Consigna

**Para la situación descripta a continuación especifique, diseñe e implemente un programa que la resuelva:**

Una empresa dedicada a la venta y reparto de viandas de almuerzo necesita un sistema de información que le permita gestionar su negocio de forma más eficaz.

Todos los días por la mañana bien temprano el grupo de cocina carga en el sistema las variedades de comidas que se ofrecen ese día y las cantidades que han preparado por la noche de cada uno de los platos. Cada plato ofrecido se identifica por su nombre, que es único dentro del menú diario.

A media mañana, el encargado de caja asigna los precios que correspondan de cada uno de los platos ingresados según los costos de sus ingredientes, la complejidad de su cocción, etc.

Como la empresa sólo vende viandas a clientes previamente registrados, quien quiera realizar un pedido de viandas debe darse de alta en el sistema ingresando su DNI, nombre, apellido, dirección y teléfono de contacto.

Cuando un cliente realiza un pedido se registran los platos solicitados del menú del día, junto con la cantidad deseada de cada uno de ellos dentro de lo que se encuentre disponible. Además, se registra en el pedido el monto total del mismo y se agrega, discriminado aparte, un 10% extra para el reparto.

En cualquier momento del día, el grupo de cocina puede modificar las comidas ofrecidas para agregar cantidades, o quitar alguna vianda que se haya ensuciado, contaminado o caído, siempre sin afectar los pedidos que ya se realizaron.

El programa a construir debe ser capaz de mantener la información descripta anteriormente, como así también debe responder a los siguientes listados: los pedidos hechos en un día, con los precios abonados y las direcciones de entrega de cada uno; y por otro lado, todos los pedidos realizados por un cliente en particular con el detalle del cliente y dirección de entrega, junto con los platos pedido, cantidades y precios individuales, y al final el total del pedido y el costo, de cada envío.

ESPECIFICACIÓN

Cargar viandas con cantidades + cargar precio individual. / Modificar cantidades

Alta, baja y modificacion de cliente.

Alta, baja y modificacion de pedido.

Listado de pedidos hechos en un dia

Listado de pedidos por cliente

DISEÑO

VIANDA

nombre: cadena

cantidad: entero

precioIndividual: entero

CLIENTE

dni: entero

nombre: cadena

apellido: cadena

direccion: cadena

telefono: cadena

PEDIDO

numeroDePedido: entero    //Autogenerado

dniCliente (cliente): entero

fechaPedido: tFecha

platos: LO (clave: tipoClaveVianda / info: cantidad: real)

montoTotal: real

ESTRUCTURAS DE DATOS

Vianda   ADT LO

tipoClaveVianda = cadena    // Nombre

tipoInfoVianda = reg

cantidad: entero

precioIndividual: entero

fin reg

Cliente  ADT ABB

tipoClaveCliente = entero        //dni

tipoInfoCliente = reg

nombre: cadena

apellido: cadena

direccion: cadena

telefono: cadena

fin reg

Pedido  ADT ABB

tipoClavePedido = entero     //numeroDePedido

tipoInfoPedido = reg

dniCliente: tipoClaveCliente

fechaPedido: tFecha

platos: tipoListaPlato

montoTotal: real

fin reg

Plato ADT LO

tipoClavePlato: cadena // Nombre

tipoInfoPlato: reg

cantidad: entero // cantidad

precioIndividual: real // Precio

fin reg

Nivel 0

Const

Var

  viandas: tipoListaViandas

  clientes: tipoArbolClientes

  pedidos: tipoArbolPedidos

  identificador: tipoClavePedido

  resp: entero

Inicio

  identificador = 0

  crearEstructuras(clientes, viandas, pedidos)                //N1-

  Repetir

    resp = menuPrincipal()                                    //N1-

    Segun resp Hacer:

      1: administrarViandas(viandas)                          //N1-

      2: abmCliente(clientes)                                 //N1-

      3: abmPedido(pedidos, identificador, clientes, viandas) //N1-

      4: listados(clientes, pedidos)                          //N1-

    Fin Segun

  Hasta que resp = 0

Fin

Nivel 1

*/\**

*¿Que hace?: Crea e inicializa las estructuras de datos*

*Pre-condiciones:*

*Pos-condiciones: clientes = C’, viandas= V’, pedidos= P’, y C’ esta creada, V’ esta creada, P’ esta creada*

*Excepciones:*

*\*/*

procedimiento **crearEstructuras**(S clientes: tipoArbolClientes, S viandas: tipoListaViandas, S pedidos: tipoArbolPedidos)

Inicio

  crear(clientes) //ADT ABB

  crear(viandas)  //ADT LO

  crear(pedidos)  //ADT ABB

Fin crearEstructuras

*/\**

*¿Que hace?: Muestra el menu principal y devuelve el número de opción seleccionada*

*Pre-condiciones:*

*Post-condiciones: menuPrincipal = N y N es la opción elegida dentro de las opciones mostradas*

*Excepciones:*

*\*/*

funcion **menuPrincipal**(): entero

Inicio

  mostrar(“Menu Principal”)

  mostrar(“1: Modulo Clientes”)

  mostrar(“2: Modulo Viandas”)

  mostrar(“3: Modulo Pedidos”)

  mostrar(“4: Listados”)

  mostrar(“0: Salir”)

  menuPrincipal = enteroEnRango(“Ingrese una opción:”, 0, 4) //U

Fin menuPrincipal

*/\**

*¿Que hace?: Muestra las opciones del modulo de viandas*

*Pre-condiciones: viandas = V*

*Post-condiciones: viandas = V’ y V’ es la lista con las acciones que se hayan realizado*

*Excepciones:*

*\*/*

procedimiento **administrarViandas**(E/S viandas: tipoListaViandas)

Var

  resp: entero

Inicio

  Repetir

    resp = menuViandas()              //N2-

    Segun resp Hacer:

      1: cargarViandas(viandas)       //N2-

      2: cargarPrecio(viandas)        //N2-

      3: menuModificarVianda(viandas) //N2-

      4: eliminarViandas(viandas)     //N2-

    Fin Segun

  Hasta que resp = 0

Fin administrarViandas

*/\**

*¿Que hace?: Muestra las opciones del modulo del cliente*

*Pre-condiciones: clientes = C, pedidos = P*

*Post-condiciones: clientes = C’ y pedidos = P y C’ = C con las acciones que se hayan realizado*

*Excepciones:*

*\*/*

procedimiento **abmCliente**(E/S clientes: tipoArbolClientes, E pedidos: tipoArbolPedidos)

Var

  resp: entero

Inicio

  Repetir

    resp = menuABMClientes()            //N2

    Segun resp Hacer:

      1: altaCliente(clientes)          //N2

      2: modificarCliente(clientes)     //N2

      3: bajaCliente(clientes, pedidos) //N2

    Fin Segun

  Hasta que resp = 0

Fin abmCliente

*/\**

*¿Que hace?: Muestra las opciones del modulo de pedidos*

*Pre-condiciones: pedidos = P*

*Post-condiciones: pedidos = P’ y P’ es el arbol de pedidos con las modificaciones que se hayan realizado*

*Excepciones:*

*\*/*

procedimiento **abmPedido**(E/S pedidos: tipoArbolPedidos, E/S identificador: tipoClavePedido, E clientes: tipoArbolClientes, E viandas: tipoListaViandas)

Var

  resp: entero

Inicio

  Repetir

    resp = menuABMPedidos()                                     //N2-

    Segun resp Hacer:

      1: altaPedido(pedidos, identificador, clientes, viandas)  //N2-

      2: modificarPedido(pedidos, clientes, viandas)            //N2-

      3: bajaPedido(pedidos, clientes)                          //N2-

    Fin Segun

  Hasta que resp = 0

Fin abmPedido

*/\**

*¿Que hace?: Muestra las opciones del modulo de listados*

*Pre-condiciones: pedidos = P, clientes = C*

*Post-condiciones:*

*Excepciones:*

*\*/*

procedimiento **listados**(E pedidos: tipoArbolPedidos; E clientes: tipoArbolClientes)

Var

  resp: entero

Inicio

  Repetir

    resp = menuListados()                                   //N2-

    Segun resp Hacer:

      1: listarPedidosDeDia(pedidos, clientes)              //N2-

      2: listarPedidosDeCliente(pedidos, clientes) //N2-

    Fin Segun

  Hasta que resp = 0

Fin listados

Nivel 2

// Viandas

*/\**

*¿Que hace?: muetra y solicita al usuario que elija entre operaciones de viandas*

*Pre-condiciones: --*

*Post-condiciones: menuViandas = M y M = 1 o M = 2 o M = 3 o M = 4 o M = 0*

*Excepciones: --*

*\*/*

funcion **menuViandas**(): entero

Inicio

  mostrar(‘Menu Modulo Viandas’)

  mostrar(‘1: Cargar Viandas’)

  mostrar(‘2: Cargar Precios’)

  mostrar(‘3: Modificar Vianda’)

  mostrar(‘4: Eliminar Viandas’)

  mostrar(‘0: Volver’)

  menuCliente = enteroEnRango(“Ingrese una opción:”, 0, 4)  //U

Fin menuViandas

*/\**

*¿Que hace?: realiza el alta de una/s vianda/s*

*Pre-condiciones: viandas = V*

*Post-condiciones: viandas = V’ y V’ es la lista de viandas con las acciones realizadas*

*Excepciones: --*

*\*/*

procedimiento **cargarVianda**(E/S viandas: tipoListaViandas)

Var

  nombreVianda: tipoClaveVianda

  cantidad: entero

Inicio

  Repetir

    nombreVianda = obtenerNombreVianda()                //N3-

    Si existeVianda(viandas, nombreVianda) Entonces     //N3-

      mostrar(“Esa vianda ya esta cargada”)

    Sino

      cantidad = obtenerCantidad()                      //N3-

      cargarViandaN(viandas, nombreVianda, cantidad)    //N3-

    Fin Si

  Hasta que No confirma(“¿Desea ingresar otra vianda?”) //U

Fin cargarVianda

*/\**

*¿Que hace?: Carga el precio de una vianda*

*Pre-condiciones: viandas = V*

*Post-condiciones: viandas = V’ y V’ es la lista de viandas con los precios cargados*

*Excepciones:*

*\*/*

procedimiento **cargarPrecio**(E/S viandas: tipoListaViandas)

Var

  nombreVianda: tipoClaveVianda

  precio: entero

Inicio

  Repetir

    nombreVianda = obtenerNombreVianda()                          //N3-

    Si existeVianda(viandas, nombreVianda) Entonces               //N3-

      precio  = obtenerPrecio()                                   //N3-

      cargarPrecioN(viandas, nombreVianda, precio)                 //N3-

    Sino

      mostrar(“Esa vianda no existe”)

    Fin Si

  Hasta que No confirma(“¿Desea ingresar otra precio de vianda?”) //U

Fin cargarPrecio

*/\**

*¿Que hace?: Modifica una vianda*

*Pre-condiciones: viandas = V*

*Post-condiciones: viandas = V’ y V’ es la lista de viandas con las modificaciones realizadas*

*Excepciones: --*

*\*/*

function **menuModificarVianda**(E/S viandas: tipoListaViandas): entero

Var

resp: entero

Inicio

  Repetir

    resp = menuModificarViandas()       //N3-

    Segun resp Hacer:

      1: agregarCantidadVianda(viandas) //N3-

      2: quitarCantidadViandas(viandas) //N3-

    Fin Segun

  Hasta que resp = 0

Fin menuModificarVianda

*/\**

*¿Que hace?: Elimina todas las viandas*

*Pre-condiciones: viandas = V*

*Post-condiciones: viandas = V’ y V’ es la lista de viandas vaciada*

*Excepciones:*

*\*/*

procedimiento **eliminarViandas**(E/S viandas: tipoListaViandas)

Var

Inicio

  Si confirma((“¿Desea eliminar TODAS las viandas?”) Entonces //U

    vaciar(viandas)                                           //ADT LO

  Fin Si

Fin eliminarViandas

// Clientes

*/\**

*¿Que hace?: muestra y solicita al usuario que elija entre operaciones de clientes*

*Pre-condiciones: --*

*Post-condiciones: menuABMClientes = M y M = 1 o M = 2 o M = 3 o M = ‘SALIR’*

*Excepciones: --*

*\*/*

funcion **menuABMClientes**(): entero

Inicio

  mostrar(‘Menu Modulo Clientes‘)

  mostrar(‘1: Alta Cliente’)

  mostrar(‘2: Modificar Cliente’)

  mostrar(‘3: Bajar Cliente’)

  mostrar(‘0: Volver’)

  menuABMClientes = enteroEnRango(“Ingrese una opción:”, 0, 3)  //U

Fin menuABMClientes

*/\**

*¿Que hace?: Realiza el alta de un cliente*

*Pre-condiciones: clientes = C*

*Post-condiciones: clientes = C’ y C’ = C con las altas que se hayan realizado*

*Excepciones:*

*\*/*

procedimiento **altaCliente**(E/S clientes: tipoArbolClientes)

  dni: tipoClaveCliente

  info: tipoInfoCliente

Inicio

  Repetir

    dni = obtenerDNI()                                  //N3

    Si existeCliente(clientes, dni) Entonces            //N3

      mostrar(“El DNI del cliente ya se encuentra cargado)

    Sino

      obtenerInfoCliente(info)                          //N3

      guardarCliente(clientes, dni, info)               //N3

    Fin Si

  Hasta que No confirma(“Desea ingresar otro cliente?”) //U

Fin altaCliente

*/\**

*¿Que hace?: Realiza la modificacion de un cliente*

*Pre-condiciones: clientes = C*

*Post-condiciones: clientes = C’ y C’ = C con las modificaciones que se hayan realizado*

*Excepciones:*

*\*/*

procedimiento **modificarCliente**(E/S clientes: tipoArbolClientes)

  dni: tipoClaveCliente

  info: tipoInfoCliente

Inicio

  Repetir

    dni = obtenerDNI()                                    //N3

    Si No existeCliente(clientes, dni) Entonces           //N3

      mostrar(“El DNI del cliente no se encuentra cargado)

    Sino

      obtenerInfoCliente(info)                            //N3

      modificarCliente(clientes, dni, info)               //N3

    Fin Si

  Hasta que No confirma(“Desea modificar otro cliente?”)  //U

Fin modificarCliente

*/\**

*¿Que hace?: Realiza la baja de un cliente*

*Pre-condiciones: clientes = C*

*Post-condiciones: clientes = C’ y C’ = C con las bajas que se hayan realizado*

*Excepciones:*

*\*/*

procedimiento **bajaCliente**(E/S clientes: tipoArbolClientes, E pedidos: tipoArbolPedidos)

  dni: tipoClaveCliente

  info: tipoInfoCliente

Inicio

  Repetir

    dni = obtenerDNI()                                      //N3

    Si No existeCliente(clientes, dni) Entonces             //N3

      mostrar(“El DNI del cliente no se encuentra cargado)

    Sino

      Si clienteTienePedidos(pedidos, dni) Entonces         //N3

        mostrar(“El cliente tiene pedidos cargados, no puede darlo de baja”)

      Sino

        eliminarCliente(clientes, dni)                      //N3

      Fin Si

    Fin Si

  Hasta que No confirma(“Desea dar de baja otro cliente?”)  //U

Fin bajaCliente

//Pedidos

*/\**

*¿Que hace?: muetra y solicita al usuario que elija entre operaciones de pedidos*

*Pre-condiciones:*

*Post-condiciones: menuPedidos = M y M = 1 o M = 2 o M = 3 o M = 0*

*Excepciones:*

*\*/*

funcion **menuABMPedidos**(): entero

Inicio

  mostrar(‘Menu Modulo Pedidos)

  mostrar(‘1: Cargar Pedido)

  mostrar(‘2: Modificar Pedido’)

  mostrar(‘3: Eliminar Pedido’)

  mostrar(‘0: Volver’)

  menuABMPedidos = enteroEnRango(“Ingrese una opción:”, 0, 3) //U

Fin menuABMPedidos

*/\**

*¿Que hace?: realiza el alta de uno o mas pedido/s*

*Pre-condiciones: pedidos = P*

*Post-condiciones: pedidos = P’ y P’ es el arbol de pedidos con las acciones realizadas*

*Excepciones:*

*\*/*

procedimiento **altaPedido**(E/S pedidos: tipoArbolPedidos, E/S identificador: tipoClavePedido, E clientes: tipoArbolClientes, E viandas: tipoListaViandas)

Var

Inicio

  Repetir

    crearPedido()                                       //N3

    obtenerInfoPedido(info, clientes, viandas)          //N3-

    guardarPedido(pedidos, identificador, info)         //N3-

    identificador = identificador + 1

  Hasta que No confirma(“¿Desea ingresar otro pedido?”) //U

Fin cargarPedido

*/\**

*¿Que hace?: Realiza la modificacion de un pedido*

*Pre-condiciones: pedidos = P*

*Post-condiciones: pedidos = P’ y P’ = P con las modificaciones que se hayan realizado*

*Excepciones:*

*\*/*

procedimiento **modificarPedido**(E/S pedidos: tipoArbolPedidos, E clientes: tipoArbolClientes, E viandas: tipoListaViandas)

  identificador: tipoClavePedido

  info: tipoInfoPedido

Inicio

  Repetir

    identificador = obtenerIdentificador()                  //N3-

    Si No existePedido(pedidos, identificador) Entonces     //N3-

      mostrar(“No se ha encontrado un pedido con el identificador dado")

    Sino

      buscar(pedidos, identificador, info)

      obtenerInfoPedido(info, clientes, viandas)            //N3-

    Fin Si

  Hasta que No confirma(“Desea modificar otro pedido?”)     //U

Fin modificarPedido

*/\**

*¿Que hace?: Realiza la baja de uno o mas pedidos*

*Pre-condiciones: pedidos = P*

*Post-condiciones: pedidos = P’ y P’ = P con las bajas que se hayan realizado*

*Excepciones:*

*\*/*

procedimiento **bajaPedido**(E/S pedidos: tipoArbolPedidos)

  identificador: tipoClavePedido

Inicio

  Repetir

    identificador = obtenerIdentificador()                //N3-

    Si No existePedido(pedidos, identificador) Entonces   //N3-

      mostrar(“No se ha encontrado un pedido con el identificador dado")

    Sino

      eliminarPedido(pedidos, identificador)              //N3-

    Fin Si

  Hasta que No confirma(“Desea dar de baja otro pedido?”) //U

Fin bajaPedido

// Listados

*/\**

*¿Que hace?: muestra y solicita al usuario que elija entre operaciones de listados*

*Pre-condiciones: --*

*Post-condiciones: menuListados = M y M = 1 o M = 2 o M = 0*

*Excepciones: --*

*\*/*

funcion **menuListados**(): entero

Inicio

  mostrar(‘Menu Modulo Listados’)

  mostrar(‘1: Listar Pedidos en un dia’)

  mostrar(‘2: Listar Pedidos de un cliente’)

  mostrar(‘0: Volver’)

  menuCliente = enteroEnRango(“Ingrese una opción:”, 0, 2)  //U

Fin menuListados

*/\**

*¿Que hace?: Realiza el listado de todos los pedidos en cierto dia*

*Pre-condiciones: pedidos = P*

*Post-condiciones:*

*Excepciones: --*

*\*/*

procedimiento **listarPedidosDeDia**(E pedidos: tipoArbolPedidos, E clientes: tipoArbolClietnes)

Var

colaPedidos: tipoColaRecorridos

fecha: tFecha

numeroDePedido: tipoClavePedidos

Inicio

    mostrarCabeceraListadosDia()                                    //N3-

    Si esVacio(pedidos) Entonces                                    //ADT ABB

      mostrar(“No hay pedidoos para mostrar”)

    Sino

      solicitarFecha(fecha)                                         //N3-

      inOrder(pedidos, colaPedidos)                                 //ADT ABB

      Mientrs No esVacia(colaPedidos) Hacer                         //ADT COLA

        frente(colaPedidos, numeroDePedidio)                        //ADT COLA

        mostrarPedidoDia(pedidos, numeroDePedido, fecha, clientes)  //N3-

        desencolar(colaPedidos)                                     //ADT COLA

      Fin Mientras

    Fin Si

  Excepciones

    cuando errorEnCola => continua(“Ocurrio un error, intente mas tarde”)

Fin listarPedidos

*/\**

*¿Que hace?: Realiza el listado de todos los pedidos de cierto cliente*

*Pre-condiciones: pedidos = P*

*Post-condiciones:*

*Excepciones: --*

*\*/*

procedimiento **listarPedidosDeCliente**(E pedidos: tipoArbolPedidos, E clientes: tipoArbolClientes)

Var

dniCliente: entero

colaPedidos: tipoColaRecorridos

numeroDePedido: entero

dirección: cadena

Inicio

    mostrarCabeceraListadoPedidosCliente()                                            //N3-

    Si esVacio(pedidos) Entonces                                                      //ADT ABB

      mostrar(“No hay pedidoos para mostrar”)

    Sino

      dniCliente = numeroEntero(“Ingrese el dni del cliente deseado”)                 //U

      direccion = obtenerDireccion(dniCliente, clientes)                              //N3-

      inOrder(pedidos, colaPedidos)                                                   //ADT ABB

      Mientras No esVacia(colaPedidos) Hacer                                          //ADT COLA

        frente(colaPedidos, numeroDePedido)                                           //ADT COLA

        mostrarPedidoCliente(pedidos, numeroDePedido, dniCliente, direccion) //N3-

        desencolar(colaPedidos)                                                       //ADT COLA

      Fin Mientras

    Fin Si

  Excepciones

  cuando errorEnCola => continua(“Ocurrio un error, intente mas tarde”)

Fin listarPedidosCliente

Nivel 3

// Viandas

*/\**

*¿Que hace?: Obtiene un nombre de vianda*

*Pre-condiciones:*

*Post-condiciones: obtenerNombreVianda = N y N es el nombre de la vianda ingresado*

*Excepciones: --*

*\*/*

funcion **obtenerNombreVianda**(): cadena

Var

Inicio

  obtenerNombreVianda = textoNoVacio()  //U

Fin

*/\**

*¿Que hace?: Busca una vianda y devuelve verdadero si esta en la lista, sino devuelve falso*

*Pre-condiciones: viandas = V, nombreVianda = N*

*Post-condiciones: existeVianda = E y E = Verdadero o E = Falso*

*Excepciones: --*

*\*/*

function **existeVianda**(E viandas: tipoListaViandas, E nombreVianda: tipoClaveViandas)

Var

  info: tipoInfoViandas

Inicio

  inicio

    recuClave(viandas, nombreVianda, info)  //ADT LO

    existeVianda = Verdadero

  Excepcion

      cuando claveNoExiste => existeVianda = Falso

  Fin

Fin existeVianda

*/\**

*¿Que hace?: Obtiene la cantidad de viandas*

*Pre-condiciones:*

*Post-condiciones: obtenerCantidad = O y O es la cantidad de viandas ingresadas*

*Excepciones: --*

*\*/*

funcion **obtenerCantidad**(): entero

Var

num: entero

Inicio

  Repetir

    num = numeroEnt()  //U

  Hasta que num > 0

  obtenerCantidad = num

Fin obtenerCantidad

*/\**

*¿Que hace?: Carga una vianda*

*Pre-condiciones: viandas = V, nombreVianda = N, cantidad = C*

*Post-condiciones: viandas = V’ y V’ es la lista de viandas con las acciones realizadas.*

*Excepciones: --*

*\*/*

procedimiento **cargarViandaN**(E/S viandas: tipoListaViandas, E nombreVianda: tipoClaveVianda, E cantidad: entero)

Const

  PrecioDef: -1

Var

  info: tipoInfoVianda

Inicio

  info.cantidad = cantidad

  info.precioIndividual = PrecioDef

  Inicio

    insertar(viandas, nombreVianda, info) //ADT LO

  Excepcion

    cuando claveExiste =>

      mostrar(“Esa vianda ya esta cargada”)

    cuando listaLlena =>

      mostrar(“Ocurrio un error inesperado, intente mas tarde”)

  Fin

Fin cargarVianda

*/\**

*¿Que hace?: Pide el precio de una vianda*

*Pre-condiciones:*

*Post-condiciones: obtenerPrecio = P y P es el precio cargado*

*Excepciones: --*

*\*/*

funcion **obtenerPrecio**(): real

Var

  num: real

Inicio

  Repetir

    num = numeroReal()  //U

  Hasta que num > 0

  obtenerPrecio = num

Fin obtenerPrecio

*/\**

*¿Que hace?: Carga el precio de una vianda*

*Pre-condiciones: viandas = V, nombreVianda = N, precio = P*

*Post-condiciones: viandas = V’ y V’ es la lista de viandas con las acciones realizadas*

*Excepciones: --*

*\*/*

procedimiento **cargarPrecioN**(E/S viandas: tipoListaViandas, E nombreVianda:tipoClaveVianda, E precio: real)

Var

  info: tipoInfoVianda

Inicio

  recuClave(viandas, nombreVianda, info)      //ADT LO

  si info.precioIndividual != -1 entonces

    mostrar(“El precio de esta vianda ya esta cargado”)

  Sino

    Inicio

      info.precioIndividual = precio

      modificar(viandas, nombreVianda, info)  //ADT LO

    Excepcion

      cuando listaLlena =>

        mostrar(“Ocurrio un error inesperado, intente mas tarde”)

    Fin

  Fin Si

Fin cargarPrecioN

*/\**

*¿Que hace?: Muestra el menu correspondiente a la modificacion de viandas*

*Pre-condiciones: --*

*Post-condiciones: menuModificarViandas = N y N=1, N=2, N=3 o N=0*

*Excepciones: --*

*\*/*

funcion **menuModificarViandas**() : entero

Var

Inicio

  mostrar(‘Menu Modificacion de Viandas’)

  mostrar(‘1: Añadir cantidad Viandas’)

  mostrar(‘2: Eliminar cantidad de Viandas’)

  mostrar(‘0: Volver’)

  menuModificarViandas = enteroEnRango(“Ingrese una opción:”, 0, 4)  //U

Fin menuModificarViandas

*/\**

*¿Que hace?: Agrega una cantidad especifica a una vianda*

*Pre-condiciones: viandas = V*

*Post-condiciones: viandas = V’ y V’ es la lista de viandas con las modificaciones realizadas*

*Excepciones: --*

*\*/*

procedimiento **agregarCantidadVianda**(E/S viandas: tipoListaViandas)

Var

nombreVianda = tipoClaveVianda

nuevaInfo = tipoInfoVianda

Inicio

  Repetir

    nombreVianda = obtenerNombreVianda()                    //N3-

    Si existeVianda(viandas, nombreVianda) entonces         //N3-

      agregarCantidadInfo(nuevaInfon nombreVianda, viandas) //N4-

    Sino

      mostrar(“Esa vianda no esta cargada”)

    Fin Si

  Hasta que No confirma(“¿Desea ingresar otra vianda?”)     //U

Fin agregarCantidadViandas

*/\**

*¿Que hace?: Quita una cantidad especifica a una vianda*

*Pre-condiciones: viandas = V*

*Post-condiciones: viandas = V’ y V’ es la lista de viandas con las modificaciones realizadas*

*Excepciones: --*

*\*/*

procedimiento **quitarCantidadViandas**(E/S viandas: tipoListaViandas)

Var

  nombreVianda: tipoClaveVianda

  nuevaInfo: tipoInfoVianda

Inicio

repetir

    nombreVianda = obtenerNombreVianda()                    //N3-

    Si existeVianda(viandas, nombreVianda) Entonces         //N3-

      quitarCantidadInfo(nuevaInfo, nombreVianda, viandas)  //N4-

      modificar(viandas, nombreVianda, nuevaInfo)           //ADT LO

    Sino

      mostrar(“Esa vianda no esta cargada”)

    Fin Si

  Hasta que No confirma(“¿Desea ingresar otra vianda?”)     //U

Fin quitarCantidadViandas

// Clientes

*/\**

*¿Que hace?: Devuelve un dni válido*

*Pre-condiciones:*

*Post-condiciones: obtenerDNI = D ; D es un dni válido*

*Excepciones:*

*\*/*

funcion **obtenerDNI**() : tipoClaveCliente

  dni: tipoClaveCliente

Incio

  Repetir

    dni = numeroEnt(“Ingrese el DNI del cliente: ”) //U

  Hasta que dni > 0

  obtenerDNI = dni

Fin obtenerDNI

*/\**

*¿Que hace?: Determina si un cliente existe o no en un árbol de clientes*

*Pre-condiciones: clientes = C; dni = D*

*Post-condiciones: existeCliente = E; E = V o F según exista o no el cliente*

*Excepciones:*

*\*/*

funcion **existeCliente**(E clientes: tipoArbolClientes, E dni: tipoClaveCliente) : booleano

  info: tipoInfoCliente

Inicio

  Inicio

    buscar(clientes, dni, info) //ADT ABB

    existeCliente = V

  Excepciones

    cuando cualquiera => existeCliente = F

  Fin

Fin existeCliente

*/\**

*¿Que hace?: Permite al usuario completar los datos de un cliente*

*Pre-condiciones:*

*Post-condiciones: info = I ; I contiene datos válidos para un cliente*

*Excepciones:*

*\*/*

procedimiento **obtenerInfoCliente**(S info: tipoInfoCliente)

Inicio

  info.nombre = textoNoVacio(“Ingrese el nombre del cliente: ”)       //U

  info.apellido = textoNoVacio(“Ingrese el apellido del cliente: ”)   //U

  info.telefono = textoNoVacio(“Ingrese el teléfono del cliente: ”)   //U

  info.direccion = textoNoVacio(“Ingrese la dirección del cliente: ”) //U

Fin obtenerInfoCliente

*/\**

*¿Que hace?: Guarda un cliente*

*Pre-condiciones: clientes = C; dni = D; C puede estar vacío o tener clientes válidos*

*Post-condiciones: clientes = C’; C’ puede tener o no un cliente nuevo con respecto a C*

*Excepciones:*

*\*/*

procedimiento **guardarCliente**(e/s clientes: tipoArbolClientes, e dni: tipoClaveCliente, e info: tipoInfoCliente)

Inicio

Inicio

insertar(clientes, dni, info) //ADT ABB

Excepciones

cdo arbolLleno => mostrar(“Ocurrio un error inesperado, reintente mas tarde”)

Fin

Fin

*/\**

*Que Hace: modifica la info de un cliente*

*Pre-condiciones: clientes = C; dni = D; info = I; C puede estar vacio o tener clientes validos*

*Post-condiciones: clientes = C'; C' = C con el cliente con su modificacion en su info*

*Excepciones:*

*\*/*

procedimiento **modificarCliente**(E/S cliente: tipoArbolClientes, E dni: tipoClaveCliente, E info: tipoInfoCliente)

Inicio

  Inicio

    modificar(cliente, dni, info) //ADT ABB

  Excepciones

    cuando claveNoExiste => mostrar ("Ocurrio un error inesperado, reintente mas tarde")

  Fin

Fin modificarCliente

*/\**

*Que Hace: verifica si el cliente tiene cargado pedidos*

*Pre-condiciones: pedidos = E; dni = D; E*

*Post-condiciones:*

*Excepciones:*

*\*/*

funcion **clienteTienePedidos**(E pedidos : tipoArbolPedidos, E dni : tipoClaveCliente) : booleano

  q: tipoCola

  k: tipoInfoPedido

  encontrado: boolean

Inicio

  crear(q)                                  //ADT COLA

  posorder(pedidos, q)                      //ADT ABB

  Repetir

    frente(q, k)                            //ADT COLA

    Si k.dniCliente = dni entonces

      encontrado = V

    Fin Si

    desencolar(q)                           //ADT COLA

  Hasta que esVacia(q) = V o encontrado = V //ADT COLA

  Si encontrado = V entonces

    clienteTienePedidos = V

  Sino

    clienteTienePedidos = F

  Fin Si

Excepciones

  cuando errorEnCola, colaVacia => mostrar ("Ocurrio un problema, intente mas tarde")

Fin clienteTienePedidos

*/\**

*Que Hace: elimina el cliente*

*Pre-condiciones: cliente = C; dni = D;*

*Post-condiciones: clientes = C’, C’ es C sin el cliente que se desea eliminar o C si no se lo encuentra*

*Excepciones:*

*\*/*

procedimiento **eliminarCliente**(E/S clientes: tipoArbolClientes, E dni: tipoClaveCliente)

Inicio

  Inicio

    suprimir(clientes, dni) //ADT ABB

  Excepciones

    cuando claveNoExiste => mostrar ("Ocurrio un error, intente mas tarde")

Fin eliminarCliente

// Pedidos

*/\**

*¿Que hace?: Elimina un pedido*

*Pre-condiciones: pedidos = P; identificador = I;*

*Post-condiciones: pedidos = P’, P’ es P sin el elemento que se desea eliminar o P si no se encuentra*

*Excepciones:*

*\*/*

procedimiento **eliminarPedido**(E/S pedidos: tipoArbolPedidos, E identificador: tipoClavePedido)

Var

  info: tipoInfoPedido

Inicio

  Inicio

    buscar(pedidos, identificador, info)  //ADT ABB

    vaciar(info.platos)                   //ADT LO

    suprimir(clientes, dni)               //ADT ABB

  Excepciones

    cuando claveNoExiste => mostrar ("Ocurrio un error, intente mas tarde")

Fin eliminarPedido

*/\**

*¿Que hace?: Guarda un pedido*

*Pre-condiciones: pedidos = P; identificador = I; P puede estar vacío o tener pedidos válidos;*

*Post-condiciones: pedidos = P’; P’ puede tener o no un cliente nuevo con respecto a P*

*Excepciones:*

*\*/*

procedimiento **guardarPedido**(E/S pedidos: tipoArbolPedidos, E identificador: tipoClavePedido, E info: tipoInfoPedido)

Inicio

  Inicio

    insertar(pedidos, identificador, info)  //ADT ABB

  Excepciones

    cuando arbolLleno => mostrar(“Ocurrio un error inesperado, reintente mas tarde”)

  Fin

Fin guardarPedido

*/\**

*¿Que hace?: Determina si un pedido existe o no en un árbol de pedidos*

*Pre-condiciones: pedidos = P; identificador = I*

*Post-condiciones: existePedido = E; E = V o F según exista o no el pedido con numero de pedido I en P*

*Excepciones:*

*\*/*

funcion **existePedido**(E pedidos: tipoArbolPedidos, E identificador: tipoClavePedido) : boolean

  info: tipoInfoPedido

Inicio

  Inicio

    buscar(pedidos, identificador, info)  //ADT ABB

    existePedido = V

  Excepciones

    cuando cualquiera => existePedido = F

  Fin

Fin existePedido

*/\**

*¿Que hace?: Devuelve un identificador válido*

*Pre-condiciones:*

*Post-condiciones: obtenerIdentificador = I ; I es un identificador de pedido (entero)*

*Excepciones:*

*\*/*

funcion **obtenerIdentificador**() : tipoClavePedido

  identificador: tipoClavePedido

Incio

  Repetir

    identificador = numeroEnt(“Ingrese el numero de pedido: ”) //U

  Hasta que identificador > 0

  obtenerIdentificador = identificador

Fin obtenerIdentificador

*/\**

*¿Que hace?: Crea un registro de tipo tipoInfoPedido*

*Pre-condiciones:*

*Post-condiciones: info = I ; I es tipoInfoPedido, I.platos es tipoListaPlatos y I.fechaPedido es tFecha*

*Excepciones:*

*\*/*

procedimiento **crearPedido** (S info: tipoInfoPedido)

Inicio

  info = nuevo tipoInfoPedido

  info.dniCliente = 0

  crear(info.platos)                //ADT LO

  info.montoTotal = 0

  info.fechaPedido = nuevo tFecha

Fin crearPedido

*/\**

*¿Que hace?: Permite al usuario completar los datos de un pedido*

*Pre-condiciones:*

*Post-condiciones: info = I ; I contiene datos válidos para un pedido*

*Excepciones:*

*\*/*

procedimiento **obtenerInfoPedido**(E/S info: tipoInfoPedido, E clientes: tipoArbolClientes, E viandas: tipoListaViandas)

  existe: booleano = F

Inicio

  Repetir

    info.dniCliente = obtenerDNI()                    //N3-

    Si No existeCliente(info.dniCliente) Entonces     //N3-

      mostrar("No se ha encontrado un cliente registrado con este DNI")

    Sino

      existe = V

    Fin Si

  Hasta que existe

  mostrar("Fecha del pedido")

  solicitarFecha(info.fechaPedido)                    //N3-

  cargarPlatos(info.platos, viandas, info.montoTotal) //N4-

Fin obtenerInfoPedido

// Listados

*/\**

*¿Que hace?: Muestra la cabecera del listado de pedidos por dia*

*Pre-condiciones:*

*Post-condiciones:*

*Excepciones: --*

*\*/*

procedimiento **mostrarCabeceraListadosDia**()

Var

Inicio

  mostrar(“ID DIA PRECIO DIRECCION”)

Fin mostrarCabeceraListadosDia

*/\**

*¿Que hace?: Solicita y devuelve una fecha*

*Pre-condiciones:*

*Post-condiciones: fecha = F y F es la fecha ingresada*

*Excepciones: --*

*\*/*

procedimiento **solicitarfecha**(S fecha: tFecha)

Var

  fecha: tFecha

Inicio

  Repetir

    fecha.dia = enteroEnRango(“Ingrese el dia”, 0, 31)        //U

    fecha.mes = enteroEnRango(“Ingrese el mes”, 0, 12)        //U

    fecha.anio = enteroEnRango(“Ingrese el año”, 2000, 2020)  //U

Hasta que esFechaCorrecta(fecha)

Fin solicitarFecha

*/\**

*¿Que hace?: Muestra un pedido si es del dia correspondiente*

*Pre-condiciones: pedidos = P, numeroDePedido = N, fecha = F*

*Post-condiciones:*

*Excepciones: --*

*\*/*

procedimiento **mostrarPedidoDia**(E pedidos: tipoArbolPedidos, E numeroDePedido: tipoClavePedidos, E fechaDeseada: tFecha, E clientes: tipoArbolCliente)

Var

info: tipoInfoPedido

fechaPedido: tFecha

direccion: cadena

Inicio

buscar(pedidos, numeroPedido, info)                         //ADT ABB

fechaPedido = info.fechaPedido

Si esMismaFecha(fechaDeseada, fechaPedido) entonces         //N4-

direccion = obtenerDireccion(info.dniCliente, clientes)   //N4-

mostrar(numeroPedido, info.fecha, info.precio, direccion)

Fin Si

Fin mostrarPedidoDia

*/\**

*¿Que hace?: Muestra la cabecera del listado de pedidos por cliente*

*Pre-condiciones:*

*Post-condiciones:*

*Excepciones: --*

*\*/*

procedimiento **mostrarCabeceraListadoPedidosCliente**()

Var

Inicio

mostrar(“DNI Direccion Plato   cantidad     precio”)

Fin mostrarCabeceraListadoPedidosCliente

*/\**

*¿Que hace?: Muestra un pedido si corresponde a cierto cliente*

*Pre-condiciones:*

*Post-condiciones:*

*Excepciones: --*

*\*/*

procedimiento **mostrarPedidoCliente**(E pedidos: tipoArbolPedidos, E numeroDePedido: tipoClavePedido, E dniClienteReq: entero, E direccion: cadena)

Var

  dniClienteActual: entero

  platos: tipoListaPlatos

info: tipoInfoPedido

envio: real

Inicio

  buscar(pedidos, numeroPedido, info)                     //ADT ABB

  dniClienteActual = info.dniCliente

  Si esMismoDNI(dniClienteReq, dniClienteActual) Entonces //N4-

    platos = info.platos

    mostrar(dniClienteReq, direccion)

    mostrarPlatos(platos)                                 //N4-

    envio = obtenerEnvio(info.montoTotal)                 //N4-

    mostrar(“TOTAL: ” + info.montoTotal , “Envio: “ + envio)

  Fin Si

Fin mostrarPedidoCliente

Nivel 4

// Viandas

*/\**

*¿Que hace?: actualiza la cantidad de viandas*

*Pre-condiciones:*

*Post-condiciones:*

*Excepciones: --*

*\*/*

procedimiento **agregarCantidadInfo**(S nuevaInfo: tipoInfoViandas, E nombreVianda: tipoClaveVianda, E viandas: tipoListaViandas)

Var

cantidadAAgregar: entero

info: tipoInfoVianda

nuevaInfo: tipoInfoVianda;

cantidad: entero;

precio: real;

Inicio

cantidadAAgregar = numeroEnt(“Ingrese la cantidad de viandas a agregar”)  //U

recuClave(viandas, nombreVianda, info)                                    //ADT LO

cantidad = info.cantidad

precio = info.precioIndividual

nuevaInfo.precioIndividual = precio

nuevaInfo.cantidad = cantidadAAgregar + info.cantidad;

modificar(viandas, nombreVianda, nuevaInfo) // ADT LO

Fin agregarCantidadInfo

*/\**

*¿Que hace?: actualiza la cantidad de viandas*

*Pre-condiciones:*

*Post-condiciones:*

*Excepciones: --*

*\*/*

procedimiento **quitarCantidadInfo**(S nuevaInfo: tipoInfoVianda, E nombreVianda: tipoClaveVianda, E viandas: tipoListaViandas)

Var

cantidadAQuitar: entero

nuevaCantidad: entero

info: tipoInfoVianda

nuevaInfo: tipoInfoVianda

cantidad: entero

precio: real

Inicio

cantidadAQuitar= numeroEnt(“Ingrese la cantidad de viandas a quitar”) //U

recuClave(viandas, nombreVianda, info)                                //ADT LO

cantidad = info.cantidad

precio = info.precioIndividual

nuevaCantidad = cantidad - cantidadAQuitar

Si nuevaCantidad >= 0 entonces

nuevaInfo.precioIndividual = precio

nuevaInfo.cantidad = nuevaCantidad

modificar(viandas, nombreVianda, nuevaInfo) // ADT LO

Sino

mostrar(“No se puede quitar esa cantidad de viandas”)

Fin Si

Fin **quitarCantidadInfo**

*/\**

*¿Que hace?: devuelve verdadero si dos fechas son iguales, en caso contrario devuelve falso*

*Pre-condiciones: fechaDeseada = FD y fechaPedida = FP*

*Post-condiciones: esMismaFecha = E y E es verdadero o E es falso*

*Excepciones: --*

*\*/*

funcion **esMismaFecha**(E fechaDeseada: tFecha, E fechaPedido: tFecha): booleano

Var

Inicio

esMismaFecha  = (fechaDeseada.dia == fechaPedido.dia) y (fechaDeseada.mes == fechaPedido.mes) y (fechaDeseada.año == fechaPedido.año)

Fin esMismaFecha

*/\**

*¿Que hace?: obtiene la direccion de un cliente*

*Pre-condiciones: dniCliente=D, clientes = C*

*Post-condiciones: obtenerDireccion = O y O es la direccoion del cliente*

*Excepciones: --*

*\*/*

funcion **obtenerDireccion**(E dniCliente: tipoClaveCliente, E clientes: tipoArbolClientes): cadena

Var

info: tipoInfoCliente

Inicio

buscar(clientes, dniCliente, info)  //ADT ABB

obtenerDireccion = info.direccion

Fin obtenerDireccion

*/\**

*¿Que hace?: compara dos DNIs y devuelve verdadero si son iguales, en caso contrario devuelve dalso*

*Pre-condiciones: dniClienteReq = DREQ, dniClienteActual=DACT*

*Post-condiciones: esMismoDNI = E y E es verdadero o E es falso*

*Excepciones: --*

*\*/*

function **esMismoDNI**(E dniClienteReq: entero, E dniClienteActual: entero): booleano

Inicio

esMismoDNI = dniClienteReq == dniClienteActual

Fin esMismoDNI

// Pedidos

*/\**

*¿Que hace?: Permite al usuario llenar una lista de platos para un pedido*

*Pre-condiciones: platos = P, P es una lista de platos (puede estar vacia o contener platos validos)*

*Post-condiciones: platos = P' ; P' es P mas los platos agregados, y/o con cantidades cambiadas*

*Excepciones:*

*\*/*

procedimiento **cargarPlatos** (E/S platos: tipoListaPlatos, E viandas: tipoListaViandas, E/S total: real)

  vianda: tipoInfoVianda

  plato: tipoClavePlato

  info: tipoInfoPlato

  cantidad: entero

  existe: booleano

Inicio

  Repetir

    plato = textoNoVacio("Ingrese el nombre del plato que desea agregar al pedido: ") //U

    Si existeVianda(viandas, plato) Entonces                                          //N3

      vianda = recuClave(viandas, plato, vianda)                                      //ADT LO

      existe = existePlato(platos, plato)                                             //N5

      Si existe Entonces

        recuClave(platos, plato, info)                                                //ADT LO

        mostrar("Este plato ya se encuentra en el pedido. Unidades: " + info.cantidad)

      Sino

        info = nuevo tipoInfoPlato

        insertar(platos, plato, cantidad)                                             //ADT LO

      Fin Si

      mostrar("El precio por unidad es de: ", vianda.precioIndividual)

      Repetir

        cantidad = numeroEntero("Ingrese la cantidad de unidades que desea agregar (max: " + viandas.cantidad + "): ")  //U

        Si cantidad > vianda.cantidad Entonces

          mostrar("No hay suficientes viandas disponibles")

          cantidad = 0

        Fin Si

      Hasta que cantidad > 0

      info.precioIndividual = vianda.precioIndividual

      info.cantidad += cantidad

      total += cantidad \* vianda.precioIndividual

    Fin Si

  Hasta que No confirma("Desea añadir otro plato?")                                   //U

Fin cargarPlatos

// Listados

*/\**

*¿Que hace?: muestra todos los platos contenidos en unn pedido*

*Pre-condiciones: platos = P*

*Post-condiciones:*

*Excepciones: --*

*\*/*

procedimiento mostrarPlatos(E platos: tipoListaPlatos)

Var

long: entero

clave: tipoClavePlato

info: tipoInfoPlato

Inicio

long = longitud(platos) // ADT LO

recuPrim(platos, clave)

desde i = 1 hasta longitud

recuClave(platos, clave, info) // ADT LO

mostrar(clave + “ “ + info.cantidad + “ “ + info.precioIndividual)

recuSig(platos, clave, clave) // ADT LO

fin desde

fin mostrarPlatos

*/\**

*¿Que hace?: obtiene el costo del envio de un pedido*

*Pre-condiciones: montoTotal = M*

*Post-condiciones: obtenerEnvio = O y O es el precio del envio*

*Excepciones: --*

*\*/*

funcion obtenerEnvio(montoTotal): real

Var

Inicio

obtenerEnvio = montoTotal \* 0.1

fin obtenerEnvio

Nivel 5

*/\**

*¿Que hace?: Determina si un plato existe o no en una lista de platos*

*Pre-condiciones: platos = P; nombre = N*

*Post-condiciones: existePlato = E; E = V o F según exista o no el plato N en P*

*Excepciones:*

*\*/*

funcion **existePlato**(E platos: tipoListaPlatos, E nombre: tipoClavePlato) : boolean

  info: tipoInfoPlato

Inicio

  Inicio

    recuClave(platos, nombre, info) //ADT LO

    existePlato = V

  Excepciones

    cuando cualquiera => existePlato = F

  Fin

Fin existePlato