Estructura

Informática II

Fecha de revisión:

Autor

Tipos de datos

Se tienen tipo de datos simples y compuestos:

Simples

- Números
- Letras
- Verdadero/falso

Compuestos (Estructurados): son combinaciones de tipos simples

- String
- Arreglos
- **Estructura**

Tipos de datos definidos por el usuario

Al momento de hacer un programa, **el usuario puede definir** sus propios tipos de datos, permitiendo

- Mayor claridad
- Aumentar el significado semántico del código
- Simplificar declaración de variables

Tipos de datos definidos por el usuario

Typedef

- Define un nuevo nombre para un tipo de dato.
- El nombre original sigue siendo válido.

typedef <tipo> <nuevo nombre>;

typedef int positivo;

Tipos de datos definidos por el usuario

```
typedef int positivo;
typedef int negativo;

int main(){
        positivo a,b;
        negativo c,d;
        a=1;
        b=2;
        c=-a;
        d=-b;
        printf("%d %d %d %d\n",a,b,c,d);
}
```

Tipos de datos definidos por el usuario

Otra forma de definir tipos de datos es componer varios datos simples en uno solo. Esto se denomina **estructura.**

Una estructura es un tipo de dato que contiene un conjunto de valores relacionados entre sí de forma lógica.

Estos valores pueden ser de distinto tipo.

Generalmente, se refiere a un concepto más complejo que un número o una letra.

Estructuras

Una estructura puede verse como una colección de variables que se referencia bajo un nombre en común.

Cada una de estas variables se denominan “miembros” de la estructura. Otras denominaciones son:

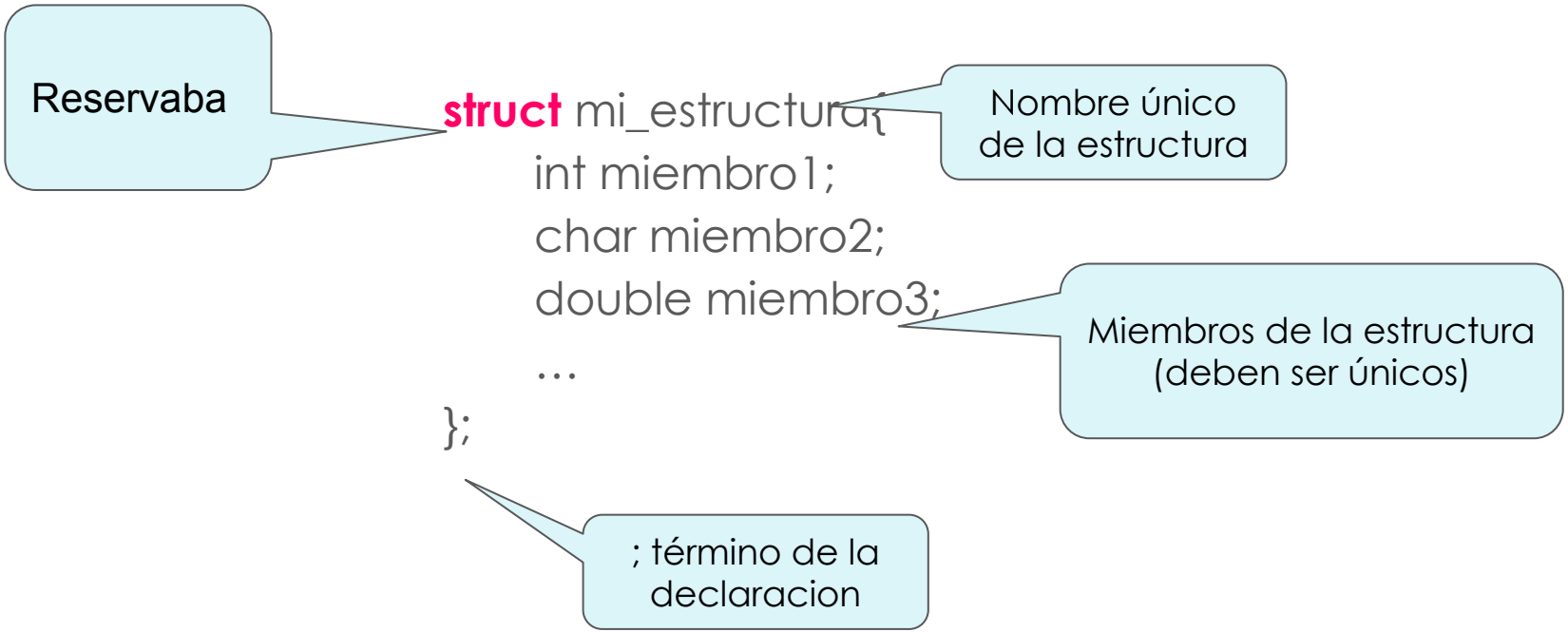
- Campo
- Elemento
- Atributo

Declaración de estructuras

La definición de una estructura se realiza fuera de cualquier función, generalmente en la parte superior del archivo. Para definir una estructura requerimos:

- Un nombre
- Una lista de miembros
- Nombre
- Tipo

Declaración de estructuras



Declaración de estructuras

La declaración de una estructura no crea variables.

Solo se define el nombre y sus miembros.

Debe estar definida para poder ser utilizada (posición en el código).

Uso de estructuras

Una vez que se ha declarado la estructura puede ser utilizada.

Para utilizarla hay que definir variables del tipo “estructura”.

Para definir estas variables se utiliza la siguiente sintaxis:

```
struct nombre_estructura nombre_variable;
```

Uso de estructuras

```
struct mi_estructura{
        int miembro1;
        char miembro2;
        double miembro3;
};
…

struct mi_estructura1;
struct mi_estructura2;
```

Dos variables del tipo mi_estructura

Operaciones con estructuras

Una vez definidas las variables, es necesario realizar operaciones con ellas.

Lo realmente útil no es la estructura, sino sus miembros.

Para acceder a los valores de los miembros de una variable de tipo estructura se utiliza el operados unario “.”.

Cada miembro es una variable común y corriente.

Operaciones con estructuras

```
struct mi_estructura{
        int miembro1;
        char miembro2;
        double miembro3;
};
…
struct mi_estructura1;
mi_estructura1.miembro1=1024;
mi_estructura1.miembro2='x';
mi_estructura1.miembro3=12.8;
```

Operaciones con estructuras

```
struct mi_estructura1;

Printf("m1=%d, m2=%d, m3=%d\n",
        mi_estructura1.miembro1=1024,
        mi_estructura1.miembro2='x',
        mi_estructura1.miembro3=12.8);
```

Ejemplo

Con tipos de datos simples

```
String nombreAlumno;
int edadAlumno;
float promedioAlumno;
```

Con estructuras

```
struct alumno{
        String nombre;
        int edad;
        float promedio;
};
```

Arreglos de estructuras

Se puede crear arreglos cuyos elementos sean variables de estructura.

Se definen de manera similar al caso común.

tipo arreglo[N]

struct estructura arreglo[N];

Arreglos de estructuras

```c
#include <stdio.h>
#define N 5

struct alumno{
      int rol;
      int promedio;
};

int main(){
      int i=0, suma = 0;
      struct alumno lista_alumno[N];
      double promedio=0;
```

Arreglos de estructuras

```
    for(i = 0; i < N; i++){
        printf("Ingrese rol y nota: ");
        scanf("%d %d", &lista_alumno[i].rol, &lista_alumno[i].promedio);
    }
    for(i = 0; i < N; i++){
        suma += lista_alumno[i].promedio;
    }
    promedio = (1.0 * suma) / N;
    printf("Promedio del curso: %.1f\n", promedio);
    return 0;
}
```