Introduction to C Programming

K Prasanna Kumar

Contents

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()
{
// Decluration 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("Substration 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;


// Deculararation 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);
    // Execlusive 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 ;

// Deculararation of Variables
printf("Eneter 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. 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 charator data 
    type is : %ld \n", sizeof(char))
    ;
return 0;
}
```

3 Scope of Variables

Definition : Scope of variable is a block or a region where a variable is defined, declared & Used.

- If the block of a region ends, the variable automatically gets destroyed
- Scope is the lifetime of the variable
- Scope is the area under which the variable is applicable or alive.

Problem 6. Basic Principle of Scoping

```
/*
 Description : Defining the scope
    of a variable by dividing the
 function into blocks using carly
    brackets "{" "}".
 Programmer : K. Prasanna Kumar
 Last Modified Date : 12 July 2021
 */

#include<stdio.h>
void main()
{
 int var1 = 24, var2 = 14;
         {
            int var1 = 3;
            printf("Value inside the
               block : %d\n", var1);
            printf("Value of the
               variable declared
               above the block : %d\n
               ", var2);
         }
 printf("Value-1 outside the block
    : %d\n", var1);
 printf("Value-2 outside the block
    : %d\n", var2);
}

/*
 Both the variable are with same
    identifier (name). But the
    opetation
 takes place in the blocks.
 Variable declared above the block
     can be accessed
 Variable inside the block cannot
    be accessed
 */
```

Problem 7. Local Variable

```
/*
 Discription : Example to
    understant local Variable
    Concept
 Definition : Variable declared
    inside a block or a function
    and accessable
 in the same block or function is
    known as local variable.
 Programmer : K. Prasanna Kumar
 Last Modified Date : 12 July 2021
 */

#include<stdio.h>
int fun(); // User defined
   function
void main()
{
         {
         // Block_1
         int A = 3;
         printf("Value of A in
            Block_1 : %d\n", A);
         }
         {
         // Block_1
         int A = 6;
         printf("Value of A in
            Block_2 : %d\n", A);
         }

         fun();
}

int fun()
{
         int A = 32;
         printf("Value of A in user
            -defined function : %d\n
            ", A);
         }

/* Variable A is a local to
   indival block and indival
   function */
```

Problem 8. Global Variable

```
/*
 Description : Example to
    understand Global Variable
 Definition : Variable outside of
    all functions is known as
    gloabal
 variable
 Programmer : K. Prasanna Kumar
 Last Modified Date : 12th July
    2021
 */


#include<stdio.h>
int B = 67;
int fun(); // User defined
   function
void main()
{
         {
         // Block_1
         int A = 3;
         printf("Value of A in
            Block_1 : %d\n", A);
        printf("Calling Global 
            Variable B in Block_1 of
             main : %d\n", B);
        }
        {
        // Block_1
        int A = 6;
        printf("Value of A in 
            Block_2 : %d\n", A);
        }
        printf("Calling Global 
            Variable B in main 
            function : %d\n", B);
        fun();
}

int fun()
{
        int A = 32;
        printf("Value of A in user
            -defined function : %d\n
            ", A);
        int C = A + B;
        printf("Value of C : %d\n"
            , C);
        printf("Value of C is 
            obtained as (local 
            variable A) + (Global 
            variable B)\n");
        }

/* Variable A is a local to
   indival block and indival
   function */
```

4 Conditional Statements

4.1 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;
}
```

4.2 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");}

}
```

4.3 Switch-Case

Problem 9. 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 10. Menu Program to do the Arithmetic Operations.

5 Iteration statements

5.1 While loop

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

Syntax :

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


#include<stdio.h>
int main()
{
int a;
printf("\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;
}
```

5.2 for loop

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

for(Initialization of incriment
   variable, Condition for True,
   Incriment)
{

Statements or Function calls

}
*/

#include<stdio.h>
int main()
{
int a;
printf("\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;
}
```

5.3 do while loop

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

do{

Statments or Function calls

}(The condition is True);

Note :
1. Semicolum (;) 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;
}
```

5.4 Infinite loop

Problem 11. 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
   codition , No Incriment
*/

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

for( ; ; )
{
printf("I will not Stop \n");
printf(" ---------------- Press \" 
   Ctrl + c\" to STIP me 
   ------------- \n");
}

return 0;
}
```

Problem 12. 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
   conditon 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 13. 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(conditon)
break;
}

*/

#include<stdio.h>
void main()
{
int a;
while(1)
{
printf("I will not Stop  untill 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;

}
}
```

6 Pointers

Problem 14. 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()
{
// Varible
int x = 5;

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

// Declartion 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;
}
```

7 Functions

Problem 15. 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 Deculararation 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 16. Program to explain Function Arguments

```
/*
Description : Program to explain
   passing Arguments in an user
   defined function
Note:-
1. Arguments of a functin are also
    called Parameters
2. Arguments are seperated 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
sufficent in function Decluration.
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 defination
are known as formal arguments
*/
printf("Add␣function␣of␣a␣&␣b␣is␣%
   d␣\n", c);
return 0;
}


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

Problem 17. Swap function using pointers

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

#include<stdio.h>
// Declartion 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;
}
```

7.1 Recursion

Problem 18. 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 Decularation
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);
// Funtion call has been done in
   the same function
        return (f);
}
}

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

7.2 Header File

Problem 19. 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␣Opertion␣can␣be␣
   performed␣using␣this␣function,␣
   since␣a<b␣\n");
}
```

8 Arrays

8.1 Uni-Dimensional Array

Problem 20. 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 Enteries of an array
   are as follows : \n {");
for (i =0; i<5; i++)
{
printf("%d, \t", arr[i]);
}
printf("\bd } \n");
return 0;
}
```

Problem 21. Program a log book to enter age of the team members in an office.

```
/*
Descritpion : 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 .. \
   nAs it is requested to terminate
   by Enterning /'0/' \n");
}
```

Problem 22. 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
           avalible in the index
           number %d \n", i);
        return;
        }
}
printf("%d is not avalible in this
   array \n", n);
}


/* Diffrence 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 23. 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 24. Accessing arrays with pointers

```
/*
Description : Accessing liear
   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 25. 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 Decularation or
   Prototype
void printing_array(int* , int);

void main()
{
// Decularation and Initilization
   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 26. Return an array from the user defined function

```
/*
Description : Defining a funtion
   that can retrun 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 Decularation 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 27. 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. Sring 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()
{
// Deculartion and Initialization
   of String
char Name[] = {'s','c','h','o','o'
   ,'l'};
/*String Needs one NULL CHARATOT
   */
// printing a string
printf("%s\n",Name);

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

return 0;
}
```

8.2 Multi-Dimensional Array

Problem 28. Print a martix

```
/*
Description : Printing a Matrix
Introduction to Muti-Dimenssional
   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] ", i, j);
scanf("%d", &mat[i][j]);
}
}

// Printing a matrixfor (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 29. 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>
// Deculararation or Prototype
void printing_matrix(int r, int c,
    int a[r][c]);

void main()
{
// Deculararation 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 30. 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>
// Deculararation or Prototype
void printing_matrix(int r, int c,
    int a[r][c]);

void main()
{
// Deculararation 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");
 }
}
```

9 Structures

Problem 31. Introduction to "C Structures"

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

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

Syntax :
* Definition :--
struct TAGLINE
{
        data_type varible_1;
        data_type varible_2;
        ..
        ..
};

* Deculration and Initialization
   :--
struct TAGLINE Stracuture_varible
   = {varible_1 , varible_2 , ..};
..

Note :
1. We can have multiple stracute
   varible for a single stracture
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 varibles
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
   attërnate methods in another
   program
```

Problem 32. 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 varible_1;
        data_type varible_2;
        ..
        ..
};

* Deculration :--
struct TAGLINE Stracture_varible;

* Initialization :--
Stracture_varible.varible_1 = ;
Stracture_varible.varible_2 = ;
..
*/

#include<stdio.h>
```

```
// Defination of stracture
struct student
{
        int id;
        char *name;
        float per_age;
};

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

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

// Initilization of stracture
   variables : --
// Complie Initlizatin
s2.id = 10;
s2.name = "Shrey␣More";
s2.per_age = 81;

// Accessing members of the
   stractures
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 33. Run time Initialization of "C Structure"

```
/*
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()
{
// Deculararation and Initilization
    of stracture variables
struct student s1 = {17,"Prasanna␣
   Kumar", 78.2};

// Alternative method
// Deculararation 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
   stractures
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 34. Alternate Method for Definition & Declaration

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

syntax :
Definiton & Deculararation :--
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 & Deculararation 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);

}
```

Problem 35. Size of a structure

```
/*
Description : Structure is a user
   defined data type. This program
   is to find the size of a
   structure
Programmer : K. Prasanna kumar
Last Modified Date : 27th May 2021
*/

#include<stdio.h>
struct student_data
{
        int S_No;
        char name;
        float percentage;
        char remarks[23];
};

void main()
{
 struct student_data s1;
 printf("Size of structure element
    1 : %ld \n", sizeof(s1.S_No));
 printf("Size of structure element
    2 : %ld \n", sizeof(s1.name));
 printf("Size of structure element
    3 : %ld \n", sizeof(s1.
    percentage));
 printf("Size of the structure
    variable s1 : %ld \n", sizeof(
    s1));
}


/*
Alter the date types of the
   elements in the structure,
compile and run the program, we
   will learn the beauti
inside the size of an structure.

Note : It is an experimented self
   learning example
*/
```

10 File handling

Problem 36. Writing data into a text file

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

#include<stdio.h>
int main()
{
// Deculararion 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("Writen to file
   Successfully \n");
return 0;
}
```

Problem 37. 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 runing
this program, if "num.txt" does
   not exists in your directory
*/

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

// Deculartation 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