TIPOS DE ARREGLOS

- Explicación Práctica -

TALLER DE LENGUAJES 1

Arreglos en C

- ANSI C90 define 3 tipos de arreglos de acuerdo a su *clase de almacenamiento*.
- Una clase de almacenamiento define el alcance (visibilidad) y el tiempo de vida de una variable dentro de un programa.
  - El alcance de una variable indica donde puede ser referenciada.
  - El tiempo de vida indica el tiempo en que una variable "existe".

Alcance y tiempo de vida

```
#include <stdio.h>
#include <stdlib.h>

int a;

void f();

int main () {
        int b;
        f();
        return 0;
}

void f(){
        int c;
        static int d;
        …
}
```

a es una variable global. Su alcance y tiempo de vida es todo el programa.

b es una variable local a la función *main*. Su alcance y tiempo de vida es la función *main*.

c es una variable local a la función *f*. Su alcance y su tiempo de vida es la función *f*.

d es una variable estática a la función *f*. Su alcance es la función *f* y su tiempo de vida es todo el programa.

Arreglos en C

ANSI C90 define 3 tipos de arreglos de acuerdo a su *clase de almacenamiento*:

- Arreglos automáticos
  - Ej. `int a[10]`
- Arreglos estáticos
  - Ej. `static int a[10]`
- Arreglos dinámicos
  - Ej. `int * a = (int *) malloc(10*sizeof(int));`

Arreglos en C

ANSI C99 incorpora un nuevo tipo de arreglo:

- Arreglos de longitud variable
  - Ej.

```
int n;
scanf("%d",&n);
int a[n];
```

  - Este tipo de arreglos se declara en forma similar a los arreglos automáticos, pero la longitud no es una expresión constante (no se puede determinar en tiempo de compilación).
  - La memoria para el arreglo se aloca cuando el programa alcanza la declaración y se desaloca cuando el bloque de código donde se encuentra termina.

Arreglos de longitud variable

Similitudes con arreglos automáticos

- Se declaran en forma similar.
- Una vez asignada la memoria, no pueden cambiar su tamaño.
- El programador no puede liberar la memoria explícitamente.

Diferencias con arreglos automáticos

- El tamaño de los arreglos automáticos debe ser una expresión constante (se debe poder evaluar en tiempo de compilación). En los arreglos de longitud variable este cálculo se demora hasta que la ejecución alcance la declaración del arreglo.

Arreglos de longitud variable

Similitudes con arreglos estáticos

- No pueden cambiar su tamaño en ejecución.
- El programador no puede liberar la memoria explícitamente.

Diferencias con arreglos estáticos

- El tamaño de los arreglos estáticos debe ser una expresión constante (se debe poder evaluar en tiempo de compilación). En los arreglos de longitud variable este cálculo se demora hasta que la ejecución alcance la declaración del arreglo.
- La memoria para los arreglos estáticos se reserva antes de que comience el programa y se libera cuando el mismo termina. La memoria para los arreglos de longitud variable se reserva cuando se alcanza la declaración y se libera cuando se desactiva el bloque de código donde se encuentra.

Arreglos de longitud variable

Similitudes con arreglos dinámicos

- No es necesario que el tamaño se conozca en tiempo de compilación.

Diferencias con arreglos dinámicos

- Los arreglos dinámicos sí pueden cambiar su tamaño en ejecución.
- El programador sí puede liberar explícitamente la memoria.
- Los arreglos dinámicos se alocan en la *heap*, mientras que todos los demás lo hacen en la pila de ejecución (*stack*). Por ese motivo, los arreglos dinámicos pueden tener tamaños mayores.

Ejercicio

Escriba un programa que lea desde teclado un número entero *n* y a continuación lea otros *n* números enteros. Almacene los *n* números en un arreglo de longitud variable y calcule su promedio.

Solución

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int i, sum=0, n;
    scanf("%d",&n);
    int a[n];
    for (i=0; i<n; i++)
        scanf("%d",a+i);
    for (i=0; i<n; i++)
        sum += a[i];
    printf("\nEl promedio es %f",(float)sum/n);
    return 0;
}
```