C Programming

Structures

Initialising a Structure

Initialising a Structure is done so in a similar way to initialising an array. Let's look at the student_rec structure and initialise it.

```
//Structure template(s)
struct student_rec
{
      int student_ID;
      char firstname[11];
      char surname[21];
      int results[5];
};
```

The above is a Structure Template and is not a variable. The template simply describes the blueprint/makeup of a structure variable for this structure.

Therefore, we first need to create a variable of this template and initialise it to contain data. We will do this as follows:

```
int main()
{
   struct student_rec student = {   1234,
                                    "Joe",
                                    "Murphy",
                                    {54, 63, 77, 90, 51}
                                };
      ...
      ...
      ...

} // end main
```

Pointers to Structures

The general format for defining a pointer to a Structure variable is:

```
struct tag_name *variable_name;
```

Using the previous code example, let's create a pointer to the structure variable.

```
struct student_rec *ptr;
```

Let's combine this into a short program and use the pointer variable to point at a student struct variable and access its contents.

```
/*
Pointer variables and Structures
*/
#include <stdio.h>
#include <string.h>

#define SIZE 5

//Structure template(s)
struct student_rec
{
      int student_ID;
      char firstname[11];
      char surname[21];
      int results[5];
};

//Function signature(s)
// ...
// …

int main()
```

```
{
    struct student_rec student = {  1234,
                                    "Joe",
                                    "Murphy",
                                    {54, 63, 77, 90, 51}
                                 };


    struct student_rec *ptr;


    //Make ptr point at the structure variable called 'student'
    ptr = & student;

    printf("\nStudent Record\n");
    printf("\nID is: %d", student.student_ID);

    //The following 2 lines of code are the same
    printf("\nID is: %d", (*ptr).student_ID);
    //can also be written
    printf("\nID is: %d", ptr -> student_ID);

    // Display the firstname
    printf("\n\nFirstname: %s", ptr -> firstname);

    // Display the surname
    printf("\nSurname is: %s", ptr -> surname);

    //Display the results
    printf("\nResults are:\n");

    for(i = 0; i < SIZE; i++)
    {
        printf("%d\n", ptr -> results[i]);
    } // end for
```

```
  return 0;

} // end main()
```

Repl 21.1: https://replit.com/@michaelTUDublin/211-Pointer-to-a-structure

Passing a structure to a function

When you pass a structure parameter to a function, it follows the same rules as passing any other regular variable parameter, i.e., you can pass the structure parameter using either (i) Pass by Value, (ii) Pass by Reference

Let's have a look at passing a Structure variable as a parameter using both Pass by Value and Pass by Reference

1. Pass by Value

```
/*
Pointer variables and Structures
*/
#include <stdio.h>
#include <string.h>


#define SIZE 5


//Structure template(s)
struct student_rec
{
   int student_ID;
   char firstname[11];
   char surname[21];
   int results[5];
};


//Function signature(s) Pass by Value, a COPY is passed
void display(struct student_rec);

int main()
{
```

```
  struct student_rec student = {  1234,
                                  "Joe",
                                  "Murphy",
                                  {54, 63, 77, 90, 51}
                               };

   //Display the contents of the structure variable student
   display(student);

   return 0;

} // end main()

/*
Function display is used to display the contents of a structure
variable parameter
*/
void display(struct student_rec stu)
{
   int i;

   printf("\nStudent Record\n");
   printf("\nID is: %d", stu.student_ID);

   // Display the first name
   printf("\n\nFirstname: %s", stu.firstname);
   // Display the surname
   printf("\nSurname is: %s", stu.surname);

   //Display the results
   printf("\nResults are:\n");

   for(i = 0; i < SIZE; i++)
   {
       printf("%d\n", stu.results[i]);
```

```
    } // end for

} // end display()
```

Repl 21.2: https://replit.com/@michaelTUDublin/212-Pass-by-Value-structures

2. Pass by Reference

Let's modify the code above, which uses Pass by Value, and this time pass the student parameter using Pass by Reference.

Here we go …

```
/*
Pointer variables and Structures
*/
#include <stdio.h>
#include <string.h>


#define SIZE 5


//Structure template(s)
struct student_rec
{
   int student_ID;
   char firstname[21];
   char surname[21];
   int results[5];
};


//Function signature(s)
Pass by Value, a COPY is passed
```

```c
void display(struct student_rec);

Pass by Reference, the address location is passed
void enter(struct student_rec *);


int main()
{
   struct student_rec student;
   // Enter the data into the structure student variable
   enter(& student);

   //Display the contents of the structure variable student
   display(student);

   return 0;
} // end main()



/*
Function enter is used to enter the contents of a structure
variable parameter
*/
void enter(struct student_rec *ptr)
{
   int i;


   printf("\nEnter student ID: ");
   scanf("%d", & ptr -> student_ID);

   //scanf("%d", & (*ptr).student_ID);

   // Clear the stdin buffer so that the 'Enter' key is not being
read
```

```
    while(getchar() != '\n');

    printf("\nEnter first name: ");
    //scanf("%s", ptr -> firstname);
    fgets(ptr -> firstname, 21, stdin);

    printf("\nEnter surname: ");
    //scanf("%s", ptr -> surname);
    fgets(ptr -> surname, 21, stdin);

    printf("\nEnter %d results\n", SIZE);

    // Enter the results
    for(i = 0; i < SIZE; i++)
    {
        scanf("%d", & ptr -> results[i]);

    } // end for

} // end enter()



/*
Function display is used to display the contents of a structure
variable parameter
*/
void display(struct student_rec stu)
{

    int i;

    printf("\nStudent Record\n");
    printf("\nID is: %d", stu.student_ID);

    // Display the first name
```

```
    printf("\n\nFirstname: %s", stu.firstname);
    // Display the surname
    printf("\nSurname is: %s", stu.surname);

    //Display the results
    printf("\nResults are:\n");

    for(i = 0; i < SIZE; i++)
    {
        printf("%d\n", stu.results[i]);

    } // end for

} // end display()
```

Repl 21.3: https://replit.com/@michaelTUDublin/213-Pass-by-Reference-structures