

C Programming

Structures

Arrays of Structures

When creating an array of Structures, each array element will be a standalone structure containing its own structure members.

For example:

```
#include <stdio.h>

#define LENGTH 11
#define S_LENGTH 21
#define SIZE 5

//Structure template(s)
struct date
{
    int day;
    int month;
    int year;
};

struct student_rec
{
    int student_ID;
    char firstname[LENGTH];
    char surname[S_LENGTH];
    int results[SIZE];
    struct date DOB;
};

// Function signature(s)
// ...

int main()
```

```
{
    int i;

    struct student_rec students[SIZE];
```

students

- student_ID - firstname - surname - results[] - DOB - day - month - year	- student_ID - firstname - surname - results[] - DOB - day - month - year	- student_ID - firstname - surname - results[] - DOB - day - month - year	- student_ID - firstname - surname - results[] - DOB - day - month - year	- student_ID - firstname - surname - results[] - DOB - day - month - year
--	--	--	--	--

0

1

2

3

4

Index (i.e. element number)

In the above example, an array called **students** is created. Each element in the array is a standalone structure containing an ID, first name, surname, set of results and a date of birth.

To enter the student_ID for each element in the array, you would write the code as follows:

```
printf("\nEnter ID for each student:");

// Enter the student ID for all elements in the array
for(i = 0; i < SIZE; i++)
{
    printf("\nEnter ID for student %d :", i+1);
    scanf("%d", & students[i].student_ID);

} // end for

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

// Enter the first name for all elements in the array
for(i = 0; i < SIZE; i++)
```

```

{
    printf("\nEnter first name for student %d :", i+1);
    fgets(students[i].firstname, LENGTH, stdin);

} // end for


printf("\nThe Student ID for each student is:");
for(i = 0; i < SIZE; i++)
{
    printf("%d ", students[i].student_ID);

} // end for


printf("\nThe Student first name for each student is: ");

for(i = 0; i < SIZE; i++)
{
    printf("\n%s", students[i].firstname);

} // end for


return 0;

} // end main()

```

Repl: 23.1: <https://replit.com/@michaelTUDublin/231-Array-of-structures>

The typedef statement

typedef allows the software developer to define a synonym (pseudo-name) for an existing data type.

For example:

```
/*
Nested Structures
*/
#include <stdio.h>

#define LENGTH 11
#define S_LENGTH 21
#define SIZE 5

typedef char STRING;

//Structure template(s)
struct date
{
    int day;
    int month;
    int year;
};

struct student_rec
{
    int student_ID;
    STRING firstname[LENGTH];
    char surname[S_LENGTH];
    int results[SIZE];
    struct date DOB;
};
```

```

// Function signature(s)
// ...

int main()
{
    int i;

    //Declare a typedef
    typedef int* INT_POINTER;

    int *ptr;
    INT_POINTER ptr2;

    STRING sentence[LENGTH] = "Hello";

    printf("\n%s", sentence);

    return 0;

} // end main()

```

Repl 23.2: <https://replit.com/@michaelTUDublin/232-Typedef>

Note:

You can create a structure variable directly from the template itself as follows:

```

struct
{
    int student_ID;
    char firstname[LENGTH];
    char surname[S_LENGTH];
    int results[SIZE];
    struct date DOB;

```

```
} stu, stu2, stu3, .. ..;
```

However, there are two points to note:

1. If the above code is declared outside any function, then both the structure template and the variable(s), i.e., stu, etc., are **Global**.
2. If the above code is declared inside any function, then both the structure template and the variable(s), i.e., stu, etc., are **Local** to that function and other functions cannot create their own structure variables of that template. Also, the local structure variable(s) cannot be passed as a parameter.

Programming pitfalls

1. You cannot compare two structures of the same structure template using a single equivalence

e.g.

```
struct student_rec
{
    int student_ID;
    char firstname[LENGTH];
    char surname[S_LENGTH];
    int results[SIZE];
    struct date DOB;
};

struct student_rec stu1, stu2;
```

Assuming both stu1 and stu2 contain their own data in each structure member, you **cannot** compare these two variables in a single if statement such as:

```
if (stu1 == stu2)
{
    ...
}
```

Instead, you need to compare each individual structure member separately. So, you might try this instead:

```
if (stu1.student_ID == stu2.student_ID)
{
    ...
}
```

2. When you are using the arrow notation with pointers to structures, there is no space between the arrow characters, i.e., use -> and not - >