Writing a simple program

PHYS2G03

McMaster University

Programming: Key Elements

In order of importance:

1. Designing the Program
2. Testing the Program
3. Writing the Program

Note: Writing has the strongest dependence on the actual Programming Language used

Actual steps:

1. **Designing the Program**
2. **Writing the Program**
3. **Testing the Program**
4. **Writing the Program**
5. **Testing the Program**

…

Testing is an ongoing process

Writing a program

1. State the problem
2. Analyse the problem: Break it down into simple steps.

   For each step:

   Decide what each step entails – what data needs to be available, what new data comes out
3. Generate the code to solve the problem, step by step

Calc Program   calc.cpp

```
#include <iostream>
int main()
{
  // Calc program
  // Takes two integers, sums them and reports the answer
  int a,b,c;

  // 1. Input 2 integers
  std::cout << "Please input two integers\n";
  std::cin >> a >> b;

  // 2. Sum a and b
  c = a + b;

  // 3. Output the results
  std::cout << "The sum of " << a << " and " << b << " equals " <<
       c << "\n";
  return 0; // success;
}
```

Program text
reflects initial goals
and design structure

Design ideas preserved
as comments:

Analysis: Major Steps

- (1) Read in 2 integers (input)
- (2) Calculate the sum of the integers
- (3) Report the answer (output)

Top-down *Structure Plan* for whole program

In class exercise: line program

Exercise: Write a program

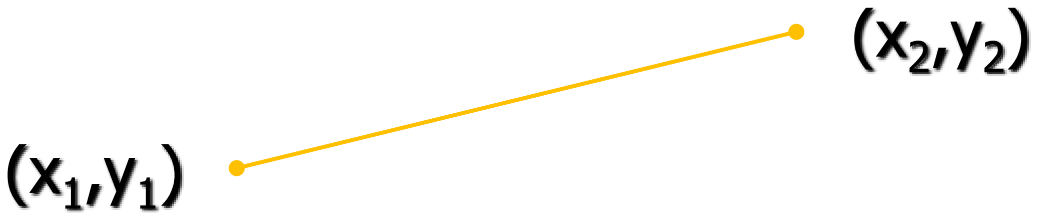
1. The problem

   Write a program to calculate the slope and intercept of a line connecting two points and report the solution. The user supplies 4 real numbers: x1,y1,x2,y2 for the coordinates of the points

Exercise: Write a program

1. The problem

   Write a program to calculate the slope, m, and intercept, c, of a line (y=mx+c)



```
m = (y2-y1)/(x2-x1)
 c = y1-m*x1
```

Note: This pseudo code is nearly valid code as written, just add ;

Line Program

2 Analysis: Major Steps

(1) Read in 4 real numbers (input)

(2) Calculate the coefficients for the line solution

(3) Report the two coefficient values (output)

Top-down *Structure Plan* for whole program

Part (2) can use the simple formulae rather than a function or anything fancy. The whole program is just one main function

Write your own line program

3. Solution

   The solution is a self-contained piece of code:
   e.g. line.cpp

```
cp –r /home/2G03/line  ~/
cd line

gedit line.cpp &
make line     OR     c++ line.cpp –o line
line
```

Line Program line.cpp

```
#include <iostream>
int main()
{
 // Line program
 // Takes two pairs of real numbers representing two points
 // Outputs the coefficients of the line that joins the points

  float x1,y1,x2,y2,m,c;

  // 1. Input x1,y1 x2,y2

  // 2. Apply the equations

  // 3. Output the results m,c

}
```

This is the line.cpp provided for you.
It will compile with make but it doesn’t do anything because the main program is nothing but comments right now!

Line Program   line.cpp

```
#include <iostream>
int main()
{
  // Line program
  // Takes two pairs of real numbers representing two points
  // Outputs the coefficients of the line that joins the points

  float x1,y1,x2,y2,m,c;

  // 1. Input 4 real numbers

  // 2. Apply the equations

  // 3. Output the results

}
```

float means real numbers

Use **std::cin** just like calc.cpp

Use coefficient formulae here

Use **std::cout** just like calc.cpp

Line Program

- Testing it out

```
make
OR
c++ line.cpp –o line
line
```

The Makefile already knows how to make a program called line by compiling line.cpp (take a look at the Makefile)

This will run your program (if the compile worked)

What if it doesn't work?
Well – try again.
How do you know what's wrong?
Take it one step at a time…

Line Program: Valid inputs?

- In general users are quite likely to give silly inputs. The specified points may be the same or otherwise problematic.
- A good program checks for crazy results rather than just producing garbage. The best way to avoid problems is to ensure the inputs are sensible.

Line Program

```
#include <iostream>

int main()
{
  // Line program
  // Takes two pairs of real numbers representing two points
  // Outputs the coefficients of the line that joins the points

  float x1,y1,x2,y2,m,c;

  // 1. Input 4 real numbers
  std::cout << "Please input two points x1 y1 and x2 y2\n";
  std::cin >> x1 >> y1 >> x2 >> y2;

  std::cout << “The user inputted x1=“ << x1 << “ y1=“ <<
     y1 << “ x2=“ << x2 << “ y2=“ << y2 << “\n”;
```

This is “print debugging” *and it works* – never under-estimate the value of knowing what the program is doing! You can delete the extra output later

Line Program Solution (i.e. Full marks version)

```
#include <iostream>

int main()
{
 // Line program
 // Takes two pairs of real numbers representing two points
 // Outputs the coefficients of the line that joins the points

 float x1,y1,x2,y2,m,c;

 // 1. Input 4 real numbers
 std::cout << "Please input two points x1 y1 and x2
     y2\n";
 std::cin >> x1 >> y1 >> x2 >> y2;

 // 1 a.  Check the input
 if (x1==x2) {
  std::cout << "Error: This line has infinite slope:
     x1=x2\n";
  return -1; // exit early and signal error
 }

 // 2. Apply the equations
 m = (y2-y1)/(x2-x1);
 c = y1 - m*x1;

 // 3. Output the results
 std::cout << "The coefficients for the line y = m x +
     c are:\n" <<
  " m = " << m << " and c = " << c << "\n";
 return 0; // success;
}
```

line_solution.cpp

Line Program Solution (i.e. Full marks version)

```
#include <iostream>

int main()
{
  // Line program
  // Takes two pairs of real numbers representing two points
  // Outputs the coefficients of the line that joins the points

  float x1,y1,x2,y

  // 1. Input 4 real numbers
  std::cout << "Please input two points x1 y1 and x2
       y2\n";
  std::cin >> x1 >> y1 >> x2 >> y2;

  // 1 a.  Check th
  if (x1==x2) {
    std::cout << "Error: This line has infinite slope:
       x1=x2\n";
    return -1; // exit early and signal error
  }
```

int main: an integer return value is expected

Tell the user input is expected

Check the input for bad values

return -1 means error

```
  // 2. Apply the equations
  m = (y2-y1)/(x2-x1);
  c = y1 - m*x1;

  // 3. Output the results
  std::cout << "The coefficients for the line y = m x +
       c are:\n" <<
    " m = " << m << " and c = " << c << "\n";
  return 0; // success
}
```

Unix convention return 0 means no errors

```
more line_solution.cpp
make line_solution
line_solution
```