

LESSON 8

ADVANCED TOPICS IN COMPUTING AND PROGRAMMING

Programming
Grade in Industrial Technology Engineering

Year 2017-2018



Universidad
Carlos III de Madrid
www.uc3m.es

TABLE OF CONTENTS:

1. EXTERNAL DATA STORAGE
 1. FILES
 2. DATABASES
2. DYNAMIC MEMORY MANAGEMENT
3. PROGRAMS USED IN ENGINEERING

EXTERNAL DATA STORAGE

The need for external data storage

- Permanent **storage of program results**
 - Data storage on the hard disk
 - Data stored in main memory of the computer is lost when the execution ends
- Data are read from disk, rather than asking the **user** to **enter** it
 - Data in files with a proper format can be read and assigned to the variables and structures of the program
- Dealing with data larger than the memory size
 - Data is stored in the hard disk, which has larger storage capabilities, and it is processed in parts

Types of external data storage

- Two main ways to store data
 - Files
 - Databases

FILES

Concept of file

- Set of information stored in any medium that can be read by a computer
- A file is identified by it's name and the description of the place in the computer where it's stored (folder's path)
- File operations
 - > Open file
 - > Read from file
 - > Write to file
 - > Close file

Binary files

- Data are stored as a sequence of bits corresponding to the binary representation of data items
- The file consists of records, each record is a copy of a variable as it is stored in memory
- To recover data one needs to know the type of variable
 - You can't just open the file with a text processor, data need to be translated
 - Only the original C program (or a similar one) will be able to use the data

Text files

- Data are converted into characters (ASCII) before writing to file
- The file consists of records, each record is a translation of the data item to text
- Any text processor can recover the data
 - A human can read and interpret (to some extent) the data
 - Other programs can use the data

Using text vs binary files

- If you plan to use your data within a C program, binary files are more efficient
 - Converting data back and forth is time consuming, specially with floats and doubles
 - Binary files use up less memory
- Choose text files if you want to share the data

Working with files in C

- C standard library `stdio` provides functions for the management of streams
- Scan/print similar to reading from keyboard/display
- Declare file variable (pointer to `FILE` type)
 - `FILE * f`
- Open file **`fopen`**
 - Specify mode: binary(`b`) /text (default) read (`r`)/write (`w`)
- Read data
 - **`fscanf`**, for text files
 - **`fread`**, for binary files
- > Print data
 - **`fprintf`**, for text files
 - **`fwrite`**, for binary files
- > Close file **`fclose`**

Sample code for working with files

```
#include <stdio.h>

// Program to write the first 100 prime numbers to file que escribe
// uses function is_prime to check if number is prime

int main(void) {

FILE *fp; // declare a variable of file datatype (FILE*)

int i, n;

//open file primes.txt, for writing
fp = fopen("primes.txt", "w");
i = 1;
n = 0;
```

Sample code for working with files (continued)

```
while (n<1000){
    if (is_prime(i)) {
        fprintf (fp, "%d\n", i); //write number to file
        n++;
    }
    i++;
}
fclose(fp); //Close file
return 0;
}
```

DATA BASES

Database

- Alternative to files for permanent information storage
- Data are organized according to the intended use
- Data follow a structure
 - Typically organized in Tables
 - example: Books database
 - Tabla libro: con título, año, id autor, id editorial,,
 - Tabla editorial: con dirección, ciudad, país
 - Tabla autor: con nombre, nacionalidad, fecha nacimiento
- In disk, data are stored as files
 - But files are not directly accessed, instead a special program (Database management system DBMS) is used to enable access to data

Relational Databases

- Most common type of database
- Data split in tables to avoid redundancy
- Tables linked through keys

Activities Table

Student	Activity1	Cost1	Activity2	Cost2
John Smith	Tennis	\$36	Swimming	\$17
Jane Bloggs	Squash	\$40	Swimming	\$17
John Smith	Tennis	\$36		
Mark Antony	Swimming	\$15	Golf	\$47

Students Table

Student	ID*
John Smith	084
Jane Bloggs	100
John Smith	182
Mark Antony	219

Activities Table

ID*	Activity*	Cost
084	Swimming	\$17
084	Tennis	\$36
100	Squash	\$40
100	Swimming	\$17
182	Tennis	\$36
219	Golf	\$47
219	Swimming	\$15
219	Squash	\$40

information linked through **keys**

Relational Databases

- Databases are managed by a database management system (DBMS)
- Examples
 - Find all books by one author in a given year
 - Find all students signed-in for tennis lessons
- The DBMS provides a query language to retrieve data from tables
 - **SQL** language (Structured Query Language)

SELECT

```
Student, Activity FROM Students, Activities
```

WHERE

```
Students.ID=Activities.ID AND
```

```
Name = "Tennis"
```


DYNAMIC MEMORY MANAGEMENT

Dynamic memory management

- Dynamic memory management is a set of techniques (supported by C functions) to optimize the amount of memory required by a program
 - Program to store the record store catalogue
 - May include 30 or 30.000 albums
 - **What we've being doing so far**
 - Declare an array of 30.000 albums
 - Use only the first 'N' elements of the array
 - 30.000 - 'N' are empty
 - >> Memory is wasted!
 - **Good solution**
 - Dynamic memory allocation and management
 - The size of the array is not assigned when it is declared
 - The size is assigned only when required, and with only the needed elements
 - >> Less memory is used (and only when needed)

C functions for dynamic memory allocation

```
<stdlib.h>
void * malloc(int n_bytes)
void * calloc(int n_elements, int size_element)
```

- malloc and calloc
 - (1) allocate the amount of memory specified as parameter
 - (2) return a pointer to the first position of this memory chunk
- The pointer can be used as the first position of an array (remember that pointers and arrays are closely related!)

C functions for dynamic memory allocation

- `void * malloc(int n_bytes)`
 - *n_bytes*: memory size (in bytes) to be allocated
 - The values of the allocated memory cells are unknown (garbage)
- `void * calloc(int n_elements, int size_element)`
 - *n_elements*: number of 'elements' of the memory chunk
 - *size_element*: size (in bytes) of each 'element' of the memory chunk
 - The values of the allocated memory cells are set to 0

C functions for memory release

- Allocated memory **must be released** when it is no longer necessary
 - It is a responsibility of the programmer to pair up conveniently memory allocation and release
 - > Very powerful feature but also error-prone
- p is a pointer to the address of a memory chunk allocated with calloc or malloc

```
<stdlib.h>
```

```
void free(void * p)
```

Dynamic memory example

```
#include <stdlib.h>
#include <stdio.h>

int main(void)
{
    int *a;
    int size, i;

    printf ("Enter vector size: ");
    scanf ("%d", &tam);

    //Allocate memory for vector a
    a = malloc(tam*sizeof(int));
    //Assign values to elements in a (remember pointer / vector )
    for (i=0; i<tam; i++)
        a[i] = i;
    //release memory
    free(a);
    a = NULL;
```

SOFTWARE PROGRAMS USED IN ENGINEERING

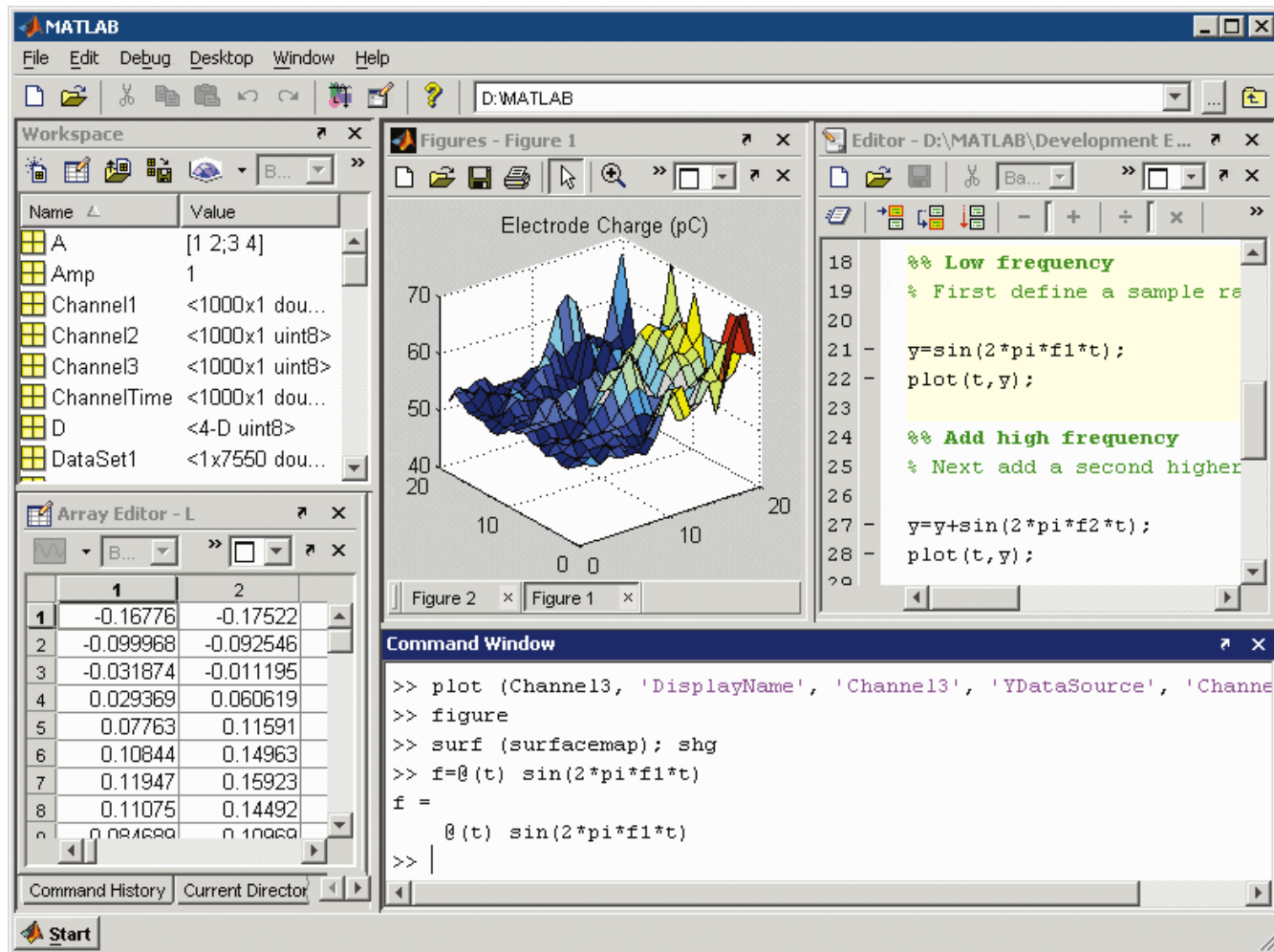
Software Programs used in Engineering

- Nearly all engineering tasks can be supported by computer software
- Engineering software, some examples
 - Matlab
 - Computer Aided Design (CAD) / Computer Aided Manufacturing (CAM)
 - Design, calculus, and simulation software
- Office tools
 - As in any other professional activity

MATLAB ®

- High-level language and interactive environment for technical computing
 - Language for algorithm development, data visualization, data analysis, and numeric computation
 - Development environment for managing code, files, and data
 - Interactive tools for iterative exploration, design, and problem solving
- Includes:
 - Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration
 - 2-D and 3-D graphics functions for visualizing data
 - Tools for building custom graphical user interfaces
 - Functions for integrating MATLAB based algorithms with external applications and languages, such as C, C++, Fortran, Java, COM, and Microsoft Excel
- <http://www.mathworks.com/products/matlab/>

Matlab interface



Computer Aided Design CAD

- **Computer Aided Design CAD**

- Tools to support design and documentation process
- The result of the CAD process is a design ready to be printed or manufactured
- Graphical information, materials, process, dimensions, tolerances, etc.

- **Advantages**

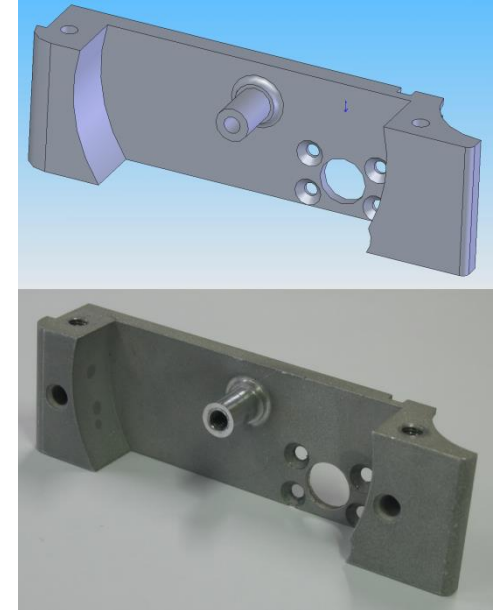
- Optimization of the design process
- Advanced functionalities not available in traditional procedures: rotations, simulation, etc.

- **2D and 3D tools**

- 2-dimension graphics: vector graphics (points, lines, arcs, polygons, etc.)
- 3-dimension graphics: solids, surfaces, etc.

CAM: Computer Aided Manufacturing

- Use of computing tools in different stages of a product manufacturing (machine parts, production, quality control)
- Typically computer aided manufacturing is coupled with computer aided design
 - Information regarding the part to build is given to the program that operates the production of the parts



Fuente: wikipedia

Design, calculus, and simulation

- Specific software programs to support different Engineering tasks
 - Industrial Organization
 - Project management
 - Manufacturing management
 - Simulation and optimization of logistics and production process
 - E.g.: Witness environment used in Quantitative Organization Methods
 - Logistics management
 - Production and logistics systems
 - Materials
 - Simulation of polymeric fluids behavior in injection molding
 - E.g.: Moldflow used in Polymer Technologies

Design, calculus, and simulation

- Electrical installations
 - Analysis and simulation of electric systems
 - E.g.: PSE/E Power System Simulator for Engineering used in Electrical Grids and Circuits
 - Planning of electricity distribution networks
 - Production and logistics systems
- Energy Technologies
 - Fluid dynamics calculus and simulation
 - E.g.: FLUENT 6.2 used in Computer-aided simulation of industrial flows
- Robotics and automation
 - Robot programming (E.g.: RAPID)
 - Integration of instrumentation systems (E.g.: LabView)
 - 3D modeling, simulation and animation of physical systems (E.g.: Roboworks)

LESSON 8

ADVANCED TOPICS IN COMPUTING AND PROGRAMMING

Programming
Grade in Industrial Technology Engineering

Year 2017-2018



Universidad
Carlos III de Madrid
www.uc3m.es