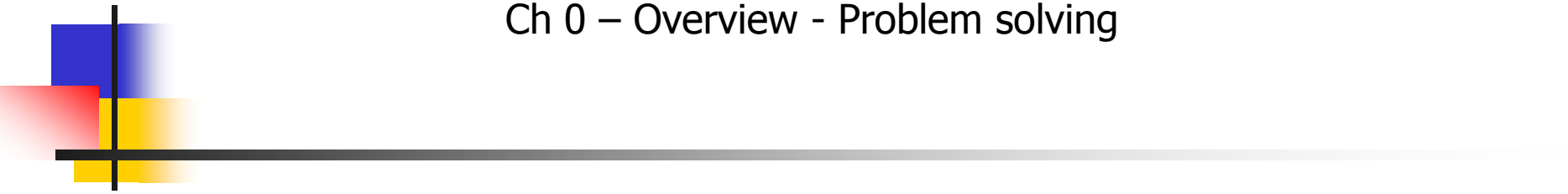
CS 1713
Introduction to Computer Programming II

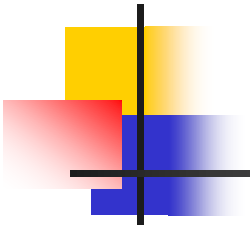
Ch 0 – Overview - Problem solving



Turgay Korkmaz

Office: SB 4.01.13
Phone: (210) 458-7346
Fax: (210) 458-4437
e-mail: korkmaz@cs.utsa.edu
web: www.cs.utsa.edu/~korkmaz

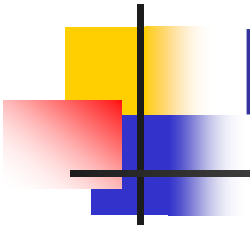
What is the goal of a programmer?



Solve problems

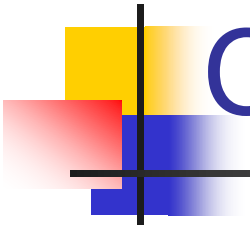
using

computing systems



Problem Solving

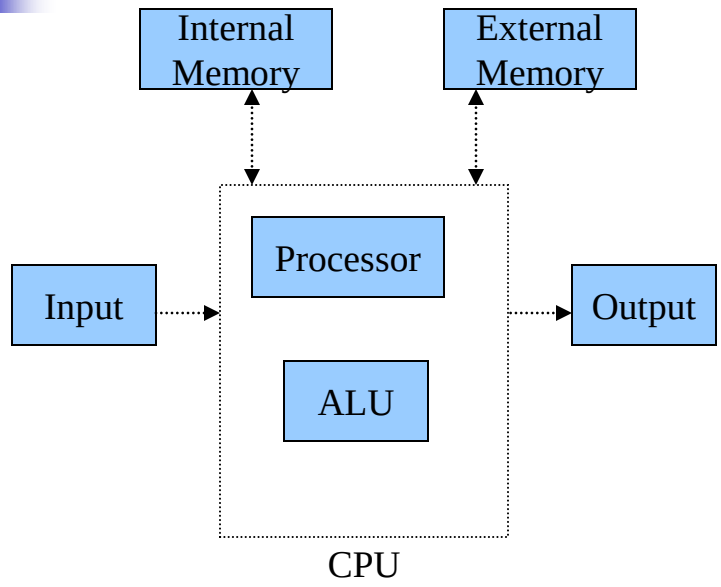
- Main part of problem solving is to figure out
  - **Algorithms** (necessary steps/instructions and their orders) and
  - **Appropriate data structures**
- Then to **code** the algorithm and the data structure in some **programming language** (we will use C)
- Computers cannot think or develop a solution! You do!
  - Computers just **follow your instructions** and do the operations faster
  - Then how do computers do many things? Even play a game, for example chess!
- For the same problem, we may come up with different and yet correct solutions. Efficiency vs. Cost



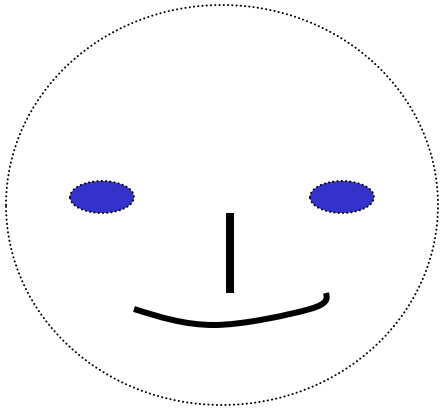
Computing System

- **Computer**: a machine that is designed to perform operations (set of instructions called *program*) to achieve a specific task (e.g., 3+4)
  - **Hardware**: computer equipment (e.g., computer, keyboard, mouse, terminal, hard disk, printer)
  - **Software**: programs that describe the steps we want the computer to perform.

Computer Hardware



- CPU - Central processing unit
- ALU - Arithmetic and logic unit
- ROM - Read only memory
- RAM - Random access memory

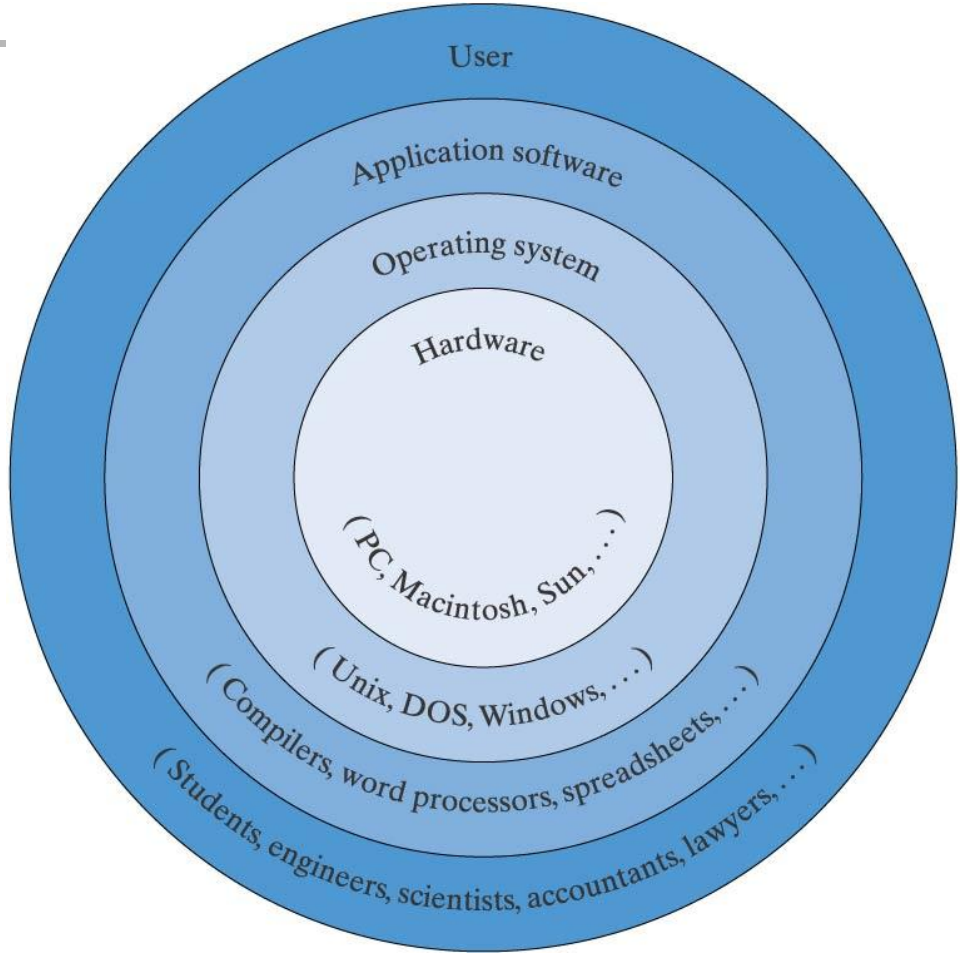


In this sense, do you think we are like a computer?

+ we have intelligence

Computer Software

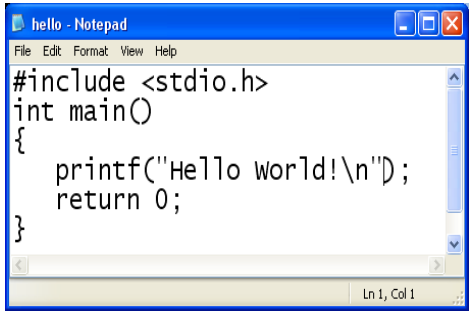
- Operating System - Provides an interface with the user
  - unix, windows, linux, ...
- Software Tools
  - word processors (MicrosoftWord, WordPerfect, ...)
  - spreadsheet programs (Excel, Lotus1-2-3, ...)
  - mathematical computation tools (MATLAB, Mathematica, ...)
- Computer Languages
  - machine language
  - assembly language
  - binary language
  - high level languages (**C**, C++, Ada, Fortran, Basic, java)
- WE WILL STUDY C PROGRAMMING LANGUAGE



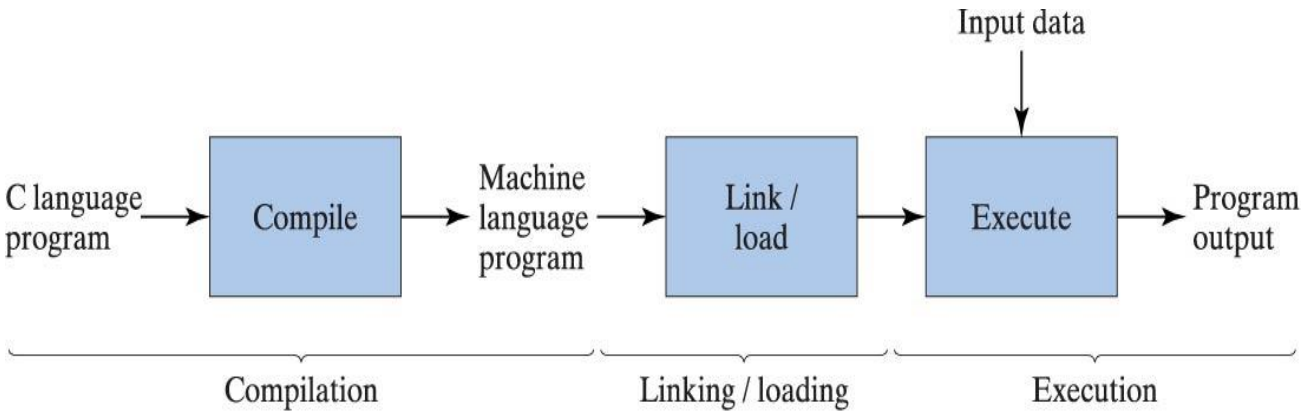
abstractions

What is C?

- General purpose, machine-independent, high-level programming language
- Developed at Bell Labs in 1972 by Dennis Ritchie
- American National Standards Institute (ANSI) approved ANSI C standard in 1989



Source file



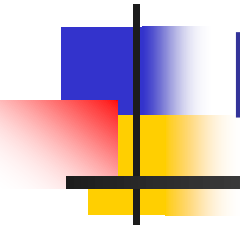
Hello World! in Linux

- Login to a linux machine
  - SSH Secure Shell (e.g., elk03.cs.utsa.edu)

```
elk03 :> mkdir myprog
elk03 :> cd myprog
elk03 :> pico hello.c
```

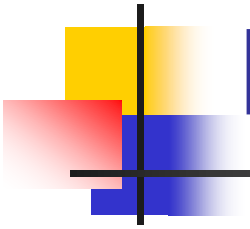
- Type your program … and save it (ctrl-o)
- Compile and execute your program

```
elk03 :> gcc hello.c –o hello
elk03 :> hello
```



PROBLEM SOLVING

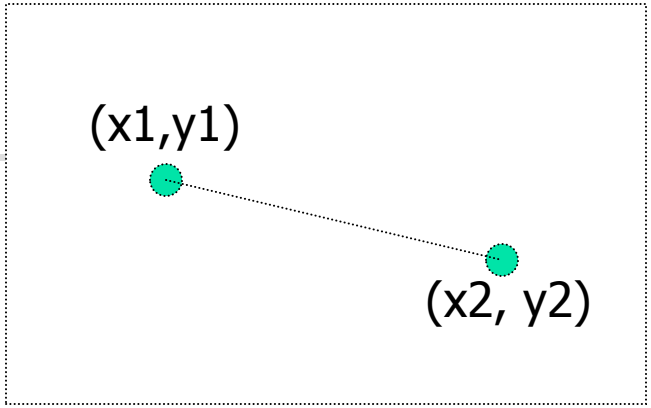
- Very Important
- If you can develop solution, then coding in C is easy…
- So, before studying C, let us C a few examples of problem solving



Problem Solving Methodology

1. **State the problem clearly**
2. **Describe the input/output information**
3. **Work the problem by hand, give example**
4. **Develop a solution** (Algorithm Development) **and Convert it to a program** (C program)
5. **Test the solution with a variety of data**

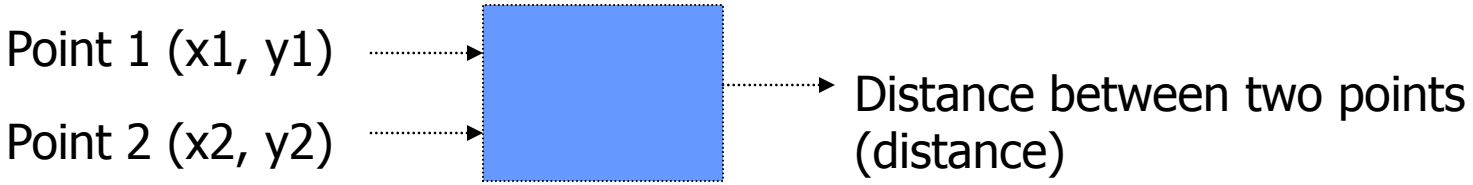
Example 1



1. Problem statement

Compute the straight line distance between two points in a plane

2. Input/output description



Example 1 (cont'd)

3. Hand example

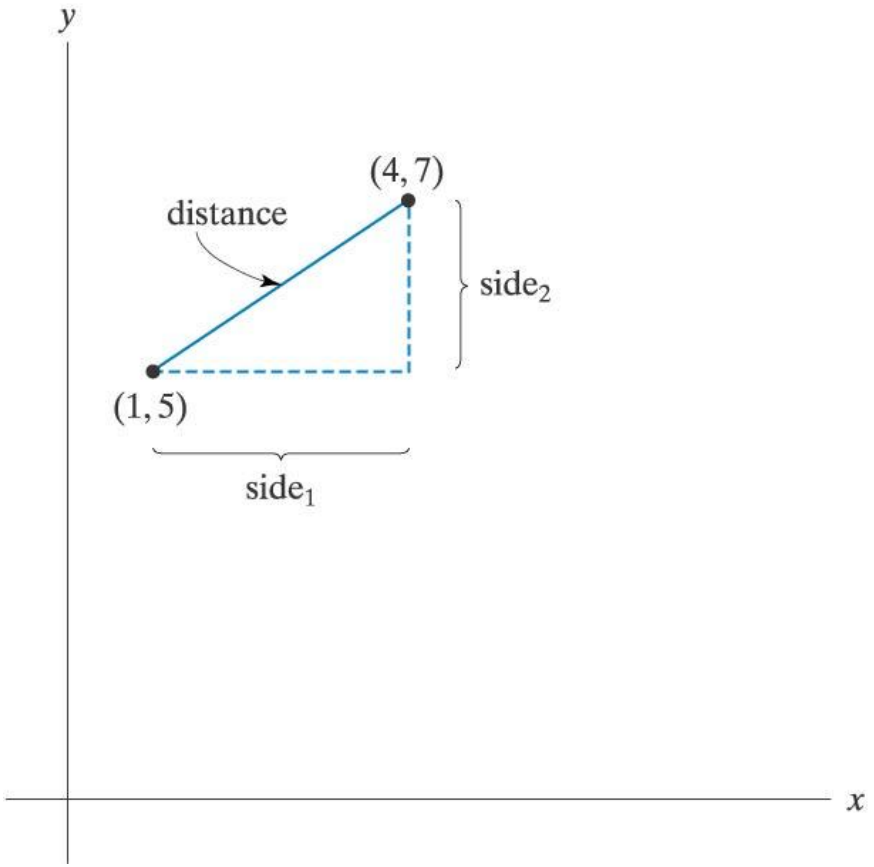
side1 = 4 - 1 = 3

side2 = 7 - 5 = 2

$$\text{distance} = \sqrt{\text{side1}^2 + \text{side2}^2}$$

$$\text{distance} = \sqrt{3^2 + 2^2}$$

$$\text{distance} = \sqrt{13} = 3.61$$



Example 1 (cont'd)

4. Algorithm development and coding

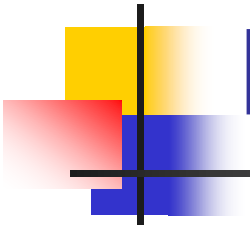
a. Generalize the hand solution and list/outline the necessary operations step-by-step

1) Give specific values for point1 (x1, y1) and point2 (x2, y2)
2) Compute side1=x2-x1 and side2=y2-y1
3) Compute $\text{distance} = \sqrt{\text{side1}^2 + \text{side2}^2}$
4) Print distance

b. Convert the above outlined solution to a program using any language you want (see next slide for C imp.)

Example 1 (cont'd)

```
/*-------------------------------------------------------------*/
/*  Program chapter1_1                                         */
/*                                                             */
/*  This program computes the                                  */
/*  distance between two points.                               */

#include <stdio.h>
#include <math.h>

int main(void)
{
   /*  Declare and initialize variables.  */
   double x1=1, y1=5, x2=4, y2=7,
          side_1, side_2, distance;

   /*  Compute sides of a right triangle.  */
   side_1 = x2 - x1;
   side_2 = y2 - y1;
   distance = sqrt(side_1*side_1 + side_2*side_2);

   /*  Print distance.  */
   printf("The distance between the two points is "
          "%5.2f \n",distance);

   /*  Exit program.   */
   return 0;
}
/*-------------------------------------------------------------*/
```



Example 1 (cont'd)

5. Testing

- After compiling your program, run it and see if it gives the correct result.
- Your program should print out

```
The distance between two points is 3.61
```

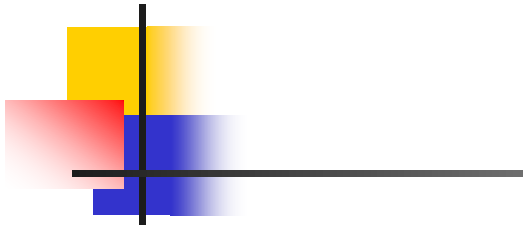
- If not, what will you do?

Modification to Example 1

How will you find the distance between two other points (2,5) and (10,8)?

```
/*-------------------------------------------------------------*/
/*  Program chapter1_1                                         */
/*                                                             */
/*  This program computes the                                  */
/*  distance between two points.                               */

#include <stdio.h>
#include <math.h>

int main(void)
{
   /*  Declare and initialize variables.  */
   double x1=1, y1=5, x2=4, y2=7,
          side_1, side_2, distance;

   /*  Compute sides of a right triangle.  */
   side_1 = x2 - x1;
   side_2 = y2 - y1;
   distance = sqrt(side_1*side_1 + side_2*side_2);

   /*  Print distance.  */
   printf("The distance between the two points is "
          "%5.2f \n",distance);

   /*  Exit program.   */
   return 0;
}
/*-------------------------------------------------------------*/
```

x1=2, y1=5, x2=10, y2=8,

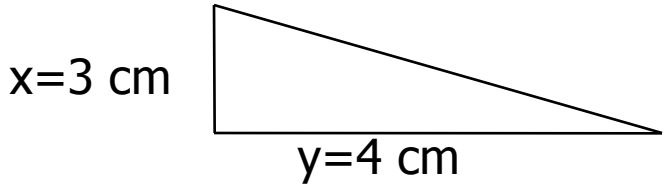


Simple examples to develop solutions

Compute the area of a triangle

1. State problem
2. I/O
3. Hand example
4. Develop solution

   and Coding
5. Testing



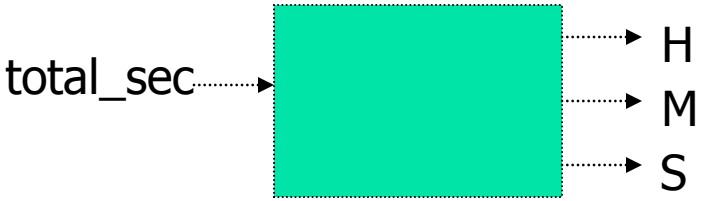
area = ½ * 3 *4 = 6 cm^2

1. Get values of x and y
2. Compute area = ½*x*y
3. Print area

Given the number of seconds, find number of hours, minutes and seconds

1. State problem
2. I/O
3. Hand example
4. Develop solution

   and Coding
5. Testing



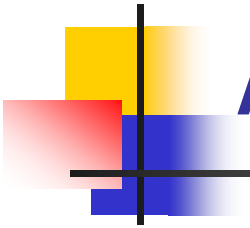
3675 seconds can be written as
1 hour 1 min 15 sec

1. Get total_sec
2. H = total_sec / 3600 (integer division)
3. M = (total_sec – (H*3600)) / 60

   M = (total_sec mod 3600) / 60
4. S = total_sec – (H*3600) – (M*60)
5. Print H hour, M min, S sec

A little bit difficult examples to develop solutions

Some problems are from **How to Solve it: Modern Heuristics** by Michalewicz and Fogel. Springer 2004.

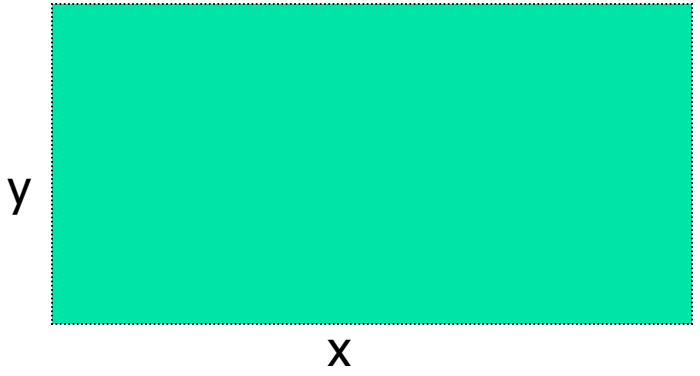


Average speed

- Suppose a car goes from city A to city B with speed of 40 mph and immediately comes back with the speed of 60 mph.
- What is the average speed?
- Can you generalize this solution and outline step by step to find average speed when the speed from A to B is X and the speed from B to A is Y?

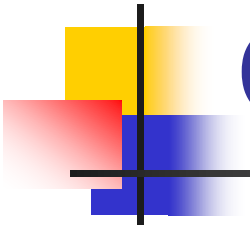
Dimensions of a rectangle ranch?

- A farmer has a rectangular ranch with a perimeter of P=110 meters and an area of A=200 square meters.
- What are the dimensions of his ranch?

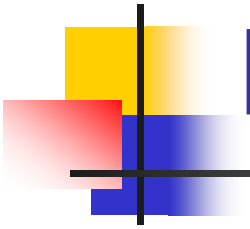


- What are the dimensions for any P and A?

Climbing a wooden post



- A snail is climbing a wooden post that is H=10 meters high.
- During the day, it climbs U=5 meters up.
- During the night, it falls asleep and slides down D=4 meters.
- How many days will it take the snail to climb the top of the post?
- Given that H > U > D. Can you generalize your solution for any H, U, and D?



Minimum number of coins

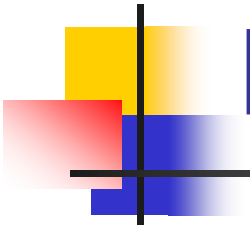
- Suppose you want to give x=67 cents to a person, what is the minimum number of coins
- You have many 25, 10, 5, 1 cents

Assign letter grades

- Suppose I have your grades as follows

name	final	midterm	avg_hw	quizze	letter
aaaa	30	20	30	4	?
bbbb	20	15	40	10	?
…					

How can I assign letter grades?



Example: Sum of numbers

Given n (for example n=1000), compute

- sum = 1+2+3+…+n
- sum_odd =1+3+5+7+…+(2n+1)
- ln2=1- 1/2 + 1/3 - 1/4 +… ± 1/n

Ten heuristics for problem solving

How to Solve it: Modern Heuristics by Michalewicz and Fogel. Springer 2004.

1. Don’t rush to give an answer, think about it
2. Concentrate on the essentials and don’t worry about the noise (tiny details)
3. Sometimes finding a solution can be really easy (common sense), don’t make it harder on yourself
4. Beware of obvious solutions. They might be wrong
5. Don’t be misled by previous experience

Ten heuristics for problem solving

How to Solve it: Modern Heuristics by Michalewicz and Fogel. Springer 2004.

6. Start solving. Don't say "I don't know how"
   - Most people don't plan to fail, they just fail to plan!
7. Don't limit yourself to the search space that is defined by the problem. Expand your horizon
8. Constraints can be helpful to focus on the problem at the hand
9. Don't be satisfied with finding **a solution**, look for better ones
10. Be patient. Be persistent!