AIM: Write a program to find the root of an equation using **BISECTION METHOD.**

We use the following functions to perform bisection program:

* **initialize( ):** It takes the polynomial from the user and stores them in a linked list, pointed by a variable.
* **operation( ):** The operation function finds the interval in which the roots of the polynomial lie.
* **verify\_roots( ):** It verifies the given interval(by the user) of the roots of the polynomial.
* **find\_roots( ):** By using the interval of the roots(either entered by the user or determined by the program), the find\_roots function finds the exact roots by taking ‘**allowed error and no. of iterations**’ as argument from the user.
* **func( ):**This function finds the solution of the entered polynomial at a particular value.

**PROGRAM**

#include<stdio.h>

#include<conio.h>

#include<math.h>

**float func(float);**

**float find\_root(float,float,float,int);**

**int verify\_roots(float,float);**

**float operation(int);**

**int initialize();**

struct node

**coeff**

**expo**

**\*next**

**p**

Each node

{

int coeff;

int expo;

struct node \*next;

};

struct node \*p=NULL;

**/\*-----------Main function-----------\*/**

void main()

{

int ch,k,iter,flag;

float r1,r2,root,ae;

clrscr();

flag=initialize();

printf("Initializated the polynomial successfully!");

start:

printf("\n======MENU====\n");

printf("1) Enter the limits of the roots of given polynomial\n");

printf("2) Let the program calculate the limits of the roots.\n");

printf("Your choice: ");

scanf("%d",&ch);

switch(ch)

{

**case 1:**

printf("\nEnter the Roots: ");

scanf("%f %f", &r1, &r2);

k=verify\_roots(r1,r2);

if(k==1)

{

printf("\nEnter the allowed error and number of iterations: ");

scanf("%f %d",&ae, &iter);

root=find\_root(r1,r2,ae,iter);

printf("\n the root is: %f",root);

}

else if (k==0)

{

goto start;

}

break;

**case 2:**

r1=operation(flag);

printf("\nEnter the allowed error and number of iterations: ");

scanf("%f %d",&ae, &iter);

root=find\_root(r1,r1+0.5,ae,iter);

printf(" the root is: %f",root);

break;

default:

printf("Please Enter a valid choice.\n");

goto start;

}

getch();

}

**/\*-----------find\_root function------\*/**

float find\_root(float r1,float r2, float ae, int n)

{

int count=0;

float k,avg\_prev=0, avg=0, aer=0;

if(n==0)

{

n=3;

printf("\nBy default, 3 iterations will be executed.");

}

if(ae==0)

{

ae=0.01;

printf("\nBy default, 0.01 is set as allowed error.");

}

printf("\n| #No\t| r1 \t| r2 \t| x \t|f(x)\t\t\t| aer \t| ae \t|\n");

printf("|=======|=======|=======|=======|=======================|=======|=======|\n");

do

{

printf("|%3d\t|", ++count);

printf("%0.5f|%0.5f|", r1,r2);

avg\_prev=avg;

avg=(r1+r2)/2;

printf("%0.5f|",avg);

k=func(avg);

printf(" %0.6f", k);

if(k>0)

{

/\*In case of recursion:

find\_root(avg,r2,ae,n-1);\*/

printf("\t(+ive) |");

r2=avg;

}

else if(k<0)

{

/\*In case of recursion:

find\_root(r1,avg,ae,n-1);\*/

printf("\t(-ive) |");

r1=avg;

}

--n;

if(n==-1)

{

n=0;

}

aer=fabs(avg-avg\_prev);

**/\***

**1) abs( ) is used to find the absolute value(i.e only positive) of an integer.**

**2) fabs( ) finds the absolute value(only positive) of floating numbers.**

**\*/**

printf("%0.5f|%0.5f|\n", aer,ae);

}

while(n!=0 || aer>ae);

**/\*This while statement will keep iterating unless any of one condition, i.e no. of iterations or allowed error both are satisfied.\*/**

printf("\nAfter completing %d iterations, ", count);

return avg;

}

**/\*--------verify\_roots function------\*/**

int verify\_roots(float r1, float r2)

{

float k,l;

k=func(r1);

printf("|f(%f)=%f \t|\n",r1, k);

l=func(r2);

printf("|f(%f)=%f \t|\n",r2, l);

if((k\*l)>=0)

{

printf("\nThe actual root of the polynomial do not lie between (%f, %f).", r1,r2);

return 0;

}

else if((k\*l)<0)

{

printf("\n The entered values have been tested. \nThe actual root lie between (%f, %f)",r1,r2);

return 1;

}

return 0;

}

**/\*-----------func function------------\*/**

float func(float i)

{

struct node \*temp2=p;

float value=0,value1=0;

while(temp2!=NULL)

{

value1=pow(i,temp2->expo);

value=value+(temp2->coeff)\*value1;

temp2=temp2->next;

}

return value;

}

**/\*--------operation function---------\*/**

float operation(int flag)

{

float k,l,i;

If the polynomial has no –ive

term: (Ex: 5x2 +4x+10) then

the flag remains 0. For such

equations we consider –ive

roots, i.e f(-5), f(-4)…

However, this is not applicable if

the equation has large values

for square indexes

(Ex: 9x2+x+10; 7x4+x3+1)

if(flag==0)

{

i=-5;

}

else

i=0;

k=func(i);

printf("|f(%f)=%f ",i, k);

if(k>0 || k==0)

printf(" (+ive) \t|\n");

else if(k<0)

printf(" (-ive) \t|\n");

l=func(i+=0.5);

printf("|f(%f)=%f ",i, l);

if(l>0 || l==0)

printf(" (+ive) \t|\n");

else if(l<0)

printf(" (-ive) \t|\n");

while((k\*l)>=0)

{

i=i+0.5;

k=l;

l=func(i);

printf("|f(%f)=%f ",i, l);

if(l>0 || l==0)

printf(" (+ive) \t|\n");

else if(l<0)

printf(" (-ive) \t|\n");

}

printf("\nThe roots lie between ( %f , %f )", i-0.5,i);

return (i-0.5);

}

**/\*--------initialize function----------\*/**

int initialize()

{

int c,e,flag=0;

struct node \*temp2;

printf("Enter the expression: ");

scanf("%dx^%d",&c,&e);

while(c!=0 || e!=0)

{

struct node\* temp=(struct node\*)malloc(sizeof(struct node));

temp->coeff=c;

if(c<0)

{

flag=1;

}

temp->expo=e;

temp->next=NULL;

if(p==NULL)

{

p=temp;

}

else

{

struct node \*temp1=p;

while(temp1->next!=NULL)

{

temp1=temp1->next;

}

temp1->next=temp;

}

scanf("%dx^%d",&c,&e);

}

return flag;

}

**Example:**

**p**

**-10**

**0**

**NULL**

**Representing: 5x3-4x2-10**

**-4**

**2**

**\*next**

**5**

**3**

**\*next**