

Introduction to C Programming

K Prasanna Kumar

CONTENTS

1	Introduction	1
2	Basic Programming	1
3	Conditional Statements	3
3.1	if-else Statement	3
3.2	if-else-if Statement	3
3.3	Switch-Case	4
4	Iteration statements	4
4.1	While loop	4
4.2	for loop	5
4.3	do while loop	5
4.4	Infinite loop	6
5	Pointers	7
6	Functions	7
6.1	Recursion	9
6.2	Header File	9
7	Memory Allocation	10
7.1	Static Memory	10
7.2	Dynamic memory	10
8	Preprocessor Directives	10
9	Arrays	10
9.1	Uni-Dimensional Array . . .	10
9.2	Multi-Dimensional Array . .	14
10	Structures	15
11	File handling	18
	References	19

Abstract—This module gives a quick recap of C Programming. C programming in Linux environment is required by a Communication Engineer, for the analysis of Openairinterface (OAI) software stack.

Author is from Dept. of Electrical Engineering, IIT Hyderabad,
Email: kprasannakumar@iith.ac.in, kprasannakumar.iith@gmail.com

1 INTRODUCTION

- 1) It is a general purpose, high level, procedural programming language
- 2) Programming file has an extension ".c" and the Linux compiler is "gcc"

2 BASIC PROGRAMMING

Problem 1. Arithmetic Operations

```

/* Arithmetic Operations using C
   Programming */
// Author : K. Prasanna Kumar
// Last Modified Date : 15/06/2020

#include <stdio.h>
int main()
{
    // Decleration of variables
    int a;
    int b;
    int Add;
    int Sub;
    int Mul;
    float Div;

    // Insialtation of Variable
    printf("Enter the Value of a: ");
    scanf("%d", &a);
    printf("Enter the value of b: ");
    scanf("%d", &b);

    /* Computation */
    // Additon operation
    Add = a + b;
    // Substraction Operation
    Sub = a - b;
    // Multiplication Operation
    Mul = a*b;
    // Division Operation
    Div = a/b;
    // Modulo Division
    int MuD =a%b;

```

```

/* Computational Output */
printf("Addition operation of a
and b is %d\n", Add );
printf("Substraction operation of a
and b is %d\n", Sub);
printf("Multiplication operation
of a and b is %d\n", Mul);
printf("Division operation of a
and b is %f\n", Div);
printf("Modulo Division operation
of a and b is %d\n", MuD);
return 0;
}

```

Problem 2. Logical Operations

```

/* Logical Operation Using C
Programming */
// Author : K. Prasanna Kumar
// Last Modified Date : 15/06/2020

```

```

#include <stdio.h>
#include <stdbool.h>

```

```

int main()
{
// Inisalization of Variables

```

```

int A;
int B;
int Add_Lg;
int Or_Lg;
int Not_Lg;
int Nand_Lg;
int Nor_Lg;
int Xor_Lg;

```

```

// Decularation of Variables
printf("Logical Value of A: ");
scanf("%d", &A);
printf("Logical Value of B: ");
scanf("%d", &B);

```

```

/* Logical Operations */

```

```

Add_Lg = A && B; // ADD Operation
Or_Lg = A || B ; // OR Operation
Not_Lg = !A ; // Not Operation of
A

```

```

Nand_Lg = !(A && B); // NAND
Operation

```

```

Nor_Lg = !(A || B); // NOR
Operation

```

```

Xor_Lg = (A && (!B)) || ((!A) && B);
// Exclusive OR operation

```

```

/* Output Display */

```

```

printf("ADD Operation of A and B
is %d\n", Add_Lg);
printf("OR Operation of A and B is
%d\n", Or_Lg);
printf("NOT Operatioon of A is %d
\n", Not_Lg);
printf("NAND Operation of A and B
is %d\n", Nand_Lg);
printf("NOR Operation of A and B
is %d\n", Nor_Lg);
printf("XOR Operation of A and B
is %d\n", Xor_Lg);

```

```

return 0;
}

```

Problem 3. Exchange of two numbers

```

/* Exchange of numbers with two
Variables */

```

```

// Author : K. Prasanna Kumar
// Last Modified Date : 15/06/2020

```

```

#include <stdio.h>

```

```

int main()
{

```

```

// Initialization of Variables

```

```

int a ;
int b;
int temp ;

```

```

// Decularation of Variables

```

```

printf("Enter the value of a: ");
;
scanf("%d", &a);
printf("Enter the value of b: ");
scanf("%d", &b);

```

```

// Swap the variables --- SWAP

```

```

    Operation
temp = a;
a = b;
b = temp;

// Output
printf("-----After SWAP
    Operation -----\n");
printf("Value of a: %d\n", a);
printf("Value of b: %d\n", b);

return 0;
}

```

Problem 4. Exchange of two number using only two variables

```

/* SWAP Operation using only two
   Variables */
// Author : K. Prasanna Kumar
// Last Modified Date : 15/06/2020

#include <stdio.h>

void main()
{
    // Initialization of Variables
    int a ;
    int b;

    // Decularation of Variables
    printf("-----SWAP
        Operation Using Two Variables
        ----- \n");
    printf("Enter the value of a: ");
    scanf("%d", &a);
    printf("Enter the value of b: ");
    scanf("%d", &b);

    // Computation for Swap Operation
    a = a + b;
    b = a - b;
    a = a -b;

    // Output Display
    printf("-----AFTER
        SWARP_OPERATION----- \n")
        ;
    printf("Value of a: %d\n", a);
    printf("Value of b: %d\n", b);
}

```

```

}

```

Problem 5. Bitwise Operations

3 CONDITIONAL STATEMENTS

3.1 if-else Statement

```

/* Introduction to IF Clause */
// Problem : Find which is
greather
// Author : K. Prasanna Kumar
// Last Modified Date : 15/06/2020

```

```

#include <stdio.h>

```

```

int main()
{
    // Initialization of Variables
    float a ;
    float b;

    // Decularation of Variables
    printf("Enter the value of a: ")
        ;
    scanf("%f", &a);
    printf("Enter the value of b: ");
    scanf("%f", &b);

    // If Logic
    if (a>=b)
    {
        printf("a is greather than or
            equal to b\n");
    }
    else
    {
        printf("a is less than b\n");
    }

    return 0;
}

```

3.2 if-else-if Statement

```

/* Basic if else if in C using
   Arthamatic Operations */

```

```
// Author : K. Prasanna Kumar
// Last Modified Date : 25/12/2020

#include <stdio.h>
void main()
{
    // Decularation of Variables
    float a;
    float b;

    // Initialization of Variables
    printf("Enter the value of a: ");
    scanf("%f", &a);
    printf("Enter the value of b: ");
    scanf("%f", &b);

    // Operation
    if (a > b)
    { printf("a is greather than b\n")
      ; }
    else if (a < b)
    { printf("a is less than b\n"); }
    else
    { printf("a is equal to b\n"); }

}
```

3.3 Switch-Case

Problem 6. Program to convert numerical value to word for 1 - 5 numbers.

```
/* Typing the word formate of the
   number */
// Author : K. Prasanna Kumar
// Last Modified Date: 28/12/2020

#include <stdio.h>
int main()
{

    // Initialization of Variables
    int num;

    // Decularation of Variables
    printf("Enter the value of number
    : ");
    scanf("%d", &num);
    printf("The word formate of the
    number %d is \t", num);
}
```

```
// Operation
switch (num)
{
    case 1:
        printf("ONE\n");
        break;
    case 2:
        printf("TWO\n");
        break;
    case 3:
        printf("THREE\n")
        ;
        break;
    case 4:
        printf("FOUR\n");
        break;
    case 5:
        printf("FIVE\n");
        break;
    default :
        printf("NOT AVALIBLE\n");
        printf("----- Please
        Enter the Valid Number
        ----- \n");
    /* "default" is optional. It is
       used to execute the statements
       in it, if
       and only if the input value does
       not match with any of the case
       */
}

return 0;
}
```

Problem 7. Menu Program to do the Arithmetic Operations.

4 ITERATION STATEMENTS

4.1 While loop

```
/*
Description : Multiplication Table
              using While Loop
Programmer : K. Prasanna Kumar
Last Modified Date : 20th April
              2020

Syntax :
```

```

Initialization of Increment
Variable
while(Condition for True)
{
    statment or function call;
    increment;
}
*/

#include <stdio.h>
int main()
{
    int a;
    printf("\n-----\n
        Multiplication Table\n
        -----");
    printf("Enter the value of a: ");
    scanf("%d", &a);

    int i = 1;
    while(i <= 20)
    {
        printf("%d x %d = %d\n", a, i, a*i);
        i = i+1;
    }

    return 0;
}

```

4.2 for loop

```

/*
Description : Multiplication table
              using for loop
Programmer : K. Prasanna Kumar
Last Modified Date : 21st April
              2021
Syntax:

for(Initialization of increment
      variable, Condition for True,
      Increment)
{
    Statements or Function calls
}

```

```

*/

#include <stdio.h>
int main()
{
    int a;
    printf("\n-----\n
        Multiplication Table\n
        -----");
    printf("Enter the value of a: ");
    scanf("%d",&a);

    for(int i=1; i<=20 ; i = i+1)
    {
        printf("%d X %d = %d\n",a,i,a*i);
    }

    return 0;
}

```

4.3 do while loop

```

/*
Description : Do-While loop basic
              example
Programmer : K. Prasanna Kumar
Last Modified Date : 20th April
              2021
Syntax :

```

```

do{

    Statements or Function calls

```

```

}(The condition is True);

```

Note :

1. Semicolom (;) **is** important after **do-while**
2. Loop Execuits at least once

```

*/

```

```

#include <stdio.h>
int main()
{
    int a;
    do
    {
        printf("Enter the value of a: ");
    }
}

```

```
scanf("%d", &a);
printf("The value of a is \t %d\n", a);
printf("The above loop gets end if a==0\n");

} while(a !=0);

printf("Program Ended as you Entered \"0\" \n");

return 0;
}
```

4.4 Infinite loop

Problem 8. Infinite loop using "for loop"

```
/*
Description : Infite loop using
for_loop
Programmer: K. Prasanna kumar
Last Modified Date : 20th April
2020
Syntax :
```

```
for( ; ; )
{
statements or function calls
}
```

Note : No Initialization , No condition , No Increment

```
*/

#include <stdio.h>
int main()
{

for( ; ; )
{
printf("I will not Stop\n");
printf("_-----_Press \"_
Ctrl+_+_c\"_to _STIP_me_
-----_\n");
}

return 0;
}
```

Problem 9. Infinite loop using "while loop"

```
/*
Description : Infite loop using
while_loop
Programmer: K. Prasanna kumar
Last Modified Date : 20th April
2020
Syntax :
```

```
while( 1 )
{
statements or function calls
}
```

Note : '1' indicates that the condition is true all the time

```
*/

#include <stdio.h>
void main()
{

while(1)
{
printf("I will not Stop\n");
printf("_-----_Press \"_
Ctrl+_+_c\"_to _STIP_me_
-----_\n");
}

}
```

Problem 10. Termination of infinite loop using "break"

```
/*
Description : Infite loop using
while_loop
Programmer: K. Prasanna kumar
Last Modified Date : 20th April
2020
Syntax :
```

```
while( 1 )
{
statements or function calls
if(condition)
break;
}
```

```

*/

#include <stdio.h>
void main()
{
    int a;
    while(1)
    {
        printf("I will not Stop until a = 0\n");
        printf("-----Enter value of a as \'0\' to stop the loop-----\n");
        printf("Enter the value of a: ");
        scanf("%d", &a);

        if(a==0)
            break;
    }
}

```

5 POINTERS

Problem 11. Program to explain pointer definition

```

/*
Description : Program to explain
              definatin of POINTER
Programmer : K. Prasanna Kumar
Last Modified Date : 24th April
              2021
*/

#include <stdio.h>
int main()
{
    // Variable
    int x = 5;

    /* Address of variable x is stored
       in variable y,
       the variable y is known as POINTER
       of x
    Note :
    1. & ---> Address of operator
    2. * --> Value of operator
    */

    // Deculation of pointer

```

```

int *y ;
y = (&x); // y stores the address
           of x
printf("Value stored in the
       address y: %d\n", *y);
printf("Value stored in the
       address of variable x: %d\n",
       *(&x));
return 0;
}

```

6 FUNCTIONS

Problem 12. Basic user defined function

```

/*
Description : Printing a line by
              calling user defined functin
Programmer : K. Prasanna kumar
Last Modified Date : 24th April
              2021
*/

#include <stdio.h>
// Function Deculation or
// Prototype of function
void printline();

void main()
{
    // Function call in the main
    // function
    printline();
    printf("Basic user defined
           Function Program\n");
    printline();
    // Note: We can call the functin N
    // number of time
}

// Function Defination or Body of
// function
void printline()
{
    printf("-----###-----\n");
}

```

Problem 13. Program to explain Function Arguments

```

/*
Description : Program to explain
              passing Arguments in an user
              defined function
Note:-
1. Arguments of a function are also
   called Parameters
2. Arguments are separated by ,
Programmer : K. Prasanna Kumar
Last Modified Date : 24th April
                2021
*/

// Header files
#include <stdio.h>

// Prototype with Parameters
int add_fun(int , int);
/*
1. Data type arguments or
   Parameters are
sufficient in function Declaration.
2. Return type of the user defined
   function depends on data type
   of the
output of user defined function.
*/

int main()
{
    int a,b,c;
    printf("Enter the value of a: ");
    scanf("%d", &a);
    printf("Enter the value of b: ");
    scanf("%d", &b);

    // Function call with arguments
    c = add_fun(a, b);
    /*
    Actual Arguments : Arguments which
                       are passed in the
    function call are known as actual
    Formal Arguments : Arguments used
                       in the function definition
                       are known as formal arguments
    */
    printf("Add function of a & b is %d\n", c);
    return 0;

```

```

}

// User Defined function
// Definition
int add_fun( int x, int y)
{
    int z = x + y;
    return z;
// Output of the function is z
}

```

Problem 14. Swap function using pointers

```

/*
Description : Swap operation using
              pointers
Programmer : K. Prasanna kumar
Last Modified Date : 24th April
                2021
*/

#include <stdio.h>
// Declaration or Prototype
void swap(int * , int *);
int main()
{
    int x,y;
    printf("Enter the value of x: ");
    scanf("%d", &x);
    printf("Enter the value of y: ");
    scanf("%d", &y);

    // Function call
    swap(&x, &y);
    /*
    1. Calling a function using values
       as arguments
    is known as call by value
    2. Calling a function using
       address as arguments
    is known as call by reference
    */
    printf("The value of x and y are %d, %d respectively\n", x, y);
}

// Definition of a function

void swap(int *a, int *b)
{

```



```
int temp;
temp = *a;
*a = *b;
*b = temp;
}
```

6.1 Recursion

Problem 15. Find the factorial of an integer

```
/*
Description : finding a factorial
of an integer using recursion
Programmer : K. Prasanna Kumar
Last Modified Date : 27th April
2021
Defination of Recursion : Calling
a function by itself is known as
recursion
*/

#include <stdio.h>
// Prototype or Declaration
int factorial(int );

void main()
{
int a;
printf("Enter the integer value
for a: ");
scanf("%d", &a);
int fact = factorial(a);
printf("Factorial of a = %d\n",
fact);
}

// function Definition
int factorial(int n)
{
int f;
if (n==0)
return (1);
else
{
f = n * factorial(n-1);
// Function call has been done in
the same function
return (f);
}
}
```

```
/*
Calling a function inside the same
function is called Recursion.
The above user defined function is
a recursive function.
Recursion runs in a loop (infinite
loop).
*/
```

6.2 Header File

Problem 16. Function call using user defined header file.

```
/*
Description : Header file with
user defined function of modulo
function
Note: This modulo function is only
for integers

Header file should be save with an
extension .h
*/

// Definition
int mod(int x , int y)
{
int Q, R;
Q = x/y;
R = x - (Q*y);
return R;
}
```

The above header file is saved with an extension .h and the main function lie in the following program

```
/*
Description : Making a user
defined Modulo function call
using
user defined head file
Programmer : K. Prasanna Kumar
Last Modified Date : 9th May 2021
*/

#include <stdio.h>
#include "modulo.h"
// Angle Brackets are not used for
user defined header file
```

```

void main()
{
    int a, b;
    printf("Enter the value of a: ");
    scanf("%d", &a);
    printf("Enter the value of b: ");
    scanf("%d", &b);

    if (a >= b)
    {
        int c = mod(a, b);
        printf("Integer Modulo Operation of a & b is: %d\n", c);
    }
    else
    printf("Modulo Operation can be performed using this function, since a < b\n");
}

```

7 MEMORY ALLOCATION

Problem 17. Size of data types

```

/*
Description : Program to explain
size of Data Types
Programmer : K. Prasanna kumar
Last Modified Date : 9th May 2021
*/

#include <stdio.h>
int main()
{
    printf("Size of an integer data type is: %ld\n", sizeof(int));
    printf("Size of an real number (float) data type is: %ld\n", sizeof(float));
    printf("Size of a character data type is: %ld\n", sizeof(char));
    return 0;
}

```

7.1 Static Memory

Problem 18. Returning address from user defined function, using static variable

```

/*
Description : Program to
illustrate the concept of
returning pointer from
a user defined function
Programmer : K. Prasanna Kumar
Last Modified Date : 24th April
2020
*/

#include <stdio.h>
// Prototype or Declaration
int* pointer_return();

void main()
{
    int *p;
    p = pointer_return();
    printf("%p\n", p);
    printf("%d\n", *p);
}

// Definition
int* pointer_return()
{
    static int a = 25;
    /* variable a will be active
    untill the main() function
    completes its exe */
    return (&a);
}

```

7.2 Dynamic memory

8 PREPROCESSOR DIRECTIVES

9 ARRAYS

9.1 Uni-Dimensional Array

Problem 19. Printing an array

```

/*
Description : Printing an Array
This program gives an introduction
to array
Programmer : K. Prasanna Kumar
Last Modified Date : 14th May 2021
*/

```

```

#include <stdio.h>

```

```

int main()
{
// Decularation of Array
int arr[5];
int i; // Index variable of arr

// Initialization of array
for (i=0; i<5; i++)
{
printf("Enter the value of arr[%d]
: ", i);
scanf("%d", &arr[i]);
}

// Printing an array
printf("\n Entries of an array
are as follows :\n");
for (i =0; i<5; i++)
{
printf("%d, \t", arr[i]);
}
printf("\b\n");
return 0;
}

```

Problem 20. Program a log book to enter age of the team members in an office.

```

/*
Description : Storing the age of
people in a team in a log.
Programmer : K. Prasanna Kumar
Last Modified Data : 24th April
2021
*/

#include <stdio.h>
void main()
{
int n;
printf("Number of Member in a team
: ");
scanf("%d", &n);
int a[n]; // Array of size n;
int i; // index variable of an
array

for(i=0; i<n; i++)
{
printf("Enter the age of roll

```

```

number %d : a[%d] = ", i+1, i);
scanf("%d", &a[i]);
}
printf("\n --- All the values are
Entered Succesfully ---\n");

// Log Book using do-while
do{
printf("\n ##### \n -----
Welcome to log Book ----\n");

printf("Enter the roll numebr:");
scanf("%d", &i);

if (i>=0 && i<=n)
{
printf("The age of roll number %d
is : %d\n", i, a[i-1]);
}
else
{
printf("--- INVALID ENTRY ---\n")
;
printf("Team Size is %d, Enter
value less than or equal to %d\n", n, n);
}

printf("Press '/' 0/' to exit the
log");
} while (i != 0);

printf("The log is terminated .. \n
As it is requested to terminate
by Enterning '/' 0/' \n");
}

```

Problem 21. Searching an array

```

/*
Description : Size of an array
Programmer : K. Prasanna Kumar
Last Modified Date : 20th April
2020
*/

#include <stdio.h>
void main()
{

```

```

int arr[] = {10, 2, 3, 4, 5, 6, 7,
             8, 9};

// Size of an array
int size = sizeof arr / sizeof arr
[0];

int n;
printf("Enter the value that to be
       searched : ");
scanf("%d", &n);

// Searching Array
for (int i=0 ; i<size ; i++)
{
    if (arr[i] ==n)
    {
        printf("The value is
               available in the index
               number %d\n", i);
        return;
    }
}
printf("%d is not available in this
       array\n", n);
}

```

/* Difference between **break** and **return** : --

A **break** statement terminates the switch **or** loop, **and** execution continues at the first statement beyond the switch **or** loop.

A **return** statement terminates the entire function that the loop is within, **and** execution continues at point where the function was called.

...

*/

Problem 22. Base address concept of array

```

/*
Description : Proving the base
              address concept of array
Programmer : K. Prasanna Kumar
Last Modified Date : 24th April

```

2020

Array Base Address Concept :
Writing the name of the array
gives the base address of the
array.

Base Address :-- Address of the
first element

*/

```

#include <stdio.h>
void main()
{
    int a[3] = {1, 2,3};

    if (&a[0] == a)
        printf("Yes, The concept is
               correct\n");
    else
        printf("The concept is incorrect\n");
}

```

Problem 23. Accessing arrays with pointers

```

/*
Description : Accessing linear
              array with help of pointers
Programmer : K. Prasanna Kumar
Last Modified Date : 24th April
              2020

```

*/

```

#include <stdio.h>
void main()
{
    int a[5] = {1, 7, 9, 21,3};
    printf("a[0] = %d\n", *a);
    /*
    since the array name gives the
    base address, *(array name)
    gives
    the value at the base address.
    Therefore, by the help of pointer
    addition concept we access any
    element of an array
    */

```

```
printf("a[1] = %d\n", *(a+1));
printf("a[4] = %d\n", *(a+4));

}
```

Problem 24. Passing linear arrays as arguments to function using pointers

```
/*
Description : User defined
              function for printing an array
Programmer : K. Prasanna Kumar
Last Modified Date : 24th April
              2020
*/

#include <stdio.h>

// Function Declaration or
// Prototype
void printing_array(int* , int);

void main()
{
// Declaration and Initialization
// of array
int arr[7] = {7, 8, 4, 2, 1, 0,
              9};

// Function call for printing an
// array
printing_array(arr, 7);

}

// Function Definition
void printing_array(int *a, int
                    Size)
{
for(int i=0; i<Size; i++)
{
printf("a[%d] = %d\n", i, *(a+i)
);
}
}
```

Problem 25. Return an array from the user defined function

```
/*
```

Description : Defining a function that can return an array as its output.

Programmer : K. Prasanna Kumar

Last Modified Date : 21th May 2021

Concept : Name of the array gives the base address of the array

```
*/
```

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

```
// Function Declaration or
// Prototype
```

```
int *initialize_array(int s);
```

```
void main()
```

```
{
int size = 5;
int *a;
a = initialize_array(size);
for (int i = 0; i < size; i++)
{
printf("%d\n", a[i]);
}
}
```

```
// Function Definition
```

```
int *initialize_array(int s)
```

```
{
static int arr[5];
for (int i = 0; i < s; i++)
{
printf("Enter the value of arr[%d] : ", i);
scanf("%d", &arr[i]);
}
return arr;
}
```

Problem 26. Printing a String

```
/*
```

Description : Printing a String and String literal

Programmer : K. Prasanna Kumar

Last Modified Date : 24th April 2020

Definitions :

1. Array : Group of Elements with

```

    same datatype
2. String : Group of Charators. So
, It can be called as charator
array
3. String literal : Sequence of
charators enclosed within a
double quotes
    Eg: "Hello_World"
Note : (') Single Quote is for
charator and (" ") Double Quote
is for string literal
*/

#include <stdio.h>
int main()
{
    // Declaration and Initialization
    of String
    char Name[] = { 's', 'c', 'h', 'o', 'o',
                    , 'l' };
    /* String Needs one NULL CHARACTOT
    */
    // printing a string
    printf("%s\n", Name);

    // Declaration and Initialization
    of String Literal
    char name[] = { "Hello_World" };
    printf("%s\n", name);

    return 0;
}

```

9.2 Multi-Dimensional Array

Problem 27. Print a matrix

```

/*
Description : Printing a Matrix
Introduction to Multi-Dimensional
Array
Program : K. Prasanna Kumar
Last Modified Date : 20th April
2020
*/

#include <stdio.h>
void main()
{
    int mat[2][3];

```

```

    int i, j; // Index Variables

    for (i=0 ; i<2 ; i++)
    {
        for(j=0 ; j<3; j++)
        {
            printf("Enter the value of mat[%d
                ][%d]\n", i, j);
            scanf("%d", &mat[i][j]);
        }
    }

    // Printing a matrix
    for (i=0 ; i<2
        ; i++)
        printf("Matrix A=\n");

    for(i=0; i<2; i++)
    {
        for(j=0 ; j<3; j++)
        {
            printf("%d\t", mat[i][j]);
        }
        printf("\n");
    }
    printf("]\n");
}

```

Problem 28. Passing matrix as arguments to function

```

/*
Description : Printing a matrix
using user defined function.
Programmer : K. Prasanna Kumar
Last Modified Date : 24th April
2020

Note : Multi-Dimension array is
array of arrays
*/

#include <stdio.h>
// Declaration or Prototype
void printing_matrix(int r, int c,
                    int a[r][c]);

void main()
{
    // Declaration and Initialization

```

```

    of Matrix
    int A[4][2] =
        {{1,2},{4,5},{7,8},{9,6}};
    printf("A=\n");
    printing_matrix(4, 2, A);
    // A is the base address of the
    matrix
}

// Definition
void printing_matrix(int r, int c,
    int a[r][c])
{
    for (int i = 0; i<r; i++)
    {
        for(int j = 0; j<c; j++)
        {
            printf("%d\t", a[i][j]);
        }
        printf("\n");
    }
}

```

Problem 29. Passing multi-dimensional arrays as arguments to function using pointer logic of multi-dimensional array

```

/*
Description : Printing a matrix
              using user defined function with
              pointers.
Programmer : K. Prasanna Kumar
Last Modified Date : 24th April
                2020

Note : Multi-Dimension array is
       array of arrays
       based on the above concept
       following formula is obtained

a[i][j] = (*(a + i)+j)

a[i][j][k] = (*(a + i) + j) + k)

*/

#include <stdio.h>
// Declaration or Prototype
void printing_matrix(int r, int c,
    int a[r][c]);

```

```

void main()
{
    // Declaration and Initialization
    of Matrix
    int A[4][2] =
        {{1,2},{4,5},{7,8},{9,6}};
    printf("A=\n");
    printing_matrix(4, 2, A);
    // A is the base address of the
    matrix
}

// Definition
void printing_matrix(int r, int c,
    int a[r][c])
{
    for (int i = 0; i<r; i++)
    {
        for(int j = 0; j<c; j++)
        {
            printf("%d\t", (*(a + i) + j)
                );
        }
        printf("\n");
    }
}

```

10 STRUCTURES

Problem 30. Introduction to "C Structures"

```

/*
Description : Progress Report of
              Student
Programmer : K. Prasanna Kumar
Last Modified Date : 14th May 2021

Note :
1. Array has the capability to
   store more than one entry but
   they should be of same data_type
2. In array the entries are known
   as elements
3. Structure has the capability
   to store more than one entry, of
   same or different data_types
4. In structure the entries are
   known as members

```

```

Syntax :
* Definition :--
struct TAGLINE
{
    data_type variable_1;
    data_type variable_2;
    ..
    ..
};

* Deculation and Initialization
:--
struct TAGLINE Stracuture_variable
= {variable_1, variable_2, ..};
..

Note :
1. We can have multiple stracute
   variable for a single structure
2. Structure can be defined any
   where in the program
*/

#include <stdio.h>
// Defination of structure
struct student
{
    int id;
    char name[20];
    float per_age;
};

int main()
{
    // Decularation and Initilization
    of structure variables
    struct student s1 = {17,"Prasanna_
    Kumar", 78.2};

    // Accessing members of the
    structures
    printf("Details_of_student_1_:_\n"
    );
    printf("ID_of_studen_1_:_%d_\n",s1
    .id );
    printf("Name_of_student_1_:_%s_\n"
    , s1.name);
    printf("Percentage_of_student_1_:_
    %float_\n", s1.per_age);

```

```

return 0;
}

// same program is explain in
  alternate methods in another
  program

```

Problem 31. Compile time Initialization of "C Structure"

```

/*
Description : Progress Report of
              Student
Programmer : K. Prasanna Kumar
Last Modified Date : 14th May 2021

```

```

Syntax :
* Definition :--
struct TAGLINE
{
    data_type variable_1;
    data_type variable_2;
    ..
    ..
};

* Deculation :--
struct TAGLINE Stracuture_variable;

* Initialization :--
Stracuture_variable.variable_1 = ;
Stracuture_variable.variable_2 = ;
..
*/

#include <stdio.h>
// Defination of structure
struct student
{
    int id;
    char *name;
    float per_age;
};

int main()
{
    // Decularation and Initilization
    of structure variables
    struct student s1 = {17,"Prasanna_

```



```

    Kumar", 78.2};

// Alternative method
// Decularation of stracture
variables :--
struct student s2;

// Initilization of stracture
variables : --
// Complie Initlizin
s2.id = 10;
s2.name = "Shrey_More";
s2.per_age = 81;

// Accessing members of the
structures
printf("Details_of_student_1_: \n"
);
printf("ID_of_studen_1_: %d\n", s1
.id );
printf("Name_of_student_1_: %s\n"
, s1.name);
printf("Percentage_of_student_1_:
%float\n\n", s1.per_age);

printf("Details_of_student_2_: \n"
);
printf("ID_of_studen_2_: %d\n", s2
.id );
printf("Name_of_student_2_: %s\n"
, s2.name);
printf("Percentage_of_student_2_:
%float\n", s2.per_age);

return 0;
}

```

Problem 32. Run time Initialization of "C Struc-
ture"

```

/*
Description : Progress Report of
Student
Programmer : K. Prasanna Kumar
Last Modified Date : 14th May 2021
Definition : A string is a user
defined datatype, that can be
used to group
elements of different datatype in
to a single type.

```

```

*/

#include <stdio.h>
// Defination of stracture
struct student
{
    int id;
    char name[20];
    float per_age;
};

int main()
{
// Decularation and Initilization
of stracture variables
struct student s1 = {17,"Prasanna_
Kumar", 78.2};

// Alternative method
// Decularation of stracture
variables :--
struct student s3;

// Initilization of stracture
variables : --
// Runtime Initlization
printf("Enter_the_details_of_
Student_3_\n");
scanf("%d%s%f", &s3.id, s3.name,
&s3.per_age);

// Accessing members of the
structures
printf("Details_of_student_1_: \n"
);
printf("ID_of_studen_1_: %d\n", s1
.id );
printf("Name_of_student_1_: %s\n"
, s1.name);
printf("Percentage_of_student_1_:
%float\n\n", s1.per_age);

printf("Details_of_student_3_: \n"
);
printf("ID_of_studen_3_: %d\n", s3
.id );
printf("Name_of_student_3_: %s\n"
, s3.name);
printf("Percentage_of_student_3_:
%float\n\n", s3.per_age);

```

```
return 0;
}
```

Problem 33. Alternate Method for Definition & Declaration

```
/*
Description : Introduction to
Structures in C.
Programmer : K. Prasanna Kumar
Last Modified Date : 14th May 2021
```

```
syntax :
Definiton & Decularation :--
struct
{
    datatype variable_1;
    datatype variable_2;
    ..
} struct_variable_1 ,
  struct_variable_2 ,
  struct_variable_3 ...;
```

```
Initilization :--
stract_variable_1.variable_1 = ;
stract_variable_1.variable_2 = ;
```

```
Note : TAGLINE is optional
*/
```

```
#include <stdio.h>
// Definition & Decularation of
Structure
struct
{
    int id ;
    char name[20];
    float per_age;
}s1;
```

```
void main()
{
// Initilization of Stracture
printf("Enter the details of
Student_1_: \n");
printf("id_: ");
scanf("%d", &s1.id);
printf("Name_: ");
scanf("%s", s1.name);
printf("Percentage_: ");
```

```
scanf("%f", &s1.per_age);
```

```
/* Initilization can be done using
any of the previous methods.
currently, we have gone with
runtime intilization */
printf("\n\n\n");
```

```
// Accessing members of the
structures
printf("Details of student_1_: \n"
);
printf("ID of studen_1_: %d\n", s1
.id );
printf("Name of student_1_: %s\n"
, s1.name);
printf("Percentage of student_1_:
%float\n", s1.per_age);

}
```

11 FILE HANDLING

Problem 34. Writing data into a text file

```
/*
Description : Creating an array in
the range [0 to 6] with
an interval of 0.1.
Programmer : K. Prasanna Kumar
Last Modified Date : 24th May 2021
*/
```

```
#include <stdio.h>
int main()
{
// Decularation and Initialization
of array
int size = 6/0.1;
float arr[size];
```

```
// Decularation of file pointer
FILE *fp;
fp = fopen("date.txt", "w");

printf("Arr_={ ");
for (int i = 0; i < size; i++)
{
    arr[i] = i*0.1;
    printf("%f\t", arr[i]);
```

```

    fprintf(fp, "%f", arr[i]);
}
printf("}\n");
fclose(fp);

printf("Written to file \n
      Successfully \n");
return 0;
}

```

Problem 35. Reading data from a text file

```

/*
Description : Scanning data from
              the file
Programmer : K. Prasanna Kumar
Last Modified Date : 24th April
              2020

Note : Create a file by name "num.
      txt" before running
this program, if "num.txt" does
      not exist in your directory
*/

#include <stdio.h>
int main()
{
    // Declaration of file pointer
    FILE *fp;

    // Declaration of variable
    float num;

    fp = fopen("num.txt", "r");

    fscanf(fp, "%f", &num);
    printf("%f\n", num);

    fclose(fp);

    return 0;
}

```

REFERENCES

- [1] Gaurav Bhorkar, "Programming Fundamentals in C (NU / MSBTE / Beginner Level)", youtube video playlist <https://www.youtube.com/playlist?list=PL90FACD026D4959BE>.
Email id : gaurav.bhorkar@gmail.com