Programming and Problem Solving through Python Language
O Level / A Level

Chapter -2 : Algorithms for Problem Solving

Type of Algorithms

The algorithm and flowchart, classification to the three types of control structures.

They are: **1. Sequence 2. Branching (Selection) 3. Loop (Repetition)**

These three control structures are sufficient for all purposes. The sequence is exemplified by sequence of statements place one after the other – the one above or before another gets executed first. In flowcharts, sequence of statements is usually contained in the rectangular process box.

Branching

The branch refers to a binary decision based on some condition. If the condition is true, one of the two branches is explored; if the condition is false, the other alternative is taken. This is usually represented by the 'if-then' construct in pseudo-codes and programs. In flowcharts, this is represented by the diamond-shaped decision box. This structure is also known as the selection structure.

Problem 1: write algorithm to find the greater number between two numbers

Step1: Start

Step2: Read/input A and B

Step3: If A greater than B then C=A

Step4: if B greater than A then C=B

Step5: Print C

Step6: End

Problem 2: write algorithm to find the result of equation: f(x) ={ -x, x < 0 ,

x, x>=0 }

Step1: Start

Step2: Read/input x

Step3: If X Less than zero then F=-X

Step4: if X greater than or equal zero then F=X

Step5: Print F

Step6: End

Problem 3: A algorithm to find the largest value of any three numbers.

Step1: Start

Step2: Read/input A,B and C

Step3: If (A>=B) and (A>=C) then Max=A

Step4: If (B>=A) and (B>=C) then Max=B

Step5:If (C>=A) and (C>=B) then Max=C

Step6: Print Max

Step7: End

Loops

The loop allows a statement or a sequence of statements to be repeatedly executed based on some loop condition. It is represented by the 'while' and 'for' constructs in most programming languages, for unbounded loops and bounded loops respectively. (Unbounded loops refer to those whose number of iterations depends on the eventuality that the termination condition is satisfied; bounded loops refer to those whose number of iterations is known before-hand.) In the flowcharts, a back arrow hints the presence of a loop. A trip around the loop is known as iteration. You must ensure that the condition for the termination of the looping must be satisfied after some finite number of iterations, otherwise it ends up as an infinite loop, a common mistake made by inexperienced programmers. The loop is also known as the repetition structure.

Examples:

Problem 1: An algorithm to calculate even numbers between 0 and 99

1. Start
2. I ← 0
3. Write I in standard output
4. I ← I+2
5. If (I <=98) then go to line 3
6. End

Problem 2: Design an algorithm which gets a natural value, n,as its input and calculates odd numbers equal or less than n. Then write them in the standard output:

1. Start
2. Read n
3. I ← 1
4. Write I
5. I ← I + 2
6. If (I <= n) then go to line 4
7. End

Problem : Design an algorithm which generates even numbers between 1000 and 2000 and then prints them in the standard output. It should also print total sum:

1. Start
2. I ← 1000 and S ← 0
3. Write I
4. S ← S + I
5. I ← I + 2
6. If (I <= 2000) then go to line 3 else go to line 7
7. Write S
8. End

Problem 4: Design an algorithm with a natural number, n, as its input which calculates the following formula and writes the result in the standard output:

S = ½ + ¼ + … +1/n

1. Start
2. Read n
3. I ← 2 and S ← 0
4. S= S + 1/I
5. I ← I + 2
6. If (I <= n) then go to line 4 else write S in standard output
7. End

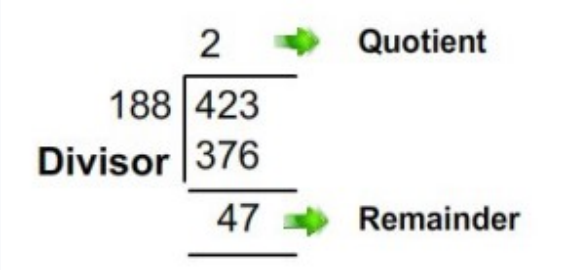
Problem 5: Factorial of Number.

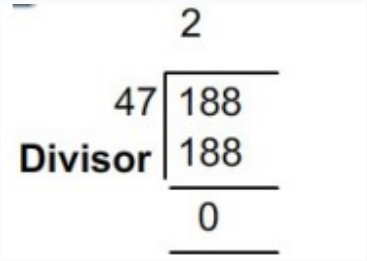
To compute factorial of a given number:
Input: Given the Number to be computed => N.
Output: Factorial of N is Fact.

Step 1: Input Number whose Factorial is to be computed: N.
Step 2: Initialize Fact =1. Assign N = M. Step
3: Check if M is greater than 1.
Yes:
Assign product of Fact and M to Fact.
Decrement M.
Repeat Step 3.
No:
Output Factorial of N is Fact.
Step 4: Stop.

Problem 6 : Greatest Common Divisor (GCD) of two number.

Step 1: Start
Step 2: Read two number for which GCD is to be found , say (A, B)
Step 3: Let A be the larger of the two numbers.
Step 4: Divide A by B.
Step 5: Get the remainder of the division operation.
Step 6: Divide the divisor of the previous division operation with the remainder.
Step 7: Repeat steps 4 , 5 ,6 till the remainder become zero.
Step 8: The divisor of the last division operation performed is the GCD of the two numbers.
Step 9: Stop





Assignments

1. Write an algorithm to print the sum of first 10 natural number.
2. Write an algorithm to print the first 10 odd number
3. Write an algorithm to print the 10 multiples of any number.
4. Write an algorithm to print

   1
   2 2
   3 3 3
   4 4 4 4
   5 5 5 5 5