IQRA NATIONAL UNIVERSITY

FUNDAMENTALS OF PROGRAMMING MID PAPER

NAME: FAREHA MEHMOOD JEHANGIRI

ID: 16051

SUBMITTED TO: DR. FAZAL E MALIK.

Question.1

a) Draw the flow chart to get two integer items from keyboard and then display to the screen their sum, difference and product.

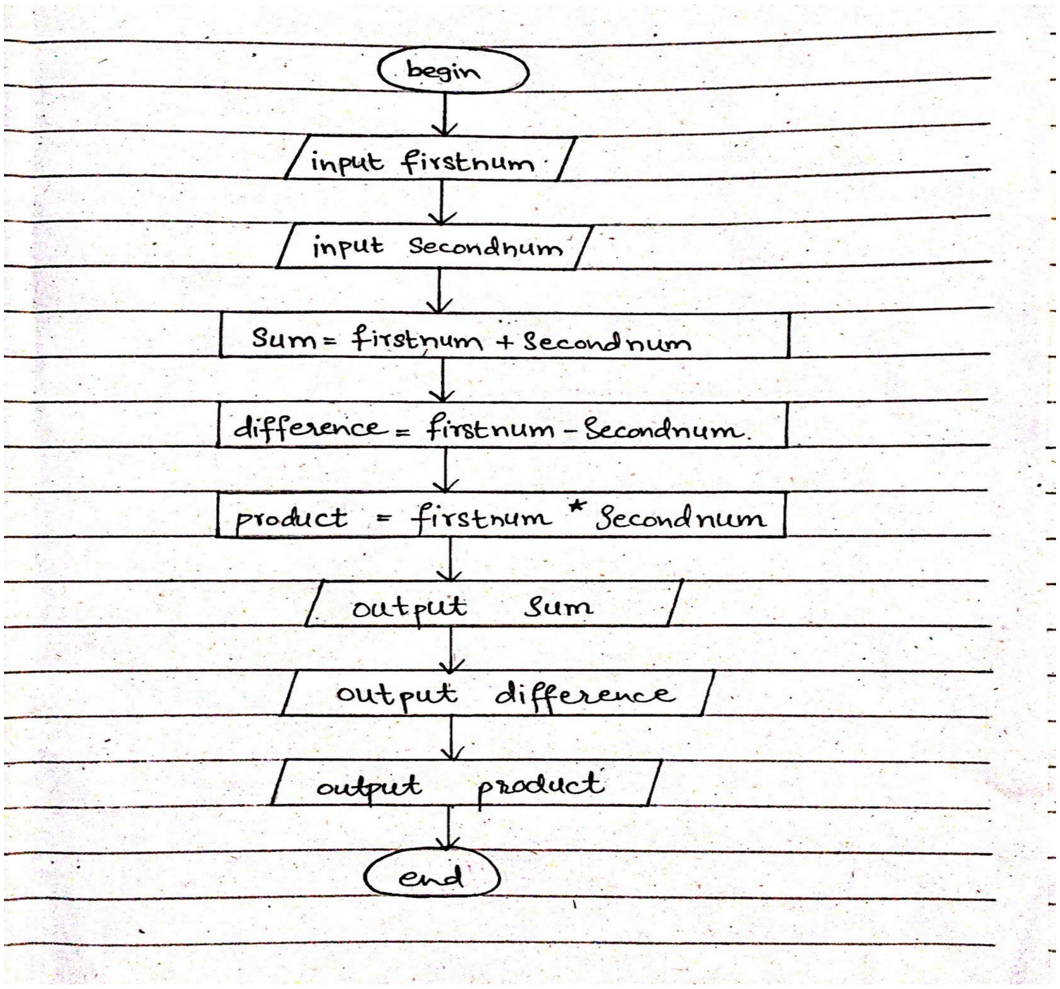
Answer:

Flowchart: A "picture" of an algorithm using specific symbols to indicate various programming constructs.

In simple words, a flowchart provides a detailed picture of the algorithm using special symbols to represent various program statements. A flowchart will always be drawn from top to bottom showing the exact order of the steps.

There are three flowchart symbols necessary for programs involving simple sequence. Begin/End of a block (written inside a horizontal oval), the process (written inside a rectangle), Input/Output (written inside a slanted rectangle or parallelogram)

Flowchart:

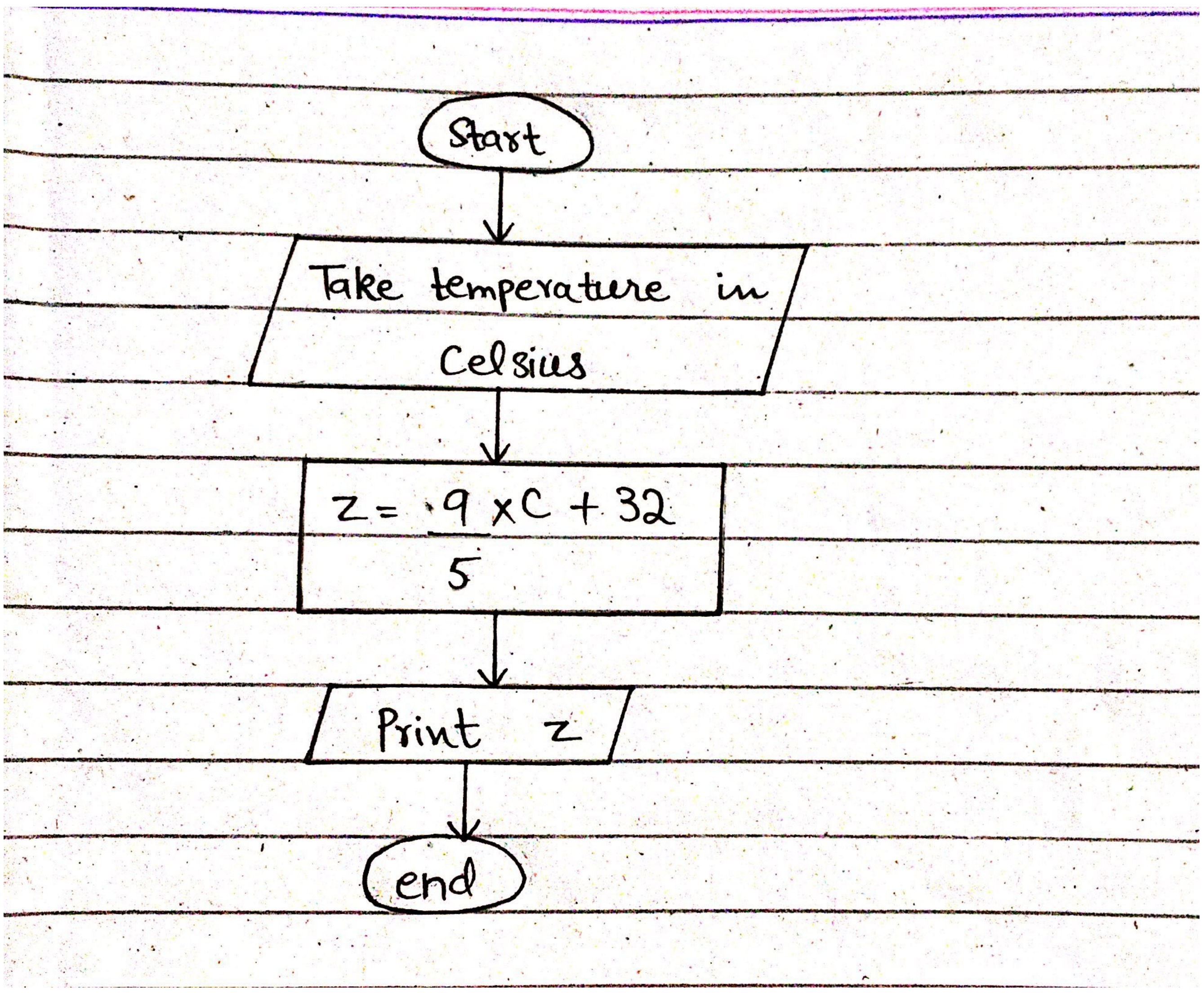

begin
input firstnum
input Secondnum
Sum = firstnum + Secondnum
difference = firstnum - Secondnum.
product = firstnum * Secondnum
output Sum
output difference
output product
end

Question.1

b) Draw the flow chart to prompt the user for a temperature in degrees Celsius (C), then convert the temperature in degrees Fahrenheit (F) using the following formula and display temperature in Fahrenheit (F) on monitor.

F=9/5 x C + 32

Flowchart:

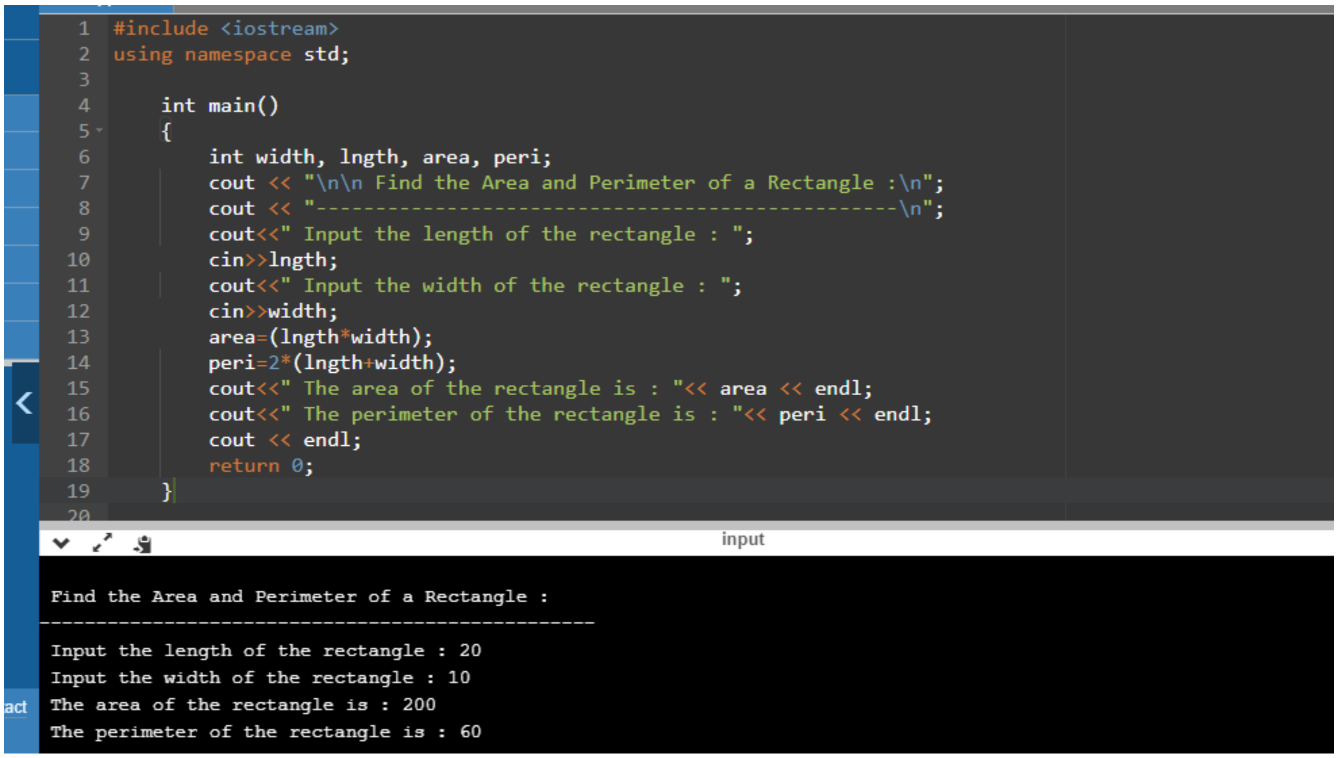


Question.2

a) Draw the flow chart and write a C++ program to find the Area and Perimeter of a Rectangle using the below formulae

Area of rectangle: height*width
Perimeter of rectangle: 2*(height + width)

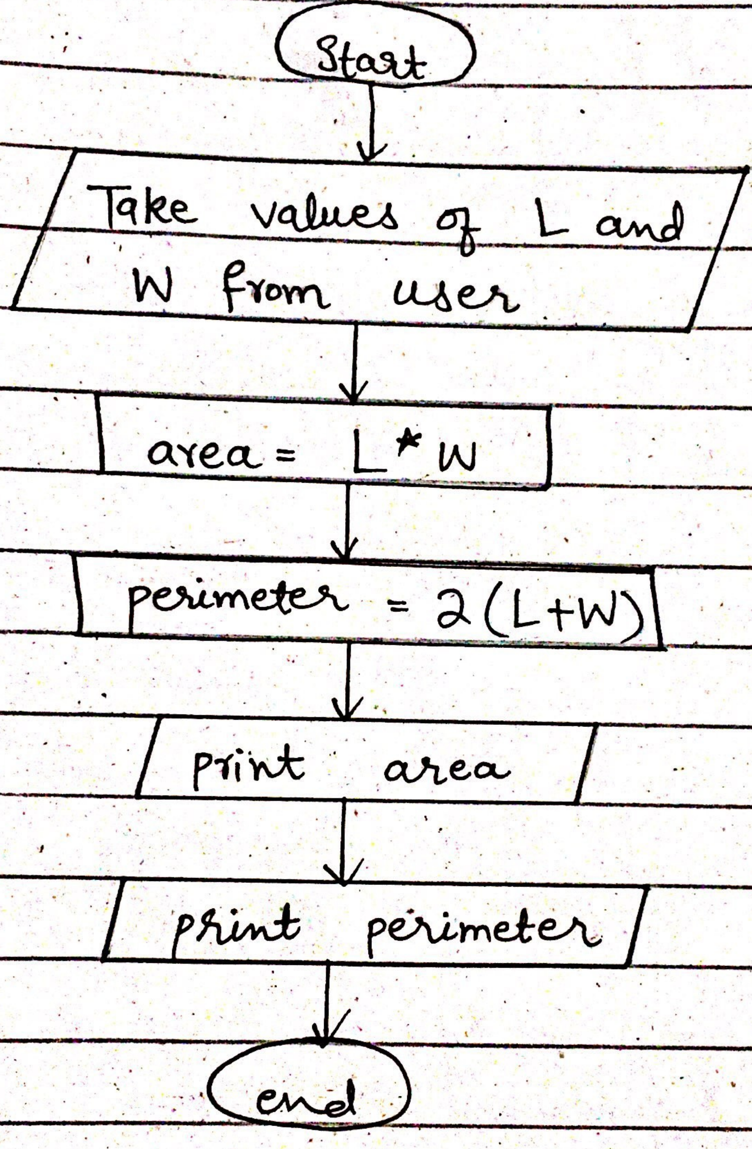
Program screenshot:



```cpp
#include <iostream>
using namespace std;

    int main()
    {
        int width, lngth, area, peri;
        cout << "\n\n Find the Area and Perimeter of a Rectangle :\n";
        cout << "--------------------------------------------------\n";
        cout<<" Input the length of the rectangle : ";
        cin>>lngth;
        cout<<" Input the width of the rectangle : ";
        cin>>width;
        area=(lngth*width);
        peri=2*(lngth+width);
        cout<<" The area of the rectangle is : "<< area << endl;
        cout<<" The perimeter of the rectangle is : "<< peri << endl;
        cout << endl;
        return 0;
    }
```

```
 Find the Area and Perimeter of a Rectangle :
------------------------------------------------
 Input the length of the rectangle : 20
 Input the width of the rectangle : 10
 The area of the rectangle is : 200
 The perimeter of the rectangle is : 60
```

Flowchart:

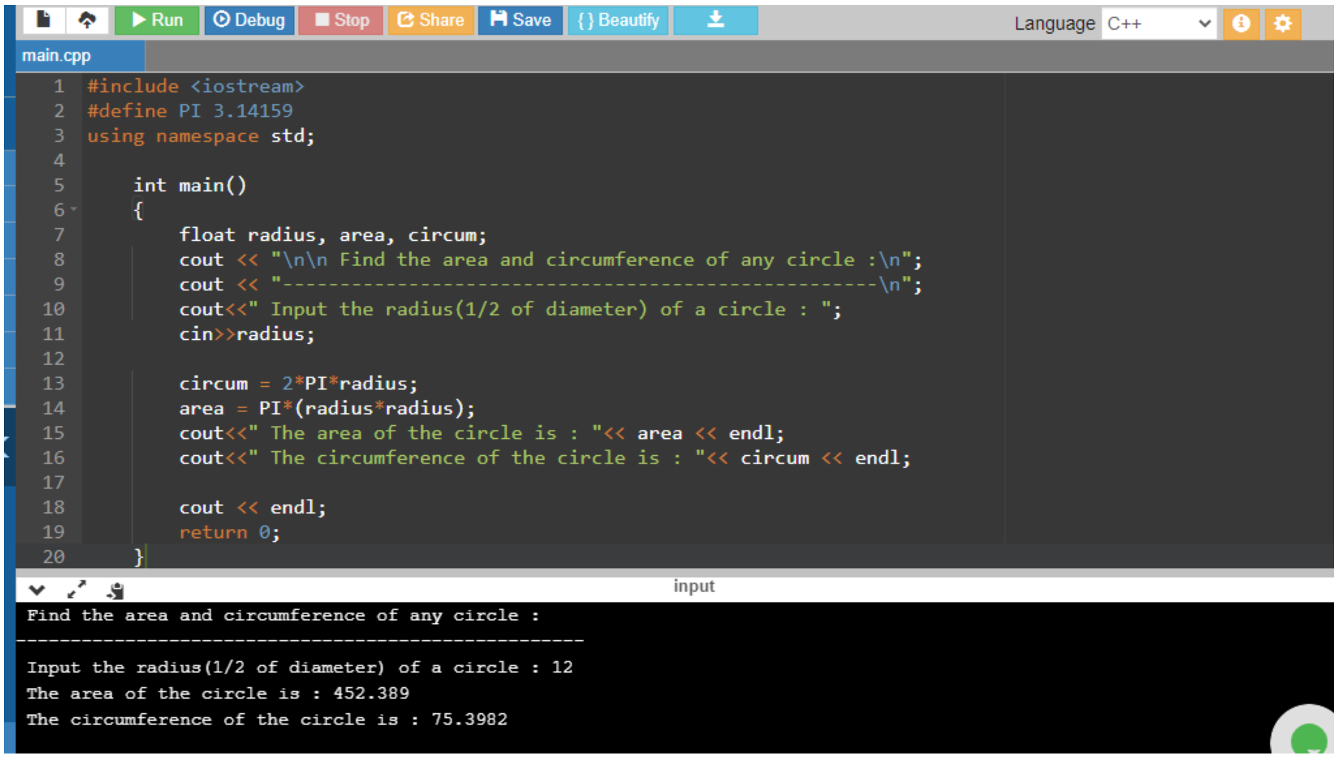

Start
Take values of L and W from user
area = L*W
perimeter = 2(L+W)
print area
print perimeter
end

Question.2

b) Draw the flow chart and write a C++ program to obtain the radius of a circle. Then program calculates the area and perimeter using the below Formulae

Area of Circle = π*R*R
Circumference formula C = 2 * π * R. where π=3.14

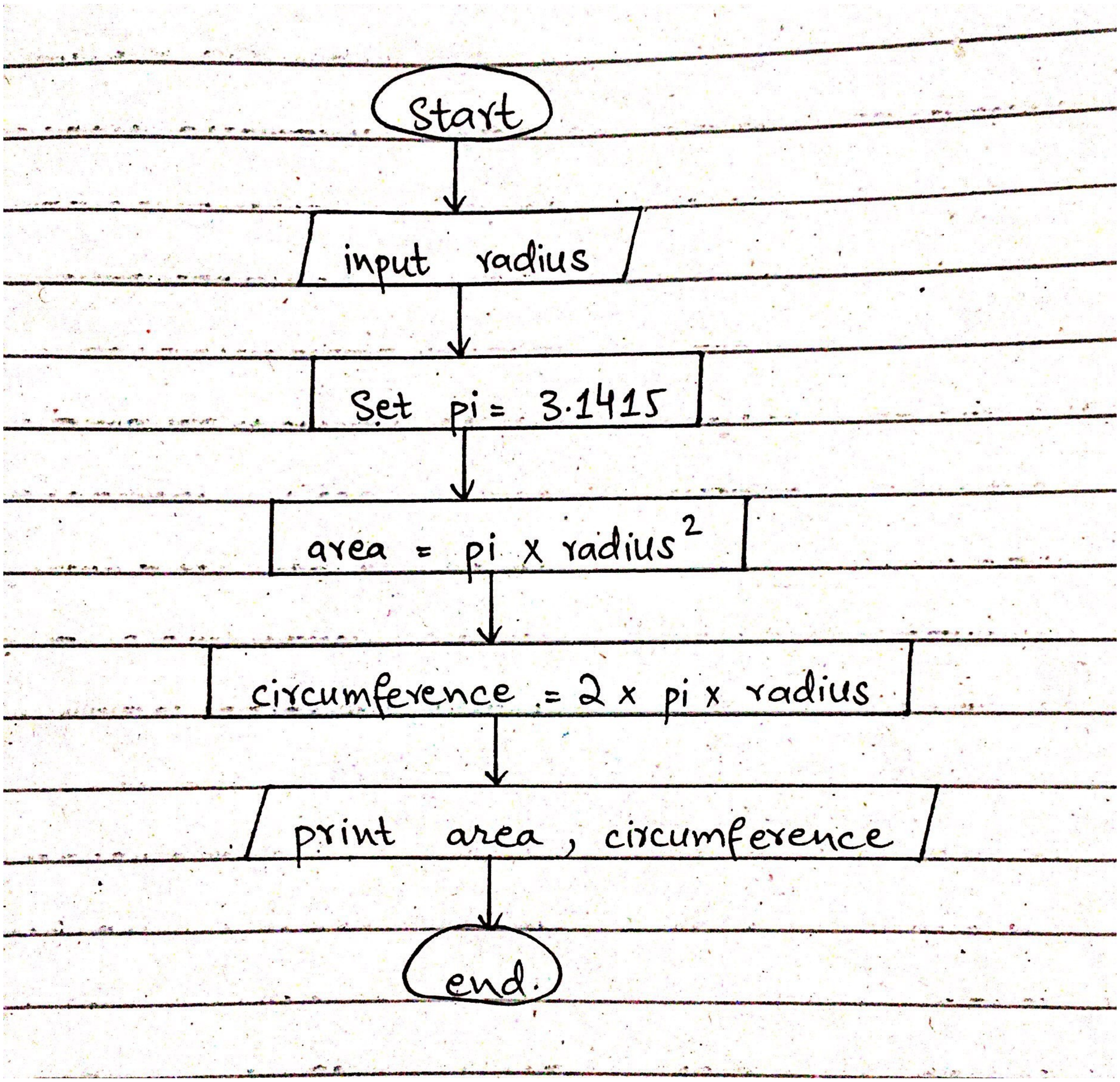
Program screenshot:



```cpp
#include <iostream>
#define PI 3.14159
using namespace std;

    int main()
    {
        float radius, area, circum;
        cout << "\n\n Find the area and circumference of any circle :\n";
        cout << "---------------------------------------------------\n";
        cout<<" Input the radius(1/2 of diameter) of a circle : ";
        cin>>radius;

        circum = 2*PI*radius;
        area = PI*(radius*radius);
        cout<<" The area of the circle is : "<< area << endl;
        cout<<" The circumference of the circle is : "<< circum << endl;

        cout << endl;
        return 0;
    }
```

```
Find the area and circumference of any circle :
---------------------------------------------------
Input the radius(1/2 of diameter) of a circle : 12
The area of the circle is : 452.389
The circumference of the circle is : 75.3982
```

Flowchart:


Start
input radius
Set pi = 3.1415
area = pi x radius^2
circumference = 2 x pi x radius
print area , circumference
end.

Question.3

a) Discuss different types of programming languages.

Answer:

Language is a medium or way of communication. In the programming terminology, language is a tool which provides a platform through which we can communicate to the computer by writing computer programs.

1. Procedural Programming Language:

The procedural programming language is used to execute a sequence of statements which lead to a result. Typically, this type of programming language uses multiple variables, heavy loops and other elements, which separates them from functional programming languages. Functions of procedural language may control variables, other than function's value returns. For example, printing out information.

2.Functional Programming Language:

Functional programming language typically uses stored data, frequently avoiding loops in favor of recursive functions. The functional programming's primary focus is on the return values of functions, and side effects and different suggests that storing state are powerfully discouraged. For example, in an exceedingly pure useful language, if a function is termed, it's expected that the function not modify or perform any o/p. It may, however, build algorithmic calls and alter the parameters of these calls. Functional languages are usually easier and build it easier to figure on abstract issues, however, they'll even be "further from the machine" therein their programming model makes it difficult to know precisely, but the code is decoded into machine language (which are often problematic for system programming).

3.Object-oriented Programming Language:

This programming language views the world as a group of objects that have internal data and external accessing parts of that data. The aim this programming language is to think about the fault by separating it into a collection of objects that offer services which can be used to solve a specific problem. One of the main principle of object oriented programming language is encapsulation that everything an object will need must be inside of the object. This language also emphasizes reusability through inheritance and the capacity to spread current implementations without having to change a great deal of code by using polymorphism.

4.Scripting Programming Language:

These programming languages are often procedural and may comprise object-oriented language elements, but they fall into their own category as they are normally not full-fledged programming languages with support for development of large systems. For example, they may not have compile-time type checking. Usually, these languages require tiny syntax to get started.

5.Logic Programming Language:

These types of languages let programmers make declarative statements and then allow the machine to reason about the consequences of those statements. In a sense, this language doesn’t tell the computer how to do something, but employing restrictions on what it must consider doing.

Question.3
b) How many translators are there to translate higher level language to machine language? Discuss.

Answer:

A high-level language is a programming language that uses English and mathematical symbols, like +, -, % and many others, in its instructions. When using the term 'programming languages,' most people are actually referring to high-level languages. High-level languages are the languages most often used by programmers to write programs. Examples of high-level languages are C++, Java and Python.

High-level language programs must be translated into machine language before they can be executed. Machine language instructions are encoded as binary numbers that are meant to be used by a machine, not read or written by people. Whereas high-level languages use a syntax that is closer to human language.

A translator, in software programming terms, is a generic term that could refer to a compiler, assembler, or interpreter; anything that converts higher level code into a language that the processor can understand, such as assembly language or machine code.

Compilers:
Compilers convert high-level language code to machine (object) code in one session. Compilers can take a while, because they have to translate high-level code to lower-level machine language all at once and then save the executable object code to memory. A compiler creates machine code that runs on a processor with a specific Instruction Set Architecture. Compilers will report errors after compiling has finished.

Interpreters:

Another way to get code to run on your processor is to use an interpreter, which is not the same as a compiler. An interpreter translates code like a compiler but reads the code and immediately executes on that code, and therefore is initially faster than a compiler. Thus, interpreters are often used in software development tools as debugging tools, as they can execute a single in of code at a time. Compilers translate code all at once and the processor then executes upon the machine language that the compiler produced. If changes are made to the code after compilation, the changed code will need to be compiled and added to the compiled code (or perhaps the entire program will need to be re-compiled.) But an interpreter, although skipping the step of compilation of the entire program to start, is much slower to execute than the same program that's been completely compiled.