# 3.3 Structures

## Objectives

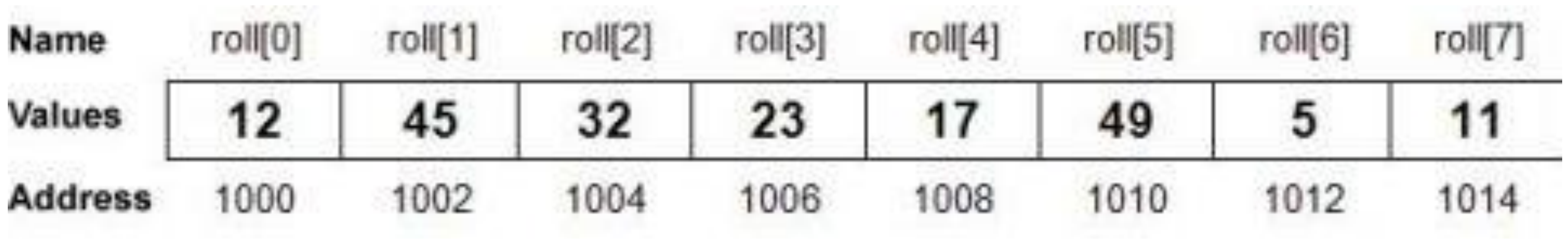
- To give an introduction to Structures
- To clearly distinguish between Structures from Arrays
- To explain the scenarios which require Structures
- To illustrate the syntax and usage of Structures with examples
- To ‘simplify’ Complex Structures

## Agenda

- Recap on Arrays
- Introduction
- Arrays vs. Structures
- Structures in detail
- Complex Structures

# Recap on Arrays

- Recap : What is an **Array** ?
  - A contiguously stored set of memory locations to be used to store elements having the same data type.
  - Example:



| Name | roll[0] | roll[1] | roll[2] | roll[3] | roll[4] | roll[5] | roll[6] | roll[7] |
|---|---|---|---|---|---|---|---|---|
| Values | 12 | 45 | 32 | 23 | 17 | 49 | 5 | 11 |
| Address | 1000 | 1002 | 1004 | 1006 | 1008 | 1010 | 1012 | 1014 |

Image Credits - http://hubpages.com/technology/Array-in-C-programming--Programmers-view

## Introduction

- What are **Structures**?
  - A structure is a collection of related elements, possibly of different types, having a single name [1].
  - A **struct** (keyword), is a complex data type declaration that defines a physically grouped list of variables to be placed under one name in a block of memory, allowing the different variables to be accessed via a single pointer, or the struct declared name which returns the same address.
  - Structures help to organize complicated data, particularly in large programs, because they permit a group of related variables to be treated as a unit instead of as separate entities.

## Syntax

- An example of a structure is the payroll record:
- The format of a structure definition is as follows:

```
struct TAG
{
    field 1;
    field 2;
    ------
};
```

OR

```
struct TAG
{
    field 1;
    field 2;
    ------
}variable;
```

- An employee is described by a set of attributes such as name, designation, employee id, salary, etc.
- A structure designed for this purpose would look like:

```
struct employee{
   int empid;
   char name[10];
   char designation[10];
   float salary;
}emp; /* emp is a variable of structure type empdata */
```

# Declaration types

- Two main ways for declaring a structure:

1. **Tagged Structure Format**

Ex : a)

```
struct employee{
int empno;
char name[10];
};
```

b)

```
struct employee{
int empno;
char name[10];
}emp;
```

- Here, 'employee' is the tag which is the identifier for the structure.
- In example 1.a), no variable is defined to be used with the structure which would make it just a template (no memory declared) which can be used later as a data type.
- In 1.b), a structure variable 'emp' is declared as a variable of type 'struct employee' and can be used readily.

## Declaration types

- In the case of **Tagged Structure Format** example 1.a), a separate variable declaration would be necessary, the format followed would be as shown below:

```
struct employee{
int empno;
char name[10];
};

struct employee emp;
```

## Declaration types

2. **Using typedef**

- The keyword 'typedef' precedes the 'struct' declaration

Format

```
typedef struct{
   field list;
}TYPE;
```

Example

```
typedef struct{
int empno;
char name[10];
}emp;
```

- An identifier (emp) in the example is required at the end of the block which acts as the type definition name.
- Example:

```
typedef struct{
int empno;
char name[10];
}emp;
emp matt; //local declaration
```

## Accessing a structure member

- To access an element which is a structure’s member, the direct selection (“.”) operator is used.
- Syntax:

```
<structure variable name>.<element name>
```

- Example:

```
struct employee{
int empno;
char name[10];
}emp;
emp.empno=2;
printf(“%d”,emp.empno);
```

- Pointers  as structure element:
  - Example:

```
struct employee{
int *empno;
char name[10];
}emp;
int emp_number;
emp.empno = & emp_number;
printf(“%d”,*emp.empno);
```

## Pointers to Structures

- To access structure elements through pointers, the pointer is to be declared as the structure type.
- For the structure employee, the pointer is declared as:

```
struct employee{
int empno;
char name[10];
}emp;
struct employee matt;
struct employee* ptr;
ptr=&matt;
```

- The address of the structure variable 'matt' will be stored in the pointer 'ptr'.
- To access the member of the 'matt' using 'ptr', the usage will be:

`(*ptr).empno` or using the indirect selection(->) `*ptr->empno`

## Complex Structures

- Structures can be built to includes variables, arrays, pointers and even structures of different data types.
- Two examples of a **nested structure** (structure within a structure).

```
struct employee{
        struct salary{
        int basic_pay;
        int DA;
        int HRA;
    }sal;
    char name[10];
};
struct employee emp;
emp.sal.DA=100;
printf("%d",emp.sal.DA);
```

Declared Inside

```
struct salary{
   int basic_pay;
   int DA;
   int HRA;
};
struct employee{
   struct salary sal;
   char name[10];
}emp;
emp matt;
printf("%d",emp.sal.DA);
```

Declared Outside

## Complex Structures

- **Structures with arrays**:
  - The character array "name" in the code snippet below is an example:

```
typedef struct{
int empno;
char name[10];
int devices[3];
}emp;
emp matt;
matt.devices[1]=1024;
```

- The way to access the individual members of the array inside the structure is illustrated  in the last line of the above code snippet.

# Complex Structures

- **Arrays of structures:**
  - We can declare an array of a particular type of structure as follows:

```
struct employee{
int empno;
char name[10];
}emp;
emp new[5];
new[1].name[1]='m';
```

- The way to access one particular element in the structure array is by using  the index, as shown in the last line of the above code snippet.

## Try it Yourself

- Predict the output:

1.

```
#include<stdio.h>
main()
{
   typedef struct{
   int empno;
   char name[10];
   }emp;
   emp e1,e2;
   e1.empno=5004;
   printf("%d\n",e2.empno);
}
```

Answer: Some value, like ‘0’ since e2’s empno is printed, which is not initialised.

# Try it Yourself

- Predict the output:

2.

```
#include<stdio.h>
main()
{
struct salary{
   int basic_pay;
   int DA;
   int HRA;
};
typedef struct{
   struct salary sal;
   char name[10];
}emp;
emp matt;
matt.salary.da=350;
printf("%d",emp.sal.DA);
}
```

Answer:
Compilation Error.
"emp" is not a variable.

## Try it Yourself

- Predict the output:

3.

```
#include<stdio.h>
#include<string.h>
main(){
   typedef struct{
   int empno;
   char name[10];
   }emp;
   emp new[5];
   emp* ptr;
   ptr=&new[1];
   strcpy(new[1].name,"matt");
   strcpy(new[2].name,"jeff");
   ptr++;
   printf("%s\n",ptr->name);
}
```

Answer: "jeff"

## Try it Yourself

- Code debugging

1.

```
struct salary{
   int basic_pay;
   int DA;
   int HRA;
};
struct employee{
   struct salary sal;
   char name[10];
}emp;
emp matt;
matt.salary.da=350;
printf("%d",emp.sal.DA);
```

Answer: change the printf statement to `printf("%d",matt.sal.DA);`

## Try it Yourself

- Code debugging

2.

```
#include<stdio.h>
main()
{
   struct employee{
   int empno;
   char name[10];
   };
   struct employee emp;
   struct employee* p;
   p=&emp;
   printf("Enter the employee number:");
   scanf("%d",&p->empno);
}
```

Answer: change the scanf statement scanf("%d",p->empno), since it is already an address.

## Try it Yourself

- Code debugging

3.

```
#include<stdio.h>
main()
{
   typedef struct{
   int empno;
   char name[10];
   }emp lucy;
   emp matt;
   emp jeff;
   p=&matt;
   printf("Enter the employee number:");
   scanf("%d",p->empno);
}
```

Answer: The structure declaration is wrong, having the identifier (emp) declared along with a variable (lucy).

## Try it Yourself

1. Write a program to find the sum of two complex numbers. The numbers are be stored as structure variables 'A' and 'B', both of type "struct number". The elements of the structure "number" are two integers namely 'real' and 'img' which stores the 'real' and 'imaginary' parts of the number respectively and they are to be read from the user.
2. Write a program to find the distance between two points. The points are to be stored as structure variables 'A' and 'B', both of type "struct point". The elements of the structure "point" are two integers namely 'x' and 'y' which stores the 'x' and 'y' coordinates of the point in X-Y plane respectively; and they are to be read from the user.
3. There are a set of students in a class whose marks in 5 subjects are to be stored to find their total and average. You have to write a program to automate this process. Use a structure to store the name, his/her marks in 5 subjects, total and average. Write a menu-driven program to accomplish this. It should present the users with the following options:
    1. Add a new student
    2. Search for a student's total using his/her name.
    3. Print the rank list of the class.
    4. Exit

## Common Programming Mistakes

- Semicolon at the end of declaration.
- The direct selection operator has higher precedence than the indirection operator.
- The typename in declaration using typedef comes after the closing brace, before the semicolon.
- Do not use the same structure name with a tag inside a structure.

## Summary

- Discussed on the following:
  - Introduction to Structures
  - Difference between Structures and Arrays
  - Syntax and usage formats of structures
  - Examples

## References

1. Computer Science : A Structured Programming Approach Using C, Behrouz A Forouzan and Richard F Gilberg.