Estructuras

- Una estructura, conocida en otros lenguajes como registro, permite agrupar en una unidad un conjunto de datos heterogéneos relacionados.
- Para poder declarar una variable de tipo estructura primero debemos describir el “molde” o “patrón” de la estructura y luego declarar la variable de este “tipo de estructura”

```
struct Empleado {//declaro el tipo
        int legajo;
        string nombre;
};

Empleado e1, e2; //defino variables
```

Declaración y definición

- Puedo declarar la estructura y definir las variables en un solo paso

```
struct Punto {
        double x;
        double y;
} pto1, pto2;
```

- Solo en C++11 y siguientes (no en C) se pueden indicar valores iniciales al declarar la estructura

```
struct Rectangulo {
        float ancho = 1.5;
        float largo;
};
Rectangulo r; //el campo ancho ya está en 1.5
```

Inicialización

- Puedo incializar valores al declarar la variable

```
Punto pto3 = { 1.0, 2.0};
Empleado e3 = { 525, "Pablo"};
```

- Puedo hacer todo junto (declarar e inicializar)

```
struct Ejemplo {
        int a;
        int b;
} vejemplo = {1, 5};
```

Uso

- Para acceder a un miembro de una estructura uso el operador . (punto) si lo que tengo es una variable

```
e1.legajo = 123;
e1.nombre = "Juan";
```

- Puedo asignar para copiar TODOS los campos

```
e2 = e1;
```

- Puedo asignar valores de modo similar a la incialización:

```
e2 = { 712, "Natxo"};
```

- Puedo pasar una estructura como parámetro o devolverla como resultado de una función

Punteros

- Si declaro un puntero a una estructura y quiero acceder a un campo, por tema de precedencias debo usar paréntesis

```
Punto *p;
p = &pto1;
(*p).x = 5.0;
```

- Como esto es engorroso se definió el operador `->` que es lo que se acostumbra utilizar.

```
p->x = 5.0; //equivalente a (*p).x = 5.0;
```

Punteros y autoreferencias

```
struct Empleado {
        int legajo;
        string nombre;
        Empleado *siguiente = nullptr;
};

int main()
{
        Empleado *pemp = new Empleado;
        cout << "Ingrese el legajo: ";
        cin >> pemp->legajo;
        cout << "Ingrese el nombre: ";
        cin >> pemp->nombre;
        pemp->siguiente = new Empleado;
        pemp->siguiente->legajo = 5;
        pemp->siguiente->nombre = "Juan";
        Empleado *ptr = pemp;
        cout << "1ro: " << ptr->nombre << " - Nro: " << ptr->legajo << endl;
        ptr = ptr->siguiente;
        cout << "2do: " << ptr->nombre << " - Nro: " << ptr->legajo << endl;
        return 0;
}
```

Salida

```
Ingrese el legajo: 1
Ingrese el nombre: Pedro
1ro: Pedro - Nro: 1
2do: Juan - Nro: 5
```

Licencia