Principal

int main(){

char opcion;

string vacio;

bool terminar = false;

do {

//system("clear");

cin.clear();

cout << "---------------------PRODUCTOS---------------------" << endl;

cout << "1. Agregar Productos" << "\n"

<< "2. Leer archivo y mostrar productos para pedido" << "\n"

<< "3. Salir" << endl;

cout << "Elija una opción (1-3)" << endl;

cin >> opcion;

switch(opcion){

case '1':

agregarArticulo();

break;

case '2':

leerArchivo();

break;

case '3':

terminar = true;

break;

default:

cout << "Opción inválida";

getchar();

getline(cin, vacio);

break;

}

}while(terminar != true);

return 0;

}

**Ejercicio # 2:**  
El director de un colegio desea realizar un programa que procese un archivo de registros correspondiente a los diferentes alumnos del centro a fin de obtener los siguientes datos:

-Nota más alta y número de identificación del alumno correspondiente.  
-Nota media del colegio.

Datos de Estudiantes:

-Identificación  
-Nombre  
-Nota

#include <fstream>

#include <iostream>

#include <string>

#include <cstdlib>

#include <ctime>

#include <stdio.h>

#include <vector>

using namespace std;

//Declaración de la estructura correspondiente a estudiantes

struct Estudiantes{

int identificacion;

string nombre;

float nota;

};

//Creación de vector de tipo Estudiante

std::vector<Estudiantes> estudiante;

//Función para agregar estudiantes

void agregarEstudiantes(){

Estudiantes \_estudiante;

//Ingreso de identificación

cout << "Ingrese la identificación del estudiante: " << "\n";

cin >> \_estudiante.identificacion;

//Ingreso del nombre del estudiante

cout << "Ingrese el nombre del estudiante: " << "\n";

cin >> \_estudiante.nombre;

//Ingreso de la nota del estudiante

cout << "Ingrese nota del estudiante: " << "\n";

cin >> \_estudiante.nota;

//Grabar Estudiante en archivo

ofstream archivoEstudiantes("ESTUDIANTES.txt", std::ios\_base::app);

archivoEstudiantes <<

\_estudiante.identificacion << "\t"<< \_estudiante.nombre << "\t" << \_estudiante.nota

<< endl;

archivoEstudiantes.close();

}

//Función encargada de extraer la nota más alta y el alumno correspondiente

void notaAlta(float \_Nota){

int \_idEstudiante;

for (int i; i <= estudiante.size(); i++){

if (estudiante[i].nota > \_Nota){

\_Nota = estudiante[i].nota;

\_idEstudiante = estudiante[i].identificacion;

}

}

cout << "---------------------------------------------------" << endl;

cout << "La identificación del alumno con la nota más alta es: " << \_idEstudiante << " con una nota de: " << \_Nota << endl;

cout << "---------------------------------------------------" << endl;

return;

}

//Función encargada de mostrar la nota promedio

void notaPromedio(){

float sumaNotas;

float promedioNota;

int noEstudiantes;

for (int i; i <= estudiante.size(); i++){

sumaNotas = sumaNotas + estudiante[i].nota;

noEstudiantes = i;

}

promedioNota = sumaNotas/noEstudiantes;

cout << "---------------------------------------------------" << endl;

cout << "La nota media del colegio es de: " << promedioNota << endl;

cout << "---------------------------------------------------" << endl;

return;

}

//Función encargada de leer archivos

void leerArchivo(){

//Declaración de variable de tipo archivo.

ifstream archivoAlumnos;

//Declaración de variable de tipo estructura de Productos

Estudiantes stEstudiantes;

//Declaración de variable Nota

float nota = 0;

//Abrir archivo Estudiantes

archivoAlumnos.open("ESTUDIANTES.txt", ios::in);

//Obtener los datos desde el archivo.

while(archivoAlumnos >> stEstudiantes.identificacion >> stEstudiantes.nombre >> stEstudiantes.nota){

//Guardar los datos obtenidos en un vector

estudiante.push\_back(stEstudiantes);

}

//Cerrar archivo.

archivoAlumnos.close();

//Invocación de la función para mostrar los productos

notaAlta(nota);

notaPromedio();

return;

}

// Principal

int main(){

char opcion;

string vacio;

bool terminar = false;

do {

cin.clear();

cout << "---------------------ESTUDIANTES---------------------" << endl;

cout << "1. Agregar Estudiantes" << "\n"

<< "2. Mostrar nota más alta y nota media del colegio" << "\n"

<< "3. Salir" << endl;

cout << "Elija una opción (1-3)" << endl;

cin >> opcion;

switch(opcion){

case '1':

agregarEstudiantes();

break;

case '2':

leerArchivo();

break;

case '3':

terminar = true;

break;

default:

cout << "Opción inválida";

getchar();

getline(cin, vacio);

break;

}

}while(terminar != true);

return 0;

}