LAB - 1

Input / Output Statements With Basic Arithmetic Operations

AIM:

To understand the working of Input & Output statements and working of Basic Arithmetic Operations.

OBJECTIVE:

- To show information to user and also get information from the user.
- To make changes to information using arithmetic operators and expressions.

PROGRAMS :

- **Experiment - 1:**

To calculate the BMI of a person.

Input / Output:

Get Height and weight from the user.
Print the calculated BMI to the user.

Code :

```
# include <stdio.h>
int  main(){
    float  weight, height;
    printf("\n Enter  your weight  in kg : ");
    scanf("%f", &weight);
    printf("\n Enter  your  height  in  m : ");
    scanf("%f", &height);
    printf("Your  BMI  is : %f", (weight/(height * height)));
    return  0;
}
```

Test Cases:

- Input : Weight = 50.0, hight = 1.7
- Output : BMI = 17.3010

- Experiment - 2:

Convert Kilometres to Miles

Input / Output:

Get kilometres from the user.

Print the calculated Mile equivalent to the user.

Code:

```
#include <stdio.h>

int main(){
   float km;
   printf("Enter Value in km: ");
   scanf ("%f", &km);
   printf("The same value in miles = %f", (km*0.621));
   return 0;
}
```

Test Cases:

- Input : Km = 100
- Output : Miles = 62.1

- Experiment - 3:

Find the discriminant of a quadratic equation.

Input/Output:

Get the co-efficients a, b & c from the user

Print the calculated discriminant to the user.

Code:

```
#include <stdio.h>

int main(){
   int A,B,C;
   printf("Enter the co-efficients , A,B & C : ");
   scanf(" %d %d %d", &A, &B, &C);
   printf("Discriminant = %d ", (B*B)-4*A*C);
   return 0;
}
```

Test Cases:

- Input : A = 1 , B = 4 , C = 1
- Output : Discriminant = 12

REMARKS :

- To remember the format specifiers for every data type .
- B^2 can be written as B*B for easier understanding and a simpler method .
- Usage of math modules would enable us to raise any value as a power to another number.

CONCLUSION :

The given Experiments helps us to understand format specifiers and getting Input as well as showing output to the user .

LAB-2

Decision STATEMENTS

AIM:

To understand the working of if. else and else if statements in C programming language.

OBJECTIVE:

- To use functions like sqrt(), pow() from math library.
- To implement if, else if and else blocks according to the requirement.

PROGRAMS:

- **Experiment-1:**

To print 'Even' or 'Odd' if entered value is greater than 0 else, to print the root of the negative number in complex form.

Input/output:

To get a number from the user.

Print Even or Odd if number positive and print root if negative.

Code:

```
#include <stdio.h>
#include <math.h
int main(){
    float Num;
    printf("\nEnter The number : ");
    Scanf("%f ", &Num);
    if (Num>0){
        if (remainder (Num, 2)==0){
            printf("\nEven");
        }else{
            printf("\n Odd");
        }
    }else if (Num<0){
        printf("\n The root is %.2f i", pow (0-Num, 0.5));
    }else{ printf("\n zero"); }
    return 0; }
```

Test Cases:

• Input : 10	• Input : - 5
• Output: Even	• Output : 2.23 i

• **Experiment - 2** :

To check if a point with co-ordinates x,y belongs inside a 2D box with opposite verteres at (0,0) and (4,4):

Input / output:

To get values of x any y from user

Print if x,y is inside or outside the box.

Code :

```
#include <stdio.h>
int main(){
    float P1[2];
    printf(" Enter the co-ordinates of x & y : ");
    scanf("%f %f", &P1[0], &P1[1]);
    if (P1[0]>0 && P1[0]<4 && P1[1]>0 && P1[1]<4){
        printf("\n The point is in the box.");
    }else{
        printf("\n The point is outside the box ");
    }
    return 0;
}
```

Test Cases:

• Input : 0.1 0.4

• Output : In the Box

• Input : 4.1 3.1

• Output : Outside the Box

• **Experiment - 3**:

To find the roots of a quadratic equation. Also find the complex roots if they exists.

Input / Output :

To get the co-efficients of the equation, A, B, c.

Print the roots (Real, Unreal)
↳ Complex

Code :

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

int  main() {
    float  a,b,c,d ;
    printf("Enter  the  co-efficients  of the equation\n");
    printf (" A , B,  C =  ");
    Scanf ("%f  %f  %f ", &a , &b , &c);
    if ( b*b - (4* a *c) >=0 ){
        d = pow ( b*b  - (4* a *c) ,0.5);
        printf ("One  root  is  = %.2f", ((0-b)+d) /2*a);
        printf ("\n Another  root  is = %.2f ",((0-b)-d) /2*a);
    }else {
        d = sqrt( 0-(b*b-(4*a*c)));
        float  RP  =  (0-b) / (2*a) ;
        float  IP  =  (0-d )/ (2*a);
        if (IP > 0) {
            printf("One  root is  =%.2f  +  i(%.2.f)", RP,IP);
            printf("\n Another  root is = %.2f - i(%.2f)", RP,IP);
        }else {
            printf("One root  is  = %.2f  - i(%.2f) \n", RP,IP);
            printf ("Another root is = %.2f + i(%.2f)", RP,IP);
        }
    }
    return 0;
}
```

Test Cases :

- Input : 1 4 1
- Output : - 0.267
- 3.732

- Input : 1 1 1
- Output : - 0.5 + 0.86
- 0.5 - 0.86

REMARKS :

- Whenever a function from libraries like math.h is used , 'gcc <filename>.c ' is not enough to compile the executable file. Use 'gcc <filename>.c - lm' to compile it.

- The executable file that is created after compiling the program can be renamed using the following command,,

```
`gcc <filename>.c -o <output-filename>.exe'
```

CONCLUSION :

The given experiments helps us to understand the working & decision statements.

7/11/23

LAB - 3

PATTERN GENERATOR AND USAGE OF ARRAYS WITH LOOPS

AIM:

To understand the logic of generating patterns and using arrays to manipulate data.

OBJECTIVES:

- To generate patterns using looping statements.
- Usage of while, do..while and for loops.

PROGRAMS :

- Experiment -1 :

To print the following patterns when Inputs A & B are 4 and '*'.

(a)

```
* * * *
* * *
* *
*
```

(b)

```
   *
  * *
 * * *
* * * *
 * * *
  * *
   *
```

Input/Output :

To get number of lines and the pattern character To Print the above patterns.

Code:

(a)

```
for (int i=0 ; i<A ; i++) {
      for (int j =A-i ;j>0 ;j--) {
              printf(" %c ",B);
      }
     printf("\n");
}
```

(b)

```
for (int i=0 ; i<A ; i++){
    for (int j= A-i ; j>0 ; j--){
        printf("  "); //2 empty spaces
    }
    for (int j=i ; j>0 ; j--){
        printf("%c   ",B); // 3 empty spaces
    }
    printf(" \n");
}
for (int i=0 ; i<A ; i++){
    for (int j=i ; j>0; j--){
        printf("  "); // 2 empty spaces
    }
    for (int j= A-i ; j>0 ; j--){
        printf("%c   ",B); // 3 empty spaces
    }
    printf ("\n");
}
```

Test Cases :

Input: A = 3 , B = "#"

```
Output: # # #
        # #
        #

            #
        #       #
    #       #       #
        #       #
            #
```

- Experiment . 2 :

To sort all the elements in an array in descending order.

Input / Output :

To get the array of numbers.

print the ordered array of numbers.

Code :

```
int *largest (int *a, int *b) {
    int *lg , ln = *a;
    lg = a;
    int d= b-a;
    for (int i=0 ; i<d; i++) {
        if (a[i] >ln ){
            lg  = &a[i];
            ln  = a[i];
        }
    }
    return lg;
}
void swap_int (int *a, int *b){
    int c =*a ;
    *a = *b;
    *b=c;
}
int main () {
    int Array [100];
    for (int i=0; i<n ;i++) {
        f = &Array [i];
        g= largest (f ,&Array [n]);
        Swap-int (f,g);
    }
}
```

Test Cases:

Input: List of ~~Arra~~ numbers

Output: The Arranged Array.

- Experiment - 8:

To generate a Time series analysis and understand the seasonal trend by finding its moving average.

Input /Output:

To get the data varying with time.
Print the Moving average for a set season time and the seasonal trend.

Code :

```
float avg (float *Start ,int length) {
    float average =0;
    for (int i=0; i<length ;i++) {
        average += start[i];
    }
    average /= length;
    return average;
}

int main() {
    float Values[100];
    float Moving_Average[100], seasonalTrend[100];
    int b, n=100;
    for (int i=0; I<(b-1)/2 ; i++) {
        printf("\n%f ", values[i]);
    }
    for (int k=0; k<100 ;k++) {
        Moving-Average [k] = avg (&values [k}, b];
        SeasonalTrend [k]= Values [k+1] - Moving Avg [k];
        printf("\n%f |%f |%f ", Values[k], MA [k], ST [k]);
    }
    return 0;
}
```

Test Cases :

- Input: Array of numbers and b= 3 (season length).
- Output: Tabular output of Moving Average and Seasonal Trend.

· Experiment - 4 :

To find the trace of a matrix, Max element and the 1-Norm of the Matrix. And to check if the square of the Matrix is same as the matrix

Input/Output:

To get the elements of the matrix

print the trace, Max element and Norm.

Code:

```
void trace(){
    int T=0;
    for(int i=0;i<25;i++){
        for(int j=0;j<25;j++){
            if(i==j){
                T+=Matrix[i][j];
            }
        }
    }
    printf("Trace = %d",T);
}
void MaxElem(){
    int max = Matrix[0][0];
    for(int i=0;i<25;i++){
        for(int j=0;j<25;j++){
            if(Matrix[i][j] >Max){
                Max = Matrix[i][j];
            }
        }
    }
    printf(" Max = %d", max);
}
void Norm(){
    int RM =0;
    for(int i=0;j<25;i++){ int sum=0;
        for(int j=0;j<25;j++){
            Sum+=Matrix[i][j]; }
        if(Sum>RM){
            ~~Row~~ RM = Sum; }
        printf(" Norm: %d ", RM);
}
int Matrix[25][25] = {} // values
int main(){

        trace();

        MaxElem();

        Norm();

}
```

Test Cases :

- Input : Nested Array (Matrix) of Elements
- Output : Trace of Matrix,
  MaxElement of Matrix,
  Norm of Matrix.,

14/11/23

REMARKS:

- Try to figure out how to create dynamic arrays.
- Using Realloc () functions to change the size of an array, while inputing values.

CONCLUSION:

The given experiments helps us to understand the working Loops and Arrays.

LAB. 4

STRINGS INTRODUCTION TO STRUCTURES

AIM:

To understand the working of strings and structures and Usage of some string manipulating functions

OBJECTIVES:

- To learn some string functions
- To Introduce in the working of structures

PROGRAMS:

- Experiment -1 :

Understanding in-built functions in <string> and<ctype>

Input/Output :

Get a sentence from the user as a string Print the given string as a title and the crazy case .

Code:

```
void  start_case (char *str_ptr){
      int i=0, w=1;
      while (str_pt [i] != '0'){
            if (w= =1){
                  str_pt[i] = toupper (str_pt [i]):

             }else {
                  str_pt [i] = tolower (str_pt [i]);

             }
             if (str_pt [i] =='*'){
                  w=1;
             }else {
                  w=0;

             } i++;
      }
}
```

```
void  crazy_case (char  *str_pt){
    int  i=0, lo=1;
    while (str_pt[i]!='\0'){
        if (lo==1){
            str_pt[i] = toupper (str_pt[i]);
        }else {
            str_pt[i] = tolower (str_pt [i]);
        }
        if (str_pt[i] ==' '){
            lo = 0;
        }else {
            lo=1;
        }i ++;
    }
}

// 1)
int  main(){
    printf("Enter  the  string :");
    gets(str_input);
    str_ptr  = &str_input[0];

 // 2)
    strcpy (str_copy, str2);
     printf ("str_copy : %s \n", str_copy);

 // 3)
    int a= strcmp (str_copy, str2);
    if (a== 0){
        printf("Str_cmp : Both  are  same ");
    }else{
        printf ("Str_cmp: Disimilar ");
    }
```

Test Cases:

Input : LALITh aDITyhyAn

Output : LAlith Adithyan
lALITH aDITHYAN

- Experiment - 2:

Remove Occurrences of a character in a string

Input / Output:

To get a string from the user and print the same string with a character removed, which is specified by the user.

Code:

```
void remove_first (char *str_pt, char x) {
        int i=0, j=0, f=1;
        while (str_pt[j] != '0') {
            if (str_pt[i]== x) {
                if (f == 1) {
                    j ++;
                    f =0;
                }
                str_pt[i]= str_pt[j];
                i++;
            } else {
                str_pt[i] = str_pt[j];
                i++;
            } j ++ ;
        }
}
void remove_all (char *str_pt, char x) {
        int i=0, j = 0;
        while (str_pt[j] != '0' ) {
            if (str_pt [i] == x) {
                j++ ;
                str_pt [i]= str_pt [j];
            } else {
                str_pt [i] = str_pt[j];
            } j++; i ++;
        }
}
```

Test Cases :

- Input : Lalith adithyan
  a

- Output : Llith adithyan
  Llith dithyn

- Experiment - 3 :

Check whether two strings have the same set of characters.

Input / Output :

To get two strings from the user and print if both strings are made of the same set of characters.

Code :

```
bool aparecium (char *str-pt-1, char *str-pt-2){
        bool flag = true;
        int i=0; j=0, f=0;
        while(str-pt-1 [i]!='\0'){
            j = 0; f=0;
            while(str-pt-2 [j] != '\0'){
                if (str_pt_1 [i] ==str-pt-2 [j]){
                    str_pt-2[j] = '~';
                    f=1;
                } j++;
            } if(f==0){
                flag = false;
                break;
            } i++;
        }
        return flag;
}
```

Test Cases :

- Input : TOM MARVOLO RIDDLE
  IMMORTAL DOVE LORD
- Output : They have the same set of characters.

· Experiment - 4

A C program that reverses the given string and returns true or false if it is a palindromme.

Input /Output:

To get String from the user

And print true if the string is a palindrome else false.

Code:

```
Void reverse_string (Char *str-pt){
    int length = strlen (str-pt) ;
    int S=0 , E = length-1;
    while (S<E){
        char  a = str_pt [S];
        str-pt[S] = str_pt[E];
        str-pt [E] = a;
        S++ ;E--;
    }
}
```

Test Cases:

.Input: "Martin "

.Output: " nitraM"

· Experiment - 5:

Program that uses structures to store personal details and prints each element.

Input /output

Get details of a person and store into a stucture . Print the saved detais.

Code :

```
struct Personal {
    int age;
    Char *name;
    char *native;
    int roll-no [10);
}:
void    print_struct (stuct Personal *mydat){
     printf(" \nYour Age is : %d", mydat -> age);
     printf(" \n Your Name is :  ");
     puts(mydat -> name);
     printf ("\nYour Native is : ");
     puts(mydat ->native);
     printf("\nYour Roll number is : ");
     for (int i=0; i<10; i++){
         printf ("%d", mydat -> roll_no[i]);
     }
}
int main(){
     struct Personal mydata;
     struct Personal *mydata_ptr;
     int a,b:
     long int r;
     char Name[25], Native [25], *NP. *VP:
     printf("Enter Age : ");
     scanf("%d", &a);
     printf("Enter Name :");
     Scanf( "%d", &b);
     gets(Name);
     NP = &Name [0] ;
     printf("Enter Native : ");
     gets(Native);
     VP = & Native [0];
     mydata.age =a;
     mydata.name = NP;
     mydata.native = VP;
     mydata_ptr = &mydata ;
     print_struct (mydata_ptr);
}
```

- REMARKS :

  - Make sure to include string.h module while using string functions.
  - Usage of membership operators ('→') while accessing data from a structure using pointers.

- Conclusion:

  The given Experiments helped us understand Strings, string functions and introduction to structures.