LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

4 Structured data types

1. <u>Struct</u>. The data type struct. Structs as parameters.
2. Arrays. The data type array. Multidimensional arrays. Arrays as parameters
3. String of chars. The data type string. Strings as parameters.
4. Examples

Prof. Juan Falgueras
1st. year ETSII

Structures

It is a collection/a binder of distinct data types under a unique name.

```
struct Name {
    type1 field1;
    type2 field2;
    type3 field3;
    …
};
```

Example

```
#include <iostream>
using namespace std;

// types
enum TMonth { JAN, FEB, MAR, APR, MAY, JUN,
              JUL, AUG, SEP, OCT, NOV, DEC };

struct TDate {
    unsigned day;
    TMonth month;
    unsigned year;
};

const TDate TODAY = {16, DEC, 2010};

int main()
{
    TDate birthD, currD;

    return 0;
}
```

Access to members

Use the dot notation:

```
theStruct.member
```

```
struct TComplex {
   float re;
   float im;
};

TComplejo b; //…

b.re = 3.0;
cout << b.im;
```

Structs can be copied!

```
struct TComplex {
            float re;
            float im;
};
TComplex a = {1, 0}, b;

b = a;

cout << "values: "
       << b.re << ", "
       << b.im << endl;
```

and passed to functions!

As they can be copied

- they can be **return**ed as results
- passed by value
- or (using &) by reference

but they cannot be compared

```
struct TDate {
    unsigned day;
    TMonth month;
    unsigned year;
};
```

```
TDate birthD, currD;
birthD = (TDate){19, AUG, 1960};

if (birthD == currD)
    cout << "IS Not legal!";
```

nor any other comparison op

they can be nested

```
struct TDate {
    unsigned day;
    TMonth month;
    unsigned year;
};
```

```
struct TEmployee {
    unsigned code;
    float salary;
    TDate joiningDate;
};
```

initialisation

```
TEmployee empl = {101, 2000, {19, AUG, 1960}};
```

access

```
empl.joiningDate.year++;
printMonth(empl.joiningDate.month);
```

struct with functions ex.

```
#include <iostream>
using namespace std;

// types
struct TTime {
    unsigned hours, mins, secs;
};

// proto
unsigned toSecs(const TTime t);
int main()
{
    TTime atime = {23, 30, 12};
    cout << "Secs: " << toSecs(atime) << endl;
    return 0;
}

unsigned toSecs(const TTime t)
{
    return t.hours * 3600 +
           t.mins * 60 +
           t.secs;
}
```

exercise

- Write a **function** that converts a number of seconds (unsigned) to a structure TTime returning it

```
struct TTime {
    unsigned hours, mins, secs;
};
```

returning structs

- Write a function that returns a TTime read from the keyboard

Exercices

- Build structs to contain:

1. A personal file, with: age, enrolment date
2. A monomial containing coefficient and grade
3. A polynomial of N monomials

- Build a structure to contain the students of a class (NMAXST=55) if we want:

1. Name and age
2. Name, age, and the name of the enrolled subjects (NMAXSUB=20)
3. … + marks in every subject