Problem solving with C

Problem solving

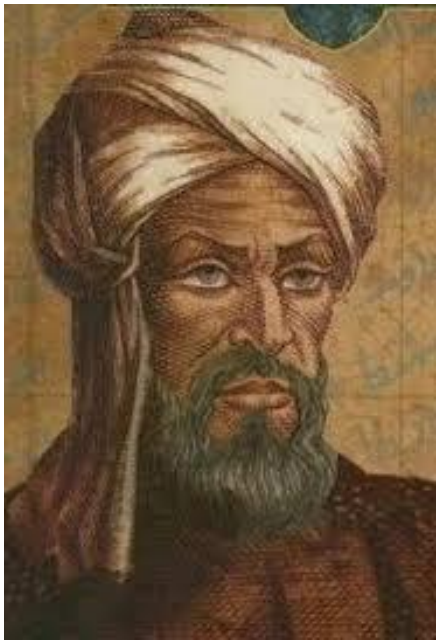
- Solving problems is the core of computer science.
- Problem solving is the process or act of finding a solution to a problem.
- Programmers must first understand how a human solves a problem, then understand how to translate this “process" into something a computer can do, and finally how to "write" the specific syntax (required by a computer) to get the job done.

Problem solving

- The core of what good programmers do is being able to define the steps necessary to accomplish a goal.
- A computer can add two numbers.
- But if you want to find the average of two numbers, this is beyond the basic capabilities of a computer. To find the average, you must:
  - First: Add the two numbers and save this result in some place.
  - Then: Divide this new number by two, and save this result in some place.
  - Finally: Use this number wherever required (or print this number).

- A problem has a well-specified method to provide a solution.
- If you can represent this method as a sequence of steps, it is called an algorithm.

Algorithm



- An algorithm is a well-defined sequence of steps to be performed to solve a problem or a task.
- The word “Algorithm” comes from the name of a 9th century Persian mathematician:

  Musa **al khwarizmi**

- Definition
  - An algorithm is a finite sequence of precise or unambiguous steps required to solve a given problem.
  - An algorithm takes zero or more inputs, processes them according to the instructions given in the steps and produces one or more meaningful outputs.

Characteristics of Algorithm

- Sequence
  - Algorithmic steps must be performed in a sequence
- Precision
  - The steps are precisely stated(defined).
  - No ambiguity
- Finiteness
  - An algorithm must always terminate after a finite number of steps.
- Input
  - An algorithm has zero or more inputs.
- Output
  - An algorithm has one or more outputs.

Examples of Algorithms

- A recipe for chocolate cake,
- Given a list of integers, tell me which ones are even,
- Finding square root of a number,
- Arranging a set of numbers in ascending order,
- Finding whether a number exists in a list or not.

Algorithm to get ready for school

1. Get up from the bed.
2. Go to washroom.
3. Brush the teeth.
4. Drink a glass of milk.
5. Take bath.
6. Have breakfast.
7. Wear the uniform.
8. Take the school bag.
9. Put on the shoes.

Algorithm for making a cup of tea

1. Take a glass of water.
2. Pour it into the bowl.
3. Put the bowl on the gas.
4. Burn the gas.
5. Add one (1) tea spoon of sugar.
6. Add one (1) tea spoon of tea powder.
7. Wait for the water to boil for two minutes.
8. Put the gas off.
9. Strain tea into a cup.
10. Add required amount of milk.
11. Tea is ready.

Ambiguous Algorithm

- Take two pieces of bread.
- Put peanut butter on one side of one piece.
- Put jelly on one side of the other piece.
- Put the pieces together.

Output of this algorithm is…

A Messy Sandwich!

- The algorithm did not specify that the pieces of bread should be put together so that the peanut butter and jelly are on the inside.
- Caution: Computers do exactly what they are told.

Example 1

- A fruit vendor sells mangoes at Rs.5 for each mango. Prepare an algorithm that will help the vendor to calculate the total amount the customer has to pay.

Algorithm

1. Accept the number of mangoes bought say N.
2. amount = N * 5.
3. Print amount.
4. Stop.

Example 2

- A child wants to calculate the area of a rectangle. Prepare an algorithm that will help the child to do this.

Algorithm

1. Accept the length and breadth of the rectangle into len and brd.
2. area = len * brd.
3. Print area.
4. Stop.

Example 3

- It is required to convert distance from kilometer to meter. Prepare an algorithm to do this.

Algorithm

1. Accpt distance in kilometers into k.
2. meters = k * 1000
3. print meters
4. Stop.

Example 4

- A bank wants to calculate simple interest of a savings account. Prepare an algorithm that will help the bank to do this.

Algorithm

1. Accept the principal, time and rate in p, t and r.
2. simple_interest = (p * t * r)/100.
3. Print simple_interest.
4. Stop.

Example

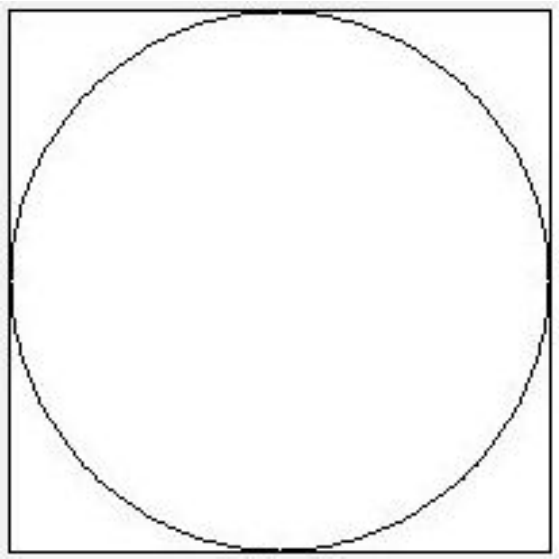
- Write an algorithm to find all roots of a quadratic equation $ax^2+bx+c=0$.
  - Input: Coefficients a, b and c

Algorithm

1. Read a,b,c
2. d = b * b - 4 * a * c
3. if d > 0 then
   Display "Roots are real and distinct"
   root1 = (-b + sqrt(d)) / (2 * a)
   root2 = (-b - sqrt(d)) / (2 * a)
   Display root1, root2
   else if d = 0 then
   Display "Roots are real and equal"
   root1 = root2 = -b / (2 * a)
   Display root1, root2
   else
   Display "Roots are imaginary"
   real = -b / (2 * a)
   imag = sqrt(abs(d)) / (2 * a)
   Display real, imag
   end if
4. Stop

Example

- A circle is inscribed in a square. Write an algorithm to find the area of the circle.

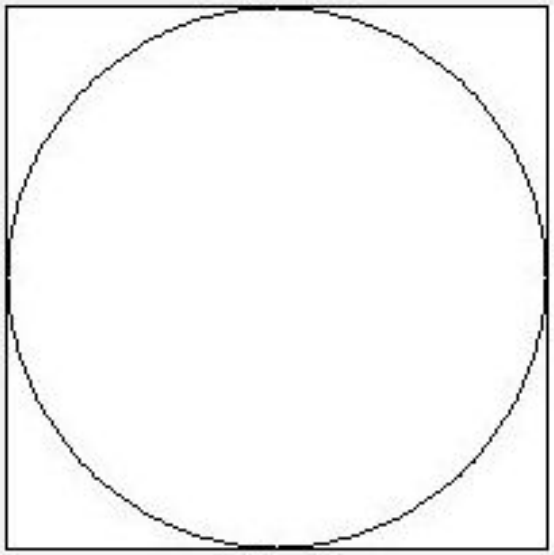


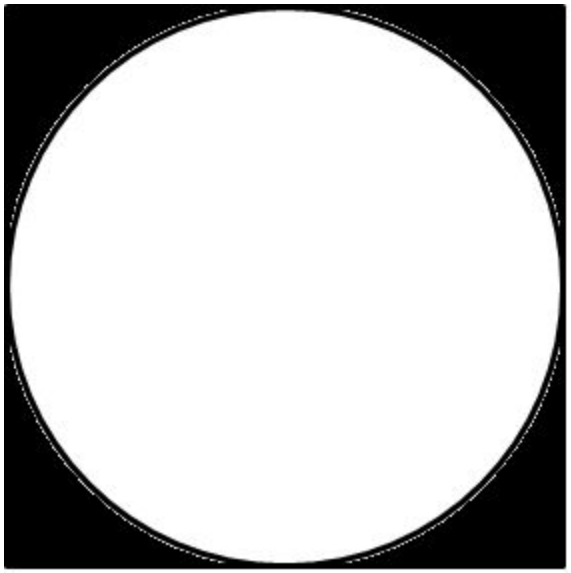
Algorithm

1. Accept side of the square in to d.
2. radius = d / 2
3. area = 3.142 * radius * radius
4. Print area
5. Stop

Example

- A circle is inscribed in a square. Write an algorithm to find the area of the square not covered by the circle.



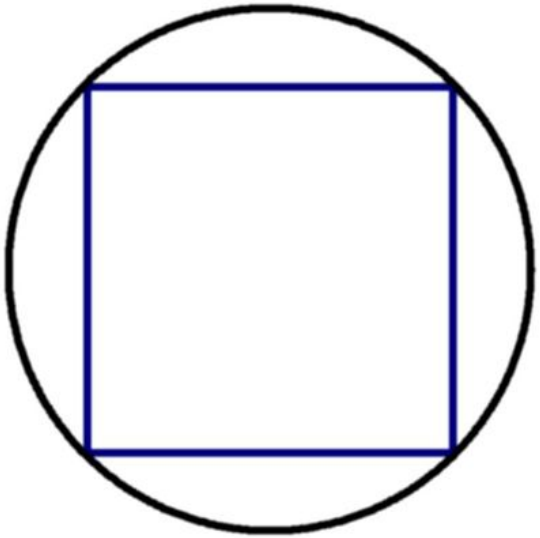


Algorithm

1. Accept side of the square in to d.
2. radius = d / 2
3. c_area = 3.142 * radius * radius
4. s_area = d * d
5. area = s_area – c_area
6. Print area
7. Stop

Example

- Given the side of a square that is inscribed within a circle, write an algorithm to calculate the area of the circle.



1. Accept the length of the side of the square as r.
2. diameter = sqrt(r * r + r * r).
3. radius = diameter / 2.
4. area = **3.142** * radius * radius.
5. Print area.
6. stop.

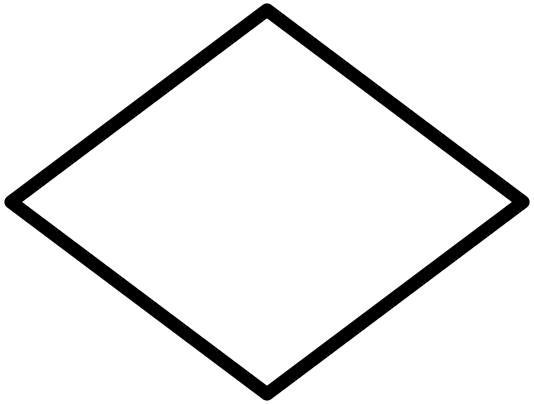
Flowchart

- A flowchart is a graphical or diagrammatic representation of an algorithm.
- Flowcharts are drawn using specific meaningful symbols such as
  - Rounded rectangle,
  - Rectangle,
  - Diamond or rhombus,
  - Circles and so on.

Flowcharting symbols

Start / Stop indicator

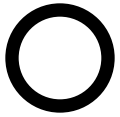
Processing step



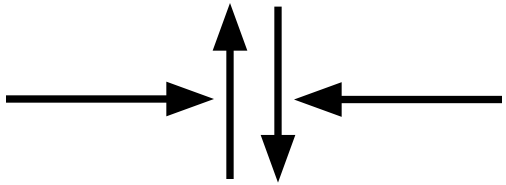
Decision Making

Flowcharting symbols

Input / Output operation



Labeled connector



Process flow

Function / Procedure call

Example

- Given two numbers, write an algorithm to find the smallest among them.

Algorithm

1. Accept two numbers into m and n.
2. if m < n then

   Print “m is the smaller one.”

   else

   Print “n is the smaller one.”

   end if
3. Stop

Example

- Write an algorithm to find the largest among three different numbers entered by user.

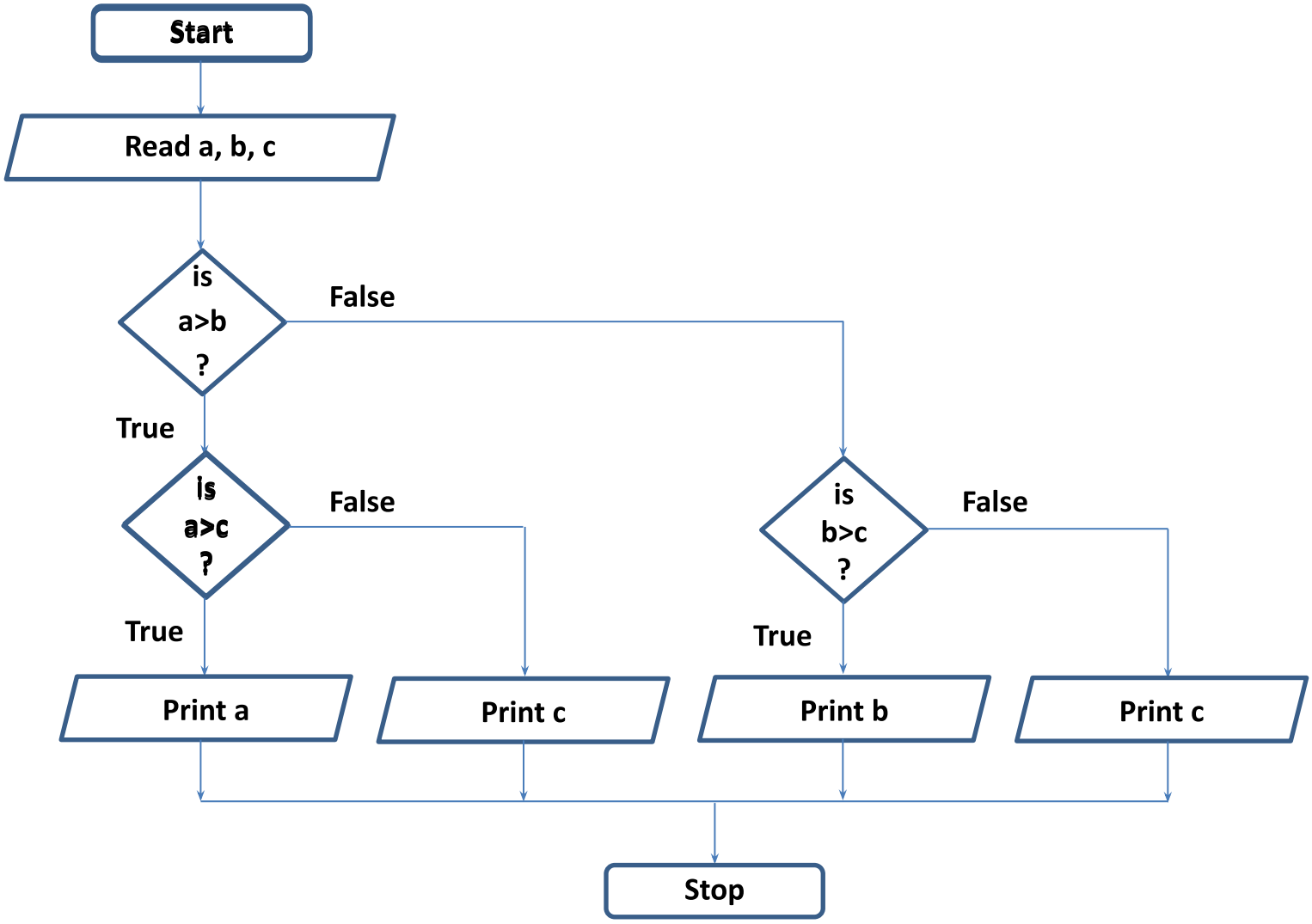
Algorithm

1. Accept three numbers into a, b and c.
2. If a > b then
    if a > c then
        Print “a is the largest number.”
    else
        Print “c is the largest number.”
    end if
else
    if b > c then
        Print “b is the largest number.”
    else
        Print “c is the greatest number.”
    end if
end if
3. Stop

- Write an algorithm to accept marks obtained by a student and display the grade. Grade is computed as per the following criteria:

Marks	Grade
90 – 100	A
70 – 89	B
50 – 69	C
< 50	D

1. Accept m
2. if m ≥ 90 then
   Print “Grade A”
   else if m ≥ 70 then
   Print “Grade B”
   else if m ≥ 50 then
   Print “Grade C”
   else
   Print “Grade D”
   end if
3. Stop


Start
Read a, b, c
is a>b ?
False
True
is a>c ?
False
True
Print a
Print c
is b>c ?
False
True
Print b
Print c
Stop

Example

- Write an algorithm to check whether a given number is odd or even.

Algorithm

1. Accept a number into n.
2. if n mod 2 equals 0 then

   Print “Number is even”

   else

   Print “Number is odd”

   end if
1. Stop

Example

- Write an algorithm to compute remainder of dividing one number by another (There is no mod operator!).

Algorithm

1. Accept two numbers into m and n.
2. quo = m / n
3. prod = n * quo
4. rem = m – prod
5. Print rem
6. Stop

3 Constructs of Structured Programming

- Sequence
  - An algorithm, and eventually a program is a sequence of instructions.
- Selection
  - Some problems cannot be solved with only a sequence of simple instructions. Sometimes we need to test a condition and follow different sequence of instructions.
  - This is called the decision (selection or branching) construct.
- Iteration or Repetition or Looping

Iteration

- Iteration is defined as the act or process of repeating.
  - For example, iteration can include repetition of a sequence of operations in order to get closer to a desired result.
- Iteration can also refer to a process wherein a computer program is instructed to perform a process over and over again repeatedly for a specific number of times or until a specific condition has been met.

- Write an algorithm to find sum of first n natural numbers

Algorithm

1. Accept the value of n
2. Initialize
   sum = 0
   i = 1
3. sum = sum + i
4. i = i + 1
5. if i ≤ n then
   goto step(3)
   end if
6. Print sum
7. Stop

Algorithm

1. Accept value for n
2. Initialize

   sum = 0

   i = 1
3. while i ≤ n do

   sum = sum + i

   i = i + 1

   end while
4. Print sum
5. Stop

Example

- Write an algorithm to multiply two numbers using repeated addition.

Algorithm

1. Accept two numbers into m and n
2. Initialize

   prod= 0

   i = 1
3. while i ≤ n do

   prod = prod + m

   i = i + 1

   end while
4. Print prod
5. Stop

Example

- Write an algorithm to find the factorial of a number entered by user.

Algorithm

1. Accept the value of n
2. Initialize

   factorial = 1

   i = 1
3. factorial = factorial * i
4. i = i + 1
5. if i ≤ n then

   goto step (3)

   end if
6. Print factorial
7. Stop

Algorithm

1. Initialize
   factorial = 1
   i = 1
2. Accept the value of n
3. while i ≤ n do
   factorial = factorial * i
   i = i + 1
   end while
4. Print factorial
5. Stop

Example

- Write an algorithm to print the numbers between 1 and 100 that are divisible by 3.

Algorithm

1. Initialize

   i = 1
2. while i ≤ 100 do

   if i mod 3 is equal to 0 then

   Print i

   end if

   i = i + 1

   end while
3. Stop

Algorithm

1. Accept m, n
2. Initialize

   i = m
3. while i ≤ n do

   if i mod 3 is equal to 0 then

   Print i

   end if

   i = i + 1

   end while
4. Stop

Example

- Write an algorithm to print all the factors of a given number.

1. Accept n
2. Initialize

   i = 1
3. while i <= n do

   if n mod i equals to 0 then

   Print i

   end if

   i = i + 1

   end while
4. Stop

Example

- Write an algorithm to compute GCD of two numbers.

Algorithm

1. Accept two numbers into m and n.
2. rem = m mod n
3. if rem ≠ 0 then

   m = n

   n = rem

   goto step(2)

   end if
4. Print n
5. Stop

Euclid's Algorithm

1. Accept two numbers into m and n.
2. rem = m mod n
3. while rem ≠ 0 do

   m = n

   n = rem

   rem = m mod n

   end while
4. gcd = n
5. lcm = (m * n) /gcd
6. print gcd
7. print lcm
8. Stop

1. Accept two numbers into a and b.
2. while a ≠ b do
    if a > b then
        a = a – b
    else
        b = b – a
    end if
   end while
3. gcd = a
4. Print gcd
5. Stop

IMPORTANT

Algorithm is not the computer program. Algorithm is just the sequence of instructions which gives a clear idea to you to write the computer program.