Advanced Procedural Programming

Lecture 4: Structures

Gemma O'Callaghan

Structures Overview

- C arrays allow you to define type of variables that can hold several data items of the same kind but **structure** is another user defined data type available in C programming, which allows you to combine data items of different kinds.
- Structures are used to represent a record.
- To keep track of your books in a library. You might want to track the following attributes about each book:
  - Title
  - Author
  - Subject
  - Book ID

Defining a Structure

```
struct [structure tag]
{
  member definition;
  member definition;
  ...
  member definition;
} [one or more structure vars];
```

```
Struct student_rec
{
    int number;
    char surname[25];
    char first_name[11];
    int results[5];
};
```


Defining a Structure

- A structure template consists of the reserved word **struct** followed by the name of the structure. The name of the structure is known as the structure tag. (e.g. *student_rec*)
- After the structure tag, each item within the structure is declared within the braces { and }.
- Each item in a structure is called a structure member.
- A structure member has a name and a data type. You can choose any name you wish for a member, provided you obey the rules for constructing valid variables names.

Declaring a Structure

- Declaring a structure does not allocate memory to the structure.
- You have just simply defined a new data type consisting of previously defined data types.
- Once you have defined the new data type you can then define variables with that type.
  - E.g. – *struct student_rec student1, student2;*
  - This defines the variables *student1* and *student2* to be of the type *struct student_rec*.
  - Both *student1* and *student2* are structure variables with 4 members, i.e. *number*, *surname*, *first_name* and *results*.

GMIT INSTITIÚID TEICNEOLAÍOCHTA NA GAILLIMHE-MAIGH EO GALWAY-MAYO INSTITUTE OF TECHNOLOGY

5

Accessing Structure Members

- The members of a structure variable can be accessed with the member selection operator "." (a dot).
- E.g. we can assign values to the member *number* of the variables *student1* and *student2* using the following:
  - *student1.number = 123;* *student2.number = 124;*
- *You can use* student1.number *and* student2.number in the same way that you use any other integer variable.
- See *BasicStructureProgram.c* example program.

GMIT INSTITIÚID TEICNEOLAÍOCHTA NA GAILLIMHE-MAIGH EO GALWAY-MAYO INSTITUTE OF TECHNOLOGY

6

Basic Structure Sample Program

```
#include <string.h>
#include <stdio.h>
main()
{
/* Declare the structure template */
struct student_rec
{
 /* Declare the members of the structure */
 int number ;
 char surname[21] ;
 char first_name[11] ;
 int scores[5] ;
} ;

/* Define two variables having the type struct student_rec */
struct student_rec student1, student2 ;
```

GMIT INSTITIÚID TEICNEOLAÍOCHTA NA GAILLIMHE-MAIGH EO GALWAY-MAYO INSTITUTE OF TECHNOLOGY

7

Basic Structure Sample Program

```
int i ;

/* Read in values for the members of student1  */
printf( "Number: " ) ;
scanf( "%d" , &student1.number ) ;
printf( "Surname: " ) ;
scanf( "%20s", student1.surname ) ;
printf( "First name: " ) ;
scanf( "%10s" , student1.first_name ) ;
printf( "Five test scores: " ) ;
for ( i= 0 ; i < 5 ; i++ )
  scanf( "%d" , &student1.scores[i] ) ;
```

GMIT INSTITIÚID TEICNEOLAÍOCHTA NA GAILLIMHE-MAIGH EO GALWAY-MAYO INSTITUTE OF TECHNOLOGY

8

Basic Structure Sample Program

```
// Now assign values to the members of student2.

student2.number = student1.number + 1 ;
strcpy( student2.surname , "Smith" ) ;
strcpy( student2.first_name , "Mary" ) ;
for ( i= 0 ; i < 5 ; i++ )
  student2.scores[i] = 100 ;
```


Basic Structure Sample Program

```
/* Display the values in the members of student1 */

printf( "\n\nThe values in student1 are:" ) ;
printf( "\nNumber is %d" , student1.number ) ;
printf( "\nSurname is %s" , student1.surname ) ;
printf( "\nFirst name is %s" , student1.first_name ) ;
printf( "\nScores are: " ) ;
for ( i= 0 ; i < 5 ; i++ )
  printf( " %d " , student1.scores[i] ) ;
```


Basic Structure Sample Program

```
/* Display the values in the members of student2 */

printf( "\n\nThe values in student2 are:" ) ;
printf( "\nNumber is %d" , student2.number ) ;
printf( "\nSurname is %s" , student2.surname ) ;
printf( "\nFirst name is %s" , student2.first_name ) ;
printf( "\nScores are: " ) ;
for ( i= 0 ; i < 5 ; i++ )
  printf( " %d " , student2.scores[i] ) ;

printf( "\n" ) ;
} //end of main
```


The Structure Tag is Optional

- The structure tag *student_rec* in line 8 of the program is optional. You can define the structure template and the structure variables together:

```
Struct
{
    int number;
    char surname[25];
    char first_name[11];
    int results[5];
} student1, student2; (Variables immediately after)
```

Copying a Structure

- You can copy an entire structure to an identical structure:
  - *student2 = student1;*
- This will assign to each member of *student2* the value of the corresponding member of *student1*.

Pointers to Structures

- The general format for defining a pointer to a structure is : *struct tag_name *variable_name;*
- *Where tag is the structure tag and variable_name is the name of the pointer variable.*
- *For example:*

  *struct student_rec *ptr;*
- You can assign *ptr* a value by using the address operator & as in:

  ptr = &student1;
  - This assigns the address of the structure variable *student1* and not the address of the structure tag *student_rec*.

Pointers to Structures

- You can refer to the members of a structure variable by using the dereferencing operator *.
- For example: *(*ptr).number* will access the student's number.
- The brackets are necessary because the selection operator . has a higher priority than the dereferencing operator *.
- Without the brackets, you are attempting to access the memory location given by *ptr.number*. This is invalid as *ptr* is not a structure and *number* is not a member of *ptr*.

Pointers to Structures

- C provides us with a much easier notation for accessing the members of a structure.
- The arrow notation -> can be used in place of the dot notation.
- The following are equivalent:

  ptr -> number and (.ptr).number
- In english, *ptr -> number* reads as "the member number of the structure pointed to by *ptr*".

Initialising a Structure Variable

- The members of a structure variable can be initialised by placing the initial value between brackets in the same way that array elements are initialised.
- *struct student_rec /* structure template */*
- *{*
- *int number ;*
- *char surname[21] ;*
- *char first_name[11] ;*
- *int scores[5] ;*
- *} ;*

Initialising a Structure Variable

- *struct student_rec student =*
- *{1234, "Jones", "John", { 50, 60 , 45 , 65, 75 } };*

Structures and Pointers – Sample Program

```
#include <stdio.h>
main()
{
    struct student_rec    /* structure template */
    {
        int number ;
        char surname[21] ;
        char first_name[11] ;
        int scores[5] ;
    } ;
```


Structures and Pointers – Sample Program

```
struct student_rec student =
{1234, "Jones", "John", { 50, 60 , 45 , 65, 75 } };
struct student_rec *ptr ;
int i ;
ptr = &student ; /* ptr points to student */
printf( "\nThe values in student are:" ) ;
printf( "\nNumber is %d" , ptr -> number ) ;
printf( "\nSurname is %s" , ptr -> surname ) ;
printf( "\nFirst name is %s" , ptr -> first_name ) ;
printf( "\nScores are: " ) ;
for ( i= 0 ; i < 5 ; i++ )
  printf( " %d " , ptr -> scores[i] ) ;
printf( "\n" ) ;
}
```

Passing a Structure to a Function

- When you a pass a structure variable to a function, you pass a copy of the member values to the function. Therefore, the values in the structure cannot be changed within the function.
- As with any variable, to change the values in a structure variable from within a function you must pass to the function a pointer to the structure variable.
- It is faster to pass a pointer to a structure than a copy of all the values in the structure variable.

Passing a Structure to a Function – Sample Program

```
#include <stdio.h>

struct student_rec  /* Global structure template */
{
    int number ;
    char surname[21] ;
    char first_name[11] ;
    int scores[5] ;
} ;
```


Passing a Structure to a Function – Sample Program

```
#include <stdio.h>
void display_student_data( struct student_rec student );
void get_student_data( struct student_rec * ptr ) ;

main()
{
    struct student_rec student ;
    struct student_rec *student_ptr ;

    student_ptr = &student ;

    /* Use a pointer to a structure variable as an argument */
    get_student_data( student_ptr ) ;

    /* Use a structure variable as an argument */
    display_student_data( student ) ;
}
```


Passing a Structure to a Function – Sample Program

```
void display_student_data (const struct student_rec student)
{
    int i;
    printf( "\nThe data in the student structure is:" ) ;
    printf( "\nNumber is %d" , student.number ) ;
    printf( "\nSurname is %s" , student.surname ) ;
    printf( "\nFirst name is %s" , student.first_name ) ;
    printf( "\nScores are: " ) ;
    for ( i= 0 ; i < 5 ; i++ )
      printf( " %d " , student.scores[i] ) ;
    printf( "\n" ) ;
}
```

Passing a Structure to a Function – Sample Program

```
void get_student_data( struct student_rec *ptr )
{
    int i ;
    printf( "Number: " ) ;
    scanf( "%d" , &(ptr->number) ) ;
    printf( "Surname: " ) ;
    scanf( "%20s" , ptr->surname ) ;
    printf( "First name: " ) ;
    scanf( "%10s" , ptr->first_name ) ;
    printf( "Five test scores: " ) ;
    for ( i= 0 ; i < 5 ; i++ )
     scanf( "%d" , &(ptr->scores[i]) ) ;
}
```


Nested Structures

- A nested structure is structure that contains another structure as one of its members.
- For example, a company employee record might consist of the employee's date of birth and start date. Both of these dates can be represented by a structure consisting of the members day, month and year.
- First declare the structure template for a date:

```
struct date
{
    int day;
    int month;
    int year;
};
```

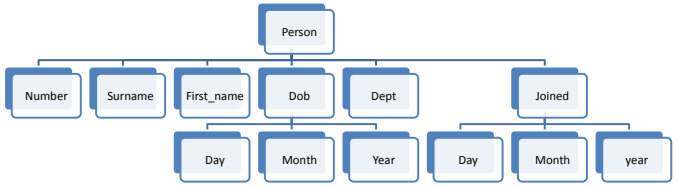

Nested Structures

- Next, the structure employee template is declared in terms of the previously declared structure template *date*.

```
struct employee
{
    long number;
    char surname[21];
    char first_name[11];
    struct date job;
    int department;
    struct date joined;
};
```


Nested Structures

- Finally, define a variable *person* of the type *struct employee*: *struct employee person;*



Nested Structures

- To access the date of birth of the employee, you would use:

  person.dob.day
  person.dob.month
  person.dob.year

- These can also be accessed by using a pointer.

```
struct employee *ptr = &person;
ptr -> dob.day;
ptr -> dob.month;
ptr-> dob.year;
```


The Typedef Statement

- ***typedef*** allows you to define a synonym for an existing data type.
- For example: typedef int * INT_POINTER;
- This creates a synonym INT_POINTER for the data type int *.
- The synonym INT_POINTER can then be used in place of int * anywhere in the program.
- For example: int *p1, *p2, *p3;
- Can now be written as INT_POINTER p1, p2, p3;
- Usually a synonym is written in capital letters.

The Typedef Statement

- You can also use typedef with structures.
- Using the previous employee example if you use typedef for the date then you could define the employee structure as follows:

```
typedef struct date DATE;
struct employee
{
    long number;
    char surname[21];
    char first_name[11];
    DATE job;
    int department;
    DATE joined;
};
```


The Typedef Statement

- Going a step further, you can write

  typedef struct employee EMPLOYEE;

- And declare the variable *person* as:
- *EMPLOYEE person;*

Arrays of Structures

- Continuing with the employee example:

  struct employee persons[5]; or

  EMPLOYEE persons[5];
- *Defines a five-element array persons. Each element of this array is of the type struct employee, with members number*, *surname*, *first_name*, *dob*, *dept*, and *joined*.
- The members *dob* and *joined* are themselves structures and have members *day*, *month* and *year*.
- Note that *persons[0].number* will access the employee number of the first employee and persons[4].joined.year will access the year of joining of the fifth employee.

GMIT
INSTITIÚID TEICNEOLAÍOCHTA NA GAILLIMHE-MAIGH EO
GALWAY-MAYO INSTITUTE OF TECHNOLOGY

33