Programming and Data Structure

Sujoy Ghose

Sudeshna Sarkar

Jayanta Mukhopadhyay

Dept. of Computer Science & Engineering.

Indian Institute of Technology

Kharagpur

Course objectives

- Learning to program in C
- Learning to program
- Problem solving using programming

Course objectives

- Learning to program in C
  - Programs should be grammatically correct

    (Learn C syntax)
  - Good programming habits

    (To prepare yourself to write and debug large software)
- Problem solving
  - You write a program to make the computer carry out the steps identified to solve a problem
  - Given a problem, find the solution (algorithm)

Problem 1

Coin balance puzzle: You are given 1000 coins. One of them is heavier than the others. You have a balance scale. Identify the heavier coin with minimum number of weighings.

Problem 2

Egg dropping puzzle

Suppose that we wish to know which stories in a 36-story building are safe to drop eggs from, and which will cause the eggs to break on landing.

How to solve the problem if you have

1. One egg?
2. Two eggs?
3. k eggs?

Problem 2

Egg dropping puzzle

We need to make a few assumptions: An egg that survives a fall can be used again.

- A broken egg must be discarded.
- The effect of a fall is the same for all eggs.
- If an egg breaks when dropped, then it would break if dropped from a higher window.
- If an egg survives a fall then it would survive a shorter fall.
- It is not ruled out that the first-floor windows break eggs, nor is it ruled out that the 36th-floor windows do not cause an egg to break.

Egg dropping puzzle solution
one egg

Solution

- Drop the egg from the first-floor window
- if it survives, drop it from the second floor window.
- Continue upward until it breaks.

In the worst case, this method may require 36 droppings.

Egg dropping puzzle solution one egg

Solution (Algorithm)

1. Curr_floor = 1
2. While egg is intact and curr_floor<=36, repeat
   a) Drop the egg from the window of curr_floor
   b) if egg survives,
      Set curr_floor = curr_floor + 1
3. Return curr_floor

In the worst case, this method may require 36 droppings.

Egg dropping puzzle solution
two eggs

- For you to solve

Problem 3

Count the number of students in this class

Basic Programming Concepts

What is a program?

- A **program** is a sequence of instructions that specifies how to perform a computation.
- The computation might be something mathematical
  - solving a system of equations
  - finding the roots of a polynomial,
- but it can also be a symbolic computation,
  - searching and replacing text in a document
  - compiling a program

Basic programming instructions

A few basic instructions appear in almost any programming language:

- **input** Get data from the keyboard/ file / some other device.
- **output** Display data on the screen or send data to a file or other device.
- **math** Perform basic mathematical operations like addition and multiplication.
- **conditional execution** Check for certain conditions and execute the appropriate sequence of statements.
- **repetition** Perform some action repeatedly, usually with some variation.

Formal and natural languages

- **Natural languages:** languages that people speak,
  - not designed by people
  - they evolved naturally.
- **Formal languages:** languages designed by people for specific applications.
  - the notation that mathematicians use is a formal language that is particularly good at denoting relationships among numbers and symbols.
  - **Programming languages are formal languages that have been designed to express computations.**

Values and types

- A **value** is one of the fundamental things - like a letter or a number — that a program manipulates.
  - 511
  - 47.51
  - ``hello class”
- These values belong to different **types**
  - int
  - float
  - Array of char (string)

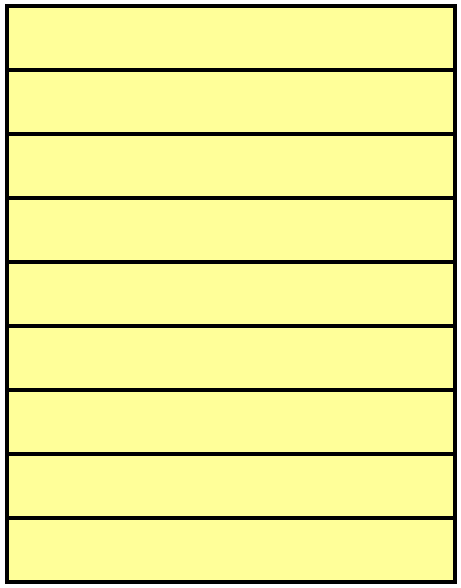
Variables

- One of the most powerful features for problem solving using computers is the ability to manipulate variables.
  - All temporary results are stored in terms of variables
  - Variables are stored in main memory.
  - A variable stores a value.

Contd.

- How does memory look like (logically)?
  - As a list of storage locations, each having a unique address.
  - Variables and constants are stored in these storage locations.
  - Variable is like a *house*, and the name of a variable is like the *address* of the house.
    - Different people may reside in the house, which is like the *contents* of a variable.

Memory map

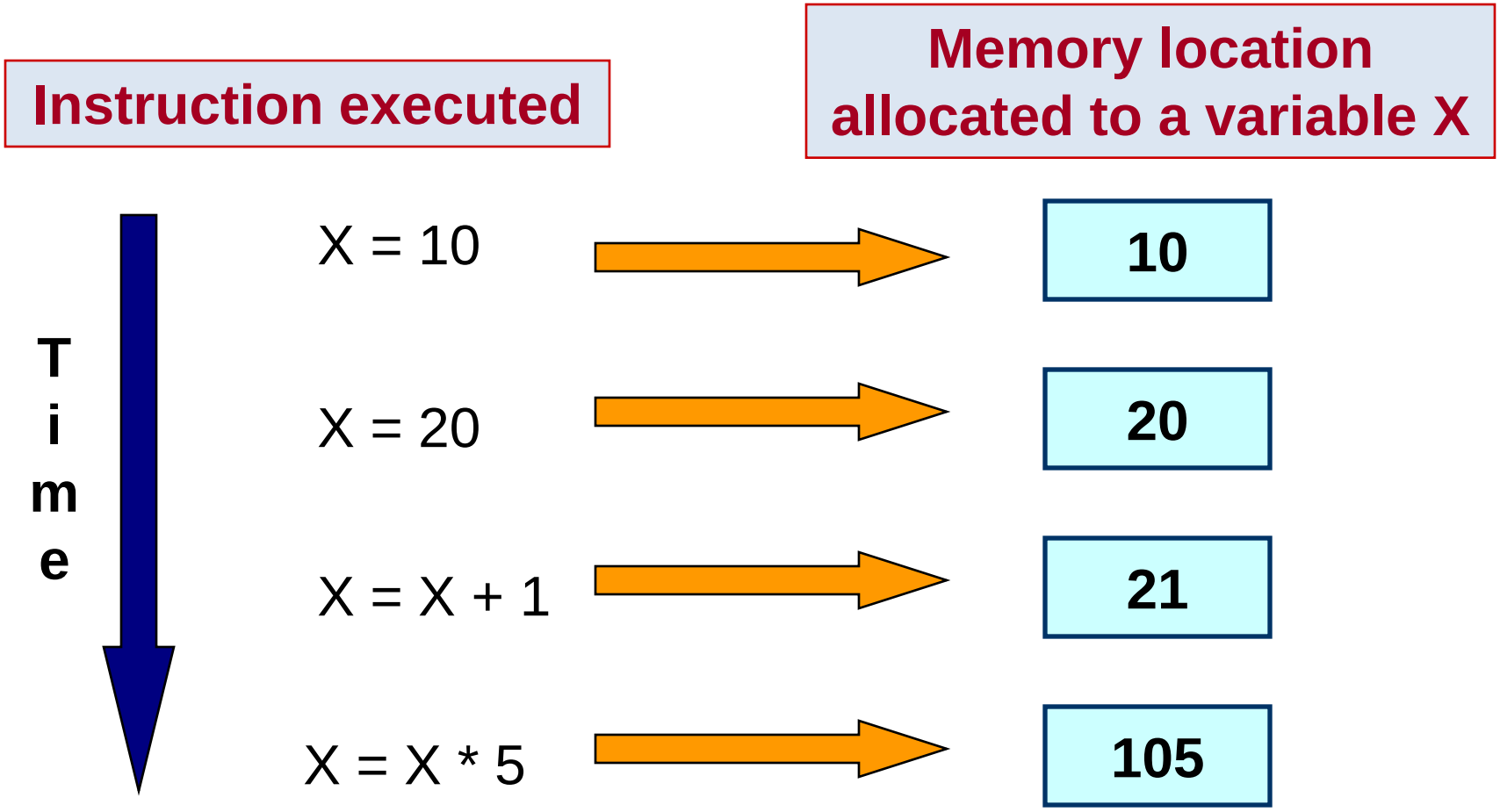


Address 0
Address 1
Address 2
Address 3
Address 4
Address 5
Address 6

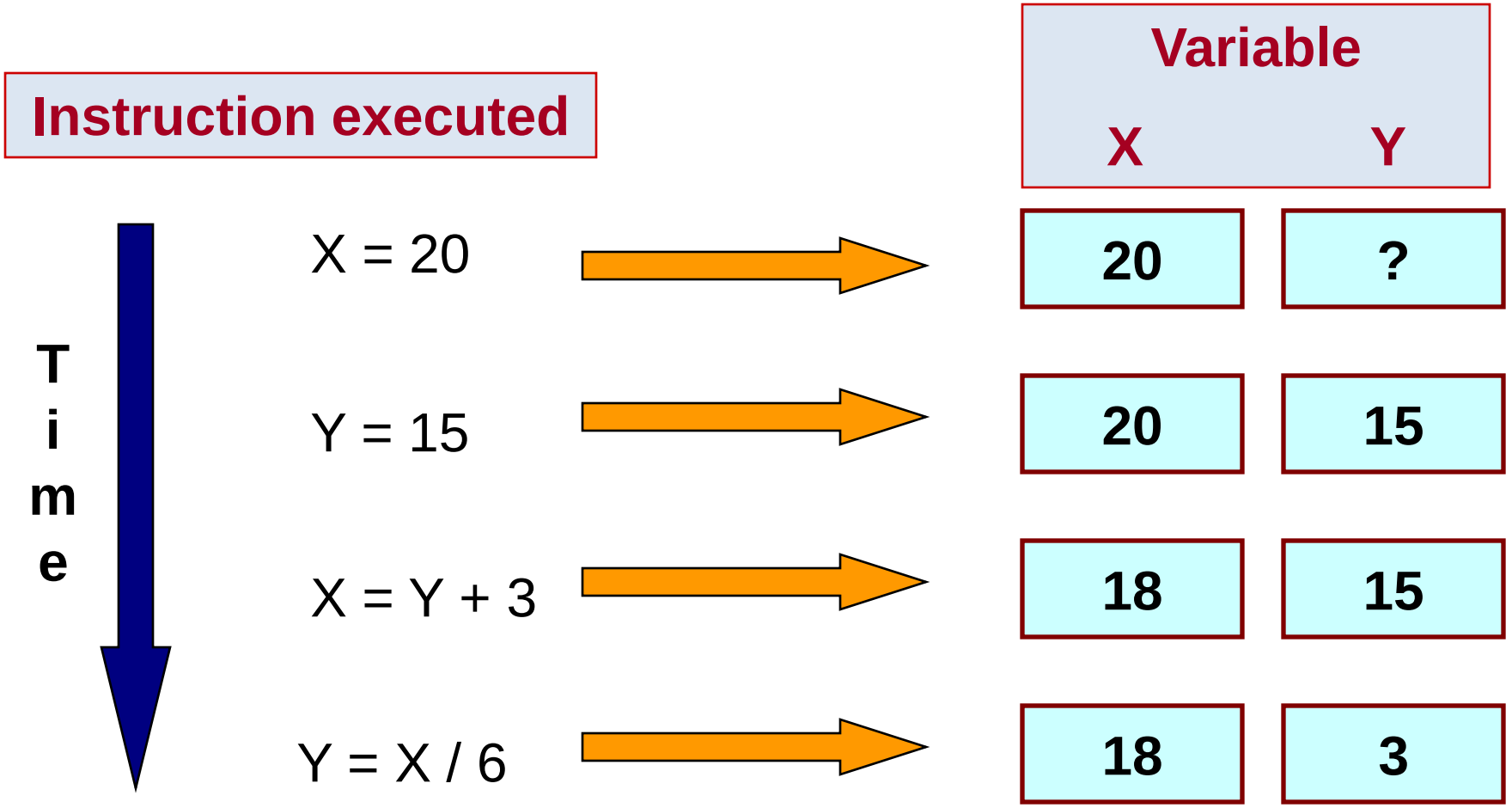
Address N-1

Every variable is mapped to a particular memory address

Variables in Memory



Variables in Memory (contd.)



Data types

- Three common data types used:
  - Integer :: can store only whole numbers
    - Examples: 25, -56, 1, 0
  - Floating-point :: can store numbers with fractional values.
    - Examples: 3.14159, 5.0, -12345.345
  - Character :: can store a character
    - Examples: ‘A’, ‘a’, ‘*’, ‘3’, ‘ ’, ‘+’

Data Types (contd.)

- How are they stored in memory?
  - Integer ::
    - 16 bits
    - 32 bits
  - Float ::
    - 32 bits
    - 64 bits
  - Char ::
    - 8 bits (ASCII code)
    - 16 bits (UNICODE, used in Java)

Actual number of bits varies from one computer to another

Problem solving

- Step 1:
  - Clearly specify the problem to be solved.
- Step 2:
  - Design an algorithm to solve the problem.
- Step 3:
  - Convert algorithm into program code.
- Step 4:
  - Compile the program into object code.
- Step 5:
  - Execute the program.

Our First Program:
Convert Centigrade to Fahrenheit

```
#include <stdio.h>
// this program takes a centigrade value as input and converts it into fahrenheit
int main( )  {
        float C;
        float F;
        printf (“Input in centigrades\n”) ;
        scanf (“%f”, &C);
        F=C*9/5+32;
        printf (Fahrenheit value = %f\n”, F) ;
        return 0;
}
```

Our First Program:
Convert Centigrade to Fahrenheit

```
#include <stdio.h>
// this program takes a centigrade value as input and converts
// into fahrenheit
int main( ) {
        float C;
        float F;
        printf (“Input in centigrades\n”) ;
        scanf (“%f”, &C);
        F=C*9/5+32;
        printf (Fahrenheit value = %f\n”, F) ;
        return 0;
}
```

- The **include ..** command specifies what family of commands the program will use.
- Lines beginning with **//** are ignored by the compiler.

  But serve to make the code readable.

Our First Program:
Convert Centigrade to Fahrenheit

```
#include <stdio.h>
// this program takes a centigrade value as input and converts it into fahrenheit
int main( ) {
        float C;
        float F;
        printf ("Input in centigrades\n") ;
        scanf ("%f", &C);
        F=C*9/5+32;
        printf (Fahrenheit value = %f\n", F) ;
        return 0;
}
```

- Every C program must have the int main ( ) and the braces { … }

Our First Program:
Convert Centigrade to Fahrenheit

```
#include <stdio.h>
// this program takes a centigrade value as input and converts it into fahrenheit
int main( ) {
        float C;
        float F;
        printf ("Input in centigrades\n") ;
        scanf ("%f", &C);
        F=C*9/5+32;
        printf (Fahrenheit value = %f\n", F) ;
        return 0;
}
```

- Declarations: This tells us that there are two variables C and F. Both of them are floating point real numbers.
- Such statements are called the Declarations since they declare the type of the variables and their names.

Our First Program:
Convert Centigrade to Fahrenheit

```
#include <stdio.h>
// this program takes a centigrade value as input and converts it into fahrenheit
int main( )  {
        float C;
        float F;
        printf (“Input in centigrades\n”) ;
        scanf (“%f”, &C);
        F=C*9/5+32;
        printf (Fahrenheit value = %f\n”, F) ;
        return 0;
}
```

Input and Output: scanf and printf

Our First Program:
Convert Centigrade to Fahrenheit

```
#include <stdio.h>
// this program takes a centigrade value as input and converts it into fahrenheit
int main( )  {
        float C;
        float F;
        printf (“Input in centigrades\n”) ;
        scanf (“%f”, &C);

        F = C*9/5+32;

        printf (Fahrenheit value = %f\n”, F) ;
        return 0;
}
```

- Assignment Statement: Causes the computation on the right to be assigned to the variable location named F.