Féidearthachtaí as Cuimse
Infinite Possibilities

Semester 2
Week 8 - Tutorial

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TU DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

Programming - Week 8 – 17th March 2025

Overview

- Structures in C (struct)
- Nested Structures
- Typedef
- Mandatory Lab Question

Structures in C (struct)

Revision

- A structure in C is a user-defined data **structure, i.e. a complex data type**. It allows you to store related data items **of different types** together.
- **Why Use Structures?**
  - C's primitive types (int, float, char) store only one type of data.
  - Structures help in grouping related information together.
  - Useful for creating complex data types, like representing a student, employee, or product.

Defining a Structure

Revision

- A structure is defined using the struct keyword:

```
struct student_rec
{
        int student_ID;
        char firstname[11];
        char surname[21];
        int results[5];
};
```

Where:

- Student_id – an integer
- Firstname - a string of max 11 characters
- Surname - a string of max 21 characters
- Results – an array if integers

Structure Members

Example 1

- Create a program to fulfill the following requirements:
- Create a structure to store the information relating to an upcoming exam. The user will enter the following data and this should be stored in the structure.
  - Module name
  - Course name
  - Course code
  - Exam time (24 hr clock)
  - Date
- Display all the information of the Exam structure.

User Input	Type
Programming	text
MSc in Coding	text
TU411	text
11:30	text
17/03/2025	text

Example 1 - solution

Example1.c > main()

```c
#include <stdio.h>

/*  Define the structure to store
    information related to the exam
*/
struct Exam {
    char module_name[50];
    char course_name[50];
    char course_code[10];
    char exam_time[10];
    char exam_date[20];
};

int main() {
    struct Exam exam;

    printf("Enter module name: ");
    fgets(exam.module_name, sizeof(exam.module_name), stdin);

    printf("Enter course name: ");
    fgets(exam.course_name, sizeof(exam.course_name), stdin);

    printf("Enter course code: ");
    fgets(exam.course_code, sizeof(exam.course_code), stdin);

    printf("Enter exam time (HH:MM, 24-hour format): ");
    fgets(exam.exam_time, sizeof(exam.exam_time), stdin);

    printf("Enter exam date (DD/MM/YYYY): ");
    fgets(exam.exam_date, sizeof(exam.exam_date), stdin);

    printf("\nExam Information:\n");
    printf("Module Name: %s", exam.module_name);
    printf("Course Name: %s", exam.course_name);
    printf("Course Code: %s", exam.course_code);
    printf("Exam Time: %s", exam.exam_time);
    printf("Exam Date: %s", exam.exam_date);

    return 0;
}
```

Output:

```
Enter module name: Programming
Enter course name: MSc in Code
Enter course code: TU411
Enter exam time (HH:MM, 24-hour format): 11:30
Enter exam date (DD/MM/YYYY): 17/03/2025

Exam Information:
Module Name: Programming
Course Name: MSc in Code
Course Code: TU411
Exam Time: 11:30
Exam Date: 17/03/2025
```

Nested Structures

- **Nested structures** refer to the concept of using one structure as a member of another structure. A structure can contain another structure as one of its fields, which allows for more complex data representations.

```c
Slide_ex_1.c > ...
#include <stdio.h>

// Define a structure for Date
struct Date {
    int day;
    int month;
    int year;
};
```

```c
// Define a structure for Person, with Date as a member
struct Person {
    char name[50];
    int age;
    struct Date dob;  // Nested structure
};
```

The Person structure has a Data structure as a member.

This populates the Date dob struct

In main create a Person

```c
    // Declare and initialize a Person object
    struct Person p1 = {"Diana Prince", 21, {10, 2, 2004}};
```

Nested Structures

```c
int main() {

    // Declare and initialize a Person object
    struct Person p1 = {"Diana Prince", 21, {10, 2, 2004}};

    // Print person's details
    printf("Name: %s\n", p1.name);
    printf("Age: %d\n", p1.age);
    printf("DOB: %d/%d/%d\n", p1.dob.day, p1.dob.month, p1.dob.year);

    return 0;
}
```

Output:

```
Name: Diana Prince
Age: 21
DOB: 10/2/2004
```

To access the data stored in the nested structure we use the dot(.) notation. p1 is the person struct instance, the dob struct is a member of the person struct.

Example 2

- Example 1 is working perfect. Thank you for coding this for us.
- Can you refactor this solution to store the date in a structure with all members of type int (e.g. day, month and year)
- The date structure must be nested in the Exam structure

Example 2 - solution

Example2.c > main()

```c
#include <stdio.h>

// Define a structure for Date
struct Date {
    int day;
    int month;
    int year;
};

struct Exam {
    char module_name[50];
    char course_name[50];
    char course_code[10];
    char exam_time[10];
    struct Date dob;
};

```

- Define the new Date structure
- Update the Exam struct to remove the day, month and year. Add a new member entry to represent our new Date struct.

Example 2 - solution

```c
int main() {
    struct Exam exam;

    printf("Enter module name: ");
    fgets(exam.module_name, sizeof(exam.module_name), stdin);

    printf("Enter course name: ");
    fgets(exam.course_name, sizeof(exam.course_name), stdin);

    printf("Enter course code: ");
    fgets(exam.course_code, sizeof(exam.course_code), stdin);

    printf("Enter exam time (HH:MM, 24-hour format): ");
    fgets(exam.exam_time, sizeof(exam.exam_time), stdin);
```

- We only have to create an Exam structure variable. The Date struct is nested in the Exam struct.
- fgets() the text data

Example 2 - solution

```c
    printf("Enter exam day (1-31): ");
    scanf("%d", &exam.dob.day);

    printf("Enter exam month (1-12): ");
    scanf("%d", &exam.dob.month);

    printf("Enter exam year (eg. 2025): ");
    scanf("%d", &exam.dob.year);

    printf("\nExam Information:\n");
    printf("Module Name: %s", exam.module_name);
    printf("Course Name: %s", exam.course_name);
    printf("Course Code: %s", exam.course_code);
    printf("Exam Time: %s", exam.exam_time);
    printf("Exam date: %d/%d/%d", exam.dob.day, exam.dob.month, exam.dob.year);
    printf("\n");

    return 0;
} // end main
```

- We use scanf() to get the int values from the user.
- Note the dot notation used to store the data in the nested Date struct.
- The dot notation is also used to access the data for display with printf().

```
Enter module name: Programming
Enter course name: MSc in Code
Enter course code: TU411
Enter exam time (HH:MM, 24-hour format): 14:00
Enter exam day (1-31): 17
Enter exam month (1-12): 3
Enter exam year (eg. 2025): 2025

Exam Information:
Module Name: Programming
Course Name: MSc in Code
Course Code: TU411
Exam Time: 14:00
Exam date: 17/3/2025
```

What is an Array of Structures?

- An **array of structures** is a way to store multiple structure variables in a single array.
- This is useful when dealing with collections of structured data, such as storing details of multiple students, employees, or exam records.

Example – array of structs

Example3.c > main()

```c
#include <stdio.h>

#define SIZE 3

// Define the structure
struct Student {
    int id;
    float grade;
};

int main() {
    // Declare an array of structures
    struct Student students[SIZE];

    // Input student details
    for (int i = 0; i < SIZE; i++) {
        printf("\nEnter details for Student %d:\n", i + 1);
        printf("Student ID: ");
        scanf("%d", &students[i].id);
        printf("Grade: ");
        scanf("%f", &students[i].grade);
    }

    // Display student details
    printf("\nStudent Records:\n");
    for (int i = 0; i < SIZE; i++) {
        printf("Student ID: %d\t", students[i].id);
        printf("Grade: %.2f\n", students[i].grade);
    }

    return 0;
}
```

Example – array of structs

Output:

```
Enter details for Student 1:
Student ID: 12345
Grade: 45

Enter details for Student 2:
Student ID: 54321
Grade: 67

Student Records:
Student ID: 12345       Grade: 45.00
Student ID: 54321       Grade: 67.00
```

- Define a structure Student with:
  - id - Integer for student ID.
  - grade - Float for student grade.
- Use an array of structures:
  - struct Student students[SIZE];
  - This stores multiple students in a single array.
- Loop for input:
  - The user enters the student ID and grade for each student.
- Loop for output:
  - The program displays all stored student details.

typedef in C

- The typedef statement in C is used to create an alias for existing data types. This makes the code easier to read and maintain by allowing custom names for complex types like structures, pointers, and arrays.
- typedef can be used with Structures

typedef in C

```c
struct Student {
    int id;
    float grade;
};

struct Student s1;  // Have to use 'struct' every time
```

```c
typedef struct {
    int id;
    float grade;
} Student;  // 'Student' is now an alias for 'struct'

Student s1;  // No need for 'struct' keyword
```

Mandatory Question – in class solution

- Using Structures, write a program to do the following:
- Design a structure template to store biographical data about a person.
- Your program must:
  a. Enter data for a person's first name, surname, date of birth, height, weight, eye colour & country of citizenship.
  b. Display the data entered.
  c. Copy the data and store it in a 2nd person's record and then modify it.
  d. Display the new data for the 2nd person.
- Hint: for part (c), when you create two variables of the same structure template, try using
  - variable_second = variable_first; This will copy ALL member data from
  - variable_first into variable_second, i.e., an exact copy.

Questions