Programming 1

Lecture 7
Structured data types.
Structs (Records)
Objectives

- Understand the concept of struct (record) data type
- Learn how to define and use complex data structures, nesting structured data types: arrays of structs
- Manage, read and display struct data types in C language
Topics

1. Struct data type
2. Arrays of structs
3. Examples
4. Information sources
The struct or record type

- The **struct type** (C language) or **record type** (other languages) is a data structure to store a finite collection of **heterogeneous** elements (the data may belong to different data types)
  - A struct usually represents a set of attributes of an entity
- Every element in the record is called **field**
  - To refer to an element in a record, use the record identifier, followed by a dot '.' and the field identifier
- Example: record with two elements

```
product.code
product

code
price
```

- Integer type variable
- Float type variable
# Examples: records

<table>
<thead>
<tr>
<th>Address</th>
<th>Book</th>
</tr>
</thead>
<tbody>
<tr>
<td>street</td>
<td>author</td>
</tr>
<tr>
<td>post code</td>
<td>title</td>
</tr>
<tr>
<td>city</td>
<td>borrowed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th>Employee data</th>
</tr>
</thead>
<tbody>
<tr>
<td>day</td>
<td>name</td>
</tr>
<tr>
<td>month</td>
<td>social security number</td>
</tr>
<tr>
<td>year</td>
<td>salary</td>
</tr>
<tr>
<td></td>
<td>address</td>
</tr>
<tr>
<td></td>
<td>birth date</td>
</tr>
</tbody>
</table>
Definition of structs (records) in C language

- To declare a variable of struct type, this type must be previously defined. There are several ways:

```c
typedef struct struct_name {
    field_type1 field_name1;
    field_type2 field_name2;
    ...
};
```

- **struct_name**: name for the defined struct. Any valid identifier
- **field_type**: type for the corresponding struct field
- **field_name**: name for the corresponding struct field. There can be as many fields as needed
Example: struct declaration

```c
typedef struct TProduct {
    int code;
    float price;
} TProduct;

TProduct p1, p2;
```

p1 and p2 are variables of struct type `TProduct`

```c
typedef struct TBook {
    bool borrowed;
    char author[30];
    char title[50];
} TBook;

TBook book1, book2;
```

Nested structured data types: a char array inside a struct

book1 and book2 are variables of struct type `TBook`

```c
typedef struct TNif{
    int number;
    char letter;
} TNif;

typedef struct TClient{
    TNif nif;
    char name[30];
} TClient;

TClient client1;
```

Nested structs
Struct initialization and access

• To initialize a struct, all its fields must be initialized, having access to each one

• To **access** a struct field, the operator `'.'` is used

```
struct_name.field_name;
```

• **struct_name**: struct identifier
• **field_name**: struct field name

• Example: accessing the code of product p1: `p1.code`;
• Example: accessing the book book1 author's initial letter: `book1.author [0];`
• Example: accessing the NIF letter of client client1: `client1.nif.letter`;
Example: struct initialization and access

```c
#include <stdio.h>

typedef struct TProduct {
    int    code;
    float  price;
} ;

TProduct  p1, p2;

p1.code = 3;
p1.price = 34.8;
p2 = p1;

TBook book1, book2;

Typedef struct TBook {
    bool borrowed;
    char author[30];
    char title[50];
    TDate borrow_date;
} ;

book1.borrowed = true;
book1.borrow_date.day = 16;
book1.borrow_date.month = 11;
book1.borrow_date.year = 2010;

strcpy(book1.author, "Quevedo");
cin.getline(book1.title, 50-1);
```

Struct assignment is allowed in C language

The structs can be nested, that is, a struct field can be of another struct type

```c
```
Topics

1. Struct data type
2. **Arrays of structs**
3. Examples
4. Information sources
Arrays of structs

- In an array of structs each element in the array is a struct

```c
typedef struct TProduct{
    int    code;
    float price;
} ;
```

```c
TProduct products[30];
```

How would you access the third product code?

```c
products[2].code
```
The structured data types can be nested (I)

• A struct field can be an array itself

typedef struct TProduct {
    int    code;
    char name [15];
    float month_price[12];
} TProduct;

TProduct products[30];

TProduct code
name
month_price

products[0]
products[1]
...
products[29]

How would you access the price in August of the fifth product?
The structured data types can be nested (II)

• A struct field can be a struct itself

```c
typedef char TString[15];
typedef struct TAddress {
    TString    street;
    int        number;
    TString    city;
} ;

typedef struct TClient {
    TString    name;
    TAddress   addr;
    TDate      register_date;
} ;

typedef TClient clients [20];
```

How would you access the last client's city?

```c
clients[19].addr.city
```

And to the year of the first client's registration?

```c
```
Topics

1. Struct data type
2. Arrays of structs
3. Examples
4. Information sources
Example 1

- Define the necessary data structures to store and process the following information:
  - A rent-a-car company wants to manage information about its car fleet (up to 200 vehicles): registration number, brand, model, purchase date and the amount of kilometres done each month during a year. The aim is to obtain vehicles which do the highest average amount of kilometres during a year (there can only be one car or several cars with the same average)
Example 1: Data structures

- Define the necessary data structures to store and process the following information:

- A rent-a-car company wants to manage information about its car fleet (up to 200 vehicles): registration number, brand, model, purchase date and the amount of kilometres done each month during a year. The aim is to obtain vehicles which do the highest average amount of kilometres during a year (there can only be one car or several cars with the same average)

  - Registration number: char array
  - Brand: char array
  - Model: char array
  - Purchase date: struct: day, month, year
  - km monthly x 12 months: array with 12 integers

array with 200 vehicles

vehicle
Example 1: Our aim is ...

- Define the necessary data structures to store and process the following information:

- A rent-a-car company wants to manage information about its car fleet (up to 200 vehicles): registration number, brand, model, purchase date and the amount of kilometres done each month during a year. The aim is to obtain vehicles which do the highest average amount of kilometres during a year (there can only be one car or several cars with the same average).

array with the index of the vehicles with the highest average (200)
Example 1: Data design

vehicle
- registration number → array of 9 chars
- brand → char array
- model → char array
- purchase date → struct: day, month, year
- km monthly × 12 months → array with 12 integers

array with 200 vehicles

```c
const int N_CARS = 200;
typedef TVehicle TCars[N_CARS];
TCars cars;
typedef struct TDate {
    int day;
    int month;
    int year;
};
typedef int TKm_month [12];
typedef struct TVehicle{
    TRegistration registration;
    TString brand;
    TString model;
    TDate purchase_date;
    TKm_month km_month;
};
typedef int THigherAverageCars[N_CARS];
THigherAverageCars higher_average_cars; // position in array cars[
int n_higher_average_cars; // amount of cars with highest average
```
Example 2

- Define the necessary data structures to store and process the following information:

- In a washing machine factory a computerized quality control wants to be established to check its prototypes. Each machine has a numerical code and a set of features: capacity (in kg), model, loading type (top/front) and the result of 10 control tests. Each control test can only have two possible results: accepted or rejected. Moreover, we need to know the inspector for each control test. We have the following information for each inspector: numerical code, name, department where he/she belongs to. The factory manufactures 25 prototypes each year.
Example 2: Revise the text

- Define the necessary data structures to store and process the following information:
  - In a washing machine factory a computerized quality control wants to be established to check its prototypes. Each machine has a **numerical code** and a set of features: **capacity** (in kg), **model**, **loading** type (top/front) and the result of **10 control tests**. Each control test can only have two possible results: accepted or rejected. Moreover, we need to know the **inspector** for each control test. We have the following information for each inspector: numerical **code**, **name**, **department** where he/she belongs to.
  - The factory manufactures **25 prototypes** each year.

```
array with 25 prototypes
{ code: integer, capacity: integer, model: char array, load: char or enum, controls: array with 10 controls, ok: boolean, inspector: { code: integer, name: char array, dept: char array } }
```
Example 2: Data design

```c
const int N_CTRL = 10;
typedef struct TMachine {
  int code;
  int capacity;
  char model[40];
  char load;
  TControl controls[N_CTRL];
};

typedef struct TControl{
  boolean ok;
  int code_inspector;
}; //the information about the inspector is not repeated for each control

typedef struct TInspector{
  int code;
  TString name;
  TString dept;
};

const int N_MACHINES = 25;
typedef TMachine TPrototypes[N_MACHINES];
typedef struct TInspector{TString name; TString dept;};

TPrototypes machines;
```
Topics

1. Struct data type
2. Arrays of structs
3. Examples
4. Information sources
Information sources

**Fundamentos de Programación**
Jesús Carretero, Félix García, y otros

✔ Capítulo 9 (Apartados 9.1.1; 9.1.2; 9.3)

**Problemas Resueltos de Programación en Lenguaje C**
Félix García, Alejandro Calderón, y otros

✔ Capítulo 2 (Apartados 7.1.1; 7.1.2; 7.2)

**Resolución de Problemas con C++**
Walter Savitch

✔ Capítulo 6 (Apartado 6.1)