República Bolivariana de Venezuela

Ministerio del PP Educación Superior

Universidad Nacional Abierta

Centro Local Metropolitano

TP de Computacion 1

Nombre: Antonio Bigott

C.I.: 3 410 173

Email: antoniobigott@gmail.com

Tlf: 02129635450/ 04164279860

Código carrera: 236

Asignatura: Computación 1(323)

Profesor: Mercedes Picón

Fecha: 30-12-2010

Lapso 2010-2

ÍNDICE

INTRODUCCIÓN

Este trabajo práctico se propone resolver un problema de una compañía ficticia que contiene grandes enseñanzas que sirven para la formación de los futuros Ingenieros de Sistemas.

Las empresas dedicadas al turismo deben tener un sistema computarizado para lograr competir en este mercado tan dinámico y atractivo.

Se trata de implementar un algoritmo que pueda ingresar datos de los clientes de la empresa TOURSVEN. Para llegar a esto, se aplica el método MAPS, el cual sistemáticamente nos llevará hasta el diseño final del programa, escrito en lenguaje PASCAL, y disponible para ser aplicado en el área de trabajo.

Todo esto es parte de la materia Computación 1, relativo a los objetivos 5,6, 7 y 8. Aceptamos las observaciones hechas por nuestra asesora del centro local metropolitano.

DESARROLLO

OBJETIVOS

5: Se resolverá el problema planteado algorítmicamente usando la metodología MAPS

6: Basándose en lo construido en el objetivo 5, se diseñará un algoritmo usando técnicas de programación estructurada, que cumplan con las especificaciones dadas, teniendo especial cuidado en hacer una buena declaración de datos y sus tipos.

7: Se probará el algoritmo diseñado en el objetivo 6 usando el concepto de Robustez.

8: Se traducirá el algoritmo obtenido a lenguaje PASCAL, a fin de obtener un programa estructurado que cumpla con los requerimientos especificados, aplicando procedimientos y funciones.

OBJETIVO 5

MAPS Son siglas inglesas que significan Metodología para Solventar Problemas Algorítmicos, en español, y es la sucesión de las siguientes etapas: Dialogo, Especificaciones, Subdivisión, Definición de Abstracciones, Codificación, Prueba y Verificación, Presentación.

ETAPA 1: EL DIALOGO

Se realizó una consulta al asesor de la materia en el Centro Local. Unido a la conversación entre compañeros de estudio, se aclaró la naturaleza del problema planteado.

Definición del problema: La agencia de viajes TOURSVEN se ha dado a la tarea de automatizar el proceso de venta de boletos con el fin de llevar un control de clientes que deseen viajar al exterior.

Datos de Entrada: Se toma como entradas:

- Nombre y apellidos del cliente
- Número de pasaporte del cliente
- Fecha de expedición del pasaporte
- Origen del pasaporte
- Fecha de expedición del certificado vacuna
- Fecha de expedición de declaración de impuesto

Las fechas caducan al año, excepto la fecha de expedición del pasaporte en caso de ser extranjero, que vence a los dos años.

Proceso: se validan los datos de entrada, de manera que sólo se graban los datos válidos según el criterio preestablecido. Se diseñará un programa en Turbo Pascal 7.0, para que usando arreglos de registros, se graben y muestren los resultados requeridos.

Datos de salida: previamente se mostrará:

- un menú que permita ingresar, modificar, eliminar datos y mostrar el listado de personas registradas como clientes.
- Un menú que permita ordenar de acuerdo al número de pasaporte de cada cliente, mostrar en pantalla los clientes que tienen

documentos No-vigentes, y visualizar el reporte estadístico de clientes.

- El reporte contiene lo siguiente:
  - Cuántos clientes tienen pasaporte venezolano
  - Cuántos clientes tienen pasaporte extranjero
  - Cuántos clientes tienen el certificado de vacuna vencido
  - Cuántos clientes tienen la declaración de impuesto vencido
  - Cuántos clientes tienen todos sus recaudos vigentes

Para verificar el vencimiento, se toma como datos sólo el año y el mes, y se considerará vencido el documento que en su fecha de vencimiento, el año y el mes coincidan o sean menores al año y mes actuales.

ETAPA 2: ESPECIFICACIONES

PRE-Condiciones:

- Nombres y apellidos de cada cliente. CRITERIO DE VALIDEZ: no puede tener números al principio. Ej.: “1carlos” no es válido, en cambio “carlos1” sí lo es.
- Número de pasaporte de cada cliente. CRITERIO DE VALIDEZ: no puede ser letras. El rango se toma como $100000 < N - pas < 99999999$
- Fecha de expedición de pasaporte. CRITERIO DE VALIDEZ: no puede ser letras. Rangos:
  - $1990 \leq \text{año} \leq \text{año } actual$
  - $1 \leq mes \leq 12$ ó $mes\ actual$
  - $1 \leq día \leq 31$
  - La fecha se introduce completa. Se colocan 4 dígitos para el año, 2 dígitos para el mes y 2 dígitos para el día, todos separados por un espacio. Ej.: el 21 de enero de 2009 se coloca $2009\ 01\ 21$. No es válido colocar $2009\ 1\ 21$, dado que para meses menores a 10 se completan con cero a la izquierda.

  - La comparación se hace con ayuda de un comando que muestra la fecha del reloj interno de la computadora, que normalmente estará ajustada a la fecha actual.
  - Si el año es bisiesto, se aceptará el día 29 de febrero.
  - Normalmente el mes de febrero no acepta los días 29, 30, 31 y los meses abril, junio, septiembre, noviembre tienen 30 días.

- Origen del pasaporte: venezolano (v) ó extranjero (e). CRITERIO DE VALIDEZ: no puede ser otro carácter diferente a éstos dos.
- Fecha de expedición del certificado de vacuna. CRITERIO DE VALIDEZ: el mismo de la fecha de expedición pasaporte.
- Fecha de declaración de impuesto. CRITERIO DE VALIDEZ: el mismo de la fecha de expedición pasaporte.
- Dado que el programa será diseñado en base a arreglos, será fijada la capacidad máxima de éstos. Ahora, en el día laboral, la cantidad de clientes varía y el usuario no puede estar todo el tiempo ajustando la capacidad máxima del arreglo de clientes. Por lo tanto, se preguntará al usuario cuántos clientes ingresará.

POST-Condiciones:

- $\text{entradas} = \emptyset \wedge \text{salida} = \text{datos válidos}$. Sólo los datos validos se graban.
- $\text{entradas} = \emptyset \wedge \text{salida} = \text{Listado}$. Aparece una tabla que indica los clientes registrados en el programa, con sus 6 características o campos cada uno.
- $\text{entradas} = \emptyset \wedge \text{salida} = \text{Reporte estadístico}$. Aparece una tabla conteniendo la suma de los clientes con pasaporte venezolano, los de pasaporte extranjero, con certificado de vacuna vencido, con declaración de impuesto vencido y los que tienen todos sus recaudos vigentes.
- $\text{entradas} = \emptyset \wedge \text{salida} = \text{No} - \text{vigentes}$. Aparece un listado que muestra los nombres y apellidos de los clientes que tienen por lo menos un documento vencido.
- $\text{entradas} = \emptyset \wedge \text{salida} = \text{Lista ordenada}$. Aparece un listado que muestra los números de pasaporte de los clientes ordenados de mayor a menor.

ETAPA 3: LA PARTICIÓN

El problema global se dividirá en base a las opciones del menú principal y submenús:

- Actualizar
  - •Ingresar
  - •Modificar
  - •Eliminar
  - •Listado
  - •Salida _menu principal
- Ordenar
- Emitir reporte No-vigentes
- Emitir reporte estadistico
- Salida _del_programa

Cada una de estas opciones, incluyendo los menús, es diseñada en base a los procedimientos, que responden al concepto de la programación estructurada.

1. Ingresar (primera vez y subsiguientes): Se leerá cuántos clientes se ingresará, ya que la capacidad total del array (arreglo) se fija a un cierto número (20), pero en el día laboral esta cantidad puede variar. Luego se leerán los datos del cliente: nombre y apellidos, n° de pasaporte, fecha expedición del pasaporte, origen del pasaporte, fecha expedición del certificado-vacuna, fecha expedición de declaración impuesto. Si un cliente no tiene un documento, por convención se graba en la fecha de expedición respectiva "1990 01 01". Esto con el fin de alentar al cliente a que consiga sus documentos, mientras que está en un archivo de espera.

   Estos datos se almacenan en un arreglo de tipo registro, que lo llamamos Clien_T. este arreglo está fijado a una capacidad definida, que puede estar entre 15 a 30 clientes por día laboral. Si la cantidad de clientes ingresados es menor que la capacidad total del arreglo, se rellenan con ceros en los campos numéricos y espacios vacios o cadenas nulas en los campos de caracteres. Si se desea ingresar cierta cantidad de clientes adicionales, se puede acceder nuevamente a esta opción. Para ello, la variable de clientes adicionales (w2) se suma a la variable de clientes existentes (w): $n = w + w2$, donde n es número total de

clientes ingresados. Esto se hace para no sobrepasar la capacidad total del arreglo de clientes. Si ocurre que la cantidad a ingresar es mayor que la capacidad del arreglo, el procedimiento no se ejecuta y se devuelve al menú. Por supuesto que la capacidad total se fija a la cantidad máxima que puede haber de clientes en temporada alta.

2. Modificar (un cliente por vez): se pregunta cuál cliente se desea modificar ingresando con precisión sus nombres y apellidos. El algoritmo de búsqueda detecta el cliente solicitado y se muestra un pequeño menú para seleccionar qué campo será modificado del registro del cliente. Se selecciona el campo a modificar según sea el caso y se releen los datos a modificar.
3. Eliminar ("borrar" el registro del cliente): se pregunta cuál cliente se debe eliminar ingresando con precisión sus nombres y apellidos. El algoritmo de búsqueda detecta el cliente solicitado y luego procede a grabar ceros en los campos numéricos y espacios vacios en los campos de caracteres.
   El número de clientes se reduce en 1.
4. Listado: se muestran todos los clientes almacenados en el registro clien_T. puede estar ordenado o no, dependiendo si se utilizó la opción ordenar. Para listar los datos, se dividió el proceso en 6 partes y se utilizó el comando "gotoxy" (ir_xy) para ubicar el cursor en las coordenadas de la pantalla.
5. Salida: el programa finaliza y se cierra la ventana respectiva, volviendo al menú principal.
6. Ordenar: este procedimiento ordena por número de pasaporte de mayor a menor a todos los clientes en el arreglo. Se usa un algoritmo de selección de la manera siguiente:
   - El primer elemento de la lista de clientes se toma como el valor variable "índice-menor"
   - Se compara con los restantes elementos de Clien_T en el campo N-pas.
   - Si algún elemento es menor que el "índice-menor", éste elemento se convierte en "índice-menor" y se sigue comparando esta variable con los restantes elementos de Clien_T.
   - Si no hay elementos menores, se coloca este índice como el último de la lista ordenada y se toma el siguiente elemento de la lista sin ordenar como el nuevo "índice-menor"
   - Se repite el procedimiento para este nuevo índice que ya tendrá $n - 1$ elementos
   - Así sucesivamente se van obteniendo la lista ordenada según el campo N-pas del arreglo de clientes.

   - Finalmente, se muestra el listado ordenado.
7. Reporte No-vigentes: Se detecta con esta opción si algún cliente presenta algún recaudo no-vigente. Esto se logra comparando las fechas de vencimiento del pasaporte, certificado de vacuna y declaración de impuesto con la fecha actual. La fecha actual se obtiene mediante un comando llamado GetDate (obtener fecha), y es la fecha del reloj interno de la computadora. La fecha de vencimiento se obtiene sumando 1 ó 2 al año de expedición, según sea el caso. Si la fecha de vencimiento de un documento es menor que la fecha actual, el documento está vencido y aparecerá el nombre del cliente involucrado en el listado del reporte. En este listado sólo aparecerán los clientes con por lo menos un documento vencido.
8. Reporte estadístico (5 datos de salida): este procedimiento arroja un listado en pantalla con los elementos siguientes: clientes con pasaporte venezolano, clientes con pasaporte extranjero, clientes con certificado de vacuna vencido, clientes con declaración de impuesto vencido, clientes con todos sus recaudos vigentes. Esto se logra comparando fechas de vencimiento y con una variable contadora, se va sumando de uno en uno cada resultado.
9. Salida del programa: el programa se cierra y se vuelve al sistema operativo de la computadora.

ETAPA 4: DEFINICIÓN DE ABSTRACCIONES

En turbo-Pascal 7.0 se utilizan los siguientes comandos:

- Lectura (read, readln) y escritura (write, writeln) de datos
- Limpieza de pantalla (clrscr), ubicación del cursor (gotoxy), leer una tecla cualquiera (readkey)
- Ciclos repetitivos como desde_hasta_hacer (for…to…do), donde la variable se incrementa automáticamente.
- Ciclos repetitivos como mientras_hacer (while…do), donde hay que incrementar la variable contadora explícitamente.
- Ciclos repetitivos como Repetir_hasta (repeat…until). Esta estructura se utiliza para construir el algoritmo de búsqueda, pues el ciclo se detiene al encontrar el dato y por lo menos se ejecuta una vez. Además, es muy especial para la validación de datos. Se utiliza un indicador de tipo booleano: al principio se almacena “true” (verdadero) en la variable “detec”, para que cambie a “false” si se detecta que el dato no es el correcto. Esto hace que se vuelva a repetir el ciclo de lectura de datos hasta que los datos sean los correctos.

- Ciclos selectivos como Si_entonces_si-no (if…then…else). Se ejecuta si la condición a que hace referencia el "si", es verdadera. Esto actúa como un detector de datos de entrada. La condicional es de las formas lógicas más utilizadas en los programas.
- Ciclos selectivos como En_caso_de_Hacer (case of…do). Estructura utilizada con preferencia en la construcción de menús. Se realizan ciertas instrucciones dependiendo del carácter que representa una opción del menú.
- Incrementar (Inc): utilizado para la variable contadora de un ciclo. Equivale a la declaración $cont := cont + 1$.
- Operadores lógicos AND, OR y NOT. Son utilizados para decidir sobre varias condiciones lógicas. Sus tablas de verdad son:

Entrada X	Entrada Y	Salida AND	Salida OR	Salida NOT X
Falso (false)	Falso	Falso	Falso	Verdad
Falso	Verdad (true)	Falso	Verdad	Verdad
Verdad	Falso	Falso	Verdad	Falso
Verdad	Verdad	Verdad	Verdad	Falso

ETAPA 5: CODIFICACIÓN

Esta etapa corresponde al objetivo 6

ETAPA 6: PRUEBA Y VERIFICACIÓN

Esta etapa corresponde al objetivo 7

ETAPA 7: PRESENTACIÓN

Los comentarios se añaden entre llaves { }. Esto se añade normalmente al seudocódigo del programa en su realización (objetivo 6 y 8).

OBJETIVO 6: PROGRAMACIÓN ESTRUCTURADA

DECLARACIÓN DE DATOS Y TIPOS

CONSTANTES

EXPLICACIÓN	ABREVIATURA	VALORES
# caracteres	Cad	60
Capacidad del arreglo de clientes	Capcid	20
Arreglo de meses (sólo para uso del comando Getdate)	Mesi	Rango de enero a diciembre
Arreglo de días (sólo para uso del comando Getdate)	Días	Rango de Domingo a sábado

TIPOS

EXPLICACIÓN	ABREVIATURA	TIPO
Rango de números ordinales desde uno	Rcap	1..*capcid*
Rango de números ordinales desde cero	Rgs	0..*capcid*
Tipo ordinal palabra	Max	Word (0..65535)
Cadena de caracteres	Modf	String[cad]
Tipo ordinal entero-largo	Psp2	Longint (-2147483648..2147483647)
Tipo lógico booleano	Logi	Boolean (true..false)
Tecla a pulsar	Op	Char (carácter)
Registro de 6 campos	Cturi	Record of *nom_ap, n_pas, f_exp, orig, f_cva, f_dim*
Nombres y apellidos	Nom_ap	*Modf*
Numero de pasaporte	N_pas	*Psp2*

Fecha expedición pasaporte	F_exp	Array[1..3] of *max*
Origen de pasaporte	Orig	*op*
Fecha certificado vacuna	F_cva	Array[1..3] of *max*
Fecha declaración impuesto	F_dim	Array[1..3] of *max*
Arreglo de registros	Clien_T	Array[*rcap*] of *cturi*
Arreglo lógico booleano	Vigent	Array[*rcap*] of *logi*
# clientes a ingresar	W,w2	*max*
Año, mes y día proporcionado por GetDate	y, m, d	*max*
números para validar	Vali7	Rango de 0..9
cadenas para validar	Valin, valif, valia, valib	*Valin:* string[1], *valif:* string[11], *valia:* string[4], *valib*: string[2]

ALGORITMO EN SEUDOCÓDIGO

Programa Servicio_Turismo {hecho en Venezuela para la empresa TOURSVEN}

PROCEDIMIENTO noms (variable L1:modf) {validación nombre-apellidos***}***
Variables locales e: vali7; st: valin;

Comienzo

e←0; leer(L1);

Repetir

cambiar_a_cadena (e, st)

Si L1[1]=st entonces

Comienzo

Escribir('no válido, inserte de nuevo'); leer(L1)

fin_Si

Incrementar (e)

hasta_que e=0

fin_noms;

PROCEDIMIENTO letras(variable D: psp2) {detecta letras en vez de números}
Variables deti: logi; cod: max; L1: valif;

Comienzo

Repetir

deti←verdad; cod←0; leer(L1)

cambiar_a_entero (L1, D, cod)

Si cod≠0 **entonces**

Comienzo

Escribir('no válido, inserte número '); deti←falso

fin_**Si**

hasta_que deti

fin_letras

PROCEDIMIENTO valipas (variable D:psp2) {validación #_pasaporte}
Variables detec: logi;

Comienzo

Repetir

detec←verdad; letras(D) {llamada al procedimiento letras}

Si (D<100 000) OR (D>99 999 999) **entonces**

Comienzo

Escribir('no válido, inserte de nuevo'); detec←falso

fin_**Si**

Hasta_que detec

Fin_valipas

PROCEDIMIENTO vori (variable o:op) {validación de origen}
Variable deti: logi;

Comienzo

deti←falso

Repetir

Leer(o)

Si (o≠'v') AND (o≠'e') **entonces**

Escribir('no válido, inserte de nuevo')

Si_no

deti←verdad

hasta_que deti

Fin_vori

PROCEDIMIENTO letr2 (variable d, e, f:max) {detecta letras para fechas}
Variables deti: logi; cd, cd2, cd3: max; m: valia; n, p: valib; c:valif;

Comienzo

Repetir

deti←verdad; cd←0; cd2←0; cd3←0

Leer (c)

m←copiar_cadena(c, 1, 4); cambiar_a_entero (m, d, cd)

n←copiar_cadena(c, 6, 2); cambiar_a_entero (n, e, cd2)

p←copiar_cadena(c, 9, 2); cambiar_a_entero (p, f, cd3)

Si [[(cd≠0) OR (cd2≠0)] OR (cd3≠0)] **entonces**

Comienzo

Escribir ('no válido, inserte números'); deti←falso

fin_**Si**

Hasta_que deti

Fin_letr2

PROCEDIMIENTO valife (y, m, d: max; variable f1, f2, f3: max) {valida fechas}
Variable detec: logi;

Comienzo

Repetir

detec←verdad; f1←0; f2←0; f3←0

letr2(f1, f2, f3) {llamada a proced. letr2}

1_**Si** (f1<1990) OR (f1>y) **entonces**

Comienzo

Escribir (‘no válido, inserte nuevo año’); detec←falso

fin_1_**Si**

2_**Si** (f1=y) AND (f2>m) **entonces**

Comienzo

Escribir (‘no válido, inserte nuevo mes<actual); detec←falso

fin_2_**Si**

3_**Si** (f2<1) OR (f2>12) **entonces**

Comienzo

Escribir (‘no válido, inserte nuevo mes’); detec←falso

fin_3_**Si**

4_**Si** (f3<1) OR (f3>31) **entonces**

Comienzo

Escribir (‘no válido, inserte nuevo día’); detec←falso

fin_4_**Si**

5_**Si** [(f2=2) AND (f3=29)] AND (f1 mod 4≠0) **entonces**

Comienzo

Escribir (‘no válido, inserte nuevo dia_feb’); detec←falso

fin_5_**Si**

6_**Si** (f2=2) AND (f3=30) **entonces**

Comienzo

Escribir (‘no válido, inserte nuevo dia_febrero’);

detec←falso

fin_6_**Si**

7_**Si** [(f2=2) OR (f2=4) OR (f2=6) OR (f2=9) OR (f2=11)] AND (f3=31)

Entonces

Comienzo

Escribir (‘no válido, inserte nuevo dia≤30’); detec←falso

fin_7_**Si**

Hasta_que detec

Fin_valife

PROCEDIMIENTO Ingresa (variable w:max, g:clien_T)
Variables w2: max

PROCEDIMIENTO Nap (entradas w, w2: max; variable g: clien_T)
{nombres_apellidos}
Variables Q: rcap; L:modf

Comienzo

Q←w+1

1_**Mientras** Q≤(w+w2) **hacer**

Comienzo

Escribir (‘introduzca apellidos y nombres de la persona numero’, Q, ‘a ingresar’)

Noms(L) {llama a proced. noms}

```
                g[Q].nom_ap←L;   Incrementar (Q)
            Fin_1_mientras
        2_mientras Q≤capcid hacer
            Comienzo
                g[Q].nom_ap←'' {cadena nula}
                Incrementar (Q)
            Fin_2_mientras
Fin_Nap

PROCEDIMIENTO Npas (entradas w, w2; variable g) {numero de pasaporte}
Variables D2: psp2; Q: rcap

Comienzo
    Q←w+1
    1_mientras Q≤(w+w2) hacer
        Comienzo
            Escribir ('introduzca el numero de pasaporte de la persona
            numero', Q, 'a ingresar')
            Valipas (D2) {llama a proced. valipas}
            g[Q].n_pas←D2; Incrementar(Q)
        Fin_1_mientras
    2_mientras Q≤capcid hacer
        Comienzo
            g[Q].n_pas←0
            Incrementar (Q)
        Fin_2_mientras
Fin_Npas
```

PROCEDIMIENTO Fexp (w, w2; variable g) {fecha_expedicion pasaporte}
Variables S: rcap; f1, f2, f3: max

Comienzo

f1←0; f2←0; f3←0; S←w+1

1_**mientras** S≤(w+w2) **hacer**

Comienzo

Escribir (‘introduzca fecha_expedicion: aaaa mm dd separado por espacio de la persona numero’, S, ‘a ingresar’)

Valife (y, m, d, f1, f2, f3) {validación de fechas}

g[S].f_exp[1]←f1; g[S].f_exp[2]←f2; g[S].f_exp[3]←f3

Incrementar (S)

Fin_1_**mientras**

2_**mientras** S≤capcid **hacer**

Comienzo

g[S].f_exp[1]←0; g[S].f_exp[2]←0; g[S].f_exp[3]←0;

Incrementar (S)

Fin_2_**mientras**

Fin_Fexp

PROCEDIMIENTO ori (w, w2; variable g) {origen de pasaporte}
Variables S: rcap; o: op

Comienzo

S←w+1

1_**Mientras** S≤(w+w2) **hacer**

Comienzo

Escribir (‘introduzca origen del pasaporte: v/e de la persona numero’, S, ‘a ingresar’)

vori(o) {llama a proced. vori, validación de origen}

```
                g[S].orig←o;   Incrementar (S)
            Fin_1_mientras
        2_mientras S≤capcid hacer
            Comienzo
                g[S].orig←' ' {carácter espacio}
                Incrementar (S)
            Fin_2_mientras
Fin_ori

PROCEDIMIENTO Fcv (w, w2; variable g) {fecha certificado de vacuna}
Variables S: rcap; g1, g2, g3: max

Comienzo
    g1←0; g2←0; g3←0; S←w+1
    1_mientras S≤ (w+w2) hacer
        Comienzo
            Escribir ('introduzca fecha_certificado_vacuna: aaaa mm dd
            separado por espacio de la persona numero', S, 'a ingresar')
            Valife (y, m, d, g1, g2, g3) {validación de fechas}
            g[S].f_cva[1]←g1; g[S].f_cva[2]←g2; g[S].f_cva[3]←g3
            Incrementar (S)
        Fin_1_mientras
    2_mientras S≤capcid hacer
        Comienzo
            g[S].f_cva[1]←0; g[S].f_cva[2]←0; g[S].f_cva[3]←0;
            Incrementar (S)
        Fin_2_mientras
Fin_Fcv
```

```
PROCEDIMIENTO Fdim (w, w2; variable g) {fecha declaracion_impuesto}
Variables S: rcap; h1, h2, h3: max

Comienzo

    h1←0; h2←0; h3←0; S←w+1

    1_mientras S≤ (w+w2) hacer

        Comienzo

            Escribir ('introduzca fecha_dec._impuesto: aaaa mm dd
            separado por espacio de la persona numero', S, 'a ingresar')

            Valife (y, m, d, h1, h2, h3) {validación de fechas}

            g[S].f_dim[1]←h1; g[S].f_dim[2]←h2; g[S].f_dim[3]←h3

            Incrementar (S)

        Fin_1_mientras

    2_mientras S≤capcid hacer

        Comienzo

            g[S].f_dim[1]←0; g[S].f_dim[2]←0; g[S].f_dim[3]←0;

            Incrementar (S)

        Fin_2_mientras

Fin_Fdim

Comienzo {programa Ingresa}

    Limpiar_pantalla {clrscr};  w2←0

    Si w≠0 entonces

        Comienzo

            Escribir ('¿cuántos clientes adicionales?')

            Leer (w2)

        Fin_Si

    Si_no comienzo
```

Escribir (‘¿cuántos clientes ingresará?’)

Leer (w2)

fin_**Si**_no

Si (w+w2) ≤capcid **entonces**

Comienzo

Nap (w, w2, g) {llamadas a los proceds. de Ingresar}

Npas (w, w2, g)

Fexp (w, w2, g)

Ori (w, w2, g)

Fcv (w, w2, g)

Fdim (w, w2, g)

w←w+w2

fin_**si**

si_no escribir (‘número total de clientes sobrepasa capacidad máxima’)

Escribir (‘presione cualquier tecla’)

leer_tecla

Fin_Ingresa

PROCEDIMIENTO buscali (entradas lista: clien_T; t: modf; variable posic: rgs) {búsqueda lineal de nombres, necesario para los proceds. Modificar o Eliminar}
Variables I: rcap; encont: logi

Comienzo

encont←falso {hasta que la persona se encuentra}; posic←0; I←1

Repetir

Si (lista[I].nom_ap)=*t* **entonces**

Comienzo

posic←I; encont←verdad

fin_**Si**

Incrementar (I)

hasta_que (encont=verdad) OR (I=1) {si I llega al máximo valor, el incremento lo devuelve al primer valor del rango}

Fin_buscali

PROCEDIMIENTO Modifica (entrada y, m, d, w: max; variable k: clien_T) {se modifican datos del cliente}
Variables mo: op; pmod: modf; pos: rgs

PROCEDIMIENTO mnomb (pos; variable k)
Variable L:modf

Comienzo

Escribir ('inserte nuevos nombres y apellidos')

Noms (L) {validación nombres y apellidos}

k[pos].nom_ap←L

Fin_mnomb

PROCEDIMIENTO mnpas (pos; variable k)
Variable D2: psp2

Comienzo

Escribir ('inserte Nuevo numero pasaporte')

Valipas (D2) {validación # pasaporte}

k[pos].n_pas←D2

Fin_mnpas

PROCEDIMIENTO mfex (y, m, d, pos; variable k)
Variables f1, f2, f3: max

Comienzo

f1←0; f2←0; f3←0;

```
        Escribir ('introduzca nueva fecha_expedicion: aaaa mm dd separado por
        espacio')

        Valife (y, m, d, f1, f2, f3) {validación de fechas}

        k[pos].f_exp[1]←f1; k[pos].f_exp[2]←f2; k[pos].f_exp[3]←f3

Fin_mfex

PROCEDIMIENTO mdor (pos; variable k)
Variable o:op

Comienzo

        Escribir ('escriba nuevo origen')

        Vori (o) {validación origen}

        k[pos].orig←o

Fin_mdor

PROCEDIMIENTO mcva (y, m, d, pos; variable k)
Variables f1, f2, f3: max

Comienzo

        f1←0; f2←0; f3←0;

        Escribir ('introduzca nueva fecha_vacuna: aaaa mm dd separado por
        espacio')

        Valife (y, m, d, f1, f2, f3) {validación de fechas}

        k[pos].f_cva[1]←f1; k[pos].f_cva[2]←f2; k[pos].f_cva[3]←f3

Fin_mcva

PROCEDIMIENTO mdim (y, m, d, pos; variable k)
Variables f1, f2, f3: max

Comienzo

        f1←0; f2←0; f3←0;

        Escribir ('introduzca nueva fecha_declaración: aaaa mm dd separado por
        espacio')
```


Digitalizado por Osmer Montilla para www.admycontuna.multiply.com

```
        Valife (y, m, d, f1, f2, f3) {validación de fechas}
        k[pos].f_dim[1]←f1; k[pos].f_dim[2]←f2; k[pos].f_dim[3]←f3
Fin_mdim
Comienzo {Modifica_principal}
        Escribir ('ingrese apellidos_nombres del cliente')
        Leer (pmod)
        Buscali (k, pmod, pos) {búsqueda lineal}
        Si pos≠0 entonces
                Comienzo
                        Escribir ('¿qué desea modificar?: nombres (N),
                        numero_pasaporte (P), f_expedicion (E), origen (G),
                        f_certificado (C), f_declaracion (D)')
                        Leer (mo)
                        En_caso_de mo hacer
                                'n': mnomb (pos, k)
                                'p': mnpas (pos, k)
                                'e': mfex (y, m, d, pos, k)
                                'g': mdor (pos, k)
                                'c': mcva (y, m, d, pos, k)
                                'd': mdim (y, m, d, pos, k)
                         Fin_en_caso_de
                Fin_Si
        Si_no  escribir ('vuelva al menú principal')
        Escribir ('1 cliente modificado de', w)
        escribir ('presione cualquier tecla')
        leer_tecla
```

Fin_Modifica

PROCEDIMIENTO verim (entradas y, m, w:max; variable p: clien_T) {verifica si ha ingresado clientes}
Comienzo

Si w≠0 **entonces**

Modifica (w, p)

Si_no escribir (‘ingrese clientes por favor’)

Leer_tecla

Fin_verim

PROCEDIMIENTO venci (entradas nac, y, m, a1, m1: max; variable sis, ven: logi) {verifica si hay fechas vencidas para n_vige y reporte}
Comienzo

ven←falso; sis←verdad; a1←a1+nac

Si a1=nac **entonces** sis←falso

Si (a1<y) AND (a1≠nac) **entonces** ven←verdad

Si (a1=y) AND (m1≤m) **entonces** ven←verdad

Fin_venci

PROCEDIMIENTO n_vige (a, me: max; regi: clien_T) {listado de no-vigentes}
Variable vn, vc, vd: vigent;

PROCEDIMIENTO n_exp (y, m: max; k: clien_T; variable venz: vigent)
Variable S: rcap; a1, t1, nac: max; indi, sis: logi

Comienzo

Desde S←1 hasta capcid **hacer**

Comienzo

a1←k[S].f_exp[1]; t1←k[S].f_exp[2]; venz[S]←falso

1_**Si** k[S].orig=’v’ **entonces** {si es venezolano…}

Comienzo

```
                nac←1
                venci (nac, y, m, a1, t1, sis, indi) {verifica fechas}
                venz[S]←indi
            fin_1_Si
           2_Si k[S].orig='e' entonces {si es extranjero…}
            Comienzo
                nac←2
                venci (nac, y, m, a1, t1, sis, indi) {verifica fechas}
                venz[S]←indi
            fin_2_Si
        Fin_desde
Fin_n_exp

PROCEDIMIENTO n_CV (y, m: max; k: clien_T; variable venc: vigent)
Variables S: rcap; cv, mv, nac: max; indi, sis: logi

Comienzo
        Desde S←1 hasta capcid hacer
         Comienzo
            cv← k[S].f_cva[1]; mv← k[S].f_cva[2]; nac←1
            venci (nac, y, m, cv, mv, sis, indi)
            venc[S]←indi
         Fin_desde
Fin_n_CV

PROCEDIMIENTO n_DI (y, m: max; k: clien_T; variable vedi: vigent)
Variables S: rcap; cd, md, nac: max; indi, sis: logi

Comienzo
        Desde S←1 hasta capcid hacer
```

```
        Comienzo
            cd← k[S].f_dim[1]; md← k[S].f_dim[2]; nac←1
            venci (nac, y, m, cd, md, sis, indi)
            vedi[S]←indi
        Fin_desde
Fin_n_DI
PROCEDIMIENTO total_vi (venz, venc, vedi: vigent; k: clien_T)
Variable S: rcap
Comienzo
    Desde S←1 hasta capcid hacer
        Comienzo
            Si [venz[S] OR venc[S]] OR (vedi[S]) entonces
                Comienzo
                    Escribir ('el cliente', k[S].nom_ap, 'tiene recaudos no vigentes')
                Fin_Si
        Fin_desde
Fin_total_vi
Comienzo {n_vige principal}
    Limpiar_pantalla
    N_exp (a, me, regi, vn)
    N_CV (a, me, regi, vc)
    N_DI (a, me, regi, vd)
    Total_vi (vn, vc, vd, regi)
    escribir ('presione cualquier tecla')
    leer_tecla
```

Fin_n_vige

PROCEDIMIENTO verino (y, m, w: max; p: clien_T)
Comienzo

Si w≠0 **entonces**

N_vige (y, m, p)

Si_no escribir (‘ingrese clientes por favor’)

Leer_tecla

Fin_verino

PROCEDIMIENTO Lorden (variable list: clien_T) {ordena el registro de clientes}
Variable L: rcap

FUNCIÓN Indm (entrada K: rcap; variable lista: clien_T): max {encuentra el índice del elemento menor}
Variables imi, indi: rcap

Comienzo

imi←1

desde indi←2 hasta K **hacer**

Si (lista[indi].n_pas)<(lista[imi].n_pas) **entonces**

imi←indi

Indm←imi

Fin_Indm

PROCEDIMIENTO Seleccs (variable lista: clien_T) {intercambia con elemento lista[K]}
Variables aux2: clien_T; me: max; K: rcap

Comienzo

Desde K←capcid descendiendo_hasta 2 **hacer**

Comienzo

me←Indm (K, lista)

```
            aux2[me]←lista[me]
            lista[me]←lista[K]
            lista[K]←aux2[me]
        Fin_desde
Fin_Seleccs
Comienzo {Lorden principal}
    Limpiar_pantalla
    Seleccs (list)
    Escribir ('lista ordenada del número pasaporte')
    Desde L←1 hasta capcid hacer
        Escribir (L, ' ', list[L].n_pas)
    escribir ('presione cualquier tecla')
    leer_tecla
Fin_Lorden
```

PROCEDIMIENTO verior (w: max; variable p: clien_T)

```
Comienzo
    Si w≠0 entonces
        Lorden (p)
    Si_no escribir ('ingrese clientes por favor')
    Leer_tecla
Fin_verior
```

PROCEDIMIENTO Elimi (entrada w: max; variable k: clien_T) {elimina registro de cliente}

```
Variable nel: modf; pos: rgs; aux: max
Comienzo
    Limpiar_pantalla
```

Escribir (‘ingrese nombre_apellidos a eliminar’)

Leer (nel)

Buscali (k, nel, pos)

Si pos≠0 **entonces**

Comienzo

k[pos].nom_ap←’eliminado’; k[pos].n_pas←0

k[pos].f_exp[1]←0; k[pos].f_exp[2]←0; k[pos].f_exp[3]←0

k[pos].orig←’ ’ {carácter espacio}

k[pos].f_cva[1]←0; k[pos].f_cva[2]←0; k[pos].f_cva[3]←0

k[pos].f_dim[1]←0; k[pos].f_dim[2]←0; k[pos].f_dim[3]←0; aux←w – 1

escribir (‘total clientes restantes: ’, aux)

Fin_**Si**

Si_no escribir (‘vuelva al menú principal’)

escribir (‘presione cualquier tecla’)

leer_tecla

Fin_Elimi

PROCEDIMIENTO veriel (w: max; variable p: clien_T)

Comienzo

Si w≠0 **entonces**

elimi (w, p)

Si_no escribir (‘ingrese clientes por favor’)

Leer_tecla

Fin_veriel

PROCEDIMIENTO Reporte (y, m: max; k: clien_T) {reporte estadístico de clientes}

Variables vz, ex, cv, di, suvi: rgs; vi, vi2, vi3: vigent

```
PROCEDIMIENTO sumaVZ (y, m: max; k: clien_T; variable sumvz, sumex: rgs;
vig: vigent) {suma pasaportes venezolanos y extranjeros}
Variables S: rcap; a1, t1, nac: max; indi, sis: logi

Comienzo

        sumvz←0; sumex←0

        desde S←1 hasta capcid hacer

          comienzo

                a1←k[S].f_exp[1]; t1←k[S].f_exp[2]

                1_Si k[S].orig='v' entonces

                  Comienzo

                        Incrementar (sumvz); nac←1;

                        Venci (nac, y, m, a1, t1, sis, indi) {verifica fechas vencidas}

                        Vig[S]←NOT indi

                  fin_1_Si

        2_Si k[S].orig='e' entonces

                  Comienzo

                        Incrementar (sumex); nac←2;

                        Venci (nac, y, m, a1, t1, sis, indi) {verifica fechas vencidas}

                        Vig[S]←NOT indi

                  fin_Si_2

          fin_desde

Fin_sumaVZ

PROCEDIMIENTO sumaCV (entradas y, m, k; variable sumcv: rgs; vig2: vigent)
Variables S: rcap; cv, mv, nac: max; indi, sis: logi

Comienzo

        sumcv←0
```

```
        Desde S←1 hasta capcid hacer
          Comienzo
                cv←k[S].f_cva[1]; mv←k[S].f_cva[2]; nac←1
                Venci (nac, y, m, cv, mv, sis, indi) {verifica fechas vencidas}
                Vig2[S]←(NOT indi) AND sis
                Si indi entonces Incrementar (sumcv)
          fin_desde
Fin_sumaCV

PROCEDIMIENTO sumaDI (entradas y, m, k; variable sumdi: rgs; vig3: vigent)
Variables S: rcap; d1, md, nac: max; indi, sis: logi

Comienzo
        sumdi←0
        Desde S←1 hasta capcid hacer
          Comienzo
                d1←k[S].f_dim[1]; md←k[S].f_dim[2]; nac←1
                Venci (nac, y, m, d1, md, sis, indi) {verifica fechas vencidas}
                Vig3[S]←(NOT indi) AND sis
                Si indi entonces Incrementar (sumdi)
          fin_desde
Fin_sumaDI

PROCEDIMIENTO sumaRV (entradas vig, vig2, vig3; variable sumvi: rgs) {suma
recaudos vigentes}
Variable S: rcap

Comienzo
        sumvi←0
        Desde S←1 hasta capcid hacer
```

Comienzo

Si [vig[S] AND vig2[S]] AND (vig3[S]) **entonces**

Incrementar (sumvi)

fin_**desde**

Fin_sumaRV

Comienzo {Reporte principal}

Limpiar_pantalla

vz←0; ex←0; cv←0; di←0; suvi←0

sumaVZ (y, m, k, vz, ex, vi)

sumaCV (y, m, k, cv, vi2)

sumaDI (y, m, k, di, vi3)

sumaRV (vi, vi2, vi3, suvi)

ir_xy (15, 5) {coordenadas de pantalla para la escritura}

escribir ('REPORTE')

ir_xy (15, 6); escribir ('clientes con pasaporte venezolano', vz)

ir_xy (15, 7); escribir ('clientes con pasaporte extranjero', ex)

ir_xy (15, 8); escribir ('clientes con certif_vacuna vencido', cv)

ir_xy (15, 9); escribir ('clientes con dec_impuesto vencido', di)

ir_xy (15, 10); escribir ('clientes con recaudos vigentes', suvi)

escribir ('presione cualquier tecla')

leer_tecla

Fin_Reporte

PROCEDIMIENTO veriep (y, m, w: max; p: clien_T)

Comienzo

Si w≠0 **entonces**

```
            Reporte (y, m, p)
        Si_no escribir ('ingrese clientes por favor')
        Leer_tecla
Fin_veriep
```

PROCEDIMIENTO Listado (p: clien_T) {listado de todos los clientes y sus campos}

```
PROCEDIMIENTO Lpas (p) {número de pasaporte}
Variables S, q: max
Comienzo
        Desde S←1 hasta capcid hacer
          Comienzo
              q←s+2
              ir_xy (1, q); escribir (p[S].n_pas)
          Fin_desde
Fin_Lpas
PROCEDIMIENTO Lori (p) {lista origen de pasaporte}
Variables S, q: max
Comienzo
        Desde S←1 hasta capcid hacer
          Comienzo
              q←s+2
              ir_xy (13, q); escribir (p[S].orig)
          Fin_desde
Fin_Lori
PROCEDIMIENTO Lnom (p) {lista nombres_ apellidos}
Variables S, q: max
Comienzo
```

```
        Desde S←1 hasta capcid hacer
          Comienzo
                q←s+2
                ir_xy (17, q); escribir (p[S].nom_ap)
          Fin_desde
Fin_Lnom

PROCEDIMIENTO Lexp (p) {lista fecha exp_pasaporte}
Variables S, q, F: max

Comienzo
        1_Desde S←1 hasta capcid hacer
          Comienzo
                q←s+2
                ir_xy (39, q)
                2_desde F←1 hasta 3 hacer
                  comienzo
                        escribir (p[S].f_exp[F]); escribir (' ') {espacio dentro de fecha}
                  fin_2_desde
          Fin_1_desde
Fin_Lexp

PROCEDIMIENTO Lcva (p)
Variables S, q, F: max

Comienzo
        1_Desde S←1 hasta capcid hacer
          Comienzo
                q←s+2
                ir_xy (51, q)
```

```
            2_desde F←1 hasta 3 hacer
              comienzo
                    escribir (p[S].f_cva[F]); escribir (' ') {espacio dentro de fecha}
              fin_2_desde
        Fin_1_desde
Fin_Lcva

PROCEDIMIENTO Ldim (p)
Variables S, q, F: max

Comienzo
        1_Desde S←1 hasta capcid hacer
          Comienzo
              q←s+2
              ir_xy (63, q)
              2_desde F←1 hasta 3 hacer
                comienzo
                    escribir (p[S].f_dim[F]); escribir (' ') {espacio dentro de fecha}
                fin_2_desde
          Fin_1_desde
Fin_Ldim

Comienzo {Listado principal}
        Limpiar_pantalla
        Ir_xy (20,1); escribir ('LISTADO DE CLIENTES')
        Ir_xy (1,2); escribir ('pasaporte  ori   nombre_apellidos  f_exp  f_cva  f_dim')
        Lpas (p)
        Lori (p)
```

Lnom (p)

Lexp (p)

Lcva (p)

Ldim (p)

escribir (‘presione cualquier tecla’)

leer_tecla

Fin_Listado

PROCEDIMIENTO Verili (w: max; p: clien_T)
Comienzo

Si w≠0 **entonces**

Listado (p)

Si_no escribir (‘ingrese clientes por favor’)

Leer_tecla

Fin_verili

PROCEDIMIENTO Salir
Comienzo

Ir_xy (20,19)

escribir (‘presione cualquier tecla’)

leer_tecla

Fin_salir

PROCEDIMIENTO Menu2 (entradas y, m, d: max; variable w: max; k: clien_T)
Variable opción: op

Comienzo

Repetir

Limpiar_pantalla

Ir_xy (20,8); escribir (‘MENÚ ACTUALIZAR’)

```
            Ir_xy (20,9); escribir ('presione una de estas teclas')
            Ir_xy (20,10); escribir ('i. Ingresar clientes')
            Ir_xy (20,11); escribir ('m. Modificar algún cliente')
            Ir_xy (20,12); escribir ('e. Eliminar datos de cliente')
            Ir_xy (20,13); escribir ('l. Emitir Listado')
            Ir_xy (20,14); escribir ('s. Salir')
            Leer (opción)
            En_caso_de opción hacer
                  'i': Ingresa (y, m, d, w, k)
                  'm': verim (y, m, d, w, k)
                  'e': veriel (w, k)
                  'l': verili (w, k)
                  's': Salir
            Si_no
              Comienzo
                  ir_xy (20,18); escribir ('opción incorrecta'); leer_tecla
              fin
            Fin_En_caso_de
      Hasta_que opción='s'
Fin_Menu2
```

PROCEDIMIENTO MenuP

```
Variables opción: op; w, y, m, d, dow: max; k: clien_T
Comienzo
      w←0
      repetir
```

```
limpiar_pantalla
obtener_fecha (y, m, d, dow) {obtiene la fecha actual del sistema
                                operativo de la computadora}
ir_xy (10,4); escribir ('Buen día!, este es Servicios_Turismo
TOURSVEN, Bienvenidos')
ir_xy (10,5); escribir ('hoy es' días[dow],',', d, '/', mesi[m], '/', y)
ir_xy (20,8); escribir ('MENÚ PRINCIPAL')
ir_xy (20,9); escribir ('presione una de estas teclas')
ir_xy (20,10); escribir ('a. Actualizar clientes')
ir_xy (20,11); escribir ('o. Ordenar clientes')
ir_xy (20,12); escribir ('e. Emitir fechas No-Vigentes')
ir_xy (20,13); escribir ('r. Emitir Reporte estadístico')
ir_xy (20,14); escribir ('s. Salir')
leer (opción)
En_caso_de opción hacer
        'a'. Menu2 (y, m, d, w, k)
        'o'. verior (w, k)
        'e'. verino (y, m, w, k)
        'r'. veriep (y, m, w, k)
        's'. Salir
Si_no
  Comienzo
        ir_xy (20,18); escribir ('opción incorrecta'); leer_tecla
  fin
Fin_En_caso_de
```

hasta_que opción=’s’

Fin_MenuP

Comienzo {programa principal}

MenuP

Fin_Servicios_Turismo

OBJETIVO 7 PRUEBA DE ROBUSTEZ Y VERIFICACIÓN

Recordemos los conceptos de corrección, robustez, amigabilidad.

Un programa es ***correcto*** si, para cualquier entrada que satisfaga la precondición, termina generando una salida que satisface sus poscondiciones.

Se dice que un programa es ***robusto*** si reúne las dos condiciones siguientes:

- Es correcto.
- Para todas las entradas que no satisfacen las precondiciones, el programa termina y produce una salida que refleja el hecho de que ha ocurrido un error en la entrada.

Se dice que un programa es ***amigable*** si reúne los requisitos siguientes:

- Es correcto
- Para todas las entradas que no se ajusten a las precondiciones, el programa indica el tipo de error de entrada y concede al usuario la oportunidad de corregirlo y continuar.

Para verificar la corrección del programa, previamente se escribe en lenguaje PASCAL y con una tabla de datos que estén dentro de las precondiciones se comprueba las salidas previamente establecidas en las poscondiciones. En este caso, se demuestra que el programa es “correcto”.

En el enunciado del trabajo practico se muestra la necesidad de validar los datos de entrada, por lo tanto, en la elaboración del seudocódigo se toma en cuenta los criterios de validez.

El algoritmo demuestra ser correcto, debido a los rangos de validez numérica, es robusto, ya que detecta datos fuera del rango, y muestra una información al usuario que lo hace amigable.

Mediante el procedimiento *noms* se detecta si el dato comienza con un número y luego la cadena de caracteres o si es totalmente numérico, en vez de

insertar letras al principio de cada nombre y apellido, que puede llevar números en otra parte si es necesario. Se muestra un mensaje invitando a teclear el tipo de datos correcto.

Con *letras* se toma la precaución de avisar si están entrando datos de tipo cadena (de caracteres). Se muestra un mensaje invitando a teclear el tipo de datos correcto, en este caso, datos numéricos.

Para el número de pasaporte, se usa el procedimiento *valipas* para saber si está en los rangos correctos.

El procedimiento *vori* es el requerido para averiguar si el carácter que se ingresa es uno de los dos previamente establecidos: 'v' si el pasaporte es origen venezolano y 'e' si el pasaporte es de origen extranjero.

Para las fechas, primero se detecta si se ingresan cadenas de caracteres o números por medio del procedimiento *letr2*.

Siendo datos numéricos de fechas, es necesario si éstas se ajustan a los rangos usuales de fecha, detallados anteriormente. De todo esto se encarga el procedimiento *valife*, que detecta además si la fecha 29 de febrero es de un año bisiesto, si se inserta un 31 de junio, que no existe en el calendario, y otros por el estilo. Se muestra un mensaje invitando a teclear el tipo de datos correcto, en este caso, datos numéricos que se ajusten a los rangos y ajustados a la realidad.

Estas verificaciones se hacen con las opciones Ingresar y Modificar del submenú.

Cuando no se han ingresado clientes, es necesario avisar que se debe ingresar. Para esto están los procedimientos *verim, veriel, verili, verior, verino* y *veriep*, que revisan antes de ejecutar algún procedimiento del menú que requiera del arreglo de clientes activo.

Para la verificación de la corrección y robustez, se probó con la siguiente tabla de datos:

Apellidos Y nombres	# pasaporte	Origen de Pasaporte	Fecha Exp. Pas.	Fecha Certif. vac.	Fecha Dec. Imp.
Blanco Bitar Carlos	3411232	V	2010 07 01	2009 05 02	2009 07 11

Bonett Borges Franco	4825316	E	2009 11 01	2010 11 01	2010 10 15
Bosch Boso Noris	5126314	E	2010 02 02	2010 01 31	2010 02 02
Cabeza Luis	5342625	V	2010 02 03	2010 01 01	2010 03 06
Cabañas Cabrera Lucia	5857324	E	2009 05 02	2009 02 02	2010 05 08
Díaz Fernández Doris	Fgty 6302127	V	2009 02 25	2009 02 02	2010 05 25
Dorta Ramos Antonio	8410331	V	2010 04 05	2010 03 27	2010 04 05
Donelli Duque Jaime	8722110	V	2010 07 04	2009 04 05	2010 07 04
Egaña Egui Manuel	10328102	E	2008 02 29	2010 12 23	2010 04 15
Escobar España Adela	10331127	E	2010 04 21	2009 04 29	2010 04 21
Escobar Espinoza Carlos	10630505	V	2010 05 03	2010 06 30	2010 05 04
Espin Dorta Angel	10646222	V	2010 05 05	2010 08 31	2009 12 13
Fariñas García Emilio	10725550	E	2010 10 05	2010 11 11	2010 04 04
García Gómez Flor	10814327	V	2010 05 13	2009 09 09	2010 10 30
Garzón Ana	11114330	V	2010 06 02	2010 08 07	2009 11 13
Garrido Gastón Ramón	11208324	V	2009 10 03	2010 10 23	2010 06 01

Gómez Grandi Ana	11320101	V	2010 07 26	2010 12 05	2010 06 14
Gómez España Juan	11342333	V	2009 09 16	2009 11 30	2010 09 29
González Atilio	11634208	E	2010 06 31	2010 05 31	2010 06 11
González Díaz Dayana	11639420	E	2010 11 31	2009 08 25	2010 06 28
Granado Grandi Víctor	11640324	V	2010 02 30	2009 09 31	2009 02 28
Graterol Greco Oscar	11642314	V	2010 10 31	2010 10 30	2010 08 17
Hernández Egaña Alicia	11682214	V	2011 10 02	2009 10 04	2010 12 07
Hidalgo Higuera Elio	11821666	E	2009 13 02	2009 11 24	2009 08 15
Ibarra Bigott Luisa	11842105	E	2010 03 42	2010 03 19	2010 08 15
Iglesias Karam Roman	11901104	E	2010 04 31	2010 07 30	2009 08 28
La Riva Ledezma Andrés	12103125	V	2010 09 31	2010 12 18	2010 06 18
Largo Leal Gertrudis	12114324	V	2010 02 31	2010 01 26	2009 10 23
Méndez Núñez Elsi	12115001	V	1982 09 06	2010 03 27	2010 10 28
526	wakm	u	2009 0 1	201051	koli

Digitalizado por Osmer Montilla para www.admycontuna.multiply.com

OBJETIVO 8 PROGRAMA CODIFICADO

El programa está escrito en lenguaje TURBO PASCAL versión 7.0, y se entrega grabado en un CD contentivo de los archivos necesarios para su correcto funcionamiento.

BIBLIOGRAFÍA

Tucker, Allen y otros (2000). Fundamentos de Informática. Editorial Mc Graw Hill. Madrid.

Joyanes Aguilar, Luis (2000). Fundamentos de programación. Editorial Mc Graw Hill. Madrid.

Joyanes Aguilar, Luis (2005). Programación en Turbo/Borland Pascal 7. Editorial Mc Graw Hill. México.

Albornoz, Alejandro (2000). Trabajo práctico de computación 1. Caracas.