Féidearthachtaí as Cuimse
Infinite Possibilities

Semester 2
Week 7 - Tutorial

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TU DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

Programming - Week 7 – 10th March 2025

Overview

- Structures in C (struct)
- Struct Examples
- Structs with Functions
- Mandatory Question

Structures in C (struct)

- A **structure** in C is a **user-defined data structure** that groups different types of variables under a single name. 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

- 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

Example 1

- Create a program to store a students name, age and grade.
- The data attributes above must be stored in a structure.
- The program should ask the user to input values for name, age and grade and store in a structure.
- Display the data in the structure back to the user using puts()

Example 1 - solution

```c
#include <stdio.h>

// Define the structure
struct Student {
    char name[50];
    int age;
    float grade;
};

int main() {
    // Declare a structure variable
    struct Student s1;

    // Assign values
    printf("Enter name: ");
    scanf("%s", s1.name);

    printf("Enter age: ");
    scanf("%d", &s1.age);

    printf("Enter grade: ");
    scanf("%f", &s1.grade);

    // Display the information
    printf("\nStudent Details:\n");
    printf("Name: %s\n", s1.name);
    printf("Age: %d\n", s1.age);
    printf("Grade: %.2f\n", s1.grade);

    return 0;
}
```

Run Program:

```
Enter name: Diana
Enter age: 21
Enter grade: 68

Student Details:
Name: Diana
Age: 21
Grade: 68.00
```

Remember:
To assign data to any structure variable member, you must use the full-stop/period operator

Example 2 - requirements

- Create a program to store the results for a programming assignment in-person exam.
- The program should store the attendance of students attending the exam.
- The program will store the following:
  - CA name
  - Date
  - Array of student numbers
- The program will continue to run allowing students to enter their student number to confirm their attendance (type exit to finish)
- Display the list of student numbers when the exam is over.

Example 2 – Problem Solving

- **Lets break the implementation into smaller parts**
  1. Create the structure of a basic C program
  2. Create the data structure
  3. Ask the user to enter the Exam Name and Date
  4. Ask the user to enter 1 student number and store in the structure.
  5. Display the 1 student in the structure
  6. Add a loop to facilitate multiple entry of student numbers
  7. Add option to end loop if exit is entered.
  8. Display all the data stored in the structure (ie. The student numbers)

Example 2 - solution

```c
#include <stdio.h>

#define MAX_STUDENTS 50 // num of students
#define MAX_LENGTH 10   // length of student num

// Structure to store exam details
struct Exam {
    char caName[MAX_LENGTH];
    char date[11]; // DD-MM-YYYY
    char studentNumbers[MAX_STUDENTS][MAX_LENGTH]; // 2D array
    int studentCount; // number signed in
};
```

```c
int main() {
    struct Exam exam;
    exam.studentCount = 0;  // Initialize student count
    char studentNum[MAX_LENGTH]; // store user input

    // Get exam details
    printf("Enter Course Assistant (CA) Name: ");
    scanf("%s", exam.caName);

    printf("Enter Exam Date (DD-MM-YYYY): ");
    scanf("%s", exam.date);

    printf("CA Name: %s\n", exam.caName);
    printf("Date: %s\n", exam.date);
}
```

Run Program:

```
Enter Course Assistant (CA) Name: CA2
Enter Exam Date (DD-MM-YYYY): 07-03-2025
CA Name: CA2
Date: 07-03-2025
```

We have currently catered for steps 1, 2 and 3 from our problem solving steps.

Example 2 - solution

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

```c
int main() {

    // Get exam details
    printf("Enter Course Assistant (CA) Name: ");
    scanf("%s", exam.caName);

    printf("Enter Exam Date (DD-MM-YYYY): ");
    scanf("%s", exam.date);
```

Run Program:

```
Enter Course Assistant (CA) Name: CA2
Enter Exam Date (DD-MM-YYYY): 01-01-1111

Enter student number (or type 'exit' to finish):
Student Number: D12345
Student Number: D54321
Student Number: exit
CA Name: CA2
Date: 01-01-1111
```

```c
    printf("\nEnter student number (or type 'exit' to finish):\n");
    while (exam.studentCount < MAX_STUDENTS) {
        printf("Student Number: ");
        scanf("%s", studentNum);

        if (strcmp(studentNum, "exit") == 0) {
            break;  // Stop loop when "exit" is entered
        }

        strcpy(exam.studentNumbers[exam.studentCount], studentNum);
        exam.studentCount++;

        if (exam.studentCount >= MAX_STUDENTS) {
            printf("Max student limit reached!\n");
            break;
        }
    }

    printf("CA Name: %s\n", exam.caName);
    printf("Date: %s\n", exam.date);
} // end main
```

We have currently catered for steps 4,5,6 and 7 from our problem solving steps. 4 and 5 were used in the incremental development of the code

Example 2 - solution

```
45      // Display attendance list
46      printf("\n--- Exam Attendance ---\n");
47      printf("CA Name: %s\n", exam.caName);
48      printf("Date: %s\n", exam.date);
49      printf("Total Students Attended: %d\n", exam.studentCount);
50      printf("Student Numbers:\n");
51
52      for (int i = 0; i < exam.studentCount; i++) {
53          printf("%s\n", exam.studentNumbers[i]);
54      }
55
56      return 0;
57
58  } // end main
```

```
Enter Course Assistant (CA) Name: CA2
Enter Exam Date (DD-MM-YYYY): 07-03-2025

Enter student number (or type 'exit' to finish):
Student Number: D12345
Student Number: D54321
Student Number: C98765
Student Number: C45678
Student Number: exit

--- Exam Attendance ---
CA Name: CA2
Date: 07-03-2025
Total Students Attended: 4
Student Numbers:
D12345
D54321
C98765
C45678
```

We have currently catered for step 8 in this iteration.

There may be a logical flaw with how the loop ends, can you see the potential issue? Can we fix this?

Pointers with Structures

- **Pointers to structures** allow us to efficiently access and modify data stored in a structure. This is especially useful when passing structures to functions without making copies.

Pointers with Structures

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TECHNOLOGICAL UNIVERSITY DUBLIN

Example3.c > ...

```c
#include <stdio.h>

// Define a structure
struct Student {
    int id;
    char name[20];
};

int main() {
    struct Student s1 = {101, "Diana"};  // Normal structure variable

    struct Student *ptr; // Declare a pointer to a structure
    ptr = &s1;  // Assign the address of s1 to ptr

    // Access structure members using a pointer
    printf("ID: %d\n", ptr->id);      // Equivalent to (*ptr).id
    printf("Name: %s\n", ptr->name);  // Equivalent to (*ptr).name

    return 0;
}
```

- Declare struct Student s1 = {101, "Diana"};
- Create a pointer struct Student *ptr;
- Assign the address of s1 to ptr using ptr = &s1;
- Access members using -> operator instead of . (dot operator).
  - ptr->id is the same as (*ptr).id
  - ptr->name is the same as (*ptr).name

Structures with Functions

- Create a program to store a student data in a structure.
- Getter and setter functions must be used to create the student struct and then display the data.
- The setter function will take in the name and id then create a Student struct and return the struct to its caller.
- The getter function will take in a struct pointer and display the student data.

Structures with Functions

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TECHNOLOGICAL UNIVERSITY DUBLIN

C Example4.c > main()

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

// Define the Student structure
struct Student {
    int id;
    char name[20];
};

// Function prototypes (setters and getters)
struct Student setStudent(int id, const char *name);
void getStudent(struct Student *s);
```

```c
int main() {
    struct Student s1;

    // Use setter function to assign values
    s1 = setStudent(101, "Diana");

    // Use getter function to retrieve and print values
    getStudent(&s1);

    return 0;
}

// Setter function: create struct and return
struct Student setStudent(int id, const char *name) {
    struct Student s;

    s.id = id;
    strncpy(s.name, name, sizeof(s.name) - 1); // Set Name

    return s;
}

// Getter function: Print student details
void getStudent(struct Student *s) {
    printf("Student ID: %d\n", s->id);
    printf("Student Name: %s\n", s->name);
}
```

Mandatory Question – in class solution

- Write a C program that enters a string from standard input and uses separate functions to do the following:
  a) Compare the string to the following string: “Hello World”.
  b) Determine if the word, i.e. substring, “is” occurs in the string entered (assuming there is at least one occurrence). Is it possible to count the number of occurrences? Hint: try the built-in string function: strstr(string_to_check, substring); This function returns a char pointer that points at the memory address of the start of the substring if found, otherwise NULL is returned.

Questions