YUVARAJA'S COLLEGE, MYSORE

(AUTONOMOUS)

A CONSTITUENT COLLEGE OF THE UNIVERSITY OF MYSORE

LABORATORY CERTIFICATE

This is to certify that smt./sri **Jeevan M N**
has satisfactorily completed
the course of **"C and PERL programming"**
in the Department of Physics,
prescribed by the **University of Mysore**
in the year **2016**

Signature of the
batch incharge(s)

Signature of the
Co-ordinator

NAME: Jeevan M N
REG NO. : YPH15307
EXAMINATION CENTER: YUVARAJA's COLLEGE
DATE OF
PRACTICAL EXAM:

TABLE OF CONTENTS

PROGRAM 1: TO CHECK WETHER THE GIVEN NUMBER IS EVEN OR ODD

ALGORITHM:

```
step 1: start
step 2: declare n remainder
step 3: accept the value of n
step 4: define remainder as REMAINDER=N%2
step 5: check whether remainder=0 or not
step 6: if the remainder is 0, print even
step 7: if the remainder is 1, print odd
step 8: stop
```

INPUT:

```
#include <stdio.h>
void main()
{
int n,remainder;
printf("enter the number to check odd or even\n");
scanf("%d",&n);
remainder=n%2;
if (remainder ==0)
printf("number is even\n");
else
printf("number is odd\n");
}
```

OUTPUT:

```
TO COMPILE: gcc-o program program.c
RUN:-$ ./Program
output:
enter the number to check odd or even
5
number is odd
enter the number to check odd or even
6
number is even
```

PROGRAM 2: TO CHECK WHETHER THE GIVEN NUMBER IS PRIME OR NOT

ALGORITHM:

```
step 1: start
step 2: declare variable n,i,flag=1
step 3: accept the value of n
step 4: perform operation n%i=0 from i=2 to i=(n+1)/2
step 5: n%i i=0 assign flag=0
step 6: if flag=0, given number is not a prime, but otherwise it’s a
    prime
step 7: stop
```

INPUT:

```
#include<stdio.h>
void main()
{
int i,n,flag=1;
printf("enter the number to check prime or not\n");
scanf("%d",&n);
for(i=2;i<=(n+1)/2;i++)
{
if(n%i ==0)
{
flag=0;
break;
}
}
if(flag)
printf("it is a prime number\n");
else
printf("it is not a prime number\n");
}
```

OUTPUT:

```
TO COMPILE: gcc-o program program.c
RUN:-$ ./Program
Output:
enter the number to check prime or not
3
it is a prime number
enter the number to check prime or not
4
it is not a prime number
```

PROGRAM 3: TO FIND THE LARGEST AND SMALLEST NUMBER

ALGORITHM:

```
Step 1: start
step 2: declare the variables
step 3: accept the size from the user and store it in variable “size”
step 4: if the user input a null input, go back to step 3
step 5: for loop from count to size
step 6: print the count and accept the input in the variable called
    “input”
step 7: check wether input is greater than largest, implies largest is
    input
step 8: check wether input is smaller than smallest, input is smallest
step 9: stop
```

INPUT:

```
#include<stdio.h>
int main()
{
double largest=-1e37;
double smallest=1e37;
double input;
int size;
int count;
printf("how many numbers you will input\t");
scanf("%d",&size);
if(size==0)
{
return 0;
}
for(count=1;count<=size;count++)
{
printf("%d\t",count);
scanf("%lf",&input);
if(input>largest)
{
largest=input;
}
if(input<smallest)
{
smallest=input;
}
}
printf("the largest number is %0.2lf\n", largest);
printf("the smallest number is %0.2lf\n", smallest);
return 0;
}
```

```
OUTPUT:
TO COMPILE: gcc program program.c
RUN:-$ ./Program
OUTPUT: how many numbers you will input: 2
3
4
smallest number is 3
largest number is 4
```

PROGRAM 4: FIBONACCI NUMBERS

ALGORITHM:

```
Step 1 : Start
Step 2 : declare fib1=0, fib2=1, fib3, n,i
step 3 : accept the input upto which fibonacci numbers are to be printed
    and store it in the variable n
step 4 : check whether the value of n is 1 if so print 0 if its not 1
    print 0 and 1 and for loop from 3 to the
Step 5 : fib3=fib1+fib2
Step 6 : replace fib1 by fib 2
Step 7 : replace fib2 by fib3
Step 8 : print the value of fib3
```

INPUT:

```
#include<stdio.h>
int main()
{
int fib1=0,fib2=1,fib3;
int n,i;
printf("Enter the number upto which fibonacci numbers can be printed
    \n");
scanf("%d",&n);
if(n==1)
printf("The fibonacci number is :%d",fib1);
else
printf("the fibonacci numbers: \n\%d\n\%d\n",fib1,fib2);
for(i=3;i<=n;i++)
{
fib3=fib1+fib2;
fib1=fib2;
fib2=fib3;
printf("%d\n",fib3);
}
return 0;
}
```

OUTPUT:

```
TO COMPILE: gcc-o program program.c
RUN:-$ ./Program
OUTPUT: Enter the number upto which you need fibonacci numbers:
3
The fibonacci numbers are
0,1,1
```

PROGRAM 5: TO FIND THE ROOTS OF A QUADRATIC EQUATION

ALGORITHM:

```
Step1:Start
Step2:Read a,b,c
Step3:If a=0,then
print “a cannot be zero”
Step4:Assign discriminant= b2-4ac
Step5:If discriminant=0,then
      Root1=Root2=-b/2a
Step6:If discriminant>0,then
      Root1=(-b+sqrt(disc))/2a
      Root2=(-b-sqrt(disc))/2a
Step7:If discriminant<0,then
      Root1=(-b+sqrt(-disc))/2a
      Root2=(-b-sqrt(-disc))/2a
Step8:Stop
```

INPUT:

```c
#include<stdio.h>
#include<math.h>
int main(void) {
double a,b,c,discriminant,root1,root2,re,im;
printf("the equation of the form;");
printf("f(x)=a*x*x+b*x+c\n");
printf("a=");
scanf("%lf",&a);
if(a==0){
printf("the value a cannot be zero");
return 1;
}
printf("b=");
scanf("%lf",&b);
printf("c=");
scanf("%lf",&c);
discriminant=(b*b)-(4*a*c);
if(discriminant==0)
{
root1=(-b)/(2*a);
printf("the equation has a single root:");
printf("%0.3lf",root1);
}
else if(discriminant>0){
root1=((-b-(sqrt(discriminant)))/(2*a));
root2=((-b+(sqrt(discriminant)))/(2*a));
printf("the roots are:\n");
printf("%0.3lf\n",root1);
printf("%0.3lf\n",root2);
}
else{
re=((-b)/(2*a));
im=((sqrt(-discriminant))/(2*a));
printf("the roots are:");
printf("%0.3lf-i/%0.3lfr:\n",re,im);
printf("%0.3lf-i/%0.3lf:\n",re,im);
}
```

```
return 0;
}



OUTPUT:
TO COMPILE: gcc -0 filename filenam.c
TO RUN: ./filename
OUTPUT:
The equation of the form f(x)=a*x*x+b*x*c
a=2  b=3   c=6
The roots are: 0.750,
               0.750+i1.561;
```

PROGRAM 6: PASCAL’S TRIANGLE

ALGORITHM:

```
Step 1: Start
Step 2: Declaration of i, long fact, main
Step 3: Enter the number of lines
Step 4: Print i-O, i<line; j=O, j < line.
Step 5: For j=0, j<i, j++.
Step 6: Print i!/(j!*i-j!)
Step 7: Print the number \n
Step 8: Declaration of number
Step 9: While i<number
Step 10: Stop
```

INPUT:

```c
#include<stdio.h>
long fact(int);
int main()
{
int line,i,j;
printf("enter the number of lines :");
scanf("%d",&line);
for(i=0;i<line;i++)
{
for(j=0;j<line-i-1;j++)
printf(" ");
for(j=0;j<=i;j++)
printf("%2ld",fact(i)/(fact(j)*fact(i-j)));
printf("\n");
}
return 0;
}
long fact(int num)
{
long f=1;
int i=1;
while(i<=num)
{
f=f*i;
i++;
}
return f;
}
```

OUTPUT:

```
TO COMPILE: gcc-o program program.c
RUN:-$ ./Program
OUTPUT:How many lines of pascal triangle you want?(0 quits):4
1
1 1
1 2 1
1 3 3 4
```

PROGRAM 7: ADDITION OF TWO MATRICES

ALGORITHM:

```
Step 1: Start
Step 2: Declare the matrices ae b, orders are m&n and i, j are the
        variables.
Step 3: Accept the oredr from the user & store it in the variable
        called m & n.
Step 4: Accept the elements of matrix from the user & store it
        in a[i][j] & print the matrix A.
Step 5: Accept the elements of matrix B from the user & store it
        in B[i][j] & print the matrix B.
Step 6: for loop from i=0 to m & increment the i by 1.
Step 7: for loop from j=0 to n & increment the j by 1.
Step 8: C[i][j]=a[i][j]+B[i][j]that is add two matrices A & B
        and store it in c.
Step 9: Print the resultant matrix c.
Step 10:Stop.
```

INPUT:

```
#include<stdio.h>
int main()
{
int a[10][10],b[10][10],c[10][10],m,n,i,j;
printf("enter the order of the first matrix \n");
scanf("%d%d",&m,&n);
printf("Enter the elements of the matrix A \n");
for(i=0;i<m;i++)
for(j=0;j<n;j++)
scanf("%d",&a[i][j]);
printf("matrix A is\n");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
printf("%d\t",a[i][j]);
}
printf("\n");
}
printf("enter the element of matrix B \n");
for(i=0;i<m;i++)
for(j=0;j<n;j++)
scanf("%d",&b[i][j]);
printf("the matrix B is \n");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
printf("%d\t",b[i][j]);
}
```

```
printf("\n");
}
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
c[i][j]=a[i][j]+b[i][j];
}}
printf("the sum of the matrices is \n");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
printf("%d\t",c[i][j]);
}
printf("\n");
}
return 0;
}
```

OUTPUT:

```
Enter the order of the first matrix 33
Enter the elements of matrix A 123456789
matrix A is
1 2 3
4 5 6
7 8 9

enter the element of matrix B 987654321
The matrix B is
9 8 7
6 5 4
3 2 1

the sum of the matrices is
10 10 10
10 10 10
10 10 10
```

PROGRAM 8: ADDING NUMBERS FROM A DATA FILE

ALGORITHM:

```
step1: start
step2: declare the size of the filename as 1024
step3: declare filename n, sum, long
step4: accept the filename of data store and store it in char filename
    called filename
step5: open the filename in read mod
step6: check whether the file is present or not using while. If it is
    present, store datas in the user defined variable caLLed temp
       sum=sum+temp
step7: increment n to add each number and to determine the average
step8: print sum and average as sum/n
step9: stop
```

INPUT:

```
#include<stdio.h>
const int MAX_FILENAME_SIZE=1024;
int main(void)
{
char filename[MAX_FILENAME_SIZE];
FILE*infile;
long int n=0;
double sum=0,temp;
printf("please input the filename\n");
scanf("%s",filename);
infile=fopen(filename,"r");
while(!feof(infile))
{fscanf(infile,"%lf",&temp);
sum=sum+temp;
n++;
}
printf("sum=%0.3lf\n",sum);
printf("average=%0.3lf\n",sum/n);
return 0;
}
```

OUTPUT:

```
TO COMPILE: gcc -o sum sum.c TO RUN:./sum
OUTPUT:
please input the filename
1
2
3
4
5
6
sum.dat
sum=21.000 average=3.500
```

PROGRAM 9: FITTING DATA IN A LINE

ALGORITHM:

```
step 1: start
step 2: declare the size of the filename as 1024
step 3: declare filename n,sum
step 4: Accept the filename of data store and store it in char filename
    called filename
step 5: open the filename in read
step 6: check whether the file is present or not using while. If it is
    present, store datas in the user defined variable
step 7: increment n to add each  to find the least square to fit in a
    file
step 8: print the least square fit
step 9:stop
```

INPUT:

```
include<stdio.h>
const int FILENAME_MAX_SIZE=1024;
int main(void)
{
char filename[FILENAME_MAX_SIZE];
FILE*infile;
int n=0;
double tempx, tempy;
double sum_xy=0,sum_x2=0,sum_x=0,sum_y=0;
double m,c;
printf("input the filename:\n");
scanf("%s",filename);
infile=fopen(filename,"r");
while(!feof(infile))
{
fscanf(infile,"%lf%lf",&tempx,&tempy);
sum_x+=tempx;
sum_y+=tempy;
sum_xy+=tempx*tempy;
sum_x2+=tempx*tempx;
n++;
}
fclose(infile);
m=(n*sum_xy-sum_x*sum_y)/(n*sum_x2-(sum_x*sum_x));
c=(sum_y*sum_x2-sum_x*sum_xy)/(n*sum_x2-(sum_x*sum_x));
printf("number read %d\n",n);
printf("the least square fit is given by the line");
printf("y=%0.10lfx+%0.10lf\n",m,c);
return 0;
}
```

OUTPUT:

```
TO COMPILE: gcc -o 1 1.c
TO RUN: ./1
```

```
OUTPUT:
input the filename: gangoo.dat
number read 13
the least square fit is given by the
liney=0.7573620391x+13.6786686464
```

PROGRAM 10: PROJECTILE MOTION

ALGORITHM:

```
Step 1: Start
Step 2: Declare velocity and angle
Step 3: Accept the velocity of the projectile is called velocity
Step 4: Check whether velocity is equal to zero or not, if so come out
    of the loop
Step 5: Accept the angle of a projectile to the variable called angle
Step 6: Check whether angle<O or angle>90, if so come out of the loop
Step 7: Declare a function to find the range, height, time and draw the
    trajectory with the user defined variable velocity &  angle with the
    data type of double.
Step 8: Calculate the range using the formula
        Range=V2sin2B/g
Step 9: Calculate the height using the formula
        Height=v2sin2B/2g
Step 10: Calculate the time using the formula Time=2Vsin9/g
step 11: Calculate the trajectory using the formula
         and return it to  print
Step 12: Declare 'R' as range velocity, angle) , counter=O cincrement=R/
    100
Step 13: Declare a file for the purpose of plotting
Step 14: Open the datas which we have entered earlier in the right mode
Step 15: for loop from counter=O , counter+=c incr
Step 16: Print the counter and trajectory to the file
Step 17: Define two variables  and b=tane
Step 18: Close the plot file
Step 19: Open the graph in gnuplot in the right mode
Step 20: Mention the path of the gnuplot, plot t!/usr/bin/gnuplot, plot
    x label as a displacement (m) and plot y label as a height (m)
Step 21: Refresh the graph using unset key
Step 22: Plot f(x)
```

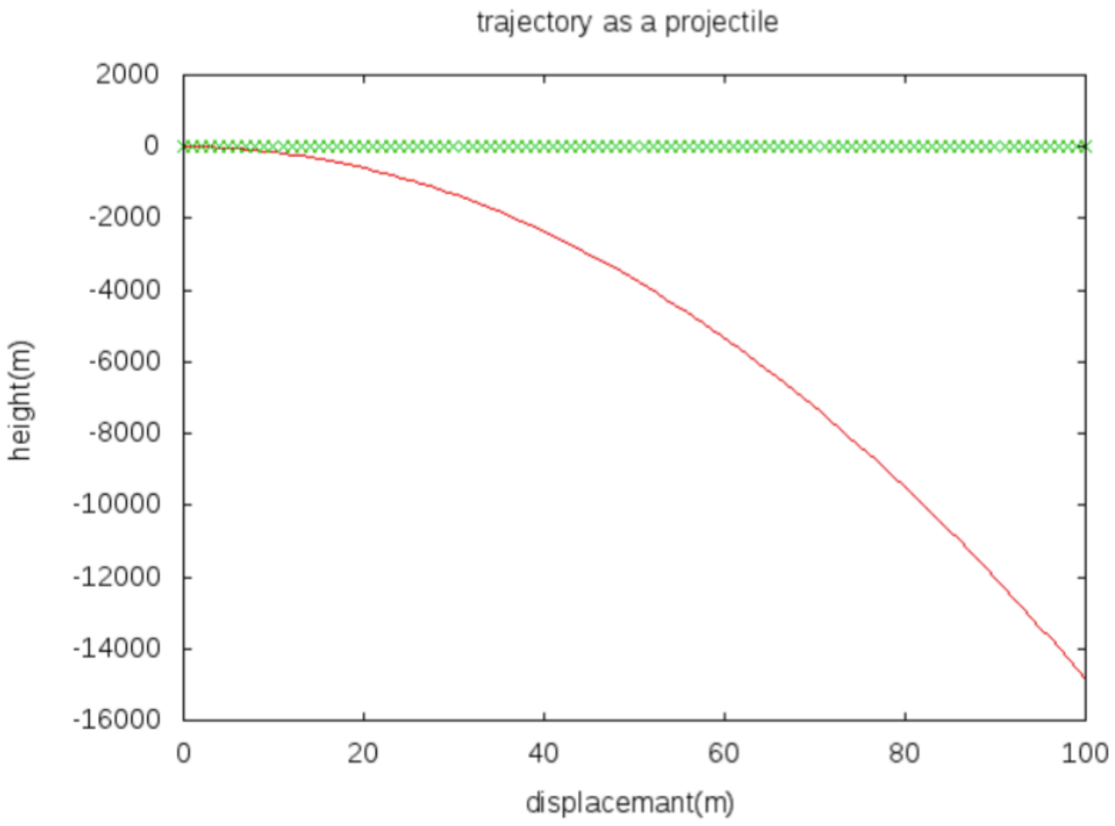
INPUT:

```
#include<stdio.h>
#include<math.h>
const double pi=3.14,g=9.8;
double range(double velocity,double angle)
{
return velocity*velocity*sin(angle*pi/180)/g;
}
double height(double velocity,double angle);
{
return velocity*velocity*sin(angle*pi/180)*sin(angle*pi/180)/(2*g);
}
double time(double velocity,double angle)
```

```
{
return 2*velocity*sin(angle*pi/180)/g;
}
double trajectory(double velocity,double angle,double x)
{
return(-g*x*x)/(2*velocity*cos(angle*pi/180)*cos(angle*pi/180));
}
void draw_trajectory(double velocity,double angle)
{
double R=range(velocity,angle),counter=0,c_incr=R/100;
FILE *plot;
plot=fopen("data.dat","w");
for(counter=0;counter<=R;counter+=c_incr)
{
fprintf(plot,"%0.5lf %0.5lf
    \n",counter,trajectory(velocity,angle,counter));
}
double a=-g/
(2*velocity*velocity*cos(angle*pi/180)*cos(angle*pi/
    180)),b=tan(angle*pi/180);
fclose(plot);
plot=fopen("graph.gpt","w");
fprintf(plot,"#!|us|bin|gnuplot\n\n");
fprintf(plot,"set title'trajectory as a projectile'\n");
fprintf(plot,"set xlabel'dispalcement(m)'\n");
fprintf(plot,"set ylabel'height(m)'\n");
fprintf(plot,"unset key\n");
fprintf(plot,"f(x)=%lf*x*x+%lf*x\n",a,b);
fprintf(plot,"set terminal png\n");
fprintf(plot,"set output 'graph.png'\n");
fprintf(plot,"plot f(x),'./data.dat'\n");
fclose(plot);
system("gnuplot -p graph.gpt");
}
int main(void)
{
double velocity,angle;
printf("if the velocity of the projectile(0quit)");
scanf("%lf",&velocity);
if(velocity==0)
return 0;
printf("Input angle");
scanf("%lf",&angle);
if((angle<=0.00)||(angle>90))
return 0;
printf("The range of a projectile is %0.3lfm/n",range(velocity,angle));
printf("The height of a projectile is %0.3lfm/
    n",height(velocity,angle));
printf("The time of flight projectile %0.3lfs/n",time(velocity,angle));
draw_trajectory(velocity,angle);
return 0;
}
```

OUTPUT:

GRAPHICAL REPRESENTATION OF TRAJECTORY AS A PROJECTILE.



PERL PROGRAM 1: TO SEARCH FOR A PATTERN IN A STRING

ALGORITHM:

```
Step 1: start
step 2: input the string
step 3: declare $PATTERN
step 4: search for pattern in a string
step 5: print the found string
step 6: stop
```

INPUT:

```
#!/usr/bin/perl
print"input a sentence:\n";
$sentence=<STDIN>;
print "input a searching string :\n";
chomp($pattern=<STDIN>);
if($sentence=~/$pattern/)
{
$pos=index($sentence,$pattern);
print"found'$pattern'at $pos\n";
}
else
{
print"$pattern not found\n";
}
```

OUTPUT:

```
$ perl ./1p.pl
input a sentence:
Rakesh M.Sc Phd
input a searching string:
j
j not found
```

PERL PROGRAM 2: SORTING WORDS IN A STRING

ALGORITHM:

```
Step 1: start
step 2: input the string
step 3: declare n
step 4: sort the string using standard keyWord called @lines
step 5: print the sorted string
step 6: stop
```

INPUT:

```
#!/user/bin/perl
print"input a string \n";
$string=<STDIN>;
@lines=split(/\s+/,$string);
print join(" ",sort(@lines));
print"\n";
```

OUTPUT:

```
$ perl ./1p.pl
input a string: one two three
three one two
```

PERL PROGRAM 3: TO FIND WETHER A NUMBER IS PRIME

ALGORITHM:

```
step 1: start
step 2: input the number n
step 3: if the number is zero it is   whole number, go to stop
step 4: declare i=2
step 5: check while ($i<=($number+I/2) )
step 6  if the remainder is O
step 7: the number is not prime
step 8: if the remainder is 1
step 9: the number is prime
```

INPUT:

```perl
#!/user/bin/perl
print "input the number:\n";
$number=<STDIN>;
if($number==0)
{
print"Zero is a whole number\n";
exit;
}
$i=2;
while($i<=($number+1)/2)
{
if($number%$i==0)
{
print"the number is not prime\n";
exit;
}
else
{
$i+=1;
}
}
print"the number is prime\n";
```

OUTPUT:

```
$ perl 3.pl
input the number :
2
 the number is prime
$ perl 3.pl
input the number :
4
the number is not prime
```

PERL PROGRAM 4: TO FIND THE NUMBER OF CHARACTERS AND WORDS IN A STRING

ALGORITHM:

```
step 1: Start declare lines—O characters—O
step 2: declare lines=0, words=0,characters=0
step 3: input the file name
step 4: check whether the file is present or not
step 5: if the file is not present print the die error "unable to open $
    filename "
step 6: if the file is present
step 7: increment the line, store the length in characters, find the
    number of words by using standard keywords
step 8: print the filename contains the words, lines , characters
step 9: stop
```

INPUT:

```
#!/usr/bin/perl
$lines=0;
$words=0;
$chars=0;
print"input the filename:\n";
$filename=<STDIN>;
chomp($filename);
if(!open$infile,"<",$filename)
{
die"error:enable to open$filename:$!\n";
}
while(<$infile>)
{
$lines++;
$chars+=length($_);
$words+=scalar(split(/\s+/,$_));
}
print"the file $filename contains $lines lines $words words & $chars
    characters\n";
```

OUTPUT:

```
perl ./4p.pl
input the filename:  raki.txt
perl program to counting the number of characters,words and the lines in
    a file.
the file raki.txt contains 1 lines 14 words& 81 characters
```

PERL PROGRAM 5: TO FIND THE ROOTS OF A QUADRATIC EQUATION

ALGORITHM:

```
Step1 : Start
Step2 : Read a, b, C
Step3 : If a-0, then print •a cannot be zero•
Step4 : Assign discriminant- -4ac
Step5 : If discriminant-o, then Root1-Root2--b/2a
Step6 : If discriminant then
        Root1- ( -b+sqrt (disc) ) /2a
        Root2- -b-sqrt (disc) ) /2a
Step7 : If discriminant<0, then
        Root1- (-b+sqrt ( -disc) ) /2a Root2- (-b-sqrt ( -disc) ) /2a
Step8 : Stop
```

INPUT:

```perl
#!\usr\bin\perl
print"input the value of a: ";
$a=<STDIN>;
print"\ninput the value of b: ";
$b=<STDIN>;
print"\ninput the value of c: ";
$c=<STDIN>;
if($a==0)
{
   print"\nthe value a cannot be zero";
   exit;
}
$discriminant=$b*$b-(4*$a*$c);
if($discriminant>0)
{
   $root1=(-$b-sqrt($discriminant))/(2*$a);
   $root2=(-$b+sqrt($discriminant))/(2*$a);
   print"the roots are :\n";
   printf"%0.3lf\n",$root1;
   printf"%0.3lf\n",$root2;
}
else
{
   $re=-$b/(2*$a);
   $im=sqrt(-$discriminant)/(2*$a);
   print"the roots are:\n";
   printf"%0.3lf - %0.3lf i \n",$re,$im;
   printf"%0.3lf + %0.3lf i \n",$re,$im;
}
```

```
OUTPUT:
$ perl ./5p.pl
input the value of a: 3
input the value of b: 2
input the value of c: 1
the roots are:
-0.333-0.471 i
-0.333+0.471 i
```

PERL PROGRAM 6: TO CHECK THE LINEAR SQUARES FITTING TO DATA IN FILE

ALGORITHM:

```
Step1: Start
Step2: Declare the variables sumx,sumy,sumx2,sumxy,m=0,n=0,c=0
Step3: Input a filename
Step4: Check whether the file is present or not if the file is not
present print error unable to open $filename.
Step5: If the file is present.
Step6: Store x&y values in the variables for tempx & tempy.
Step7: Define the variables sumx and store the tempx value in it.
Step8: Define the variables sumy and store the tempy value in it.
Step9: Define the variables sumx2 and store the tempx & tempx
value in it
Step10: Define the variables sumxy and store the tempx & tempy
value in it.
Step11: Increment n value
Step12: m= n∗∑xy−∑x∗∑y n∗∑x2−(∑x)2
Step13: c= ∑y∗∑x2−∑x∗∑xy
n∗∑x2−(∑x)2
Step14: Print the m & c value
step15 :stop
```

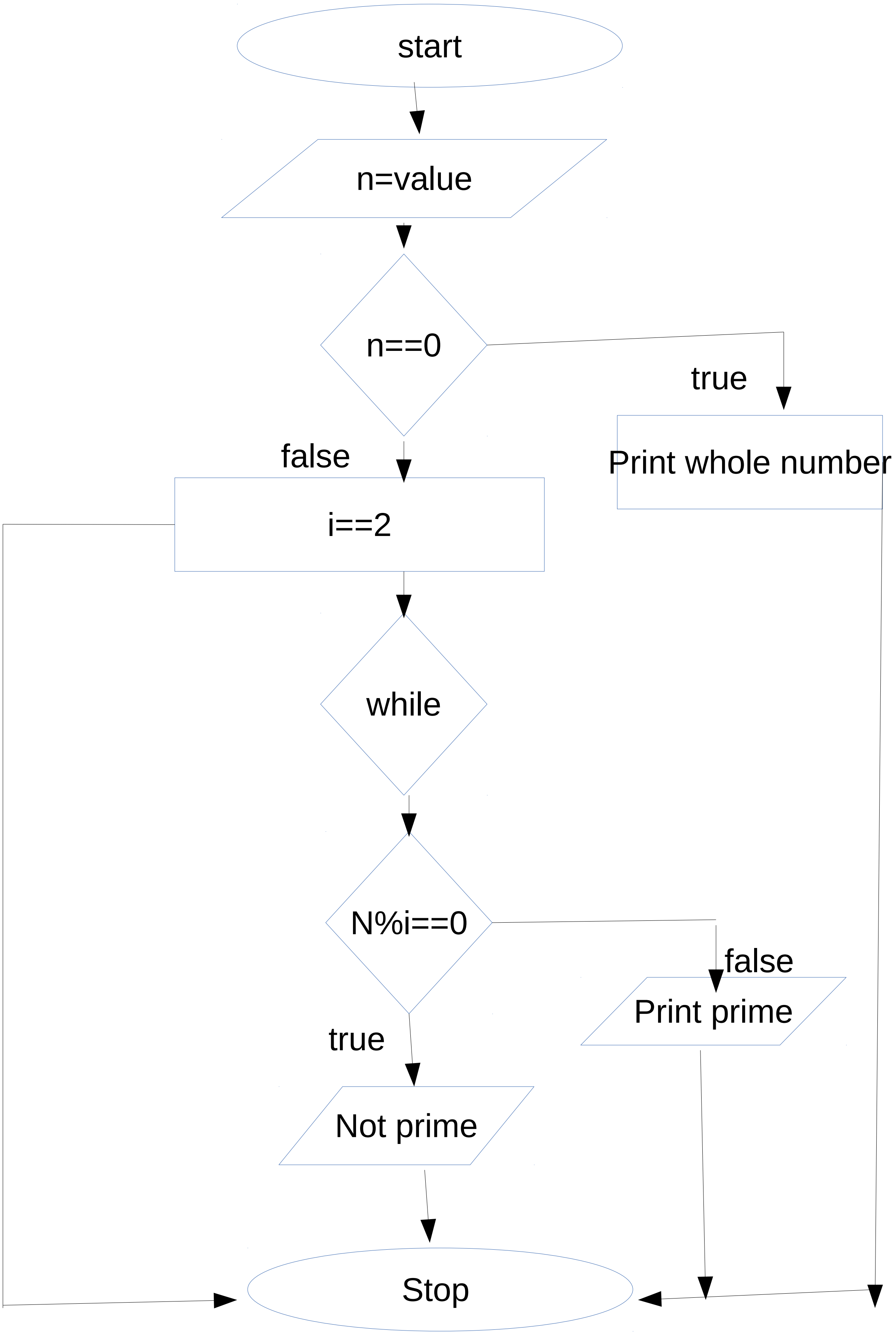
INPUT:

```
#!usr/bin/perl
$sum_x=0,$sum_y=0,$sum_x2=0,$sum_xy=0;
$m=0,$c=0;
print"input a filename $:\n";
$filename=<STDIN>;
chomp($filename);
if(!open $infile,"<", $filename)
{
die"error:unable to open filename:$ !\n";
}
while(<$infile>)
{
chomp($_);
my @toks=split(/[t ]+/,$_);
my $tempx = $toks[0];
my $tempy = $toks[1];
$sum_x+=$tempx;
$sum_y+=$tempy;
$sum_x2+=$tempx*$tempx;
$sum_xy+=$tempx*$tempy;
$n++;
}
close($infile);
print "\n",$sum_x,"\n",$sum_y,"\n",$sum_x2,"\n",$sum_xy,"\n";
$m=($n*$sum_xy-$sum_x*$sum_y)/($n*$sum_x2-$sum_x*$sum_x);
```

```
$c=($sum_y*$sum_x2-$sum_x*$sum_xy)/($n*$sum_x2-$sum_x*$sum_x);
print"n numbers read \n",$n,"\n";
print "the least squares fit is given by \n"
print "y=mx+c \n";
print "slope= \n", $m, "\n";
print "intercept= \n", $c, "\n";
```

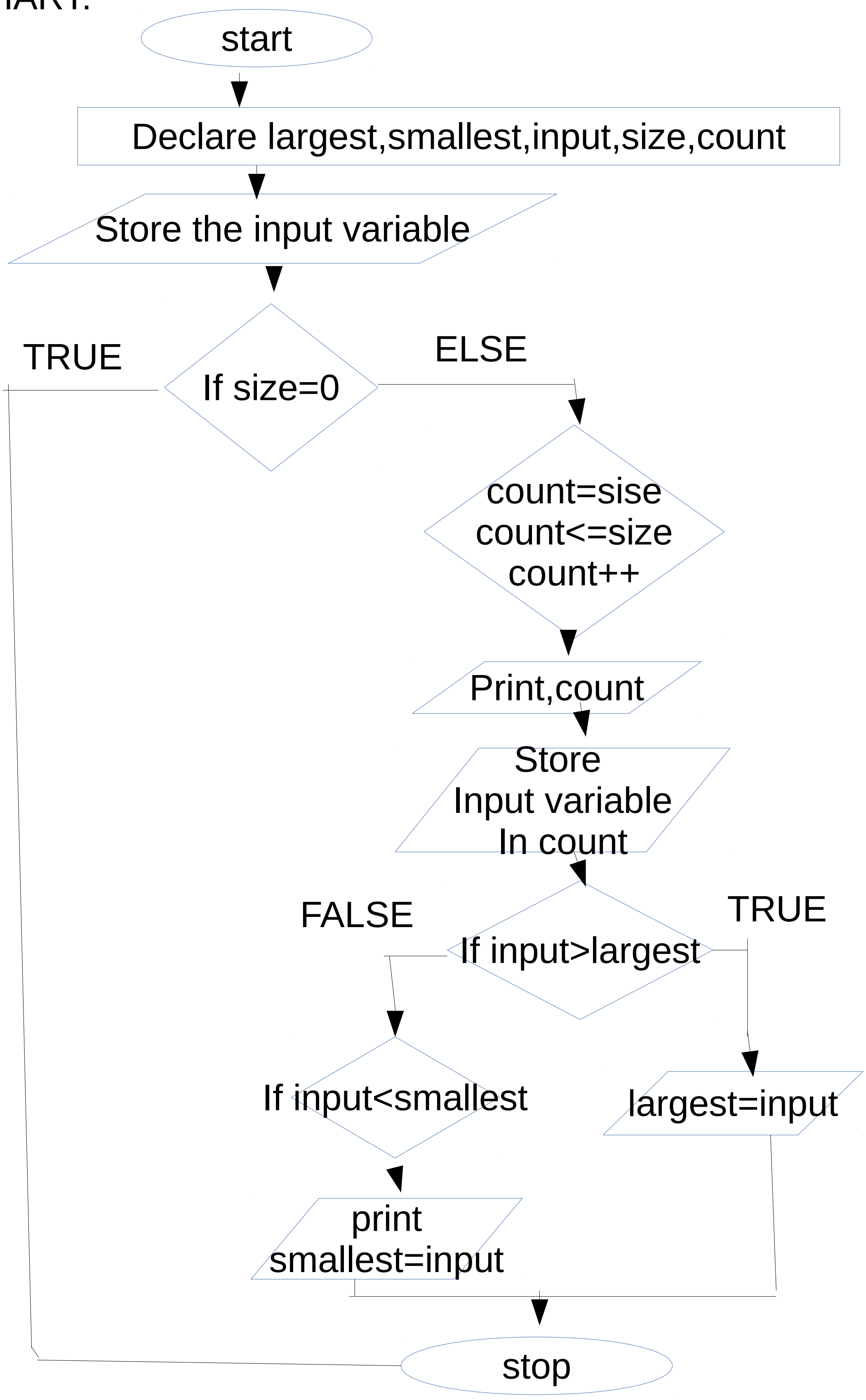
```
OUTPUT:
input the filename: abc.dat
numbers read 12
The least square fit is given by the line
y=0.798
```

PERL PROGRAM: TO FIND WETHER A NUMBER IS PRIME



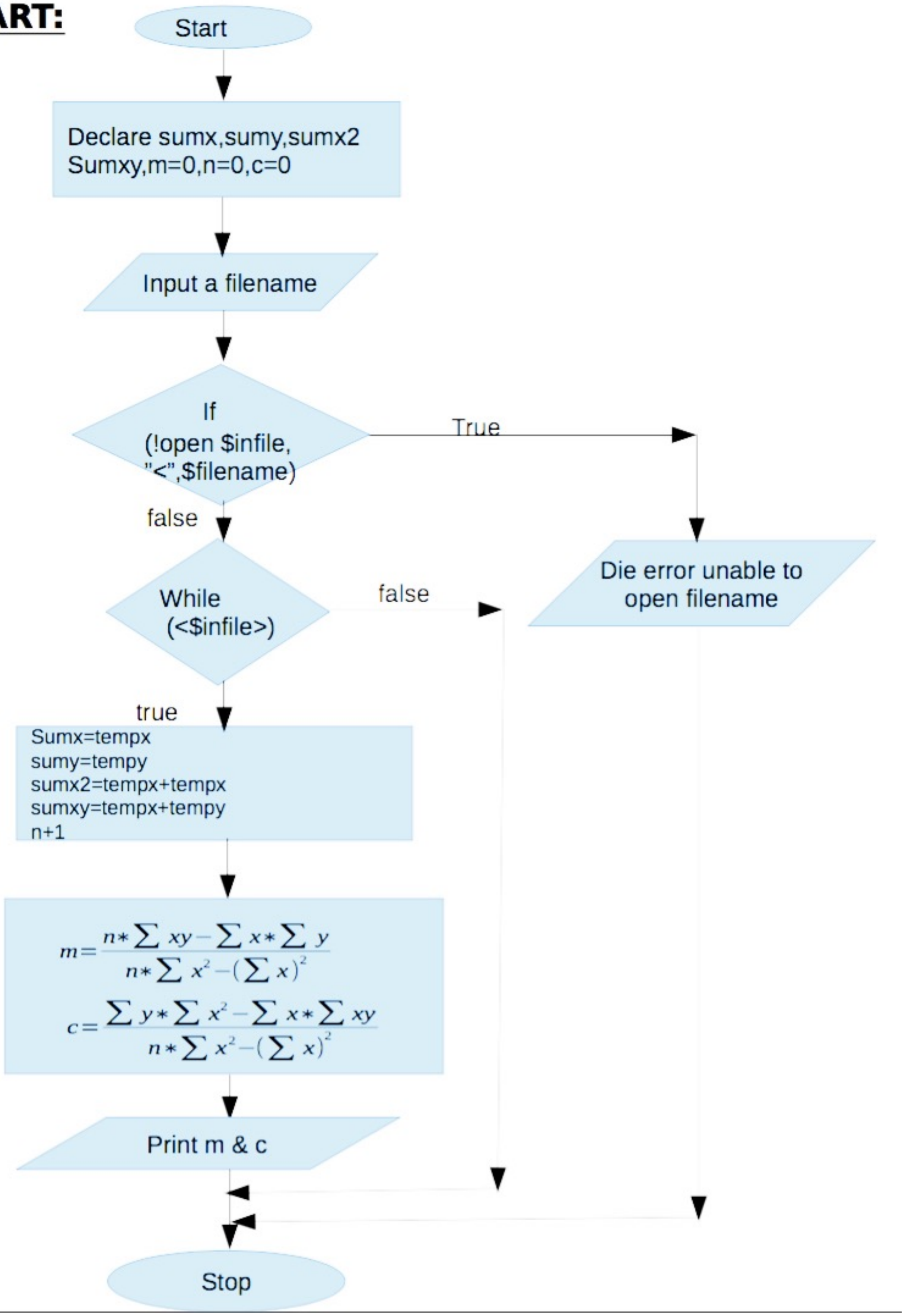
PROGRAM TO FIND LARGEST AND SMALLEST NUMBER IN THE INPUT SET

FLOWCHART:



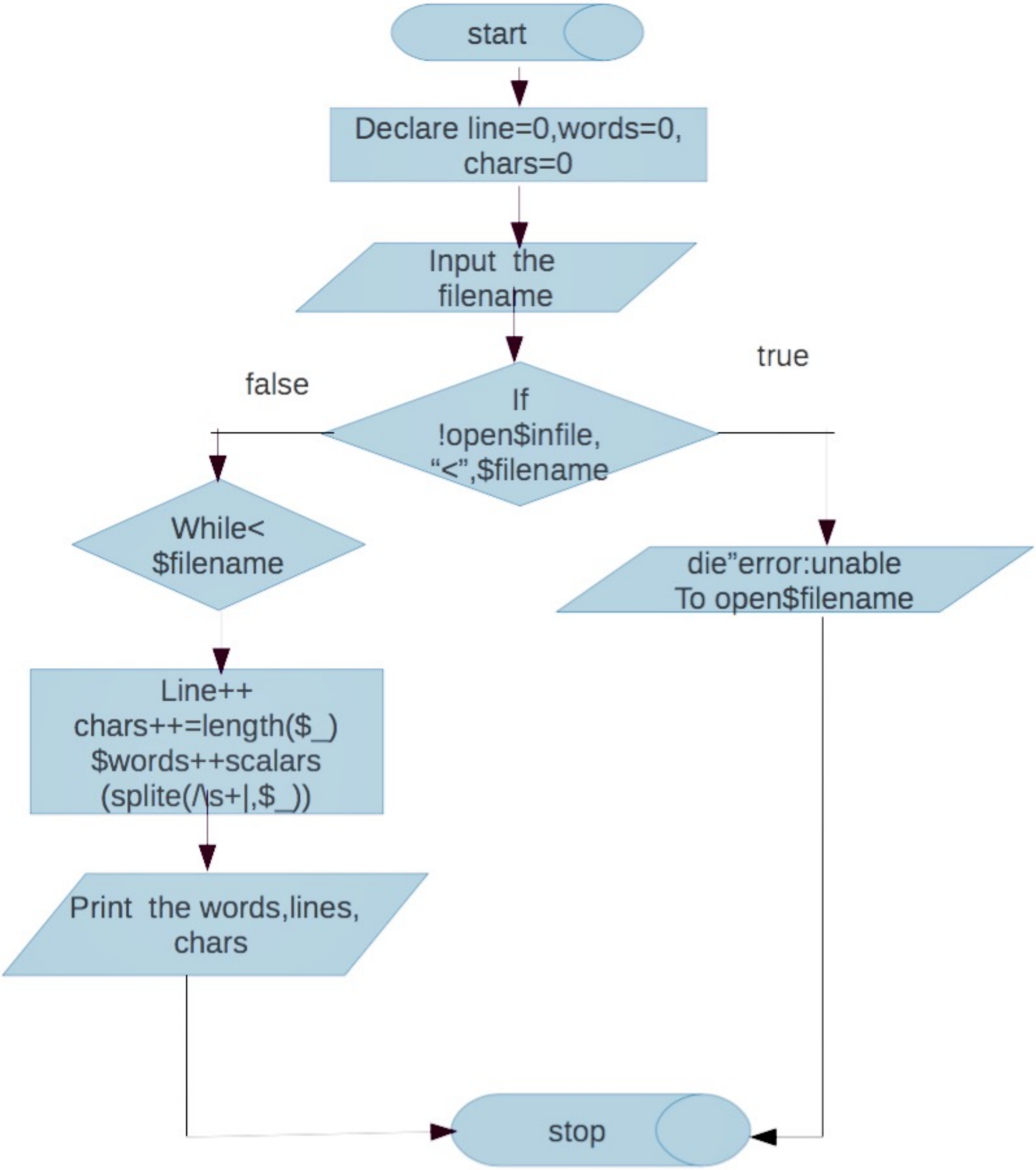
PERL PROGRAM TO CHECK THE LINEAR SQUARES FITTING TO DATA IN FILE

FLOWCHART:

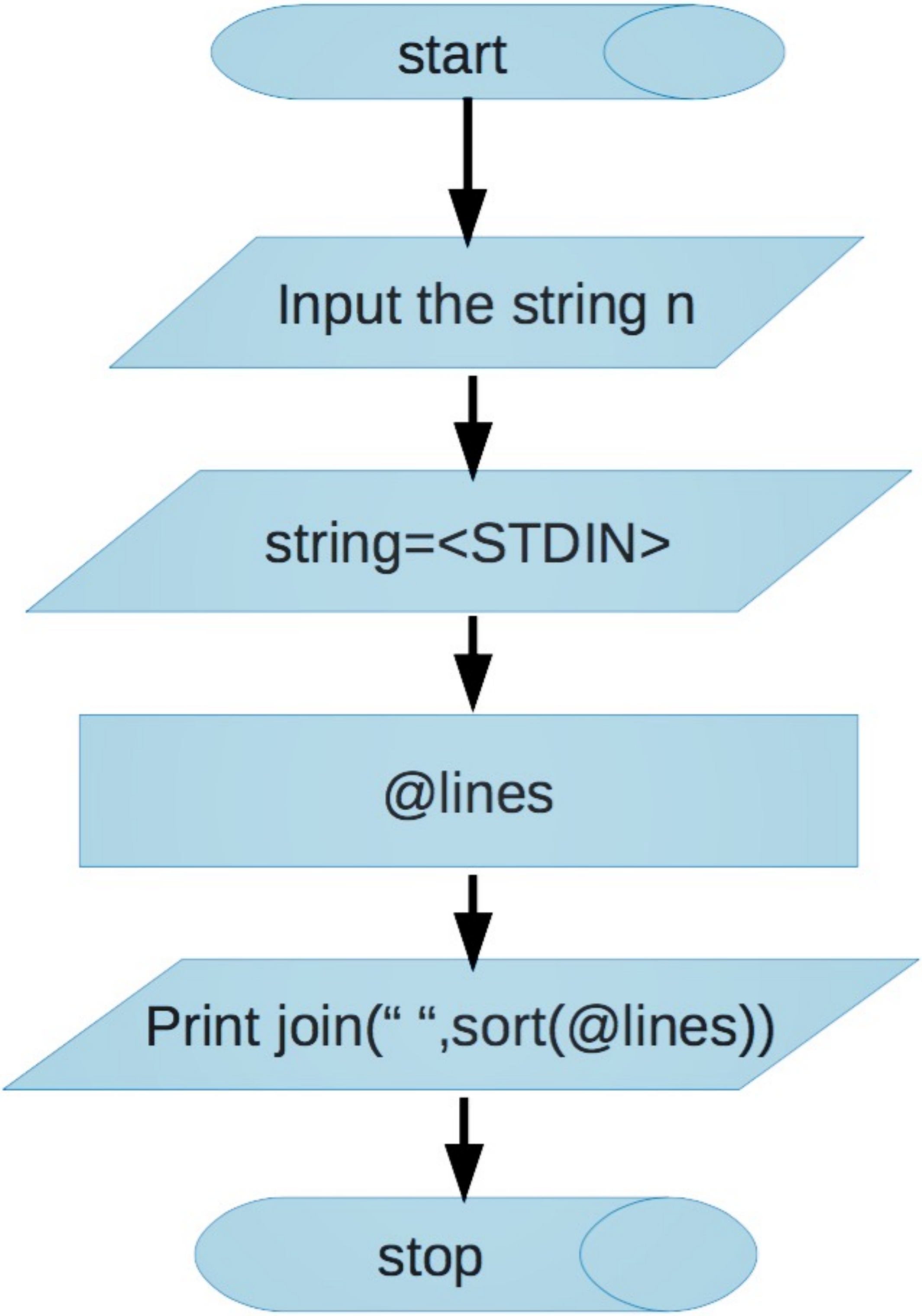


PERL PROGRAM: NO. OF WORDS IN A FILE

FLOWCHART:

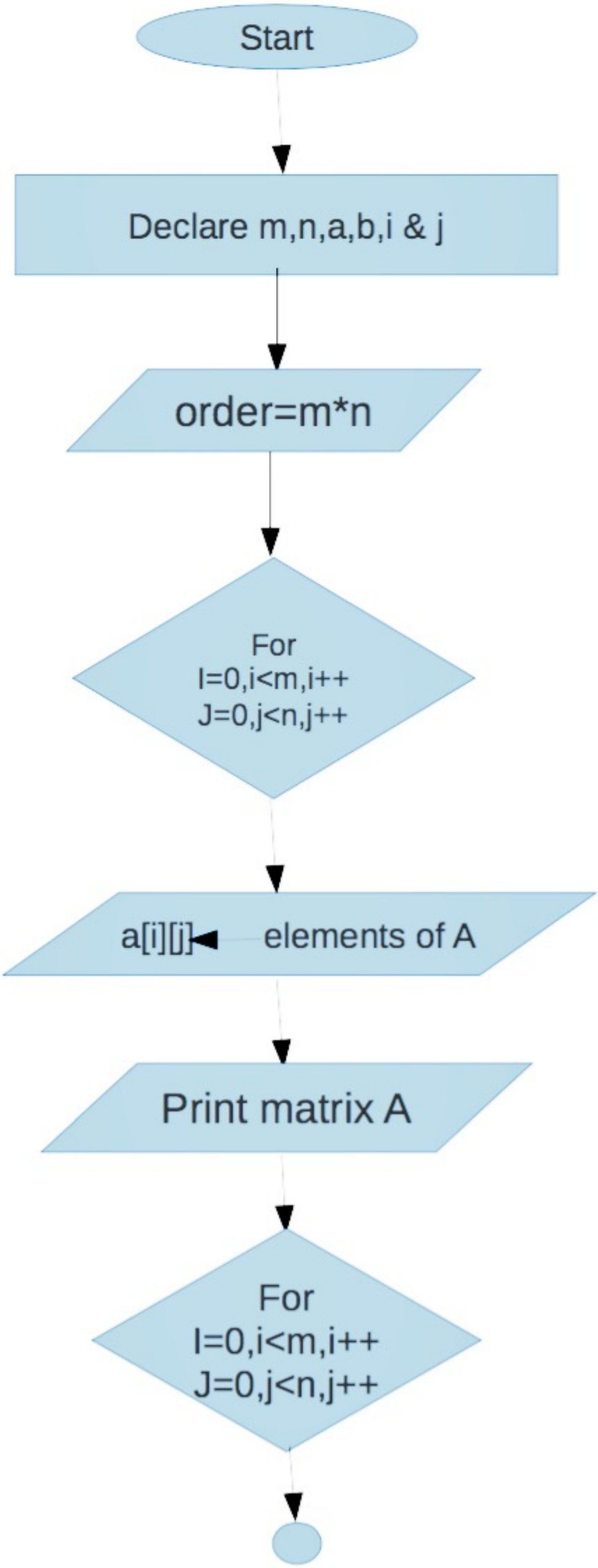

start
Declare line=0,words=0, chars=0
Input the filename
If !open$infile, “<”,$filename
false
true
While< $filename
die”error:unable To open$filename
Line++ chars++=length($_) $words++scalars (splite(/\s+|,$_))
Print the words,lines, chars
stop

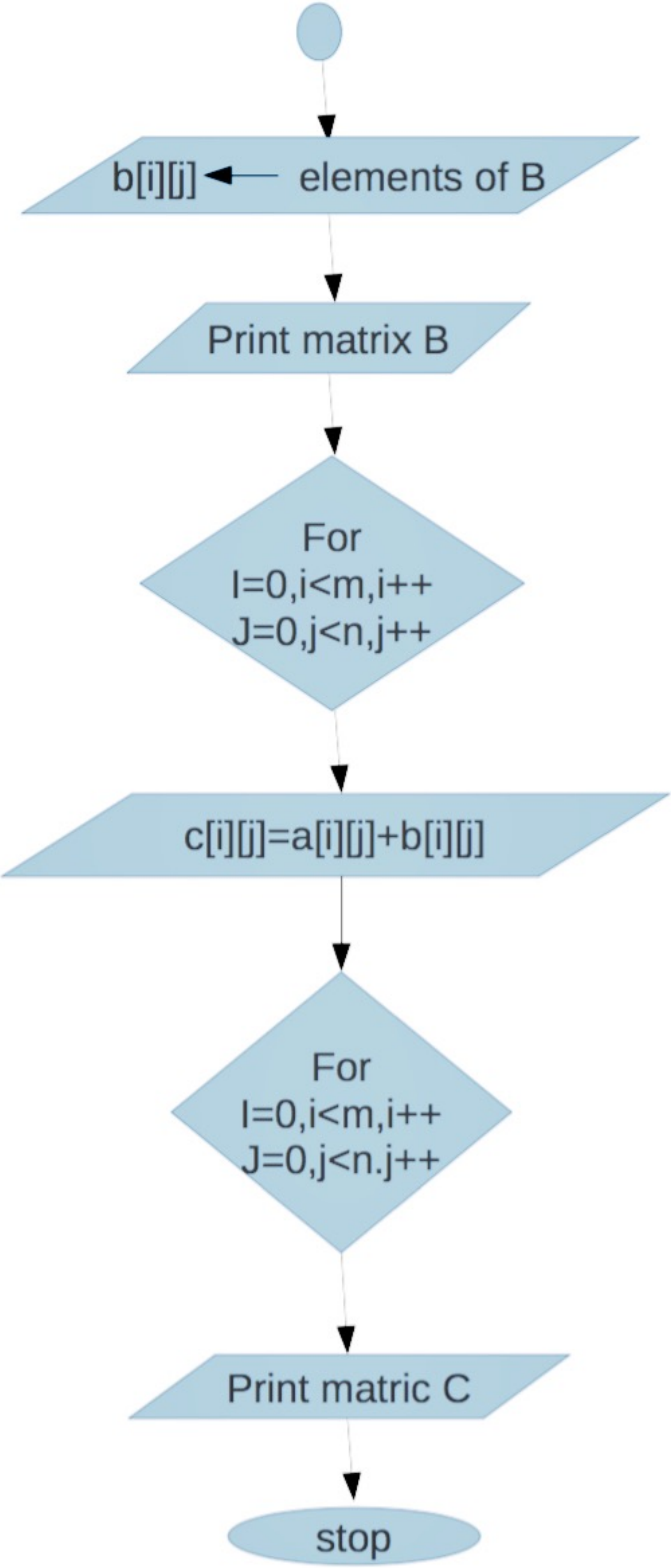
PERL PROGRAM: A PROGRAM TO SORT STRINGS



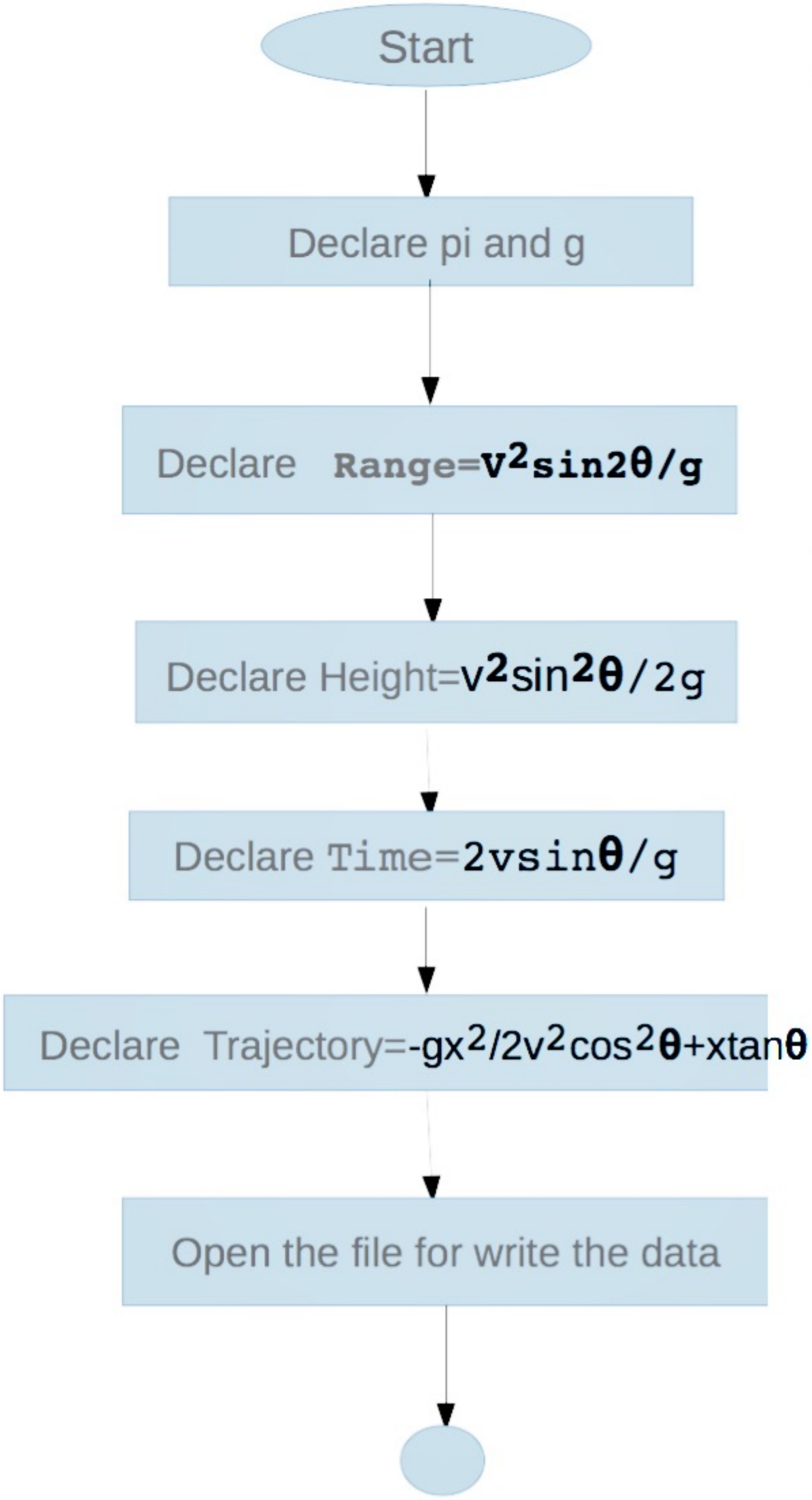
C PROGRAMME TO ADD TWO M X N MATRICES

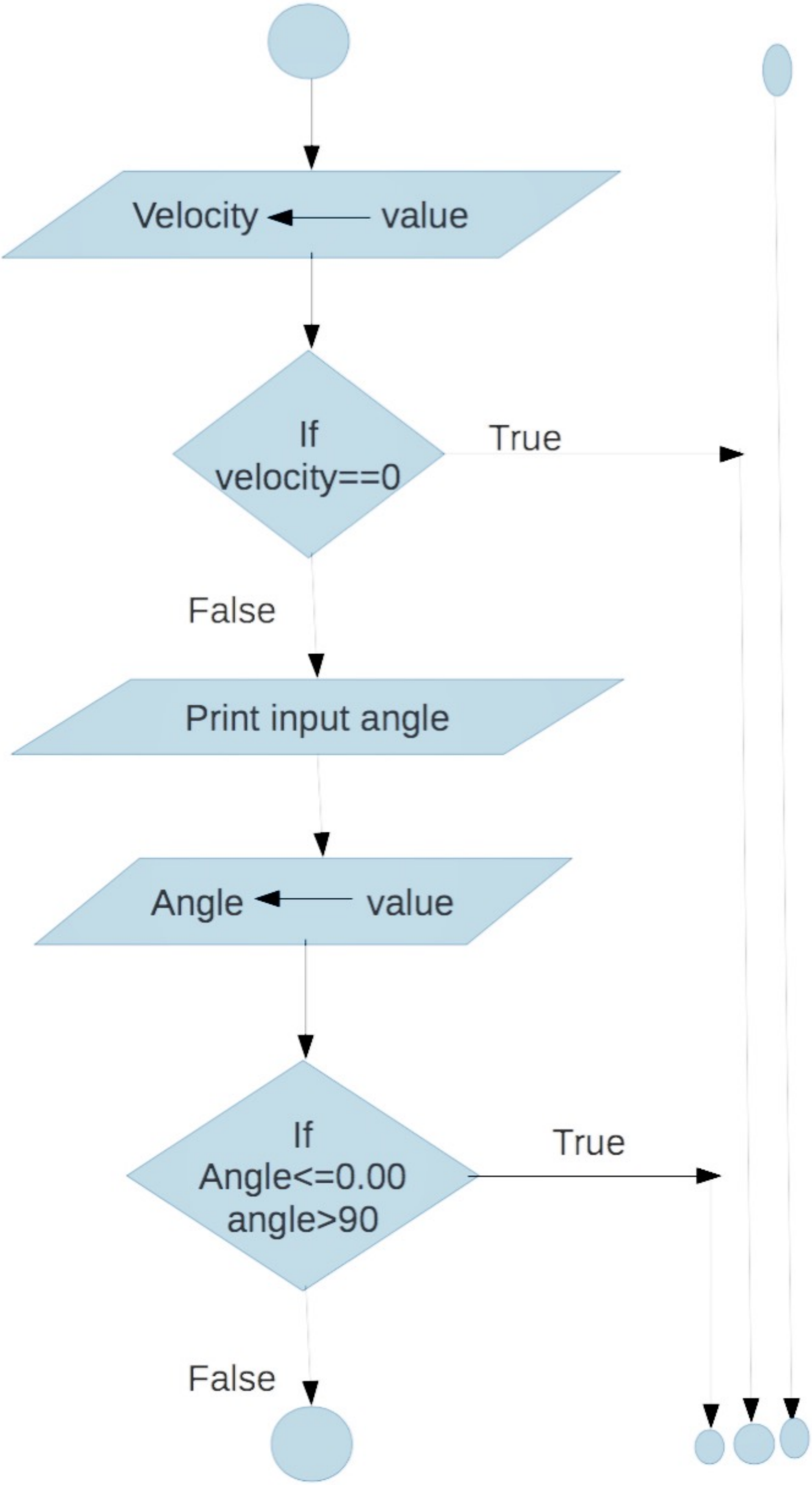
FLOW CHART:

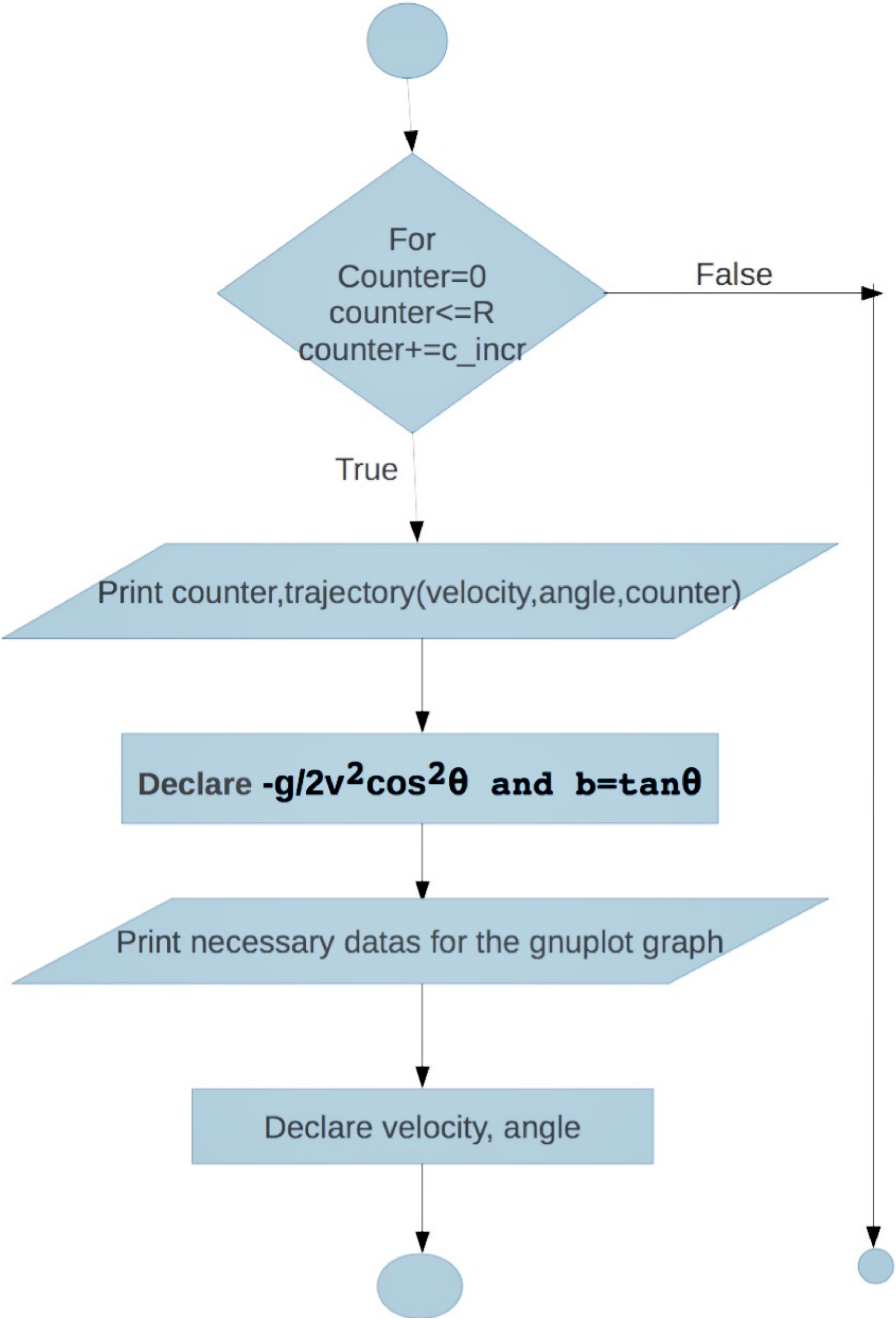



b[i][j]
elements of B
Print matrix B
For
I=0,i<m,i++
J=0,j<n,j++
c[i][j]=a[i][j]+b[i][j]
For
I=0,i<m,i++
J=0,j<n.j++
Print matric C
stop

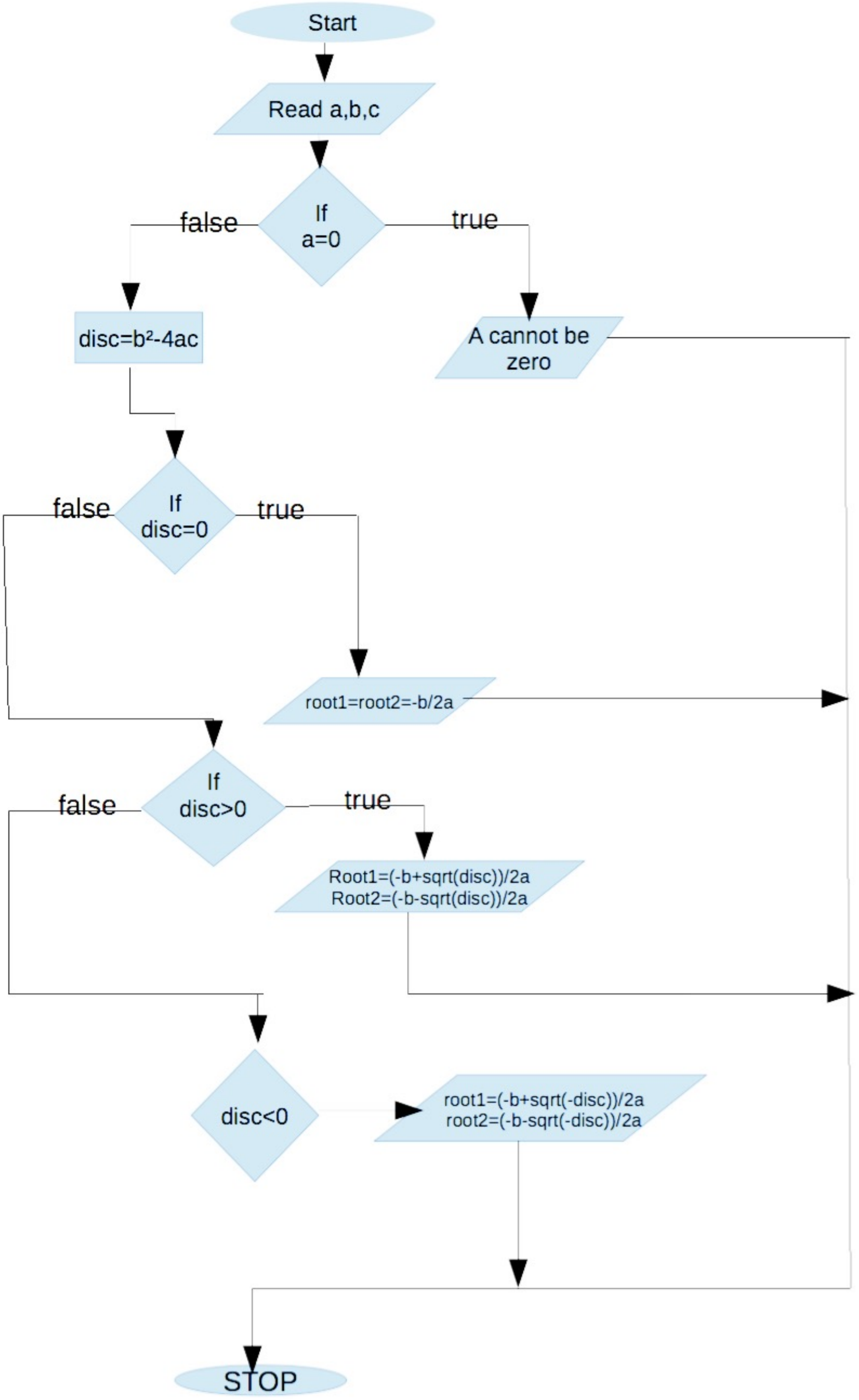
C: PROJECTILE MOTION


Start
Declare pi and g
Declare Range=v²sin2θ/g
Declare Height=v²sin²θ/2g
Declare Time=2vsinθ/g
Declare Trajectory=-gx²/2v²cos²θ+xtanθ
Open the file for write the data


Velocity ← value
If velocity==0
True
False
Print input angle
Angle ← value
If Angle<=0.00 angle>90
True
False

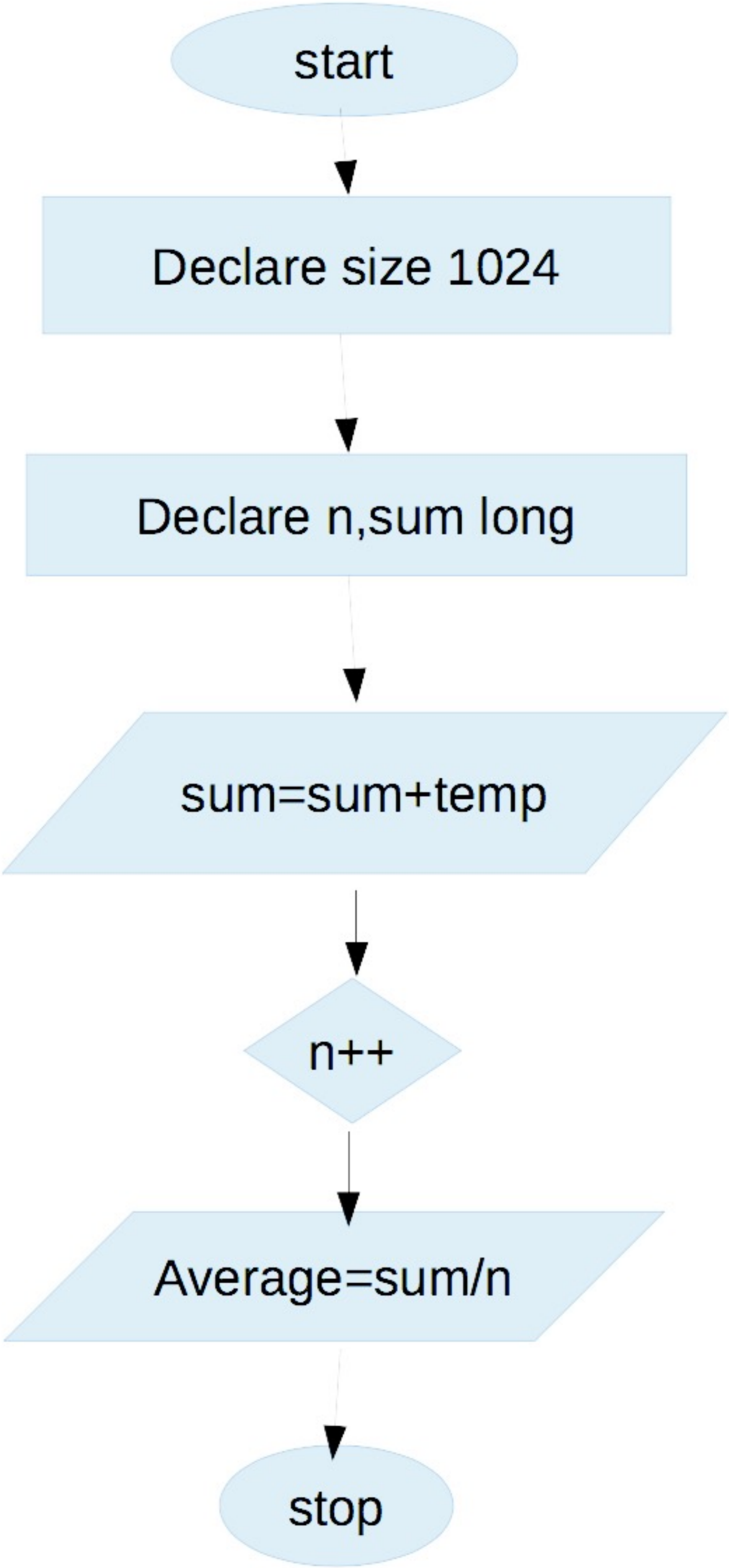

For
Counter=0
counter<=R
counter+=c_incr
False
True
Print counter,trajectory(velocity,angle,counter)
Declare -g/2v²cos²θ and b=tanθ
Print necessary datas for the gnuplot graph
Declare velocity, angle

C PROGRAM: TO FIND THE ROOTS OF A QUADRATIC EQUATION



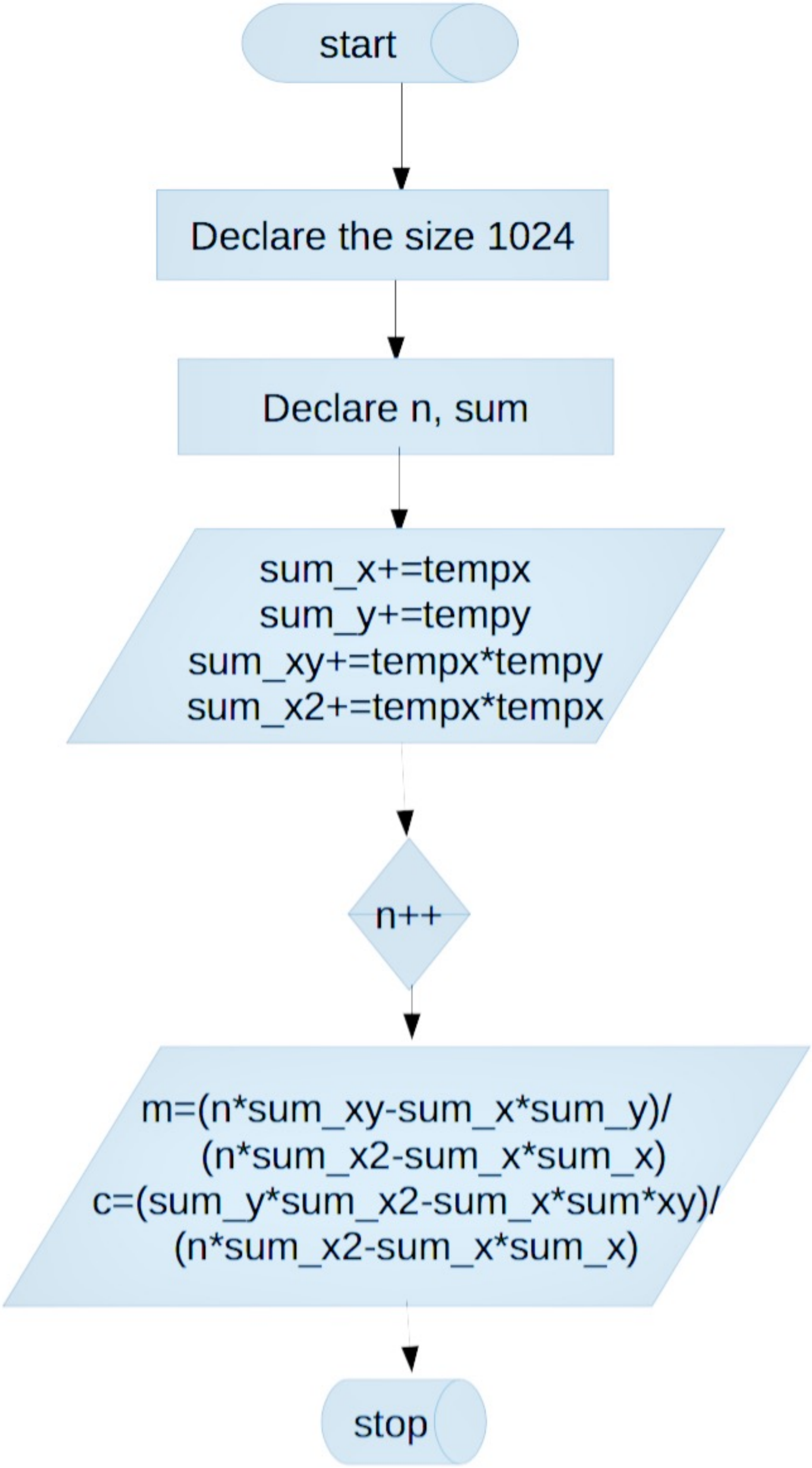
FLOWCHART TO FIND SUM AND AVERAGE OF THE GIVEN DATA

FLOWCHART:



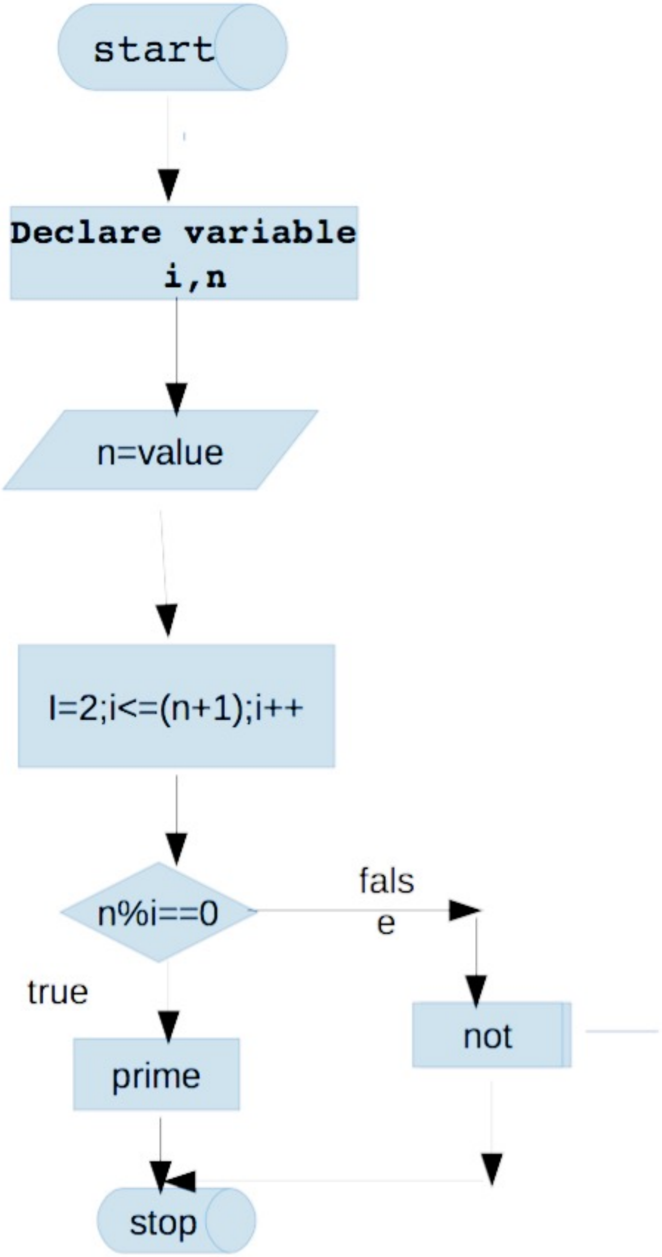
C PROGRAM FLOWCHART TO LINEAR LEAST-SQUARES FITTING TO DATA IN A FILE

FLOWCHART:



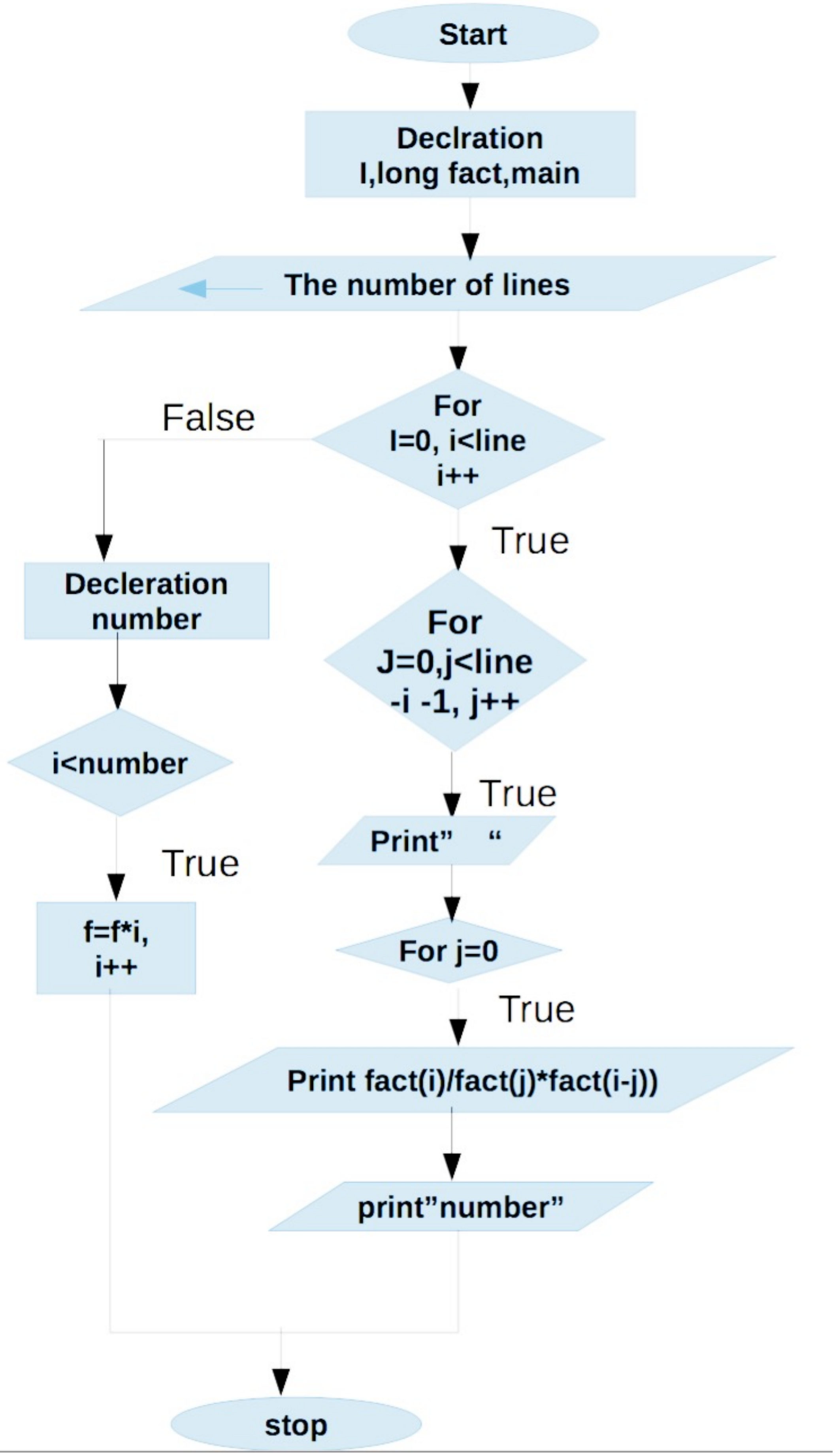
C PROGRAM TO CHECK THE WHETHER THE GIVEN NUMBER IS PRIME OR NOT

FLOWCHART:



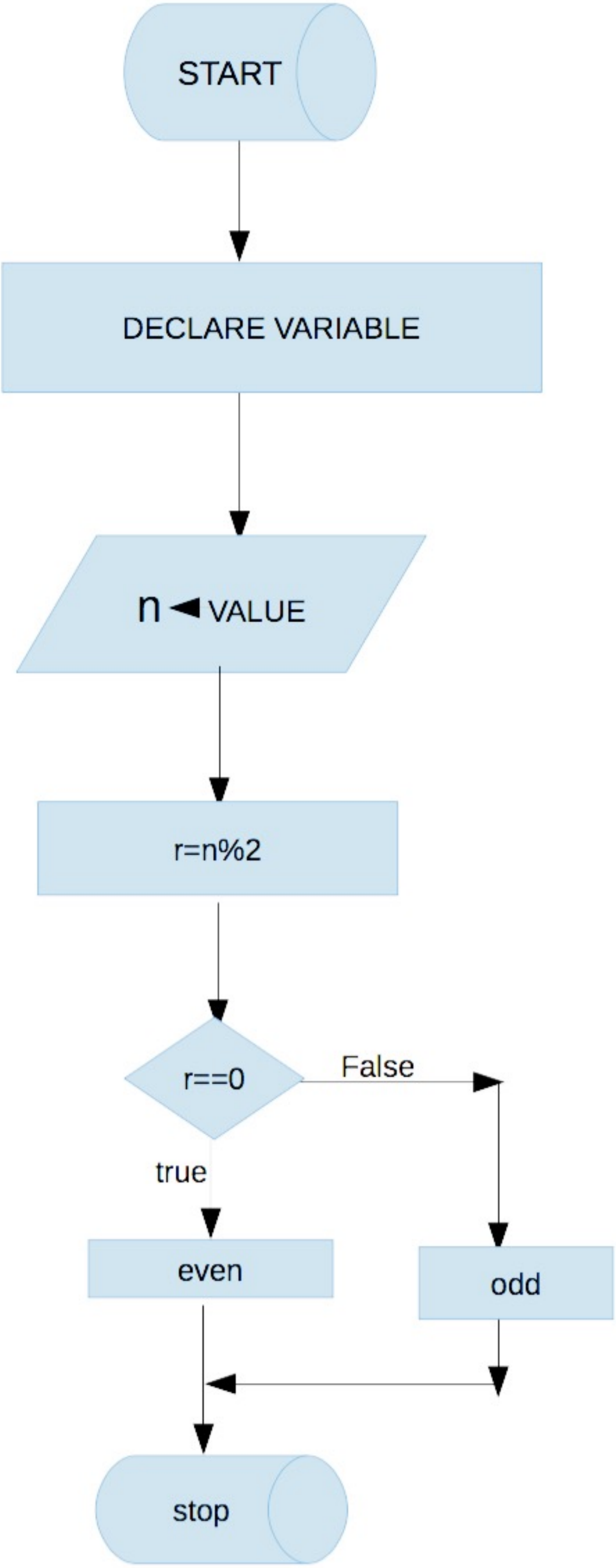
C PROGRAM TO GENERATING PASCAL TRIANGLE

FLOW CHART:



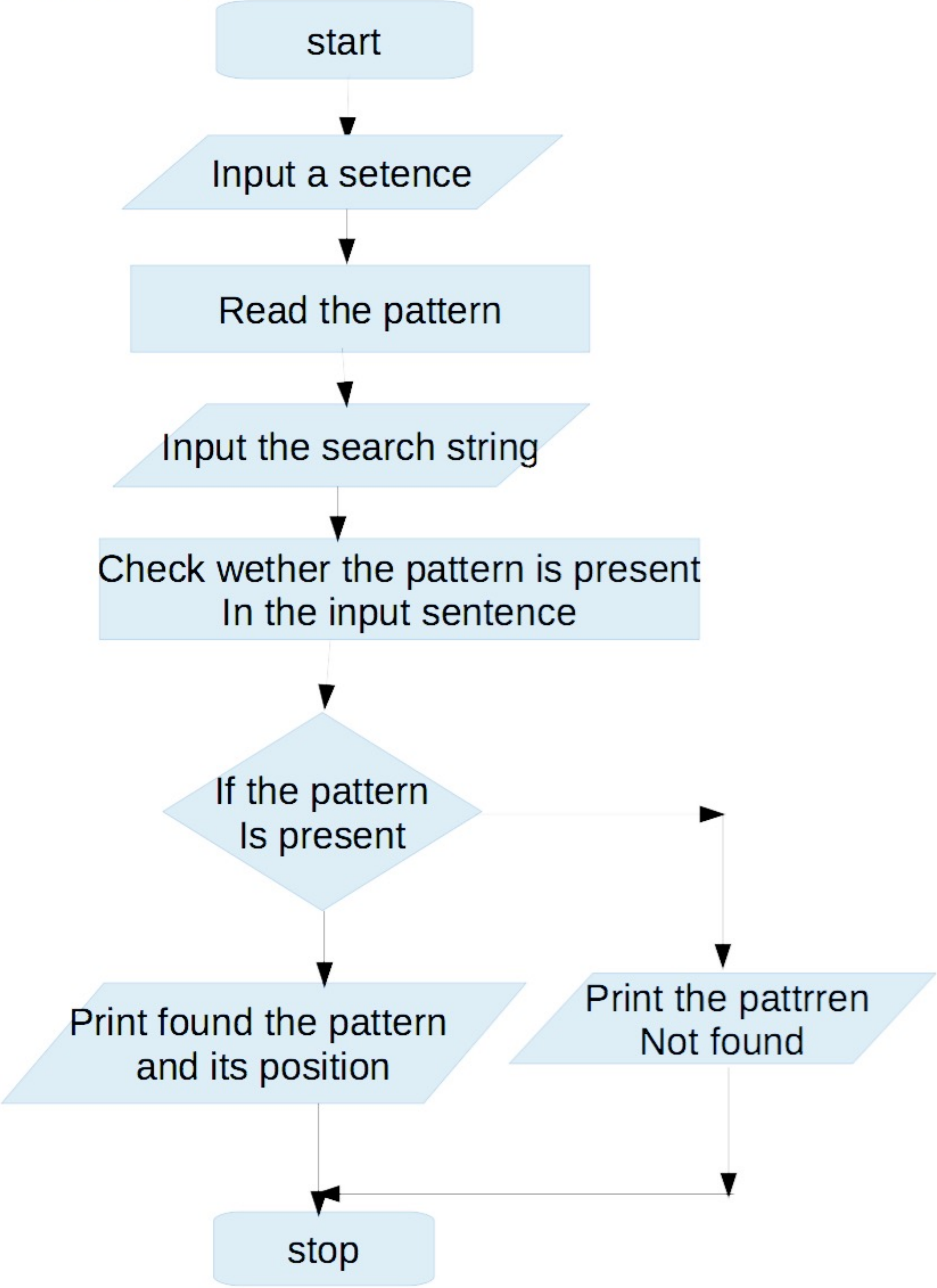
c PROGRAM TO CHECK THE WHETHER THE GIVEN NUMBER IS ODD OR EVEN

FLOWCHART:



PERL PROGRAM TO SEARCHING FOR PATTERN IN A STRING

FLOWCHART:


start
Input a setence
Read the pattern
Input the search string
Check wether the pattern is present
In the input sentence
If the pattern
Is present
Print found the pattern
and its position
Print the pattrren
Not found
stop