United International University

Dept. of Electrical and Electronic Engineering (EEE)

Course No. : **EEE 122**
Course Title: **Structured Programming Laboratory**

Lab Sheet 8
Structure in C

Outcomes

After finishing this lab students should be able to ...

1. Create and use structures, unions and enumerations.
2. Pass structures to functions by value and by reference.
3. Use `typedef` to create aliases for existing type names.
4. Manipulate data with the bitwise operators.

Contents

1 C - Structure **1**
1.1 USES OF STRUCTURES IN C: . . . 1
1.2 Defining a Structure in C . . . 2
1.3 Structure variable declaration . . . 3
1.4 Accessing members of a structure . . . 3
1.5 Keyword typedef while using structure . . . 3
1.6 Structures within structures . . . 4
1.7 Passing structures to a function . . . 4

2 Programming Examples **7**

3 Practice session **10**

4 Lab Assignments **11**

1 C - Structure

1.1 USES OF STRUCTURES IN C:

i. C Structures can be used to store huge data. Structures act as a database.

ii. C Structures can be used to send data to the printer.

iii. C Structures can interact with keyboard and mouse to store the data.

iv. C Structures can be used in drawing and floppy formatting.

v. C Structures can be used to clear output screen contents.

vi. C Structures can be used to check computers memory size etc.

Structure is a collection of variables of different types under a single name.

Arrays allow to define type of variables that can hold several data items of the same kind. Similarly **`structure`** is another user defined data type available in C that allows to combine data items of different kinds.

Structures are used to represent a record.
For example: You want to store some information about a person: his/her name, citizenship number and salary. You can easily create different variables **`name, citNo, salary`** to store these information separately.

However, in the future, you would want to store information about multiple persons. Now, you'd need to create different variables for each information per person: **`name1, citNo1, salary1, name2, citNo2, salary2`**

You can easily visualize how big and messy the code would look. Also, since no relation between the variables (information) would exist, it's going to be a daunting task.

A better approach will be to have a collection of all related information under a single name **`Person`**, and use it for every person. Now, the code looks much cleaner, readable and efficient as well.

This collection of all related information under a single name **`Person`** is a structure.

1.2 Defining a Structure in C

Keyword **`struct`** is used for creating a structure.

Syntax of structure

```
struct structure_name {
data_type member1;
data_type member2;
.
.
data_type memeber;
};
```

Note: Don't forget the semicolon `};` in the ending line.

We can create the structure for a person as mentioned above as:

```
struct person{
  char name[50];
  int citNo;
  float salary;
};
```

This declaration above creates the derived data type **`struct person`**.

Here is the another way you would declare Book structure

```
struct books {
  char  title[50];
  char  author[50];
  char  subject[100];
  int   book_id;
};
```

This declaration above creates the derived data type **struct books**.

1.3 Structure variable declaration

When a structure is defined, it creates a user-defined type but, no storage or memory is allocated.

For the above structure of a person, variable can be declared as:

```
struct person{
  char name[50];
  int citNo;
  float salary;
};

int main(void){
  struct person person1, person2, person3[20];
  return 0;
}
```

Another way of creating a structure variable is:

```
struct person{
  char name[50];
  int citNo;
  float salary;
} person1, person2, person3[20];
```

In both cases, two variables **person1, person2** and an array **person3** having 20 elements of type **struct person** are created.

1.4 Accessing members of a structure

There are two types of operators used for accessing members of a structure.

i. Member operator(.)

ii. Structure pointer operator(-¿)

Any member of a structure can be accessed as:

```
structure_variable_name.member_name
```

Suppose, we want to access salary for variable **person2**. Then, it can be accessed as:

```
person2.salary
```

1.5 Keyword typedef while using structure

Writing **struct structure_name variable_name**; to declare a structure variable isn't intuitive as to what it signifies, and takes some considerable amount of development time.

So, developers generally use typedef to name the structure as a whole. For example:

```
typedef struct complex{
  int imag;
  float real;
} comp;

int main(void){
  comp comp1, comp2;
}
```

Here, **typedef** keyword is used in creating a type **comp** (which is of type as **struct complex**).

Then, two structure variables **comp1** and **comp2** are created by this **comp** type.

1.6 Structures within structures

Structures can be nested within other structures in C programming.

```
struct complex{
  int image_value;
  float real_value;
};

struct number{
  struct complex comp;
  int real;
} num1, num2;
```

Suppose, you want to access **image_value** for **num2** structure variable then, following structure member is used.

```
num2.comp.imag_value
```

1.7 Passing structures to a function

There are mainly two ways to pass structures to a function:

1. Passing by value
2. Passing by reference

Passing structure by value

A structure variable can be passed to the function as an argument as a normal variable.

If structure is passed by value, changes made to the structure variable inside the function definition does not reflect in the originally passed structure variable.

Example : C program to create a structure student, containing name and roll and display the information.

```
#include <stdio.h>
struct student{
  char name[50];
  int roll;
};
```

```
void display(struct student stu);
// function prototype should be below to the structure declaration
//otherwise compiler shows error

int main(void){
  struct student stud;
  printf("Enter student's name: ");
  scanf("%s", &stud.name);
  printf("Enter roll number:");
  scanf("%d", &stud.roll);
  display(stud);   // passing structure variable stud as argument
  return 0;
}
void display(struct student stu){
  printf("Output\nName: %s",stu.name);
  printf("\nRoll: %d",stu.roll);
}
```

Output:

```
Enter student's name: Kevin Amla
Enter roll number: 149
Output
Name: Kevin Amla
Roll: 149
```

Passing structure by reference

The memory address of a structure variable is passed to function while passing it by reference.

If structure is passed by reference, changes made to the structure variable inside function definition reflects in the originally passed structure variable.

Example: C program to add two distances (feet-inch system) and display the result without the return statement.

```
#include <stdio.h>
struct distance{
  int feet;
  float inch;
};
void add(struct distance d1,struct distance d2, struct distance *d3);

int main(void){
  struct distance dist1, dist2, dist3;

  printf("First distance\n");
  printf("Enter feet: ");
  scanf("%d", &dist1.feet);
  printf("Enter inch: ");
  scanf("%f", &dist1.inch);

  printf("Second distance\n");
  printf("Enter feet: ");
  scanf("%d", &dist2.feet);
```

```
  printf("Enter inch: ");
  scanf("%f", &dist2.inch);

  add(dist1, dist2, &dist3);

  //passing structure variables dist1 and dist2 by value
  //whereas passing structure variable dist3 by reference
  printf("\nSum of distances = %d\'-%.1f\"", dist3.feet, dist3.inch);

  return 0;
}
void add(struct distance d1,struct distance d2, struct distance *d3) {
  //Adding distances d1 and d2 and storing it in d3
  d3->feet = d1.feet + d2.feet;
  d3->inch = d1.inch + d2.inch;

  // if inch is greater or equal to 12, converting it to feet.
  if (d3->inch >= 12) {
    d3->inch -= 12;
    ++d3->feet;
  }
}
```

Output:

```
First distance
Enter feet: 12
Enter inch: 6.8
Second distance
Enter feet: 5
Enter inch: 7.5

Sum of distances = 18'-2.3"
```

2 Programming Examples

Example: 1

Description: Write a C program to add two distances entered by user. Measurement of distance should be in inch and feet. (Note: 12 inches = 1 foot)

Source Code

```c
#include <stdio.h>
struct Distance{
  int feet;
  float inch;
} dist1, dist2, sum;

int main(void){
  printf("1st distance\n");

// Input of feet for structure variable dist1
  printf("Enter feet: ");
  scanf("%d", &dist1.feet);

// Input of inch for structure variable dist1
  printf("Enter inch: ");
  scanf("%f", &dist1.inch);

  printf("2nd distance\n");

// Input of feet for structure variable dist2
  printf("Enter feet: ");
  scanf("%d", &dist2.feet);

// Input of feet for structure variable dist2
  printf("Enter inch: ");
  scanf("%f", &dist2.inch);

  sum.feet = dist1.feet + dist2.feet;
  sum.inch = dist1.inch + dist2.inch;

  if (sum.inch > 12) {
//If inch is greater than 12, changing it to feet.
    ++sum.feet;
    sum.inch = sum.inch - 12;
  }

// printing sum of distance dist1 and dist2
  printf("Sum of distances = %d\'-%.1f\"",
                  sum.feet, sum.inch);
  return 0;
}
```

Output

```
1st distance
Enter feet: 12
Enter inch: 7.9
2nd distance
Enter feet: 2
Enter inch: 9.8
Sum of distances = 15'-5.7"
```

Example: 2

Description: Write a program to store and access id, name and percentage for student.

Source Code

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

struct student {
  int id;
  char name[20];
  float percentage;
};

int main(void) {
  struct student record = {0}; //Initializing to null

  record.id=1;
  strcpy(record.name, "Raju");
  record.percentage = 86.5;

  printf(" Id is: %d \n", record.id);
  printf(" Name is: %s \n", record.name);
  printf(" Percentage is: %f \n", record.percentage);
  return 0;
}
```

Output

```
Id is: 1
Name is: Raju
Percentage is: 86.500000
```

Example: 3

Description: Write a C program that is used for Demonstrates using arrays of structures.

Source Code

```
#include <stdio.h>
/* Define a structure to hold entries. */
struct entry {
  char fname[20];
  char lname[20];
  char phone[10];
};
struct entry list[4];
int i;
int main(void){
  /* Loop to input data for four people. */
  for (i = 0; i < 4; i++){
    printf("\nEnter first name: ");
    scanf("%s", list[i].fname);
    printf("Enter last name: ");
    scanf("%s", list[i].lname);
    printf("Enter phone in 123-4567 format: ");
    scanf("%s", list[i].phone);
  }
  printf("\n\n");
  for (i = 0; i < 4; i++){
    printf("Name: %s %s", list[i].fname, list[i].l
    printf("\t\tPhone: %s\n", list[i].phone);
  }
  return 0
}
```

Output

```
Enter first name: Bradley
Enter last name: Jones
Enter phone in 123-4567
format: 555-1212

Enter first name: Peter
Enter last name: Aitken
Enter phone in 123-4567
format: 555-3434

Enter first name: Melissa
Enter last name: Jones
Enter phone in 123-4567
format: 555-1212

Enter first name: Deanna
Enter last name: Townsend
Enter phone in 123-4567
format: 555-1234

Name: Bradley Jones
Phone: 555-1212

Name: Peter Aitken
Phone: 555-3434

Name: Melissa Jones
Phone: 555-1212

Name: Deanna Townsend
Phone: 555-1234
```

3 Practice session

Sl	Source Code

Practice 1	`#include <stdio.h>`
	`struct A{`
	`  int marks;`
	`  char grade;`
	`} A1;`
	`int main(void){`
	`  struct A B1;`
	`  A1.marks=80;`
	`  A1.grade='A';`
	`  printf("Marks=%d\t",A1.marks);`
	`  printf("Grade=%c\n",A1.grade);`
	`  B1=A1;`
	`  printf("Marks=%d\t",B1.marks);`
	`  printf("Grade=%c\n",B1.grade);`
	`  return 0;`
	`}`

Practice 2	`#include <stdio.h>`
	`struct tag{`
	`  int i;`
	`  char c;`
	`};`
	`void func(struct tag v);`
	`int main(void){`
	`  struct tag var={2,'s'};`
	`  func(var);`
	`  return 0;`
	`}`
	`void func(struct tag v){`
	`  printf("%d %c\n",v.i,v.c);`
	`}`

4 Lab Assignments

1. Declare the C structures for the following scenario:
    i. College contains the following fields: College code (2characters), College Name, year of establishment, number of courses.
    ii. Each course is associated with course name (String), duration, number of students. (A College can offer 1 to 50 such courses)
2. Write a C program using array of structure to create employee records with the following fields: **`emp-id, name, designation, address, salary`** and display it.

Acknowledgment

First OBE version, prepared by:
B.K.M. Mizanur Rahman,
Assistant Professor,
Department of EEE, UIU

Second Update , prepared by:
Nazmul Alam,
Part Time Faculty,
Department of EEE, UIU