

# Programming For Engineers

## Structure Data Type

by

Wan Azhar Wan Yusoff<sup>1</sup>, Ahmad Fakhri Ab. Nasir<sup>2</sup>  
Faculty of Manufacturing Engineering  
wazhar@ump.edu.my<sup>1</sup>, afakhri@ump.edu.my<sup>2</sup>



# 0.0 Chapter's Information

- **Expected Outcomes**
  - To create and use structure data type programming.
- **Contents**
  - 1.0 Structure data type
  - 2.0 Nested structure data type
  - 3.0 Array of structure data type
  - 4.0 typedef structure data type
  - 5.0 Structure and Pointer
  - 6.0 Summary



# 1.0 Structure Data Type


- We know that an array data type can group many elements of similar data type under one variable. For example, when we define an array `int score[10]`, we have grouped together 10 integer values under variable `score`. However, an array cannot group elements of different data types under one variable. For example, we cannot group 5 integers and 3 doubles under one array variable.
- A structure is a derived data type that groups elements of different data types under one variable. We can group integer, character, double etc. under one variable.
- We give an example below to show the idea of structure data type.



# 1.0 Structure Data Type

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

int main()
{
    //Create new data type
    struct student {
        int ID;
        char name[100];
        double CGPA;
    };
    //Declare Ali as struct student
    struct student Ali;
    //Set values to Ali
    Ali.ID = 224;
    strcpy(Ali.name, "Ali Bin Ahmad");
    Ali.CGPA = 3.14;
    //Display values using dot operator
    printf("Name:\t%s\n", Ali.name);
    printf("ID:\t%d\n", Ali.ID);
    printf("CGPA:\t%f\n", Ali.CGPA);
    return 0;
}
```

 "D:\Wan-Azhar-Wan-Yusoff\WAWY-Programming-for-Engineers\Course Notes\Assessm

```
Name:  Ali Bin Ahmad
ID:    224
CGPA:  3.140000
```

```
Process returned 0 (0x0)  execution time : 0.122 s
Press any key to continue.
```



# 1.0 Structure Data Type

- In the previous program, we define student structure by using a C keyword “struct”. Then, we group 3 different elements inside the structure namely, ID, name and CGPA. Notice that, we group 3 different data types i.e. an integer, an array of 100 characters (string) and a double. Notice also that we define structure data type called “student” as a new data type. This does not create the variable. It is only the data type. The variable is created when we declare the variable such as “struct student Ali”. We can say that Ali is a struct student data type.



# 1.0 Structure Data Type

- How do we access the structure variable? We use “dot” operator to access the variable. For example, if we want to set the ID value for Ali, we write “Ali.ID = 0224;”. We need to be very careful for string. We cannot write “Ali.name = Ali Bin Ahmad;”. **THIS IS WRONG!** In C, we use a function strcpy() in <string.h> to set string variable. Thus, we write strcpy(Ali.name, “Ali Bin Ahmad”); This will copy the string.



## 2.0 Nested Structure Data Type

- We can nest structure data type by making structure data type inside another structure data type. A student is part of university employee. Thus, we can make a student inside university employee data type together with staff data type. We give an example below.





## 2.0 Nested Structure Data Type

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    struct student {
        int ID;
        char name[100];
        double CGPA;
    };
    struct staff {
        int ID;
        char name[100];
        double salary;
    };
    struct employee{
        struct staff wawy;
        struct student ali;
        int employeeNumber;
    };
    struct employee ump;
    strcpy(ump.wawy.name, "Wan Yusoff");
    ump.wawy.ID = 224;
    ump.wawy.salary = 2500.0;
    ump.ali.ID = 231;
    strcpy(ump.ali.name, "Wan Ali");
    ump.ali.CGPA = 3.14;
    printf("UMP Staff: %s\tID: %d\tSalary: %f\n", ump.wawy.name, ump.wawy.ID, ump.wawy.salary);
    printf("UMP Student: %s\tID: %d\tCGPA: %f\n", ump.ali.name, ump.ali.ID, ump.ali.CGPA);
    return 0;
}
```

F:\Wan-Azhar-Wan-Yusoff\WAWY-Programming-for-Engineers\WAWY\_C\_Programming\_Tutor

```
UMP Staff: Wan Yusoff    ID: 224 Salary: 2500.000000
UMP Student: Wan Ali    ID: 231 CGPA: 3.140000

Process returned 0 (0x0)    execution time : 0.016 s
Press any key to continue.
```





## 2.0 Nested Structure Data Type

- In the previous example, we nest the “struct student” and “struct staff” into “struct employee”. Then, we can include student and staff into our employee structure.



# 3.0 Array of Structure Data Type

- We can also make collection of similar structure data type by using an array of structure. We show an example of such array below.

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

```
int main()
```

```
{
```

```
    struct student {
```

```
        int ID;
```

```
        char name[100];
```

```
        double CGPA;
```

```
    };
```

```
    struct student FKP[2];
```

```
    FKP[0].ID = 0224;
```

```
    FKP[0].CGPA = 3.14;
```

```
    strcpy(FKP[0].name, "Ali Bin Ahmad");
```

```
    FKP[1].ID = 0324;
```

```
    FKP[1].CGPA = 3.67;
```

```
    strcpy(FKP[1].name, "Abu Bin Ahmad");
```

```
    int i;
```

```
    for (i=0;i<2;i++)
```

```
    {
```

```
        printf("Name: %s\tID: %d\tCGPA: %f\n", FKP[i].name, FKP[i].ID, FKP[i].CGPA);
```

```
    }
```

```
    return 0;
```

```
}
```

```
F:\Wan-Azhar-Wan-Yusoff\WAWY-Programming-for-Engineers\WAWY_C_Programming_Tu
```

```
Name: Ali Bin Ahmad      ID: 148 CGPA: 3.140000
```

```
Name: Abu Bin Ahmad      ID: 212 CGPA: 3.670000
```

```
Process returned 0 (0x0)   execution time : 0.047 s
```

```
Press any key to continue.
```



## 3.0 Structure Data Type

- In the previous example, we create two collection of student structure data type. We then, fill up the data structure and display them on the screen. Notice the use of subscript to access the data structure.



# 4.0 typedef Structure Data Type

- Some programmers prefer to use “struct student Ali” to remind us that Ali is a “struct student” type. However, some practice use STUDENT Ali so that we know Ali is a typedef STUDENT. What they do is to declare struct as typedef struct? We give an example below.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    struct student {
        int ID;
        char name[100];
        double CGPA;
    };
    typedef struct {
        int ID;
        char name[100];
        double salary;
    }STAFF;
    struct student Ali;
    STAFF wawy;
    Ali.CGPA = 2.54;
    wawy.salary = 3425.45;
    printf("CGPA Ali = %f\n",Ali.CGPA);
    printf("Salary wawy = %f\n",wawy.salary);
    return 0;
}
```

F:\Wan-Azhar-Wan-Yusoff\WAWY-Programming-for-Engineers\WAWY\_C\_Programming

```
CGPA Ali = 2.540000
Salary wawy = 3425.450000

Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```



## 4.0 typedef Structure Data Type

- In the above example, we DEFINE two structure data types: student and staff. But, in the second definition, we use typedef so that we can use the name during declaration. We usually use CAPITAL LETTER for typedef definition. Now, for declaration we use:

```
struct student Ali;  
STAFF wawy;
```

- Using typedef, it is easier to declare because it is shorter. But, by using struct, we are reminded about the variable is a struct data type.



# 5.0 Structure Data Type and Pointer

- As with any function calling, we can either pass variables (parameters) to a function as values or as references. When we pass by reference, we pass the top address of the parameter. Similarly, when we pass a structure to a function, we can pass either as value or as reference (address). An example below illustrates the point.





# 5.0 Structure Data Type and Pointer

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int ID;
    char name[100];
    double salary;
}STAFF;

int main()
{
    STAFF wawy;
    wawy.salary = 2500.0;
    printf("Before function, the Salary: %f\n",wawy.salary);
    initialize(wawy);
    printf("After function, the Salary: %f\n",wawy.salary);
    return 0;
}

void initialize(STAFF wawy)
{
    wawy.salary = 1800.0;
    printf("During function, the Salary: %f\n",wawy.salary);
}
```

"F:\Wan-Azhar-Wan-Yusoff\WAWY-Programming-for-Engineers\New folder\wawyStructF

```
Before function, the Salary: 2500.000000
During function, the Salary: 1800.000000
After function, the Salary: 2500.000000
```

```
Process returned 0 (0x0)   execution time : 0.033 s
Press any key to continue.
```



## 5.0 Structure Data Type and Pointer

- In the previous program, we pass a structure STAFF to the function initialize(). But, we pass by value. Then, we change the value of wawy salary. We notice that because we only pass by value, the value of wawy salary does not change outside the function, as shown by the output. In order to change the value, we need to pass the STAFF as reference so that the function has the address. An example below shows the passing by reference.



# 5.0 Structure Data Type and Pointer

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int ID;
    char name[100];
    double salary;
}STAFF;

int main()
{
    STAFF wawy;
    wawy.salary = 2500.0;
    printf("Before function, the Salary: %f\n",wawy.salary);
    initialize(&wawy);
    printf("After function, the Salary: %f\n",wawy.salary);
    return 0;
}

void initialize(STAFF* wawy)
{
    wawy->salary = 1800.0;
    printf("During function, the Salary: %f\n",wawy->salary);
}
```

□ "F:\Wan-Azhar-Wan-Yusoff\WAWY-Programming-for-Engineers\New folder\wawyStruct

```
Before function, the Salary: 2500.000000
During function, the Salary: 1800.000000
After function, the Salary: 1800.000000
```

```
Process returned 0 (0x0)   execution time : 0.019 s
Press any key to continue.
```



# 5.0 Structure Data Type and Pointer

- In the previous example, we pass the structure STAFF to the function by reference i.e. &wawy. The receiving function also receives the structure as pointer i.e. STAFF\* wawy.

## IMPORTANT POINT!!!!!!

- We access structure pointer NOT BY USING DOT OPERATOR. We access pointer structure by using an ARROW OPERATOR i.e.” ->” a dash and a greater than symbol. For example, wawy->salary = 1800.0 instead of wawy.salary = 1800.0. THIS IS AN IMPORTANT DIFFERENCE.



## 6.0 Summary

- In this note, we have learned about a very important data type which is the structure data type. We know that a structure data type is a collection of different data type including another structure data type and also an array of data type. We also know that sometimes it is easier to define as typedef so that when we declare a structure data type, we don't have to type the word "struct". Lastly, a very important C language is to use the pointer to struct as a mechanism to pass address to function. We use different notation to access the member data i.e. the arrow operator -> as the operator for structure pointer. We will learn more.



# 6.0 Summary

- Competencies that you need to know about struct data type: You must be able to:
  1. Define structure data type
  2. Define pointer structure data type.
  3. Define typedef structure data type.
  4. Access members of structure data type using dot operator.
  5. Access members of structure data type using arrow operator.
  6. Use nested structure data type.
  7. Use array of structure data type
  8. Pass structure data type to function and return a structure from a function.

