10) Reverse of a Number

Title :- write a C program to print Reverse of a number.

Aim :- To write a C program to print Reverse of a number.

Algorithm :-

Step1 : Start

Step2 : Declare number (i.e) num

Step3 : Sum = 0

Step4 : rem = num % 10

Sum = Sum * 10 + rem

num = num/10

Step5 : If (num > 0) then

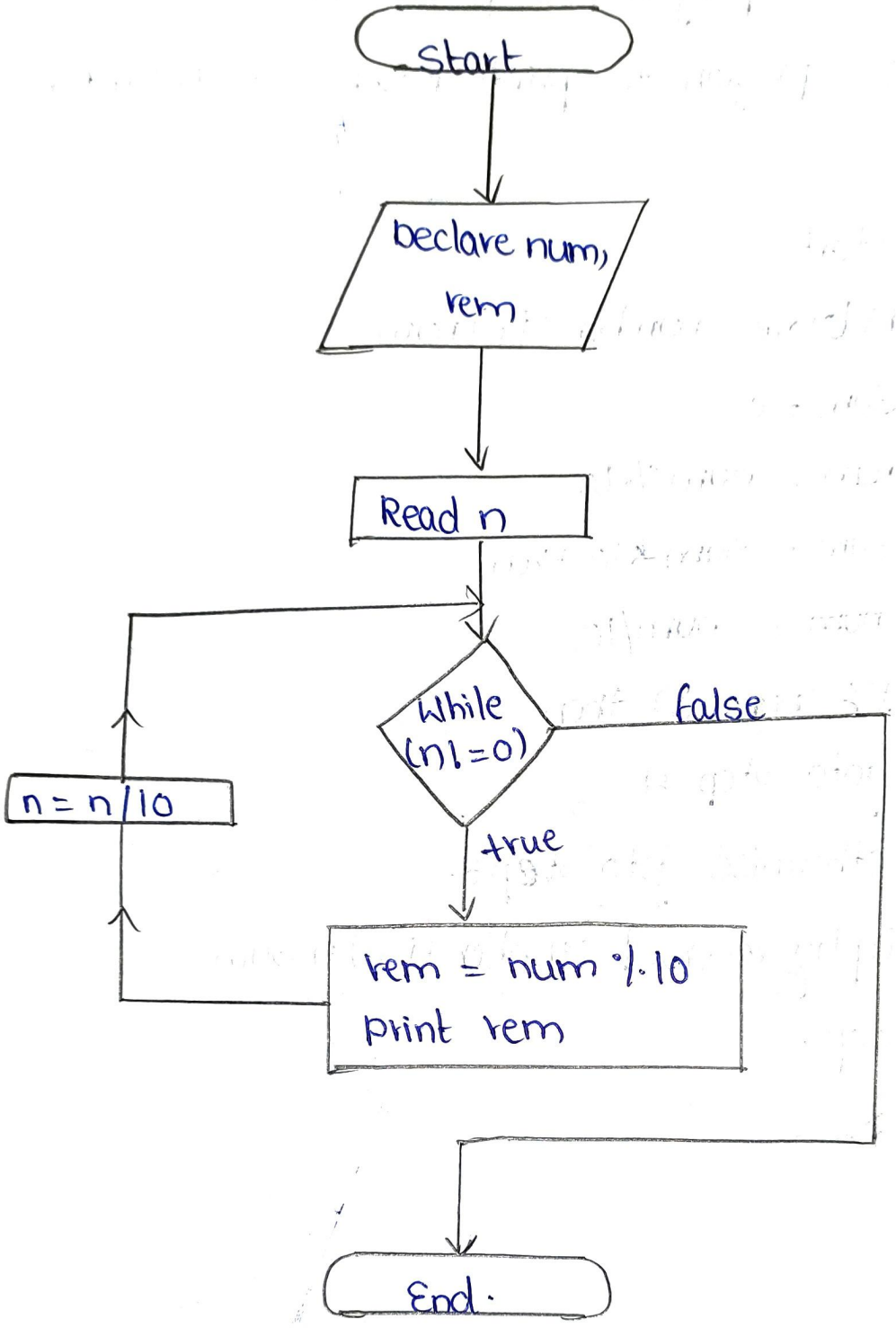
goto step 4

otherwise goto step6.

Step6 : Display reversed number (i.e) Sum

Step7 : Stop.

Flow chart :-


Start
Declare num, rem
Read n
While (n!=0)
false
true
n = n/10
rem = num %.10
print rem
End.

Source code :-

```
#include <stdio.h>
#include <conio.h>
void main( )
{
 int num, rem ;
  clrscr( );
printf ("Enter any number to print reverse of it \n");
scanf ("%od", &num);
 while (num != 0)
{
 rem = num %.10
 printf ("%od", rem);
 num = num/10 ;
 }
 getch( );
}
```

Sample Input :-

Enter any number to print reverse of it 678

Sample output :-

876

Result :-

Armstrong Number

11) Title :- write a C program to check the given number is Armstrong or not

Aim :- To check the given number is Armstrong or not.

Algorithm :-

Step1 : Start

Step2 : Accept number (i-e) num

Step3 : sum ← 0

Step4 : temp ← num

Step5 : rem ← num %. 10

sum ← sum + rem * rem * rem

num ← num/10

Step6 : if (num > 0) then goto step 5

Otherwise

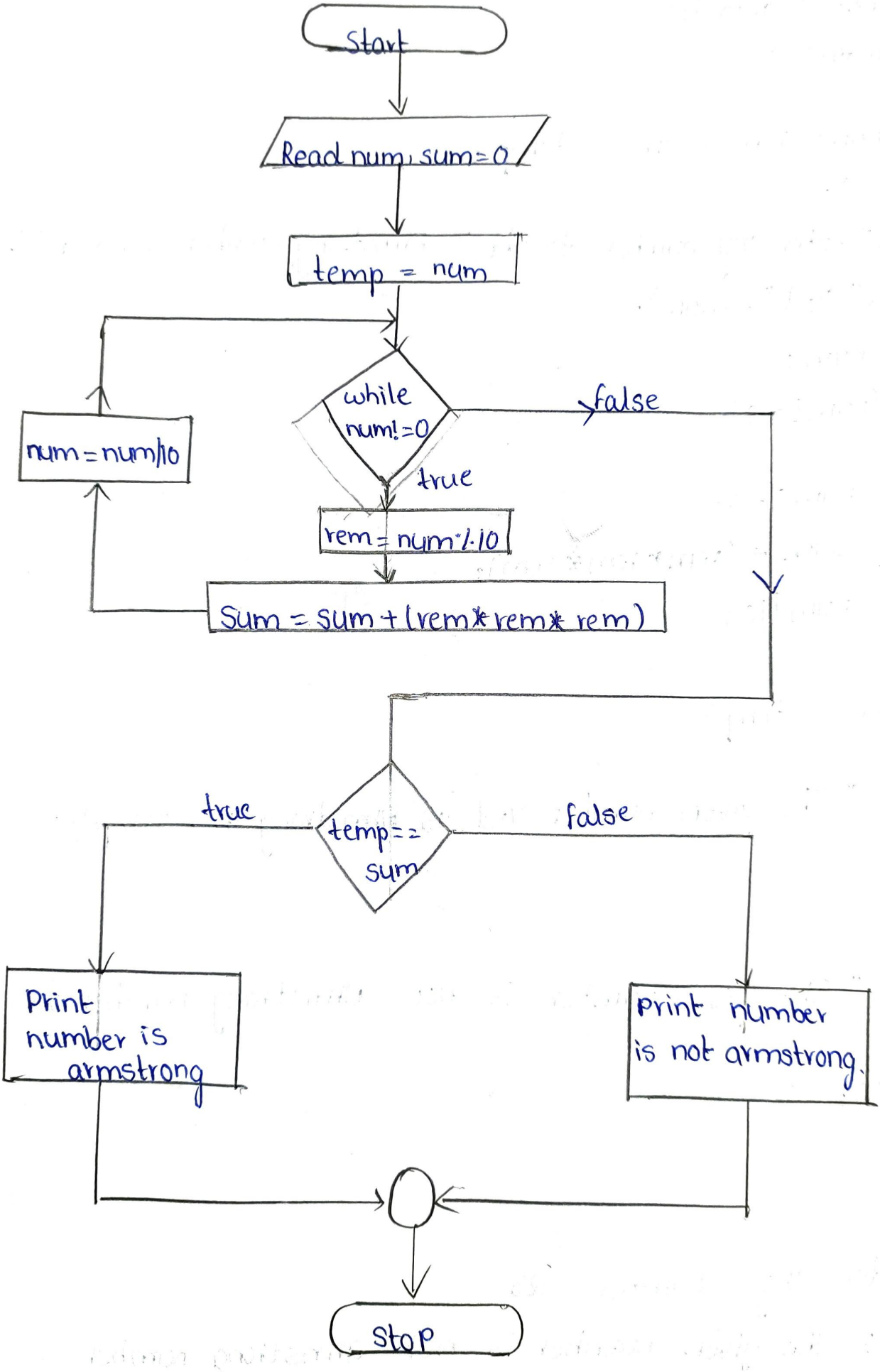
goto step 7

Step7 : if (sum == temp) then display number is armstrong number otherwise goto 8

Step8 : Display number is not armstrong

Step9 : Stop.

Flow chart :-


Start
Read num, sum=0
temp = num
while num!=0
false
true
num = num/10
rem = num%10
Sum = sum + (rem*rem*rem)
temp == sum
true
false
Print number is armstrong
print number is not armstrong.
Stop

Source code :-

```
#include <stdio.h>
#include <conio.h>
void main( )
{
int num, rem, sum=0, temp;
clrscr( );
Printf ("Enter any number to check Armstrong number or not \n");
Scanf ("%od", &num);
temp =num;
while (num!=0)
{
rem = num %.10;
Sum = Sum + (rem * rem * rem);
num = num /10;
}
if (sum = = temp)
{
printf (" The given number %d is Armstrong \n", temp);
}
else
{
printf (" The given number is not Armstrong number");
}
getch( );
}
```

Input

Enter any number 23

Output :- The given number is not armstrong number.

Program scope

12. Title :- write a C program to demonstrate program scope?

Aim :- To write a C program to demonstrate program scope.

Algorithm :-

Set x ← 50 (Global variable)

step1 : Start

step2 : set x ← 15

Step3 : Print ("x value inside main function y", x)

Step4 : call function A( )

Step5 : call function B( )

Step6 : call function A( )

Step7 : call function B( )

Step8 : print "x value inside mainfunction" x

Step9 : Stop

function :- define function A( )

Stepa :- Set x ← 25

Stepb : print "x value inside function A is" x

stepc : x ← x++

Stepd : print "x value inside function A is "x.

function : define function B( )

Stepa : Print "x value inside function B is "x

Stepb : Set x ← x * 10

Stepc : Print "x value inside function B is "x.

Source code :-

```
# include <stdio.h>
# include <conio.h>
void function A( );
void function B( );
int xc = 50;
void main( )
{
int xc = 5;
clrscr ( );
Printf (" xc value inside main function is %od \n", xc);
function A( );
function B( );
function A( );
function B( );
Printf (" xc value inside the main function is %od \n", xc);
getch( );
}
void function A( )
{
int xc = 25;
printf (" xc value inside function A is %od \n", xc);
xc++;
printf (" xc value inside function A is %od \n", xc);
}
void function B( )
{
printf (" xc value inside function B is %od \n", xc);
 xc = xc * 10;
printf (" xc value inside function B is %od \n", xc);
}
```

Source code :-

Output :-

x value inside main function is 5
x value inside function A is 25
x value inside function A is 26
x value inside function B is 50
x value inside function B is 500
x value inside function A is 25
x value inside function A is 26
x value inside function B is 500
x value inside function B is 5000
x value inside the main function is 5.

Result :-

13a] Call by value (or) Pass by value.

Title :- write a C program to demonstrate call by value

Aim :- To write a c program to demonstrate call by value.

Algorithm :

Step 1 : start

Step 2 : Input number as x

Step 3 : Input number as y

Step 4 : print " The values before swaping ", x, y

Step 5 : swap (x, y)

Step 6 : Print " The values after swaping ", x, y

Step 7 : Stop.

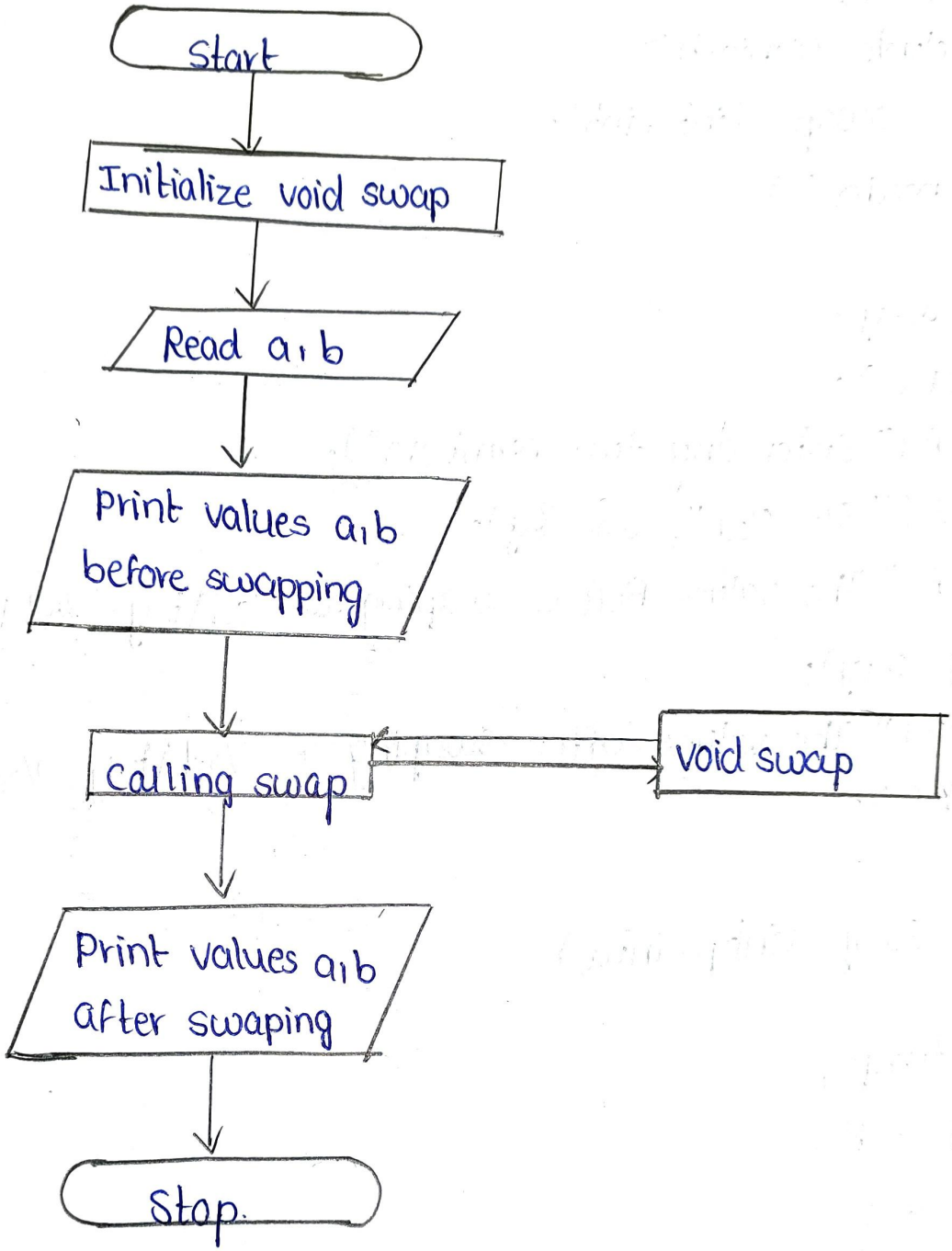
=> swap (x, y)

Step a : temp ← x

Step b : x ← y

Step c : y ← temp.

Flow chart :-


Start
Initialize void swap
Read a, b
Print values a, b before swapping
Calling swap
void swap
Print values a, b after swaping
Stop.

Source code :-

```
# include <stdio.h>
# include <conio.h>
void swap (int , int);
void main ( )
{
int x, y ;
clrscr ( );
printf (" Enter any two numbers");
scanf ("%od %od", &x, &y);
printf (" The values Before swaping x = %od \t y = %od \n", x, y);
swap (x, y);
printf (" The values after swaping x = %od \t y = %od \n", x, y);
getch ( );
}
void swap (int p, int q)
{
int temp ;
temp = p ;
p = q ;
q = temp ;
}
```

Sample input :- Enter any two numbers 78 98

sample output :- The value Before swaping x = 78 y = 98

The values after swaping x = 78 y = 98.

Result :-

13b) Call by Reference (or) pass by reference :-

Title :- write a C program to demonstrate call by Reference

Aim :- To write a C program to demonstrate call by Reference.

Algorithm :-

Step1 : Start

Step2 : Input a number as x

Step3 : Input a number as y

Step4 : print " The values before swapping " x, y

Step5 : swap (&x, &y)

Step6 : print " The values after swapping " x, y

Step7 : Stop

=> Swap (*p, *q)

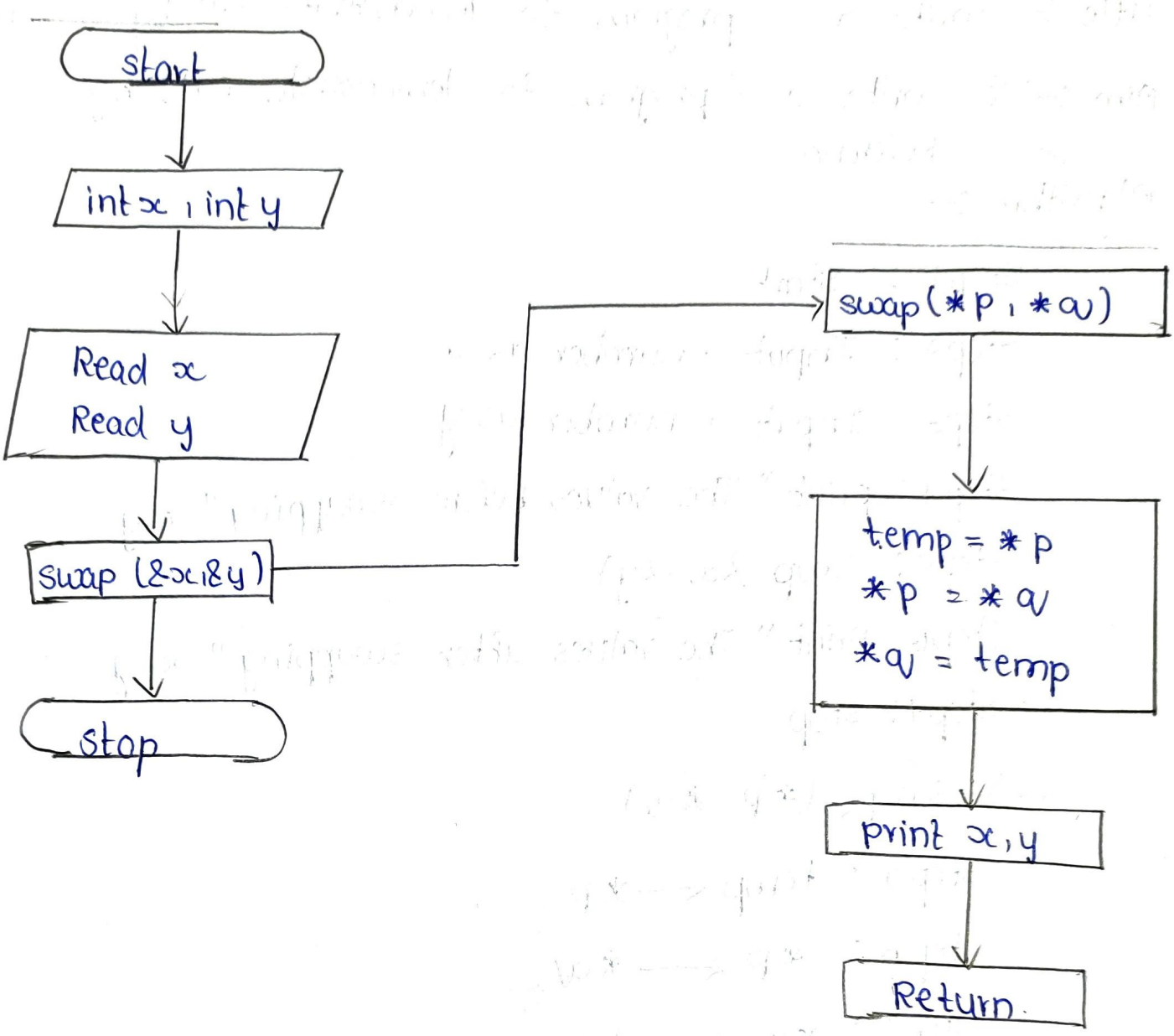
Step a : temp ← *p

Step b : *p ← *q

Step c : *q ← temp

Step d : return value to main function.

Flow chart :-


start
int x , int y
Read x
Read y
swap (&x,&y)
stop
swap (*p , *q)
temp = *p
*p = *q
*q = temp
print x, y
Return

Source code :-

```
#include <stdio.h>
#include <conio.h>
void swap (int * , int *);
void main ()
{
int x, y;
clrscr ();
printf ("Enter any two numbers");
scanf ("%od %od", &x, &y);
printf ("The values Before swaping x=%od \t y=%od \n", x, y);
swap (&x, &y);
printf ("The values after swaping x=%od \t y=%od \n", x, y);
getch ();
}
void swap (int *p, int *av)
{
 int temp;
 temp = *p;
 *p = *av;
 *av = temp;
}
```

Sample Input :- Enter any two numbers 56 78

Sample output :-

The values before swaping x=56 y=78

The values after swaping x=78 y=56

Result :-