ARREGLOS DE ESTRUCTURAS

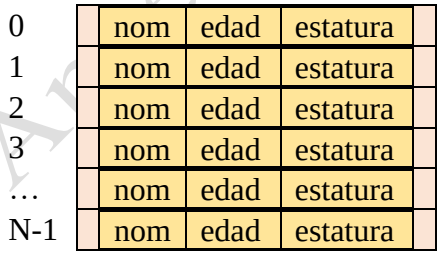
Así como podemos necesitar un conjunto de datos simples (enteros, flotantes, etc.) también se llega a requerir un conjunto de estructuras como un grupo de personas, de alumnos, de productos, etc., de hecho, es más común en las aplicaciones actuales el uso de conjuntos de estructuras que de tipos simples, debido a esto la relación entre arreglos y estructuras es muy natural, solo nos queda conocer la sintaxis para su uso.

* Declaración de arreglos de estructuras

Partiendo de que ya tenemos la estructura persona (PER) un arreglo unidimensional de personas se declararía así:

```
PER arrPer[N];
```

Esquemáticamente lo podemos visualizar así:



Como se puede ver en el esquema se obtiene un conjunto de datos donde cada uno de ellos contiene los elementos de una persona, se conservan las características del arreglo es decir cada dato-persona está indizado(referenciado por una posición).

Como parámetro

```
____ funX( PER arrP[], int n, …)
```

En el uso de arreglos de estructuras como parámetros se conserva la característica de que siempre *pasan por referencia*.

* Acceso a los datos de un arreglo de estructuras

No hay que perder de vista que es un **arreglo** de *estructuras*, por lo tanto "*manda*" primero la notación de arreglo (arr[…]) y luego la de estructura (.campo), juntándolas se obtiene:

arrEstructuras[pos] . campo

Por ejemplo, para el arreglo de personas:

arrPer[pos].nom
arrPer[pos].edad
arrPer[pos].estatura

Con base en la estructura *persona* que hemos estado usando, vamos a escribir algunas *funciones* para manejar un arreglo de personas.

a) Función para capturar los datos de un arreglo con n personas.

```
void capturaArrPer(PER arrPer[], int n)
{
 int i;

 for(i=0; i<n; i++)
 {
   printf("No. %d\n", i);
   printf("Nombre: ");
   gets(arrPer[i].nom);
   printf("Edad: ");
   scanf("%d", &arrPer[i].edad);
   printf("Estatura: ");
   scanf("%f", &arrPer[i].estatura);
 }
}
```

b) Función para mostrar los datos del arreglo de n personas.

```
void muestraArrPer(PER arrPer[], int n)
{
  int i;

  printf("No.\tNombre\t\tEdad\tEstatura\n");
  for(i=0; i<n; i++)
    printf("%d\t%s\t\t%d\t%.2f\n", i, arrPer[i].nom, arrPer[i].edad, arrPer[i].estatura);
}
```

c) Función para calcular el promedio de estatura.

```
float promEstatura(PER arrPer[], int n)
{
   int i;
   float suma=0, prom;

   for(i=0; i<n; i++)      //Sumar
     suma += arrPer[i].estatura;

   prom = suma / n;

   return(prom);
}
```

d) Función para contar el No. de mayores y menores de edad.

```
void cuentaMayMen(PER arrPer[], int n, int *cMay, int *cMen)
{
  int i;

  *cMay = *Men = 0;
   for(i=0; i<n; i++)
     if(arrPer[i].edad >17)
          (*cMay)++;      // *cMay +=1;     o       *cMay = *cMay + 1;
     else
          (*cMen)++;
}
```

e) Encontrar (“entregar”) a la persona(nombre) con mayor estatura. (Algoritmo del mayor/menor)

supuestoMayor=10
nomMayor =”Carlos”
6 (Luis)
9 (María)
4
7
10 (Carlos)
5

```
void encuentraMasAlto(PER arrPer[], int n, char nomMayor[])
{
  int i;
  float sMayor;

  sMayor=arrPer[0].estatura;
  strcpy(nomMayor, arrPer[0].nom);
  for(i=1; i<n; i++)
    if(arrPer[i].estatura > sMayor)
    {
      sMayor=arrPer[i].estatura;
      strcpy(nomMayor, arrPer[i].nom);
     }
}
```

f) Escriba una función para calcular el ***No. de personas*** que hay en cada rango de edad (en bloques de 10 años).

0-9
10-19
20-29
30-39
40-49
50-59
. . .
90 . . .

```
void cuentaRangos(PER arrPer[], int n, int arrCont[])
{
  int i;

  for(i=0; i<10; i++)
     arrCont[i]=0;

  for(i=0; i<n; i++)
    if(arrPer[i].edad < 10)                  // 0 . .  9
       arrCont[0]++;
    else
      if(arrPer[i].edad < 20)                // 10 . .  19
        arrCont[1]++;
      else
        if(arrPer[i].edad < 30)              // 20 . .  29
          arrCont[2]++;
        else
          if(arrPer[i].edad < 40)            // 30 . .  39
            arrCont[3]++;
          else
              . . .                          //Agregar los rangos que faltan
            else                             // 90 . . .
              arrCont[9]++;
}
```

El ciclo podría simplificarse como:

```
  for(i=0; i<n; i++)
    if(arrPer[i].edad < 90)
      arrCont[arrPer[i].edad/10]++;
    else
      arrCont[9]++;
```