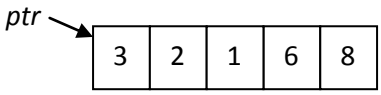
Apuntes de Arreglos Dinámicos: enteros, estructuras y estructuras dinámicas (metadatos)

1.- Arreglo dinámico de enteros

```
int *ptr;
printf("Teclea el tamaño del arreglo");
scanf("%d",&n);
ptr=(int*)malloc(sizeof(int)*n);
```

Si n=5, tenemos un arreglo que permite guardar hasta 5 valores enteros, por ejemplo:



Lo interesante aquí es que si comparamos un arreglo dinámico de este tipo comparado con un arreglo tradicional (p.ej. int a[5]), es que el arreglo dinámico representado por *ptr*, pude crecer en tiempo de ejecución, de ahí el nombre de arreglo dinámico.

Una función para capturar los datos del arreglo dinámico puede ser de la siguiente forma:

```
/* o es el número de elementos o tamaño del arreglo */
void captura(int *dato, int o)
{
    int a;
    for (a=0;a<o;a++)
    {
       printf("teclea el valor %d",a+1);
       scanf("%d",(dato+a));
    }
}
```

2.- Arreglo dinámico de estructuras

```
typedef struct{
      char titulo[80];
      char autor[80];
      int anio;
      int duracion;
}disco;

   disco *cd;
   int n;

   printf("Dame el tamaño del arreglo:  ");
   scanf("%d", &n);
```

```
cd=(disco*)malloc(sizeof(disco)*n);
```

En este caso tenemos un arreglo dinámico de estructuras, es decir podemos guardar la información de n discos (n=3):

cd →

Para Siempre Vicente Fernández 2008 40min	Incondicional Luis Miguel 1995 50min	Like a Virgin Madona 1998 70min

Una función para capturar los elementos del arreglo dinámico de estructuras sería de la siguiente forma:

```
void captura(disco *cd, int n)
{
    int i;
    for(i=0;i<n;i++)
     {
          printf("\n\n*****Datos del disco # %d*****\n\n", i+1);
          printf("\nDame el nombre del disco: ");
          fflush(stdin);
          gets((cd+i)->titulo);
          printf("\nDame el nombre del autor:");
          fflush(stdin);
          gets((cd+i)->autor);
          printf("\nDame el año de lanzamiento:");
          fflush(stdin);
          scanf("%d",&(cd+i)->anio);
          printf("\nDame la duración del disco(segundos): ");
          fflush(stdin);
          scanf("%d",&(cd+i)->duracion);
     }
}
```

3.- Arreglos dinámicos de estructuras dinámicas (metadatos)

Ahora considerar que se quiere una estructura que permita cambiar el número de campos (también conocidos como atributos) en tiempo de ejecución, es decir que en vez de los campos: titulo, autor, anio y duracion se quieren otros al momento de ejecutar el programa. En este caso requerimos una estructura de datos que permita manejar un conjunto de campos de una estructura, que aquí llamamos “Diccionario” que utiliza la estructura “MetaDato”:

```
typedef struct
{
   char nombre[20];   //nombre del atributo
   int tipo;          //tipo del atributo
   int tam;           //tamaño del atributo en bytes
   char clave;        //si forma parte de la clave o no.
}MetaDato;
```

```
typedef struct
{
  int numDatos;                 //No. de atributos
  MetaDato * metaDatos;     //Apuntador al arreglo "dinámico" de metadatos
}Diccionario;
```

La estructura Diccionario permite crear una estructura dinámica (metadatos). Estrictamente hablando la estructura Diccionario lo que tiene es un arreglo dinámico de estructuras de tipo Metadato y el número de elementos de dicho arreglo.

Una vez creado el Diccionario, ahora necesitamos capturar los datos de la estructura dinámica que representa el Diccionario, esto es, crear un arreglo dinámico de estructuras dinámicas. Para esto utilizamos la siguiente estructura:

```
typedef struct
{
  int numDatos;        //No de registros
  void *datos:         //apuntador al arreglo "dinámico" de datos
  int tam;             //tamaño del registro
}Datos;
```

Ahora la función de captura de los datos se puede realizar de la siguiente forma:

```
void capturaDatos(Diccionario d, Datos *dat)
{  int i;
  void * dir;
    dat->numDatos = 5;
    printf("\nNo de datos a capturar: "); // %d\n",dat->numDatos);
    scanf("%d",&dat->numDatos);
    dat->tam = tamRegistro(d);   /* ver abajo la función */
    dat->datos = malloc(dat->tam * dat->numDatos);
    dir = dat->datos;
    for(i=0; i<dat->numDatos; i++)
    {
      capturaRegistro(d, dir);
      dir = (char *)dir + dat->tam;
    }
}
```

Primero requerimos el Diccionario por que es quién define la estructura dinámica, y un apuntador a Datos para la información que queremos capturar. Como no sabemos a priori el tipo de datos para cada elemento del Diccionario, es necesario un apuntador genérico (void*).

```
int tamRegistro(Diccionario d)
{
   int i, tam = 0;
    for(i = 0; i< d.numDatos; i++)
       tam+=d.metaDatos[i].tam;
   return(tam);
}
```

La función tamRegistro es importante para poder determinar el tamaño de la estructura dinámica y así poder desplazarse entre elemento y elemento.

```
void capturaRegistro(Diccionario d, void *dir)
{  int i;
    for(i = 0; i<d.numDatos; i++)
    {    printf("%s: ", d.metaDatos[i].nombre);
         switch(d.metaDatos[i].tipo)
         {
             case 0: //char
                 scanf("%c", (char *) dir);
                 break;
             case 1: //int
                 scanf("%d",(int *) dir);
                 break;
             case 2: // float
                 scanf("%f",(float*)dir);
                 break;
             case 3: //string
                 scanf("%s", (char *)dir);
         }
         dir = (char *)dir + d.metaDatos[i].tam;
    }
}
```

Notar que en capturaRegistro el scanf se hace dependiendo del tipo de datos dentro de la estructura dinámica representada por Diccionario. También cabe resaltar que “dir” es un apuntador genérico y el avance del apuntador al siguiente elemento dentro de la estructura dinámica se hace a través de d.metaDatos[i].tam