Universidad Abierta Interamericana

Campus Virtual UAIOnline

PROGRAMACIÓN ESTRUCTURADA

LECTURA

UNIDAD 6

ESTRUCTURAS DE DATOS

Autor de contenidos:
Nicolás Battaglia

OBJETIVOS

Esperamos que usted, a través del estudio de esta unidad, adquiera capacidad para:

- Almacenar datos.
- Elegir la estructura más conveniente para resolver la situación que el programa presente.
- Cargar esas estructuras en C.
- Manejar y operar los datos dentro de esas estructuras.
- Extraer datos de la estructura.

PRESENTACIÓN

Usted inicia el estudio de las últimas clases de la asignatura. Hemos realizado un largo recorrido y se va acercando la fecha de cierre y acreditación de los aprendizajes.

En esta unidad veremos las **estructuras de memoria** es decir, aquellas estructuras en las que se almacenan los datos durante la ejecución de un programa –recuerde que al cerrarse un programa los datos contenidos en el mismo se borran, se pierden.

En esta asignatura siempre utilizaremos estructuras de **tamaño fijo**, estructuras estáticas para cuyo uso siempre es necesario conocer, previamente, su tamaño.

Estas estructuras son las conocidas como **arrays o arreglos.** Los *arreglos* podrán ser de una dimensión, como el caso de los **vectores,** o multidimensionales como las **matrices**.

Es posible utilizar **arrays** dinámicos u otras estructuras para guardar en memoria datos y trabajar con ellas modificándolas, usándolas y hasta mejorando la perfomance del programa, pero su abordaje será contenido de otra asignatura.

Hay muchos principios que deberá considerar a la hora de trabajar con estas estructuras (vectores y matrices); uno de ellos es que los datos dentro de las mismas deben ser siempre del mismo tipo – esto por dogma de casi todos los lenguajes de programación

Para guardar datos utilizaremos a lo sumo matrices tridimensionales ya que de más dimensiones se torna peligroso y difícil su manejo y control.

A través de ejemplos y ejercicios le explicaremos las distintas operaciones que se pueden realizar con estas estructuras (obtención de máximos, mínimos, ordenamiento, etcétera)

Tipos de datos estructurados

Los dos tipos de datos estructurados más importantes son el array (vectores y matrices) y los registros.

En un arreglo las componentes son del mismo tipo y se accede a ellas mediante un número correlativo llamado subíndice.

En los registros en cambio los componentes pueden ser de diversos tipos, especificadas por distintos nombres, y a su conjunto se lo puede acceder mediante distintas técnicas que veremos más adelante.

Vectores

Los vectores son estructuras de memoria que poseen un tamaño conocido de ocurrencias y en general almacenan datos de un mismo tipo.

Dado un vector de nombre vect de 10 ocurrencias que almacena valores numéricos

12	45	56	78	15	23	56	15	1000	41
1	2	3	4	5	6	7	8	9	10

la forma de escribir un vector sería

vect [subíndice]

Donde el subíndice sería normalmente un número entero, representado por un dato, una variable o una constante.

El valor del subíndice puede ser desde 0, 1 o un valor arbitrario, eso dependerá del lenguaje a utilizar.

En el ejemplo anterior tenemos que

Si la posición o valor del subíndice es 6

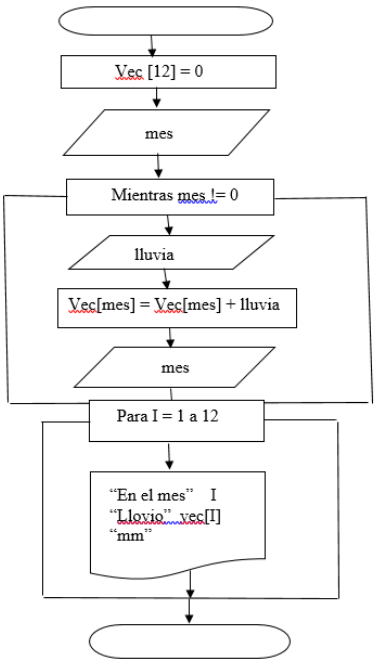
El valor guardado en el vector en esa posición es 23

Carga directa de un vector

Sea un lote de datos con los siguientes campos

mes	tipo numérico de 1 a 12
lluvia-caida	tipo numérico

Desea cargar un vector con dicha información.



Como vemos en este ejemplo se utilizó como subíndice de carga un dato del lote que era numérico, entero y conocido

En seudo código seria

Comienzo

Defino vec (12)

Ingresar “ingrese el mes y la cantidad de lluvia caida”

Ingresar mes,lluvia

Hacer hasta mes = 0

    Vec(mes)=vec(mes)+lluvia

    Ingresar “ingrese el mes y la cantidad de lluvia caida”

    Ingresar mes,lluvia

Repetir

Fin

En lenguaje *C* , DEV C++, seria:

```
//*********************************************************************
 SIN FUNCIONES    ( ACCESO DIRECTO SEGUN LA VARIABLE MES )
//*********************************************************************
```

```c
#include <stdio.h>
#include <conio.h>
#include<iostream>
#include<stdlib.h>

int main(void)
{
 int vec[12]={0};       // se inicializa todo el vector en cero
 int mes,lluvia,i;
 system("cls");

 do{                                          // conviene validar el dato
   printf("\nIngresar mes : ");              // cuando se ingresa directamente a
   scanf("%d",&mes);                         // una posicion de memoria, para
   }
while(mes < 0 ||  mes > 12);                 // NO invadir memoria NO asignada.

 while(mes)
 {
  printf("\nIngresar cantidad de lluvia caida : ");
  scanf("%d",&lluvia);

  vec[mes-1]+=lluvia;                    // es lo mismo que vec[mes-1]=vec[mes-1]+lluvia;

  do{
     printf("\nIngresar mes : ");
     scanf("%d",&mes);
     }while(mes < 0 ||  mes > 12);
  }
 system("cls");
 printf("\n TOTAL DE LLUVIA CAIDA SEGUN EL MES \n\n\n");

 for( i=0;i<12;i++)
   printf( "en el mes %2d llovio %6d mm de lluvia \n",i+1,vec[i]);

 printf("\n\n\n TIPEE UNA TECLA PARA FINALIZAR  " );
system("pause");
}
```

CON FUNCIONES (ACCESO DIRECTO SEGUN LA VARIABLE MES)

//**

#include <stdio.h>

#include<conio.h>

#include<iostream>

#include<stdlib.h>

void carga(int[]);

void suma(int[],int,int);

void informe(int[]);

```
int main()
{
int vec[13]={0};
system("cls");
carga(vec);
system("cls");
informe(vec);
}
void carga(int x[13])
{
	int mes, lluvia;
	do
		{
			printf("ingrese el mes");
			scanf("%d",&mes);
		}while(mes<0||mes>12);

	while(mes!=0)
		{
			printf("ingrese cantidad de lluvia caida");
			scanf("%d",&lluvia);
			suma(x,mes,lluvia);
			do
			{
			printf("ingrese el mes");
			scanf("%d",&mes);
			}while(mes<0||mes>12);
		}
}
```

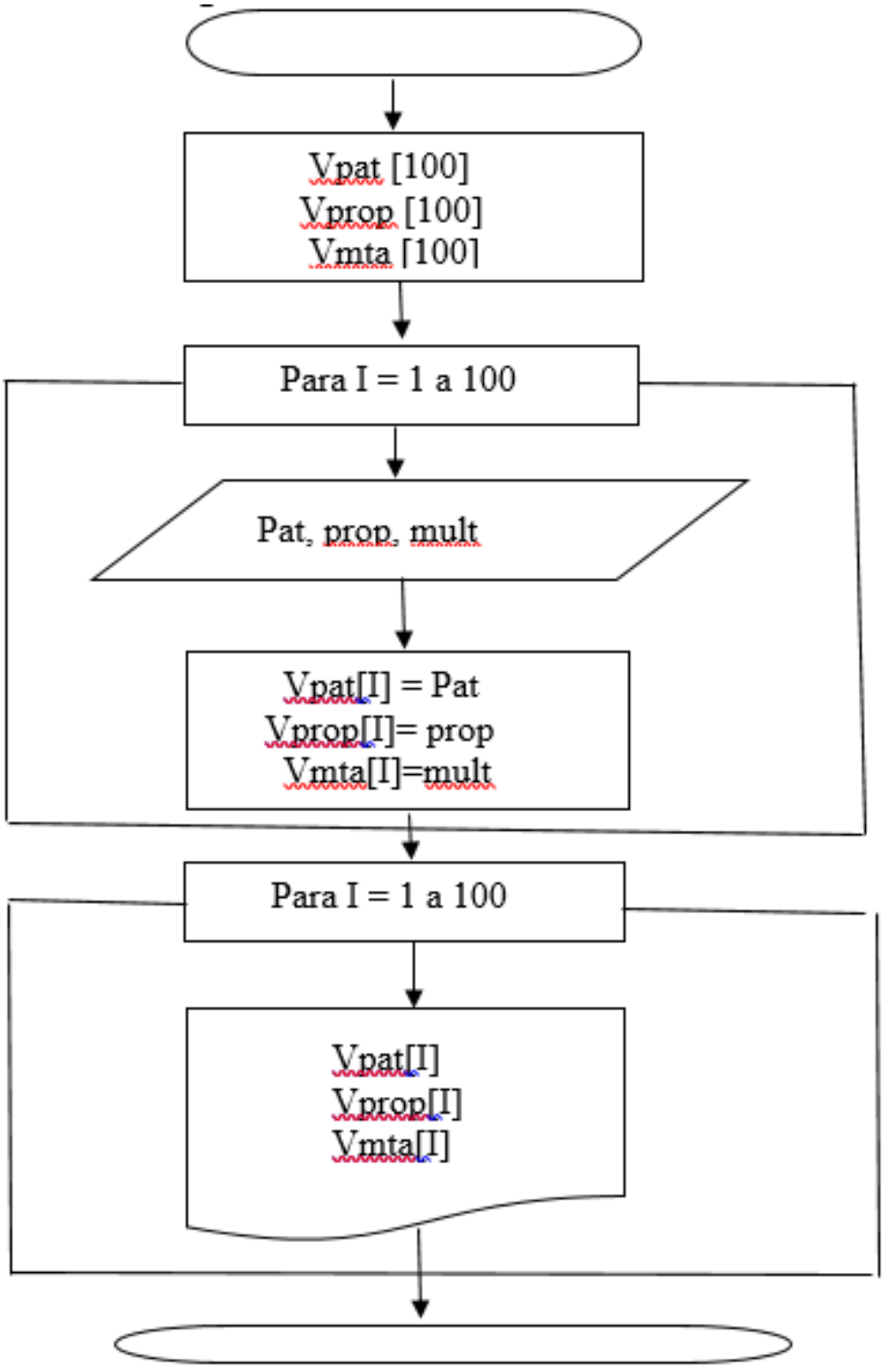
```
void suma(int x[13],int y, int z)
{
        x[y]= x[y]+ z;
}


void informe(int x[13])
{
        int i;
        for (i=1;i<=12;i++)
                {
                        printf("en el mes %d llovio %d mm\n",i,x[i]);
                }
}
```

Carga indirecta de un vector

Sea un archivo secuencial con los siguientes campos

Patente	alfanumérico
Propietario	alfanumérico
Multa	numérico

Y se sabe que se confeccionaron 100 multas en el día, si deseamos cargar este archivo en vectores, vemos que no podemos utilizar ninguno de sus campos como subíndice, por lo que deberemos hacer lo siguiente:


Vpat [100]
Vprop [100]
Vmta [100]
Para I = 1 a 100
Pat, prop, mult
Vpat[I] = Pat
Vprop[I]= prop
Vmta[I]=mult
Para I = 1 a 100
Vpat[I]
Vprop[I]
Vmta[I]

En lenguaje *C*, DEV C++, seria:

//**

SIN FUNCIONES (ACCESO SECUENCIAL)

//**

```c
#include <stdio.h>
#include <conio.h>
#include<iostream>
#include<stdlib.h>

#define N 3

int main()
{
 char Vpat[N][7], Vprop[N][31];
 float Vmta[N];
 int i;
 system("cls");
 for(i=0;i<N;i++)
 {
  printf("\nIngresar Patente : ");
  scanf("%s",&Vpat[i]);
  fflush(stdin);              // se limpia el buffer por el <ENTER>
  printf("\nIngresar Propietario : ");
  scanf("%[^\n]",&Vprop[i]);       // cuando se ejecuta se leer  una cadena de

  printf("\nIngresar la Multa :"); // caracteres de longitud no determinada,
  scanf("%f",&Vmta[i]);            // no mas de 79 caracteres
 }
 system("cls");
 printf("\n\n LOS DATOS INGRESADOS SON ");
 printf ( "\n     PATENTE     PROPIETARIO                    MULTA   \n");

 for(i=0;i<N;i++)
  printf("\n      %-6s     %-30s  %8.2f", Vpat[i],Vprop[i],Vmta[i]);  // no mas de 79 caracteres

 printf("\n\n\n TIPEE UNA TECLA PARA FINALIZAR  " );
 system("pause");
}
```

//**

CON FUNCIONES (ACCESO SECUENCIAL)

//**

```c
#include <stdio.h>
#include <conio.h>
#include<stdlib.h>
#define N 3

////////////////////// DECLARACIONES O PROTOTIPOS DE FUNCIONES
void carga_datos(char [][7],char[][31],float[]);
void informe (char [][7],char[][31],float[]);
void FIN (void);

int main()
{
 char Vpat[N][7], Vprop[N][31];
 float Vmta[N];
 system("cls");
 carga_datos(Vpat,Vprop,Vmta);
system("cls");
 informe(Vpat,Vprop,Vmta);
 FIN();
}

////////////////////// DEFINICIONES DE FUNCIONES

void carga_datos(char x[][7],char y[][31],float z[])
{
 int i;
 for(i=0;i<N;i++)
 {
  printf("\nIngresar Patente : ");
  scanf("%s",&x[i]);
  fflush(stdin);
  printf("\nIngresar Propietario : ");
  scanf("%[^\n]",&y[i]);
  printf("\nIngresar la Multa :");
  scanf("%f",&z[i]);
 }
}
```

```
38
39  ////////////////////////////////////////////////////
40  void informe (char x[][7],char y[][31],float z[])
41  {
42   int i;
43   printf("\n\n LOS DATOS INGRESADOS SON ");
44   printf ( "\n     PATENTE     PROPIETARIO                    MULTA   \n");
45
46   for(i=0;i<N;i++)
47    printf("\n      %-6s     %-30s  %8.2f", x[i],y[i],z[i]);
48  }
49
50  ////////////////////////////////////////////////////
51  void FIN(void)
52  {
53    printf("\n\n\n TIPEE UNA TECLA PARA FINALIZAR  " );
54    getch();
55  }
```

Matrices

Las matrices son arrays de dos o más dimensiones según soporte el lenguaje a utilizar.

Para nuestro estudio utilizaremos matrices bidimensionales

Mantienen básicamente las mismas características en lo que concierne al tipo de subíndices a utilizar, su representación y su contenido.

En lugar de manejar un solo subíndice manejaremos dos que nos representarán las filas y columnas de una matriz

45	65	12	58	47	69
15	48	59	23	52	42
78	84	9	120	451	784
65	85	53	42	73	95

En este ejemplo tenemos que la matriz llamada MATSDO tiene las siguientes características

Tiene 4 filas y 6 columnas MATSDO (4,6) MATSDO[4][6] la definición

Depende del lenguaje

Si la fila vale 2 y la columna 5 el valor de MATSDO (2, 5) es 52, esto también depende del lenguaje, este resultado sería en COBOL que comienzan las filas y columnas en 1, en C seria 784 pues comienzan las filas y columnas en 0

Sus formas de carga y de posible búsqueda de un dato en ella es similar a la de los arrays o vectores unidimensionales, pudiendo combinarse en sus subíndices una constante y un dato, una constante y una variable, dos datos, etc.

Vectores asociados a una matriz

Supongamos que me dan los siguientes lotes de datos

Lote depósito
codDep alfanumérico

Nombre alfanumérico

Lote artículo
Codart alfanumérico

Descripción alfanumérico

Me dicen que son 1000 artículos y 10 depósitos

Luego me dan el lote de datos del stock

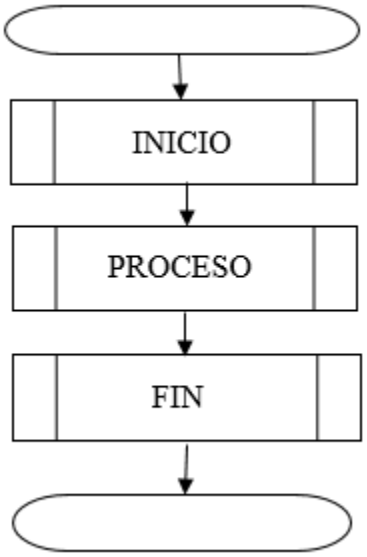
Deposito alfanumérico

Articulo alfanumérico

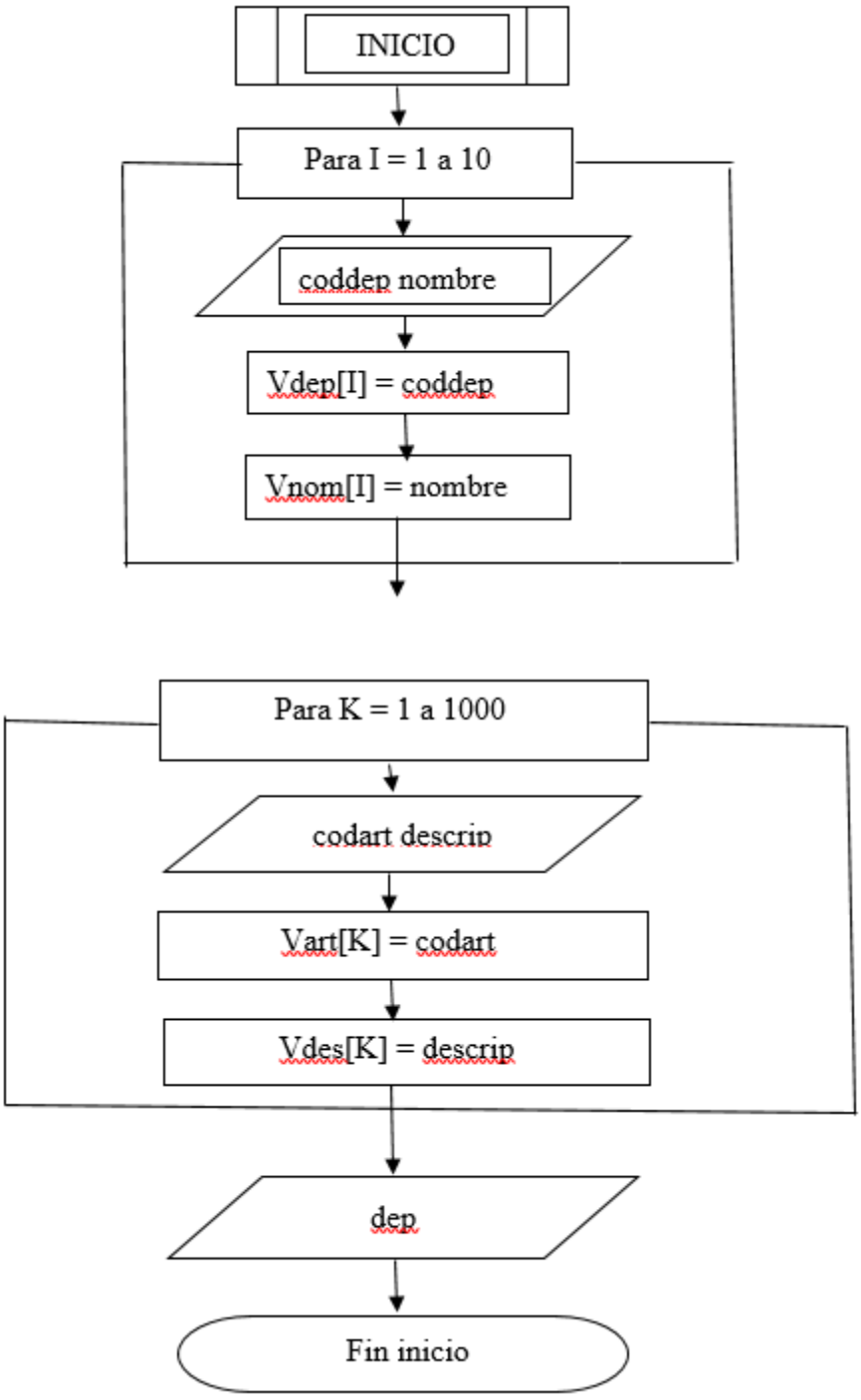
Stock numérico

Pues bien lo primero que deberé hacer es cargar los archivos depósito y artículo en sendos vectores, uno de 10 posiciones y el otro de 1000 posiciones

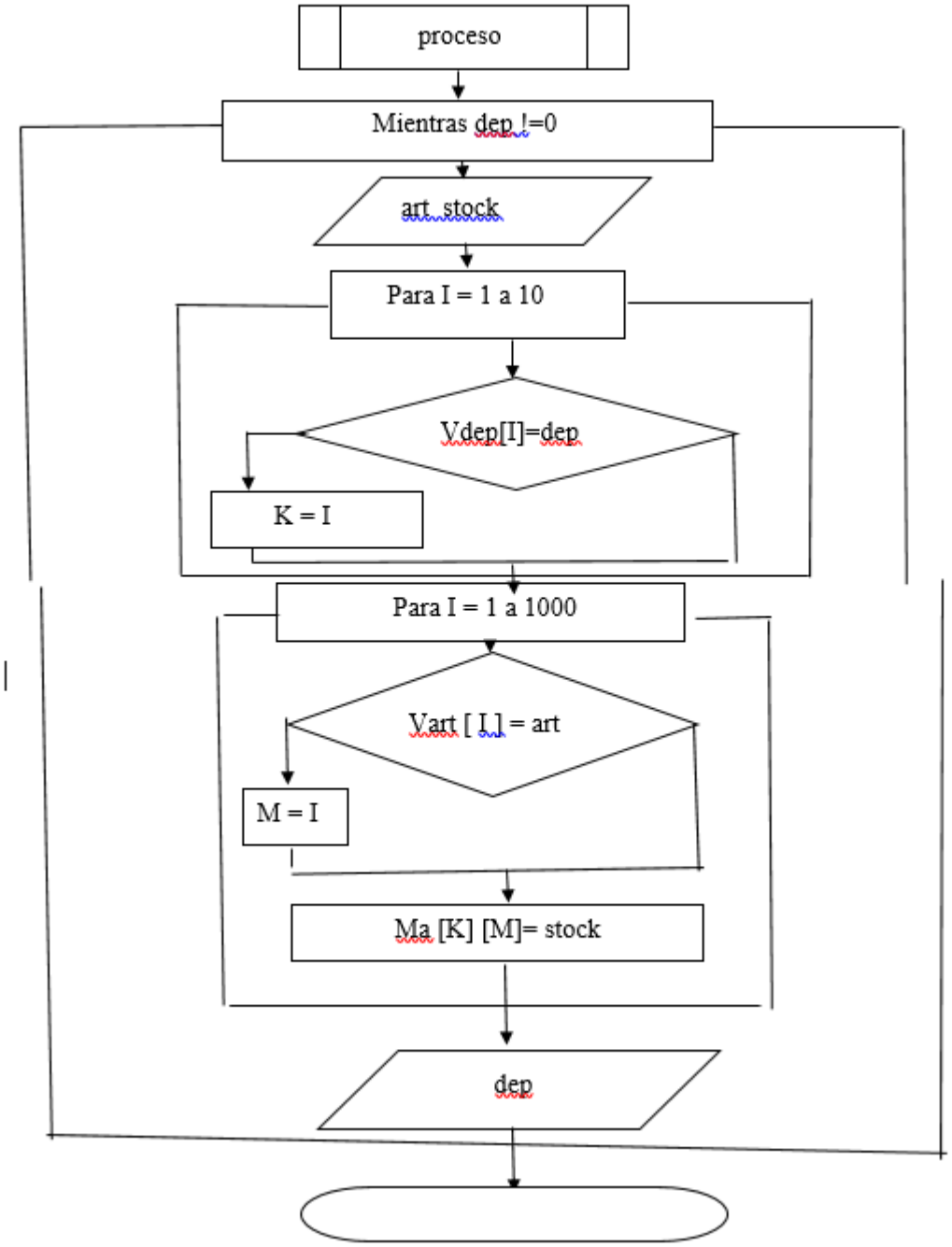
Vamos a dividiré el programa en 3 partes a fin de poder manejarlo mejor



Dentro de inicio


INICIO
Para I = 1 a 10
codden nombre
Vdep[I] = codden
Vnom[I] = nombre
Para K = 1 a 1000
codart descrip
Vart[K] = codart
Vdes[K] = descrip
dep
Fin inicio

Ahora bien para poder cargar el archivo de stock debo hacer que exista una relación entre las posiciones donde guarde los depósitos y los artículos en los vectores y la matriz que voy utilizar, es por ello que deberé obtener los subíndices de la matriz de estos vectores asociados, para ello en el proceso tendré que hacer lo siguiente:



En seudo codigo seria

```
Comienzo
Para I =1 a 10
        Ingresar “ingrese el código de deposito”
        Ingresar coddep
        Ingresar”ingrese el nombre del deposito”
        Ingresar nombre
        Vdep (I)=coddep
        Vnom (I)=nombre
Próximo
Para k = 1 a 1000
        Ingresar “ingrese el código del articulo”
        Ingresar codart
        Ingresar “ingrese la descripción del articulo”


        Ingresar descripción
        Vart (K)=codart
        Vdes (K)=descripción
Próximo
Ingresar “ingrese el deposito”
Ingresar dep
Hacer mientras dep != 0
        Ingresar art
        Ingresar stock
        Para I =1 a 10
```

```
            Si Vdep (I)=dep entonces
                K=I
            Fin si

        Próximo
        Para I=1 a 1000
            Si Vart(I) = art entonces
                M=I
        Fin si
        Próximo
        Ma(K,M) = stock
        Ingresar “ingrese el deposito”
        Ingresar dep
Repetir
Fin
```