Juan Carlos Alcausa Luque. 2021.

TIPOS DE DATOS ESTRUCTURADOS

1. Structs (registros)

Consisten en una serie de tipos de datos (que no tienen por qué ser iguales), unidos en un mismo tipo. Por ejemplo, si queremos dar una fecha completa, con día, mes y año, lo podemos hacer agrupando estos tres datos bajo un mismo tipo, un **struct**. Ejemplo:

```
//para poder dar el mes con palabras usamos un enum.
enum Tmes{
        enero, febrero, marzo, abril, mayo, junio, julio, agosto, septiembre, agosto, septiembre,
        octubre, noviembre, diciembre
};
struct Tfecha {
        int dia;
        Tmes mes;
        int anyo;
}
```

Una vez definido el struct, SE HACE FUERA DE MAIN, se pueden declarar constantes y variables de ese tipo:

```
const TFecha hoy={22, diciembre, 2022};                    **LLAVES SIMPLES AQUI {}.
Tfecha cumple;
```

Se puede acceder a un elemento de la struct, poniendo primero el nombre de esta y luego el del miembro al que se quiere acceder. Por ejemplo, para asignarle el valor 5 al día cumple se haría así:

```
cumple.dia=5;
```

Existen otros operadores que se pueden utilizar con las structs, pero NO SE PUEDEN COMPARAR STRUCTS (ni == ni > ni <), solo asignación (=).

Tampoco se puede leer ni escribir directamente sobre la struct, sino que hay que hacerlo componente a componente. Para leer o imprimir una struct podemos hacer una función tipo void que realiza esa tarea, pasando mediante REFERENCIA la struct.

```
void escribir_fecha (Tfecha &cumple)
{
        cin>>cumple.dia>>cumple.mes>>cumple.anyo;
}
```

Y para imprimir igual:

```
void imprimir_fecha (Tfecha &cumple)
{
        cout<<cumple.dia<<cumple.mes<<cumple.anyo;
}.
```

2. Arrays

Un array es básicamente una matriz. Siendo más rigurosos, es un número fijo de elementos del mismo tipo. Estos tipos pueden ser cualesquiera queramos como int, float o double.

- TAMAÑO de un array: a.size() *Antes del punto se pone el nombre que le hemos dado al array al declararlo.
- Leer e imprimir un array del teclado:

```
typedef array<int, M> TVector;
void readArr(&TVector);
void printArr(const &TVector);
int main()
{
	TVector a;
	readArr(a);
	print (a);
	cout<<endl;
}
void readArr(&TVector)
{
	for (int i=0; i<M;++i)
		cin>>a[i]>>” “;
}
void printArr(const &TVector)
{
	for (int i=0; i<M;++i)
		cout<<a[i]<<” “;

}
```