Module 5
(Structures, Pointers and Preprocessor Directives)
Pointers

1. **What is pointer in C? What are the benefits of using pointers? How to declare and initialize pointer. Explain with an example.**
2. **Develop a C program to swap two numbers using pointer**.
3. **Write a program in C to find the sum, mean and standard deviation of all elements in an array using pointers**.

Structures

4. **What is structure in C? With an example Explain How to define and declare structure. Compare array and structure**
5. **Implement structures to read, write and compute average marks and the students scoring above and below the average marks for a class of N students.**
6. **Using nested structure Develop a C program to read and display the details of 100 employees**.
7. **Develop a C program to Add two complex numbers using structures**
8. **With an example explain array of structure and nested structure? Also explain how to define and declare it**
9. Using nested structure Develop a C program to read and display the details of 100 students.

Preprocessor Directives

10. **What is preprocessor directives? Explain different categories of pre-processor directives used in C.**

1. What is pointer in C? What are the benefits of using pointers? How to declare and initialize pointer. Explain with an example.

A Pointer is a variable that holds the address of another variable

Advantages of using Pointer (Why pointer is required or Benefit of using pointer):

- It allows to use dynamic memory allocation
- It help to implement call by reference technique
- Helps to return more than one value from function
- It provide direct access to memory
- It reduces storage space of program
- It improve execution speed of program
- Help to build complex data structures such as linked list, tree, graph etc.

How to declare Pointers?

Syntax for Declaration:

```
data_type *pointer_name;
```

Example:

```
 int *ptr;
float *p;
char *cptr;
```

How to Initialize Pointers? Assigning value to pointer is called pointer initialization.

Syntax***:*** ***pointername=&variableName;***

Example:

```
 int *p1;/* declaring pointer*/
 Int x,y; ;
x=50;
p1=&x; /* initialize pointer p1, ie; storing address of x in p1*/
y=*p1; /* getting value from address in p1 and storing it in b*/
```

We have used two operators *** and &:**

- *: Content of the specified address
- & : Address

Example:

```
#include<stdio.h>
void main()
{
int *p;
int x=10,y=20;
printf("Original: x=%d\t, y=%d\n",x,y);
p=&x;
y=*p;
printf("Now Changed values: x=%d\t, y=%d\n",x,y);
printf(" p=%u\n",p);
printf("&p=%u\n",&p);
printf("*p=%d\n",*p);
printf("*(&p)=%u\n",*(&p));
printf("Address of x=%u\n",&x);
printf("Address of y=%u\n",&y);
}
```

Output:

```
Original: x=10  , y=20
Now Changed values: x=10 , y=10
p=3271248256
 &p=3271248264
  *p=10
   *(&p)=3271248256
Address of x=3271248256
Address of y=3271248260
```

2. *Write a C Program to swap Two number using Pointer*

```
#include <stdio.h>
void swap(int * n1, int * n2)
{
   int temp;
   temp = *n1;
   *n1 = *n2;
   *n2 = temp;
}
int main()
{
   int a = 15, b = 100;
   printf("Before Swapping: a=%d\t, b=%d\n",a,b);
   swap( &a, &b);
   printf("After Swapping: a=%d\t, b=%d\n",a,b);
}
```

Output:

```
Before Swapping: a=15   , b=100
Swapping: a=100   , b=15
```

3. **Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of n real numbers.**
   [Pointer to array]

Program:

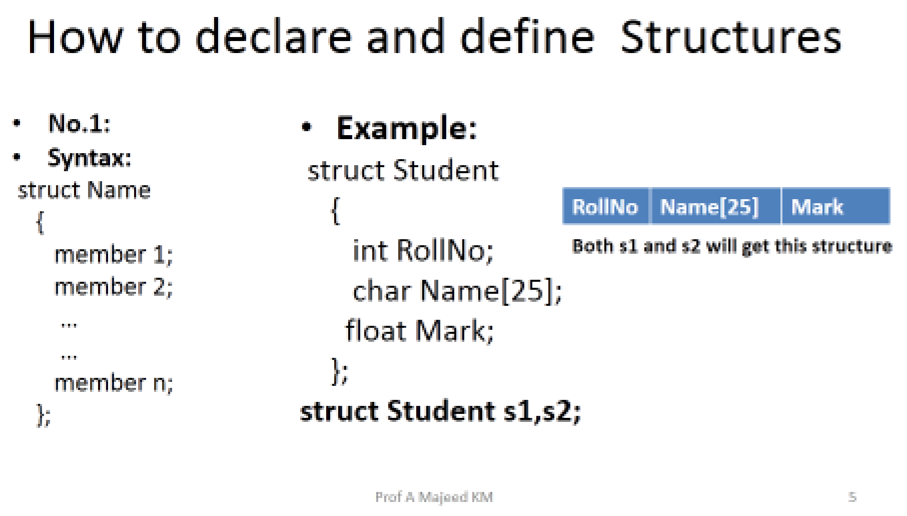
```
#include<stdio.h>
#include<math.h>
main()
{
  float a[10], *ptr, mean,var, std, sum=0, sumstd=0;
  int n,i;
  printf("Enter the no of elements\n");
 scanf("%d",&n);
  printf("Enter the array elements\n");
  for(i=0;i<n;i++)
   {
     scanf("%f",&a[i]);
   }
  ptr=a;  // initialization of a pointer to array  (ie;  ptr=&a[0] )
  for(i=0;i<n;i++)
   {
      sum=sum+ *ptr; // Calculate sum
      ptr++; // move to next array element
   }
  mean=sum/n;//   Calculate Mean
  ptr=a; // initialization of a pointer to array (ie;  ptr=&a[0] )

  for(i=0;i<n;i++)
   {
     sumstd=sumstd + pow((*ptr - mean),2);
     ptr++;  // move to next array element
   }
  var=sumstd/n;
  std= sqrt(var); //Calculate standard deviation
  printf("Sum=%f\t",sum);
  printf("Mean=%f\t",mean);
  printf("Standard deviation=%f\t",std);
}
```

Out put:

```
Enter the no of elements
6
Enter the array elements
12 13 44 34 44 32
Sum=179.000000     Mean=29.833334     Standard deviation=13.069260
```

4. What is structure in C? With an example Explain How to define and declare structure. Compare array and structure

A structure is collection of variety of elements which can be of different data types.
A structure is collection of elements with different data types.
How to declare and define Structures
Before declaring structure we have to **define** structure. We can define structure by using ***struct*** keyword. There are **three method** for defining and declaring structure. They are shown below.
No.1: Defining and declaring structures separately-Here first we have to define structure and then we can declare structure by using struct keyword. Syntax and example shown below (Defining Structure with name: student, and declaring structure variable s1 and s2).



How to declare and define Structures

- **No.1:**
- **Syntax:**

```
struct Name
  {
    member 1;
    member 2;
     ...
     ...
    member n;
  };
```

- **Example:**

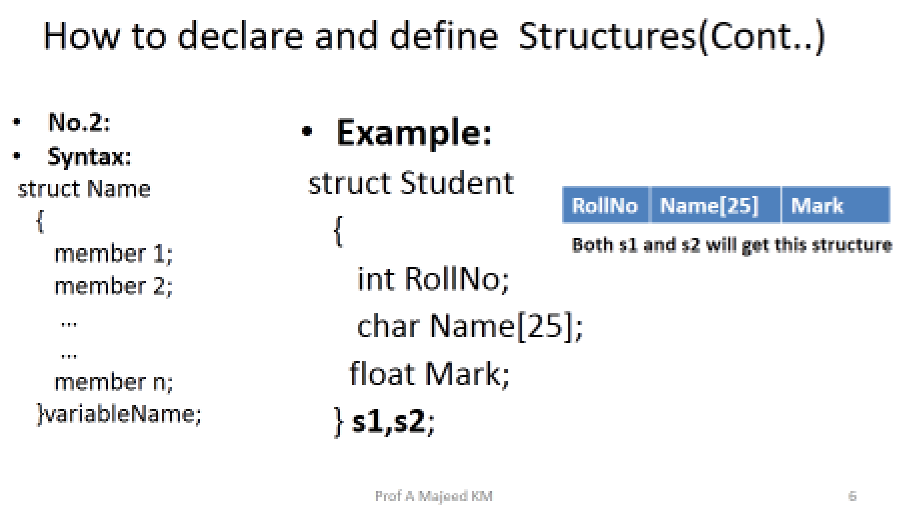
```
struct Student
  {
    int RollNo;
    char Name[25];
   float Mark;
  };
struct Student s1,s2;
```

RollNo	Name[25]	Mark

Both s1 and s2 will get this structure

Prof A Majeed KM 5

No.2: Defining and declaring structures together-Here we have to define and declare structure together by using struct keyword. Syntax and example shown below



How to declare and define Structures(Cont..)

- **No.2:**
- **Syntax:**

```
struct Name
  {
    member 1;
    member 2;
     ...
     ...
    member n;
  }variableName;
```

- **Example:**

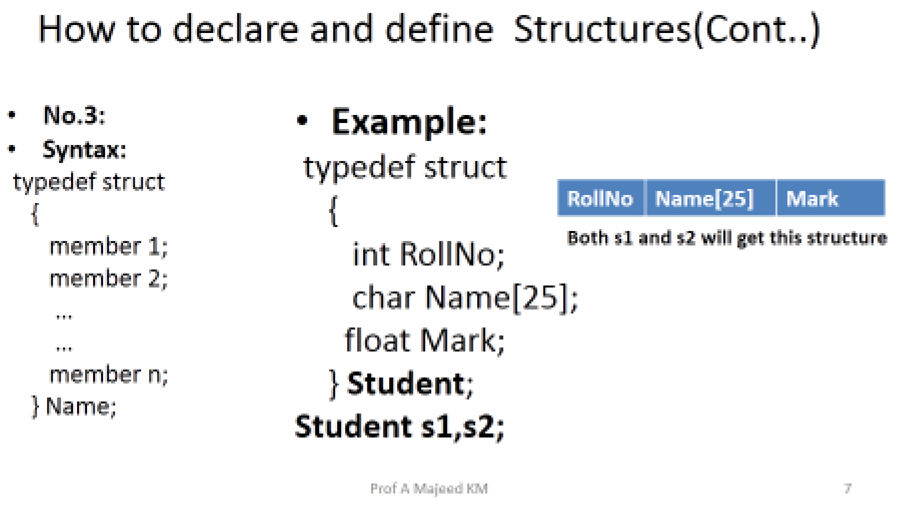
```
struct Student
  {
    int RollNo;
    char Name[25];
   float Mark;
  } s1,s2;
```

RollNo	Name[25]	Mark

Both s1 and s2 will get this structure

Prof A Majeed KM 6

No.3: **Defining and declaring structure by using typedef-**Here we are defining user defined structure with typedef keyword. Then we are declaring structure by using user defined structure type. Its syntax and example shown below.



How to declare and define Structures(Cont..)

- **No.3:**
- **Syntax:**

```
typedef struct
 {
   member 1;
   member 2;
   ...
   ...
   member n;
 } Name;
```

- **Example:**

```
typedef struct
  {
    int RollNo;
    char Name[25];
    float Mark;
  } Student;
Student s1,s2;
```

RollNo	Name[25]	Mark

Both s1 and s2 will get this structure

Prof A Majeed KM 7

Arrays and Structure-Comparison:

1) Array is a collection of elements with same data types. Structure is a collection of elements with different data types
2) Array elements can be accessed by the index placed within []. Structure elements can be accessed with the help of. (Dot) operator
3) To represent array, Array name is followed by []. To represent structure, a keyword struct has to be used
4) Example: for array:

int a[20];

Example for structure:

```
struct Student
 {
   int RollNo;
   char Name[25];
 } s1;
```

5. **Implement structures to read, write and compute average marks and the students scoring above and below the average marks for a class of N students.**

Program:

```
#include<stdio.h>
struct Student // Defining structure
       {
          int rollNo;
          char name[25];
         int mark;
       } s[100];// Declaring array of structure

 int main()
{
  int n,i,avg,sum=0;
  printf("Enter the no.of students\n");
   scanf("%d",&n);
   for(i=0;i<n;i++) // to Read student details one by one
   {
     printf("Enter Roll No ,Name and Mark of Student:\n " );
     scanf("%d%s%d",&s[i].rollNo, s[i].name,&s[i].mark);
     sum=sum+s[i].mark;// calculate total mark
   }
   avg=sum/n; // calculate average mark
   printf("Average Mark= %d\n",avg);
   printf("Students with mark greater than average:\n");
   printf("\n Roll_No    Name  Mark\n");
   for(i=0;i<n;i++)
   {
      if(s[i].mark>=avg)
       printf("%d  %s    %d\n", s[i].rollNo,s[i].name,s[i].mark);
   }
    printf("Students with mark below average :\n");
    printf("\n Roll_No    Name  Mark\n");
    for(i=0;i<n;i++)
    {
     if(s[i].mark<avg)
         printf("%d  %s    %d\n", s[i].rollNo,s[i].name,s[i].mark);
    }
  }
```

6. **Using nested structure Develop a C program to read and display the details of 100 employees.**

```
#include<stdio.h>
struct Date
  {
    int dd;
    int mm;
    int yyyy;
  };
struct Employee
  {
    int EmpID;
    char Name[25];
    struct Date DOJ;  // Nested structure
  } s[100];  //Array of structure
void main()
{
 int n,i;
 printf("Enter the no.of Employee\n");
 scanf("%d",&n);
 for(i=0;i<n;i++)
 {
   printf("Enter EmpID, Name and DOJ of Employee %d\n",i+1);
   scanf("%d%s%d%d%d",&s[i].EmpID,s[i].Name,&s[i].DOJ.dd,&s[i].DOJ.mm,&s[i].DOJ.yyyy);
 }
 printf("Student Detail:\n"), printf("\n EmpID    Name  DOJ\n");
for(i=0;i<n;i++)
 {
   printf("%d  %s    %d-%d-%d\n", s[i].EmpID,s[i].Name,s[i].DOJ.dd,s[i].DOJ.mm,s[i].DOJ.yyyy);
 }
}
```

7. Develop a C program to Add two complex numbers using structures

```
#include <stdio.h>
struct complex
{
  int real, img;
};
main()
{
  struct complex a, b, c;
  printf("Enter a and b where a + ib is the first complex number.\n");
 scanf("%d%d", &a.real, &a.img);
 printf("Enter c and d where c + id is the second complex number.\n");
 scanf("%d%d", &b.real, &b.img);
 c.real = a.real + b.real;
 c.img = a.img + b.img;
 printf("Sum of the complex numbers: (%d) + (%di)\n", c.real, c.img);
 }
```

8. With an example explain array of structure and nested structure? With an example explain how to define and declare it

Array of structure:

- Array of structures is nothing but collection homogeneous of structures.
- This is also called as structure array in C.
- If you wish to maintain the information of ‘n‘ employees or Students then you need to declare an array of structure

Example for defining and declaring array of structure

```
struct Student
{
   int RollNo;
   char Name[25];
   float Mark;
} s[10];  //  defining and declaring Array of structure
```

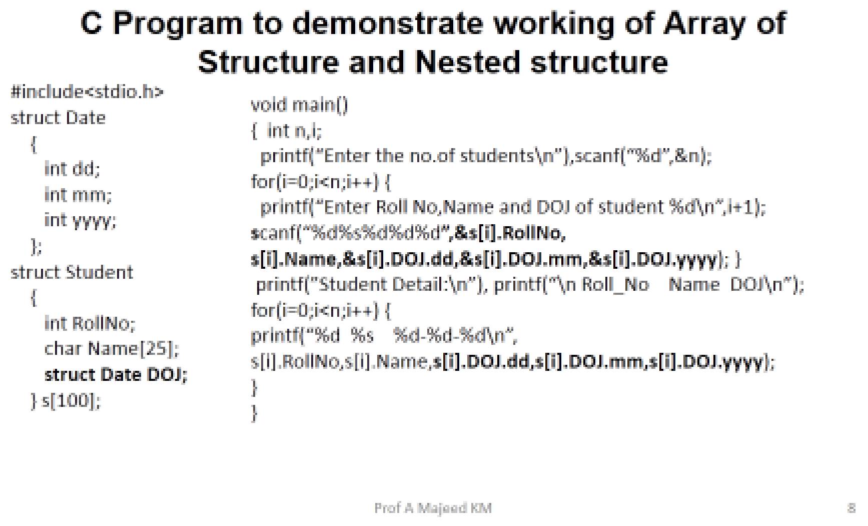
Nested Structure:

- A structure is collection of elements with different data types.
- ***A structure present within another Structure is called Nested Structure***

Example for nested structure:

```
struct Date                      struct Student
  {                                {
    int dd;                          int RollNo;
    int mm;                          char Name[25];
    int yyyy;                       struct Date DOJ;  //Nested structure
  };                               } s1,s2;
```

9. Using nested structure Develop a C program to read and display the details of 100 students.



C Program to demonstrate working of Array of Structure and Nested structure

```
#include<stdio.h>
struct Date
  {
    int dd;
    int mm;
    int yyyy;
  };
struct Student
  {
    int RollNo;
    char Name[25];
    struct Date DOJ;
  } s[100];
```

```
void main()
{  int n,i;
  printf("Enter the no.of students\n"),scanf("%d",&n);
for(i=0;i<n;i++) {
  printf("Enter Roll No,Name and DOJ of student %d\n",i+1);
scanf("%d%s%d%d%d",&s[i].RollNo,
s[i].Name,&s[i].DOJ.dd,&s[i].DOJ.mm,&s[i].DOJ.yyyy); }
 printf("Student Detail:\n"), printf("\n Roll_No    Name  DOJ\n");
for(i=0;i<n;i++) {
printf("%d  %s    %d-%d-%d\n",
s[i].RollNo,s[i].Name,s[i].DOJ.dd,s[i].DOJ.mm,s[i].DOJ.yyyy);
}
}
```

Prof A Majeed KM 8

Output:

```
Enter the no.of students
3
Enter Roll No,Name and DOJ of student 1
1001 sachin 22 5 1999
Enter Roll No,Name and DOJ of student 2
1002 Richard 22 2 2000
Enter Roll No,Name and DOJ of student 3
1003 Ahmed 20 5 2018
Student Detail:

Roll_No    Name  DOJ
1001  sachin    22/5/1999
1002  Richard    22/2/2000
1003    Ahmed    20/5/2018
```

10. What are preprocessor directives? Explain different categories of pre-processor directives used in C.

- Before a C program is compiled in a compiler, source code is processed by a program called ***preprocessor***. This process is called ***preprocessing***.
- **Commands used in preprocessor are called preprocessor directives.** Preprocessor directives are placed in the source program before the main line**. They begin with "#" symbol.**
- **Categories of Preprocessors directives**
    1) File inclusion directives
    2) Macro substitution directives
    3) Compiler control directives

- **File inclusion Directives(#include** directive)
    - An external file containing functions or macro definition can be included as part of program so that we need not rewrite those functions or macro definitions
    - Puts copy of file in place of directive
    - This is achieved by Two forms
        - **#include <filename>**
            - For standard library header files
            - Example: **#include<stdio.h>**
        - **#include "filename"**
            - Searches in current directory
            - Normally used for programmer-defined files
            - Example: **#include "test.c"**
- **Macro substitutions (#define)**
    - Macro substitution is a process where an identifier in program is replaced by a predefined string composed of one or more tokens. The preprocessor accomplishes this task under the direction of ***#define*** statement. This statement usually known as ***macro statements or macros.***
    - ***There are*** <u>three forms</u> ***of macro substitutions. They are***
        - ***Simple macro substitution***
        - ***Argumented macron substitution***
        - **Nested macro substitution**
    - **Simple macro substitution:** It is also used to define **symbolic constants**.
        - Constants represented as symbols
        - When program compiled, all occurrences replaced
        - Format
            - **#define *identifier replacement-text***
            - **Example: #define PI 3.14159**
        - Everything to right of identifier replaces text
        - **#define PI 3.14159** - Replaces **PI** with **"3.14159"**

- **Argumented macro substitution *(***Macro with argument)
    - The preprocessor permit us to define more complex and more useful form of replacement
    - It takes the form: **#define identifier(f1,f2,f3,…, fn) string**
        - Where ***identifier f1,f2,…fn*** are formal macro argument that are similar to formal argument in function definition
        - When a macro is called, the preprocessor substitute the string, replacing the formal parameter with actual parameter, Hence the string behave like a template.
        - Example: **#define cube(x)   (x*x*x)**

          If the statement, ***Volume = cube (side);*** appear later in the program then the preprocessor would expand this statement to ***Volume = (side*side*side);***
- **Nested Macros substitution:**
    - We can also use one macro in definition of another macro and that's known as **nested macros**
    - Example:
        - #define **M** 5
          #define N **M***5 //nested macro
- **Undefining a macro:**
    - A defined macro can be undefined, using the statement
        - **#undef identifier**
        - This is useful when we want to restrict the definition only to a particular part of the program

- **Compiler control directives:**
    - **While developing large program, you may face many problems or different situation.**
        - One solution to these problem is to develop different program to suit the needs of different situation.
        - Another solution is to develop a single, comprehensive program that include all optional code and then direct the compiler to skip over certain parts of source code when they are not required.
        - C preprocessor offer a feature known as **Conditional compilation, which can be used to switch on or off a particular line or group of lines in a program. They are**
            - ***#ifdef*** *to check whether macro is defined or* not
            - ***#ifndef*** to check whether macro is not defined yet
            - ***#undef*** to undefined macro
            - ***#endif*** to represent end of ***#if***
            - ***#if, #else*** more general form to represent constant-expression

- Example:
```
#include "DEFINE.H"
#ifndef TEST
#define TEST 1
#endif
```
- **#DEFINE.H** is a header file that is supposed to contain the definition of **TEST** macro. The directive **#ifndef TEST** searches for the definition of **TEST** in the header file and if not defined, then all the lines between **#ifndef** and corresponding **#endif** directives are left **active** in the program. In case if the **TEST** has been defined in the header file then the **#ifndef** condition **becomes false**, therefore the **#define TEST 1 is ignored**.
- **ANSI addition:**
  - **#elif** provide alternative test facility(if-else-if sequence)
  - **#pragma** Specifies certain instructions
  - **#error** Stop compilation when an error occurs
  - **#** Stringizing operator
  - **##** Token pasting operator

- ***Example: #elif directives***
```
#if expression1
      Statement1;
#elif expression2
      Statement2;
      ……………….
#elif expression n
      Statement n;
#endif
```
- ***Example: #pragma directives***
  - **#pragma name** *where name of the pragma we want*
- ***Example for #error directives:***
  - ***#error message*** ***where message is any error message***