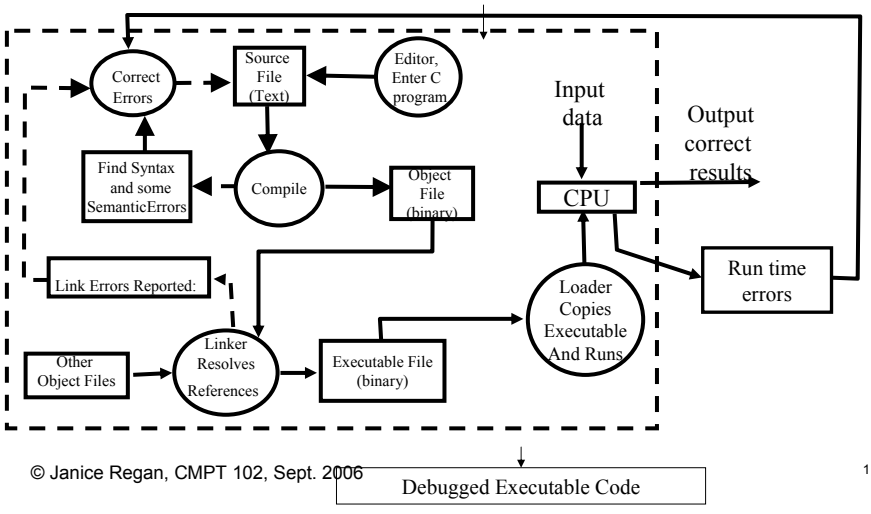
CMPT 102
Introduction to Scientific Computer Programming

The software development method
algorithms

© Janice Regan, CMPT 102, Sept. 2006

Processing a C Program



© Janice Regan, CMPT 102, Sept. 2006

Getting to step 1: writing the program

- Writing and debugging the C program implementing an application is only a small part of designing and building an application.
- ***The software development method***
  - puts writing the program into a larger context.
  - provides a valuable guidelines to help you develop usable and efficient applications
  - Saves time and effort during development of computer applications.

Software Development Method

1. Specify the problem requirements
2. Analyze the problem
3. Design an algorithm to solve the problem and tests to demonstrate the algorithm is working correctly
4. Implement the algorithm in a programming language
5. Test and verify that the implemented algorithm (program) is working correctly
6. Maintain and update the program

Algorithms and Programs

- An ***algorithm*** is a finite set of instructions that is a step-by-step solution for a problem
- A computer can be instructed to implement many (not all) algorithms with a finite number of instructions
- A ***program*** is a set of computer instructions that implements an algorithm
- There are different ways to implement the same algorithm
- There may be many different algorithms to solve a given problem

© Janice Regan, CMPT 102, Sept. 2006

Software Development: 1

- **Specify the problem requirements:**
  - State the problem clearly
  - What does your application need to do?
  - Under what conditions does it need to do it?
  - Do you need more information to specify the problem?
- **Analyze the problem**:
  - Define inputs and outputs
  - Specify what the important quantities are
  - Outline and relationships between important quantities

© Janice Regan, CMPT 102, Sept. 2006

Software Development: 2

- Design an algorithm to solve the problem:
  - Determine a set of steps that will take the inputs and produce the outputs. (an algorithm)
  - Use top down design:
    - Start by identifying the large tasks that need to be done
    - Then, consider each task, and make a more detailed set of steps to accomplish it, this is called ***refining your algorithm***.
    - As you refine your algorithm be sure to consider special cases
  - Design tests to demonstrate that the algorithm "works" in all cases

© Janice Regan, CMPT 102, Sept. 2006

Software Development: 3

- Implement the algorithm in a programming language
  - Use the steps we discussed to write the program
  - Debug until you have an executable program
- Test to verify that the algorithm is working correctly
  - Locate and correct run time errors, and logic errors
  - Use the tests you developed while designing your algorithm to verify the correctness of your algorithm and its implementation
- Maintain and update the program
  - Correct errors found in regular use
  - Add features not anticipated at time of design

© Janice Regan, CMPT 102, Sept. 2006

Why Should I Write Algorithms?

- A computer program solves a scientific problem with a computer
  - a simulation problem
  - a data analysis application
  - a control system
- To write a computer program you need to know the series of steps your are implementing to solve your problem, You need to know your algorithm!

Important for Algorithms and Programs

- The sequence or order of the steps is usually of critical importance in writing a correct algorithm
- You must be exact when specifying an algorithm that is to be translated into a computer program
- Example: What is the difference between A*(B+C) and (A*B)+C?
  - Be careful, the computer will do exactly what you ask, even it is not what you really want it to do!!

Software Development: Example

- Initial problem statement
  - Think of the "big picture" description of the problem
  - For our example: Find the intersection of two lines
- Begin analysis of the problem
  - First we need to know what is the input to the problem and what is the desired result
    - Input: two lines
    - Output:
      - the points where the two lines intersect
      - A message saying the lines do not intersect

Choose a representation: 1

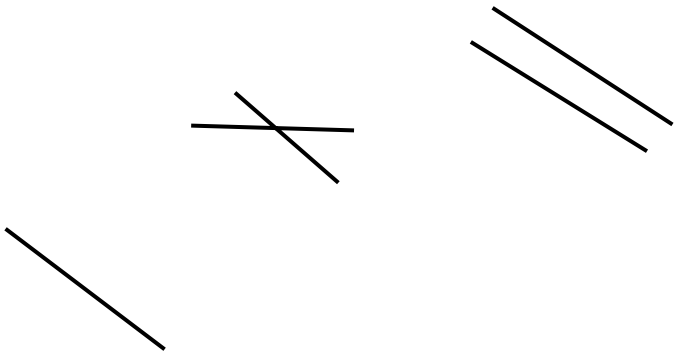
- Must represent the two lines and the intersection points in a consistent way.
  - Use a mathematical coordinate system in which points and lines can be easily represented. What are our choices?
    - Cartesian x,y:
    - Polar r, Θ
  - Are there particular reasons to choose one of these
    - Is one more familiar or easier to use?
    - Does one make it easier to solve the problem?
  - Let's choose Cartesian it's more familiar

Choose a representation: 1

- We must choose a way to express our points and lines
  - In Cartesian Coordinates a point is (x, y)
  - What about an equation for a line
    - $Ax + By + C = 0$ Input A, B and C
    - $y = mx + b$ Input m and b
      $m = (y - y_1)/(x - x_1)$
    - $y - y_1 = m(x - x_1)$ Input m and (x, y)
    - $y - y_1 = \frac{(y_2 - y_1)}{(x_2 - x_1)}(x - x_1)$ Input (x_1,y_1) and (x_2,y_2)

© Janice Regan, CMPT 102, Sept. 2006

Special Cases?



© Janice Regan, CMPT 102, Sept. 2006

Defining input and output

- INPUT for each line
  - Two points on the first line (x_{12},y_{12}) and (x_{11}, y_{11})
  - Two points on the second line (x_{22},y_{22}) and (x_{21}, y_{21})
- OUTPUT: the intersection of the two lines
  - Common case: a single point (x,y)
  - Special cases inherent to problem;
    - Parallel lines
      - With no points in common: no intersection
      - With all points in common: the same line

Continuing the Analysis

- What needs to be done to the input to determine the output? Design an algorithm
- Determine a series of steps that will transform the input data into the output results.
  - Do an example of the most common case first!
  - Then try examples of any special cases
  - If necessary modify or redesign your series of steps so it can handle all special cases

The Common Case

- Use an example and step through the problem by hand to help you find the algorithm
- Choose a set of example data to illustrate the common case (2 lines intersecting in 1 point)
  - line 1: $(x_{11},y_{11}) = (3,7)$, $(x_{12},y_{12}) = (1,4)$
    line 2: $(x_{21},y_{21}) = (4,4)$, $(x_{22},y_{22}) = (2,3)$
- Next:
  - Consider how we could solve the problem by hand

Developing an Algorithm

- Calculate the expected results for each test case by hand
  - You should know intermediate results so you can debug your code later by checking against your hand calculations
  - You can use your hand calculation to determine what the steps are to transform the inputs into the outputs

Solution by hand: 1

- Substitute the slope and point into the equation for each line
  - $(x_{11},y_{11}) = (3,7)$, $(x_{12},y_{12}) = (1,4)$

    $y-7=[(4-7)/(1-3)]*(x-3)=1.5x-4.5 \qquad y-1.5x-2.5=0$
  - $(x_{21},y_{21}) = (4,4)$, $(x_{22},y_{22}) = (2,3)$

    $y-4=\{(3-4)/(2-4)\}*(x-4)=0.5x-2 \qquad y-0.5x-2=0$
- Equate the two resulting equations

$$y-1.5x-2.5=y-0.5x-2$$

© Janice Regan, CMPT 102, Sept. 2006

Solution by hand: 2

- Determine the value of the x coordinate of the intersection point

$$-x-0.5=0 \qquad x=-0.5$$

- Determine the value of the y coordinate of the intersection point

$$x=-0.5 \quad y-1.5(-0.5)-2.5=0 \quad y=7/4$$

© Janice Regan, CMPT 102, Sept. 2006

Generalize hand example: 1

- Substitute the slope and point into the point slope equation for each line
  - Points on line 1: (x_{11},y_{11}) (x_{12},y_{12}) slope m_1
  - Points on line 2: (x_{21},y_{21}) (x_{22},y_{22}) slope m_2

$$y - y_{21} - \frac{(y_{22} - y_{21})}{(x_{22} - x_{21})}(x - x_{21}) = m_2(x - x_{21}) = 0$$

$$y - y_{11} - \frac{(y_{12} - y_{11})}{(x_{12} - x_{11})}(x - x_{11}) = m_1(x - x_{11}) = 0$$

© Janice Regan, CMPT 102, Sept. 2006

Generalize hand example: 2

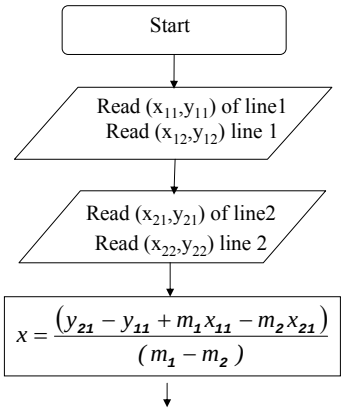
- Equate the two resulting equations and determine the value of the coordinates of the intersection point

$$x = \frac{(y_{21} - y_{11} + m_1 x_{11} - m_2 x_{21})}{(m_1 - m_2)}$$

$$y = y_{21} + m_2(x - x_{21})$$

© Janice Regan, CMPT 102, Sept. 2006

Flow chart (1): common case



Read (x_{11},y_{11}) for line 1

Read (x_{12},y_{12}) for line 1

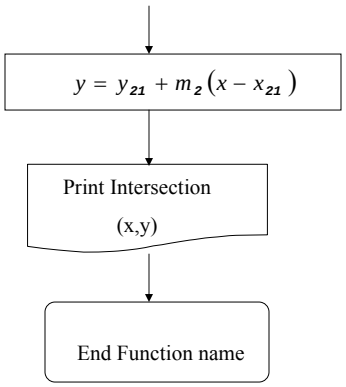
Read (x_{21},y_{21}) for line 2

Read (x_{22},y_{22}) for line 2

Find x value of intersection point

© Janice Regan, CMPT 102, Sept. 2006

Flow chart (2): common case



Find y value of intersection point

Print Intersection (x,y)

End Function name:

© Janice Regan, CMPT 102, Sept. 2006

Algorithm

- Read (x11,y11) for line 1
- Read (x12,y12) for line 1
- Read (x21,y21) for line 2
- Read (x22,y22) for line 2
- Find x value of intersection point
- Find y value of intersection point
- Print Intersection point (x, y)

Does this always work?

- No, to determine the value of x we are using the following equation

$$x = \frac{(y_{21} - y_{11} + m_1 x_{11} - m_2 x_{21})}{(m_1 - m_2)}$$

  - For parallel lines the slopes are equal so $m_1 = m_2$
  - If we use the equation above $m_1 - m_2 = 0$, we are dividing by 0
  - We can determine if the lines are parallel by comparing the slopes
    - Parallel lines can be coincident
    - Parallel lines that are not coincident never intersect
    - How do we tell the difference in our algorithm?

Hand examples: parallel lines

- For parallel lines

$$0 = y_{21} - y_{11} + m(x_{11} - x_{21}) \qquad or \quad m_1 = m_2 = m = \frac{(y_{11} - y_{21})}{(x_{11} - x_{21})}$$

  - So if two points, each chosen from one of the two lines form a line with the same slope as each line, the lines are coincident
  - Otherwise the lines do not intersect
- In both cases we skip the steps to determine the x and y coordinates
- The desired output is to print the message "two identical lines" or "no intersection"

Does this algorithm always work?

- No, to determine the value of x we are using the following equation

$$x = \frac{(y_{21} - y_{11} + m_1 x_{11} - m_2 x_{21})}{(m_1 - m_2)}$$

  - For parallel lines the slopes are equal so $m_1 = m_2$
  - If we use the equation above $m_1 - m_2 = 0$, we are dividing by 0
  - We can determine if the lines are parallel by comparing the slopes
    - Parallel lines can be coincident
    - Parallel lines that are not coincident never intersect
    - How do we tell the difference in our algorithm?

Example 1 Using Parallel Lines

- Starting with the two lines
  - $(x_{11},y_{11}) = (6,6)$, $(x_{12},y_{12}) = (3,3)$
    $(x_{21},y_{21}) = (4,5)$, $(x_{22},y_{22}) = (3,4)$
    - Slope of lines 1 and 2 show lines are parallel

$$m_1 = \frac{(3-6)}{(3-6)} = 1 \qquad m_2 = \frac{(4-5)}{(3-4)} = 1$$

    - Slope of line with one point from each line

$$m = \frac{(y_{11} - y_{21})}{(x_{11} - x_{21})} = \frac{(6-5)}{(6-4)} = \frac{1}{2} \neq m_1$$

  - The lines do not intersect

© Janice Regan, CMPT 102, Sept. 2006

Example 2 Using Parallel Lines

- Starting with the two lines
  - $(x_{11},y_{11}) = (4,4)$, $(x_{12},y_{12}) = (3,3)$
    $(x_{21},y_{21}) = (1,1)$, $(x_{22},y_{22}) = (5,5)$
    - Slope of lines 1 and 2 show lines are parallel

$$m_1 = \frac{(3-4)}{(3-4)} = 1 \qquad m_2 = \frac{(5-1)}{(5-1)} = 1$$

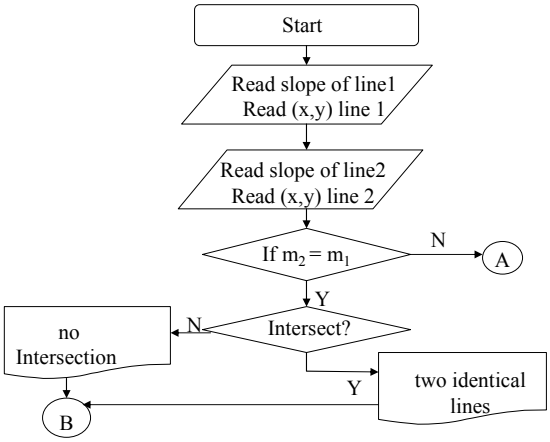
    - Slope of line with one point from each line

$$m = \frac{(y_{11} - y_{21})}{(x_{11} - x_{21})} = \frac{(4-1)}{(4-1)} = 1 = m_1$$

  - The lines are coincident

© Janice Regan, CMPT 102, Sept. 2006

Refined Flow chart (1)



Start Function name:

Read (x_{11},y_{11}) for line 1

Read (x_{12},y_{12}) for line 1

Read (x_{21},y_{21}) for line 2

Read (x_{22},y_{22}) for line 2

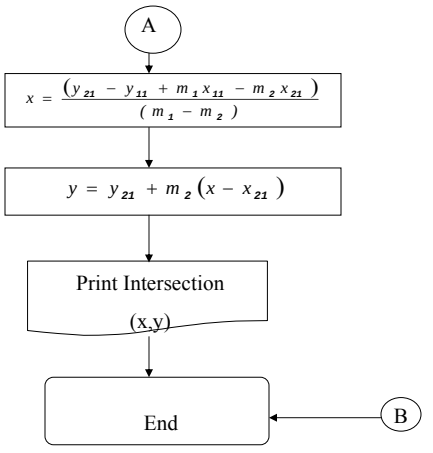
if m2 = m1 the lines are parallel
if lines intersect

$$\left(m_1 = m_2 = m = \frac{(y_{11} - y_{21})}{(x_{11} - x_{21})}\right)$$

Print "two identical lines."
If lines do not intersect
Print "no intersection"

© Janice Regan, CMPT 102, Sept. 2006

Refined Flow chart (2)



If the lines are not parallel ($m2 \neq m1$)

Find the x value of intersection point

Find y value of intersection point

Print Intersection (x,y)

End

© Janice Regan, CMPT 102, Sept. 2006

Refined Algorithm (1)

- Start
- Read x and y values for the first point on line 1, (x_{11}, y_{11})
- Read x and y for the second point on line 1, (x_{12}, y_{12})
- Read x and y values for the first point on line 2, (x_{21}, y_{21})
- Read x and y for the second point on line 2, (x_{22}, y_{22})

© Janice Regan, CMPT 102, Sept. 2006

Refined Algorithm (2)

- If the lines are parallel (*m2* = *m1*)
  - if the lines are coincident (if $\left(m_1 = m_2 = m = \frac{(y_{11} - y_{21})}{(x_{11} - x_{21})}\right)$)
    print the message "two identical lines"
  - if the lines do not intersect
    print the message "no intersection"
- if the lines are not parallel (*m2* ≠ *m1*)
  - Find x value of intersection point
  - Find y value of intersection point
  - Print Intersection point (x, y)
- End

© Janice Regan, CMPT 102, Sept. 2006

Does this refined algorithm always work?

- No, to determine the value of slope we are using the equations

$$m_1 = \frac{(y_{12} - y_{11})}{(x_{12} - x_{11})} \qquad m_2 = \frac{(y_{22} - y_{21})}{(x_{22} - x_{21})}$$

- If $x_{12} - x_{11} = 0$ or $x_{22} - x_{21} = 0$ then the x coordinate of both points defining a line are the same (the line is vertical)
- For vertical lines the slope is infinite so we cannot use any equations containing slope
- This special case is caused by the way we represent the problem in terms of slope

How do we fix it? (1)

- Slope of a vertical line is infinite.
- An infinite value cannot be represented
- A value that cannot be represented cannot be used in an equation
  - We need to compare slopes to determine if the lines are parallel

    Rather that testing if $m_1 = m_2$ test if
    $(x_{22}-x_{21})(y_{12}-y_{11}) = (x_{12}-x_{11})(y_{22}-y_{21})$
  - If the lines are parallel and one is vertical then both are vertical

    If both lines are vertical they are coincident if $x_{11}=x_{21}$

How do we fix it? (2)

- An value that cannot be represented cannot be used in an equation
  - We need to determine the x coordinate of the intersection.

    Since one of the lines is vertical, all points on that line have the same x coordinate. Therefore, the x coordinate of the vertical line is the x coordinate of the intersection.
  - We need to determine the y coordinate of the intersection

    Substitute the x coordinate into the equation of the line that is not vertical to determine the corresponding y coordinate

© Janice Regan, CMPT 102, Sept. 2006

Special Case: One Vertical line

- One vertical line $(x_{21},y_{21}) = (4,2)$, $(x_{22},y_{22}) = (4,5)$

  one non-vertical line $(x_{11},y_{11}) = (1,4)$, $(x_{12},y_{12}) = (3,3)$
  - Any two points on the vertical line will have the same x coordinate
- Consider the equation for the non-vertical line in this case

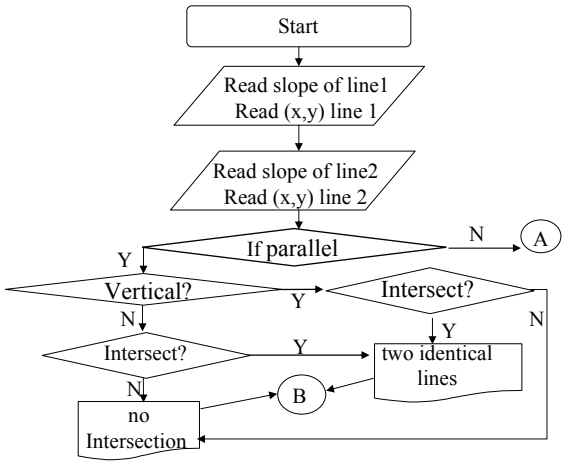
$$y - y_{11} = \frac{(y_{12} - y_{11})}{(x_{12} - x_{11})}(x - x_{11})$$

  - The x coordinate of the intersection point is the x coordinate of any point on the vertical line, say $x = x_{21}=4$
- Determine the value of the y coordinate of the intersection point

$$y = y_{11} + \frac{(y_{12} - y_{11})}{(x_{12} - x_{11})}(x - x_{11}) = 4 + \frac{(3-4)}{(3-1)}(4-1) = 4 + 0.5*3 = 11/2$$

© Janice Regan, CMPT 102, Sept. 2006

Refined Flow chart (1)



Start Function name:

Read (x_{11},y_{11}) for line 1

Read (x_{12},y_{12}) for line 1

Read (x_{21},y_{21}) for line 2

Read (x_{22},y_{22}) for line 2

if the lines are parallel

[if $(x_{22}-x_{21})(y_{12}-y_{11})=(x_{12}-x_{11})(y_{22}-y_{21})$]

If lines vertical and intersect

Print "two identical lines"

if non-vertical lines intersect

Print "two identical lines."

If vertical lines do not intersect

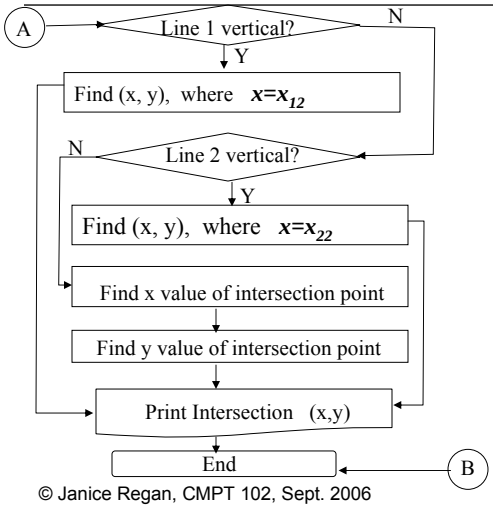
Print "no intersection"

If non- vertical lines do not intersect

Print "no intersection"

© Janice Regan, CMPT 102, Sept. 2006

Refined flowchart (2)



If line 1 is vertical.

Find (x, y), where $x = x_{12}$

. .

If line 2 is vertical

Find (x, y), where $x = x_{12}$

. .

If neither line1 nor line 2 is vertical

Find x value of intersection point

Find y value of intersection point

Print Intersection (x,y)

End

© Janice Regan, CMPT 102, Sept. 2006

Refined Algorithm (1)

- Start
- Read x and y values for the first point on line 1, (x_{11}, y_{11})
- Read x and y for the second point on line 1, (x_{12}, y_{12})
- Read x and y values for the first point on line 2, (x_{21}, y_{21})
- Read x and y for the second point on line 2, (x_{22}, y_{22})

© Janice Regan, CMPT 102, Sept. 2006

Refined Algorithm (2)

- if (x_{22}-x_{21})(y_{12}-y_{11})=(x_{12}-x_{11})(y_{22}-y_{21}) the lines are parallel
  - If (x_{12}=x_{11}) the lines are vertical
    - If (x_{12}=x_{22}) the lines are coincident
      - print the message “two identical lines”
  - If (x_{12}=x_{11}) the lines are vertical
    - If (x_{12}≠x_{22}) the lines do not intersect
      - print the message “no intersection”
  - If (x_{12}≠x_{11}) the lines are not vertical
    - If $(m_1 = m_2 = m = (y_{11} - y_{21})/(x_{11} - x_{21}))$ the lines are coincident
      - print the message “two identical lines”
  - if (x_{12}≠x_{11}) the lines are not vertical
    - If $(m_1 = m_2 \neq m = (y_{11} - y_{21})/(x_{11} - x_{21}))$ the lines do not intersect
      - print the message “no intersection”

© Janice Regan, CMPT 102, Sept. 2006

Refined Algorithm (3)

- if $(x_{22}-x_{21})(y_{12}-y_{11}) \neq (x_{12}-x_{11})(y_{22}-y_{21})$ the lines are not parallel
  - If $x_{12} - x_{11} = 0$ line 1 is vertical, line 2 is not
    - $x = x_{12}$ $\quad y = y_{21} + \frac{(y_{22} - y_{21})}{(x_{22} - x_{21})}(x - x_{21})$
  - if $x_{22} - x_{21} = 0$ line 2 is vertical, line 2 is not
    - $x = x_{22}$ $\quad y = y_{11} + \frac{(y_{12} - y_{11})}{(x_{12} - x_{11})}(x - x_{11})$
  - If $x_{12} - x_{11} \neq 0$ and if $x_{22} - x_{21} \neq 0$ neither line vertical
    - $x = \frac{(y_{21} - y_{11} + m_1 x_{11} - m_2 x_{21})}{(m_1 - m_2)}$
    - $y = -y_1 + m_1 x - m_1 x_1$
  - Print the intersection point (x, y)

Verifying Algorithms

You have written your algorithm, is it ready to be translated into a program?

- Verify that it gives the desired results.
- Verify that all special cases are handled
- Verify that the algorithm ends after the outputs are determined

You have a series of items to verify, you also have made a good start on determining what tests need to be included in your ***test plan***.

Constructing a test plan

- Now we have the information needed to construct our test plan
- A test plan is a list of tests that help us determine when our algorithm (or program) has met the requirements.
- For our example the test plan would consist of a list of sets of input data along with correct answers (output values)
- The sets of data include at least one for the common case and one for each of the special cases that have been identified.
- To minimize work use the examples we used for hand calculation

Test Plan (1)

- Test the common case
  - Set 1: $(x_{11},y_{11}) = (3,7)$, $(x_{12},y_{12}) = (1,4)$ and $(x_{21},y_{21}) = (4,4)$, $(x_{22},y_{22}) = (2,3)$
  - Set 1 should give result of (-1/2 , 7/4)
- Test for the nonintersecting parallel line case
  - $(x_{11},y_{11}) = (6,6)$, $(x_{12},y_{12}) = (3,3)$ and $(x_{21},y_{21}) = (4,5)$, $(x_{22},y_{22}) = (3,4)$
  - Should give a result of 'no intersection'
- Test for the case of coincident parallel lines
  - $(x_{11},y_{11}) = (4,4)$, $(x_{12},y_{12}) = (3,3)$ and $(x_{21},y_{21}) = (1,1)$, $(x_{22},y_{22}) = (5,5)$
  - Should give a result of 'two identical lines'

Test Plan (2)

- Test for the case of line 1 vertical
  - $(x_{11},y_{11}) = (7,2)$, $(x_{12},y_{12}) = (7,5)$ and $(x_{21},y_{21}) = (1,3)$, $(x_{22},y_{22}) = (4,6)$
  - Should give result of (7, 9)
- Test for the case of line 2 vertical
  - $(x_{11},y_{11}) = (1,4)$, $(x_{12},y_{12}) = (3,3)$ and $(x_{21},y_{21}) = (4,2)$, $(x_{22},y_{22}) = (4,5)$
  - Should give result of (4,11/2)

© Janice Regan, CMPT 102, Sept. 2006

Test Plan (3)

- Test for parallel coincident vertical lines
  - $(x_{11},y_{11}) = (3,4)$, $(x_{12},y_{12}) = (3,3)$ and $(x_{21},y_{21}) = (4,4)$, $(x_{22},y_{22}) = (4,5)$
  - Should give result 'no intersection'
- Test for non intersecting parallel vertical lines
  - $(x_{11},y_{11}) = (3,4)$, $(x_{12},y_{12}) = (3,3)$ and $(x_{21},y_{21}) = (4,4)$, $(x_{22},y_{22}) = (4,5)$
  - Should give a result of 'two identical lines'

© Janice Regan, CMPT 102, Sept. 2006

Writing Algorithms

- You will succeed in writing algorithms if you
  - **Follow the Software development process**
    - First think about the problem, its input data and required results (output)
    - Next determine a series of steps that will transform the input data into the output results
    - Then enumerate all the special cases that the must be handled
    - If necessary modify or redesign your series of steps so that all special cases are handled
    - Verify your algorithm

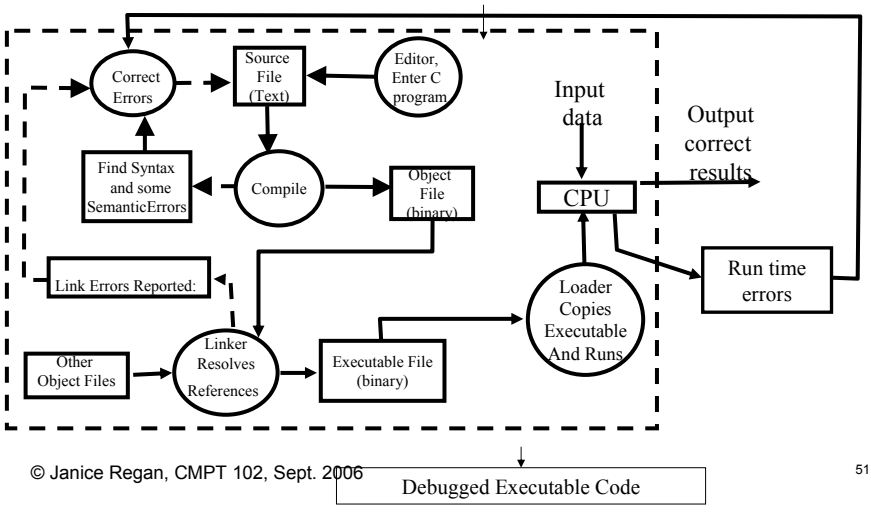
Choices

- There may be several algorithms to solve a given problem
  - Which algorithm is the best?
  - How do we chose?

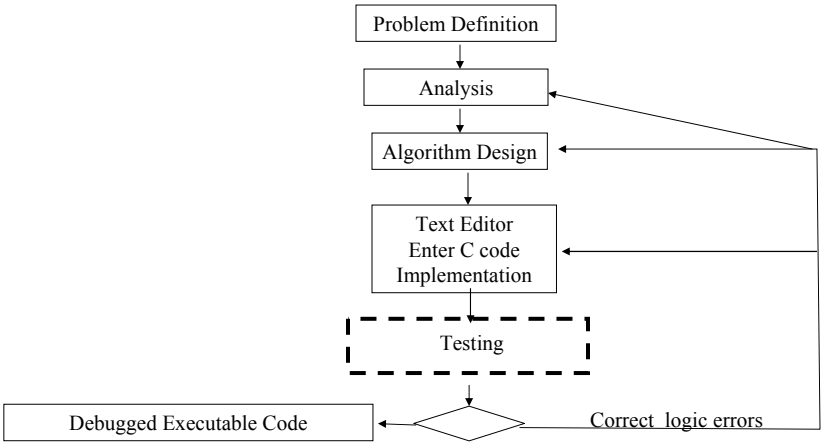
Properties of Good Algorithms

- Efficiency
- Simplicity
- Precision
- Effectiveness
- Generality
- Levels of Abstraction
- Correctness
- Finiteness
- Maintainability

Processing a C Program



Problem-Analysis-Coding-Execution Cycle



© Janice Regan, CMPT 102, Sept. 2006

Programming Methodologies

- Structured design
  1. A problem is divided into smaller sub-problems
  2. Each sub-problem is solved
  3. The solutions of all sub-problems are then combined to solve the problem

© Janice Regan, CMPT 102, Sept. 2006

Programming Methodologies

- In Object oriented design (OOD), a program is a collection of interacting objects
- An object consists of data and operations
- Steps in OOD:
  1. Identify objects
  2. Form the basis of the solution
  3. Determine how these objects interact

© Janice Regan, CMPT 102, Sept. 2006