16.216
ECE Application Programming

Instructor: Dr. Michael Geiger

Fall 2011

Lecture 20: Functions

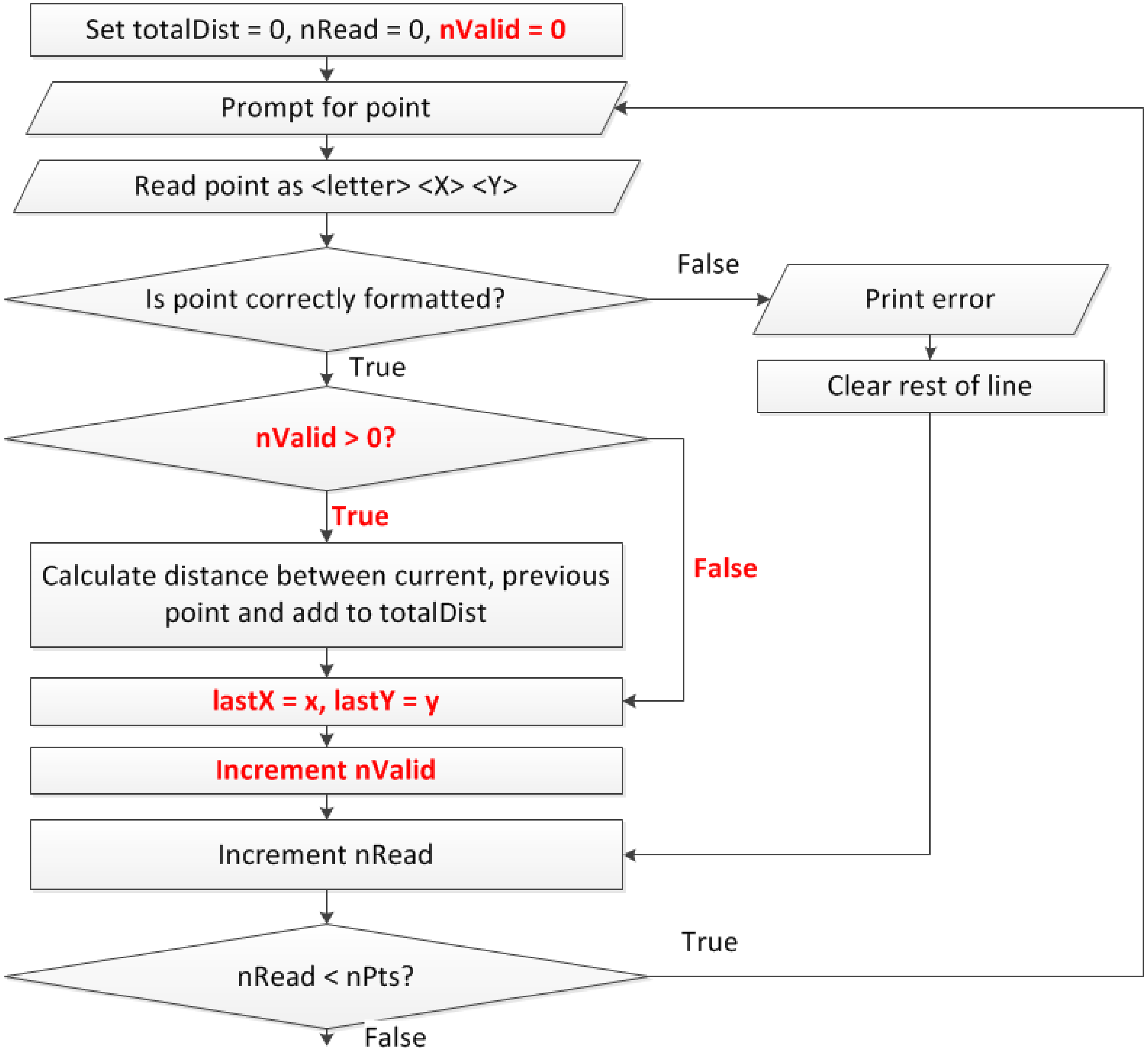
Lecture outline

- Announcements/reminders
  - Program 5 due tomorrow
  - Program 6 to be posted shortly
- Today
  - Brief review of PE3 solution
  - Functions

Reading list of points

- Issues
  - In order to calculate distances, must be able to store (x, y) values from both current and previous iteration
  - We can't calculate distance if we don't have at least two valid points → can't do if:
    - It's the first loop iteration
    - The new point isn't valid (correctly formatted)
    - All previous iterations produced invalid points
- Solution: counter for # of valid points (`nValid`)
  - Increment each time valid point is read
    - Store current point (`x, y`) for next iteration (`lastX`, `lastY`) under same conditions
  - Only calculate distance if:
    - Current point is valid
    - We have at least one previously read valid point (`nValid > 0`)

Flow chart: reading list of points, #2


Set totalDist = 0, nRead = 0, nValid = 0
Prompt for point
Read point as <letter> <X> <Y>
Is point correctly formatted?
False
Print error
Clear rest of line
True
nValid > 0?
True
False
Calculate distance between current, previous point and add to totalDist
lastX = x, lastY = y
Increment nValid
Increment nRead
nRead < nPts?
True
False

Character input

- One other issue: when reading point, likely using following scanf() call:
  - `scanf(“%c %lf %lf”, &letter, &x, &y);`
- %c format specifier will read any character
  - If previous input line was correctly formatted, next character left in input will be a newline!
    - Valid input example (newlines shown; input underlined):

```
Enter total # of points: 2\n
Enter a point: A 1 2\n
Enter a point: B 3 5\n
```

- Solution: clear newline after each valid input

Translating flow charts to code:

- Covered process for reading total # points on Friday
- Started process for handling single point
- See posted file (16216f11_PE3_soln.c) for rest of code

Functions

- Functions used to break problem down into small, "bite-sized" pieces.
  - Make code more manageable and readable
  - Identify reusable pieces
- Functions have an optional type of return value, a name, and optional arguments
- Functions return at most, ONE value
- Functions must be either "prototyped" or declared prior to use. Good programming practices requires all functions to be prototyped.

Functions

type of value returned

name of function

parameters of function (variables in)

```
double hyp(double a, double b)
{
  double sum, result;
  sum = a*a + b*b;
  result = sqrt(sum);
  return result;
}
```

Single value returned by function

Alternate way of writing above function

```
double hyp(double a, double b)
{
  return sqrt(a*a + b*b);
}
```

Functions - complete program

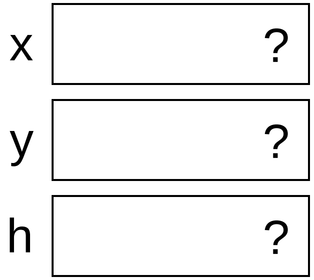
```
#include <stdio.h>
#include <math.h>
double hyp(double a, double b);
void main()
{
  double x,y,h;
  printf("Enter two legs of triangle: ");
  scanf("%lf %lf",&x,&y);
  h=hyp(x,y);
  printf("Trgle w legs %lf and %lf has hyp of %lf\n",x,y,h);
}
double hyp(double a, double b)
{
  double sum, result;
  sum = a*a + b*b;
  result = sqrt(sum);
  return result;
}
```

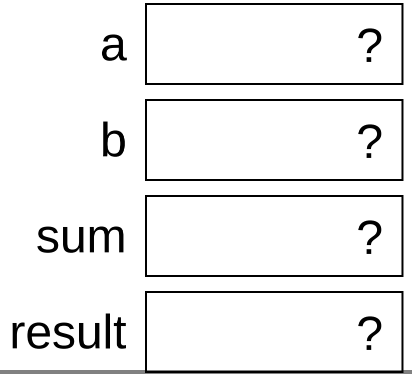
prototype (note semi-colon)

actual function definition
(NO semi-colon)

Functions - scope

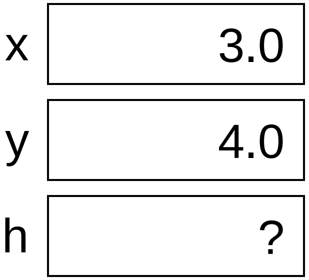
```
#include <stdio.h>
#include <math.h>
double hyp(double a, double b);
void main()
{
  double x,y,h;
  printf("Enter two legs of triangle: ");
→ scanf("%lf %lf",&x,&y);
  h=hyp(x,y);
  printf("Trgle w legs %lf and %lf has hyp of %lf\n",x,y,h);
}
double hyp(double a, double b)
{
  double sum, result;
  sum = a*a + b*b;
  result = sqrt(sum);
  return result;
}
```


x ?
y ?
h ?

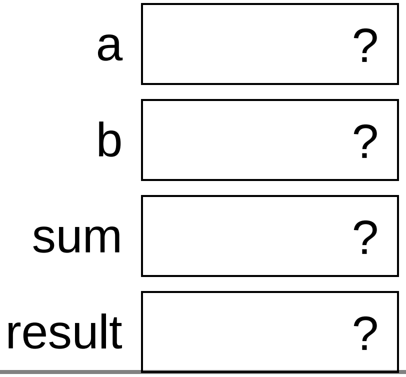

a ?
b ?
sum ?
result ?

Functions - scope

```
#include <stdio.h>
#include <math.h>
double hyp(double a, double b);
void main()
{
  double x,y,h;
  printf("Enter two legs of triangle: ");
  scanf("%lf %lf",&x,&y);
  h=hyp(x,y);
  printf("Trgle w legs %lf and %lf has hyp of %lf\n",x,y,h);
}
double hyp(double a, double b)
{
  double sum, result;
  sum = a*a + b*b;
  result = sqrt(sum);
  return result;
}
```



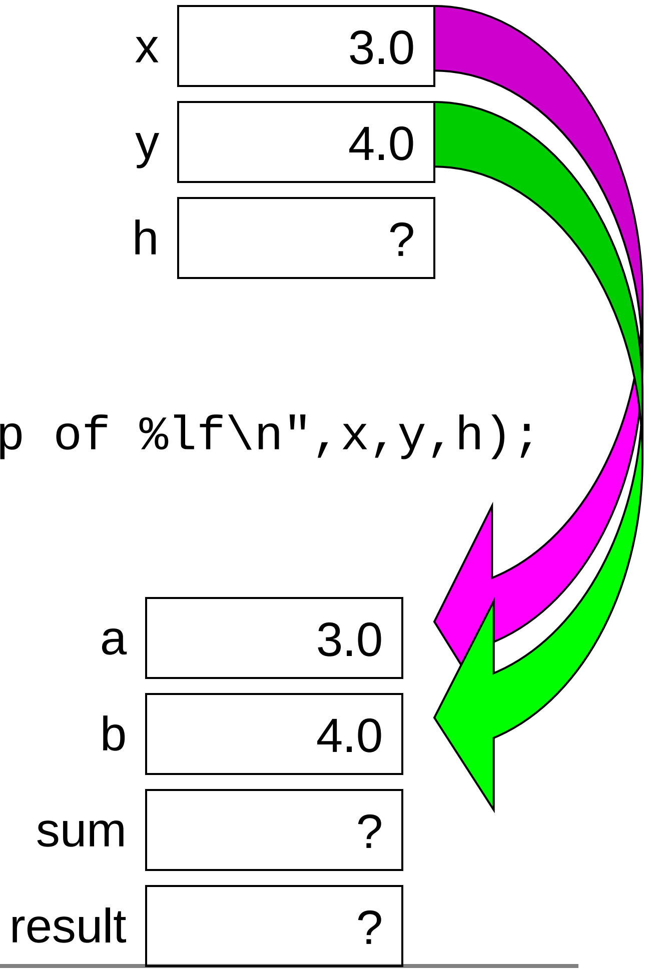
x	3.0
y	4.0
h	?



a	?
b	?
sum	?
result	?

Functions - scope

```
#include <stdio.h>
#include <math.h>
double hyp(double a, double b);
void main()
{
  double x,y,h;
  printf("Enter two legs of triangle: ");
  scanf("%lf %lf",&x,&y);
  h=hyp(x,y);
  printf("Trgle w legs %lf and %lf has hyp of %lf\n",x,y,h);
}
double hyp(double a, double b)
{
  double sum, result;
  sum = a*a + b*b;
  result = sqrt(sum);
  return result;
}
```



x 3.0

y 4.0

h ?

a 3.0

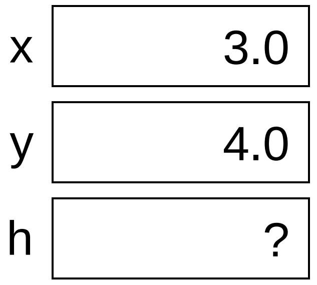
b 4.0

sum ?

result ?

Functions - scope

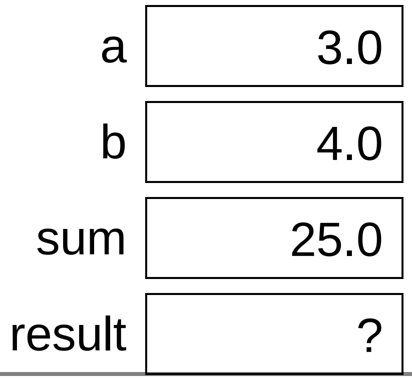
```
#include <stdio.h>
#include <math.h>
double hyp(double a, double b);
void main()
{
  double x,y,h;
  printf("Enter two legs of triangle: ");
  scanf("%lf %lf",&x,&y);
  h=hyp(x,y);
  printf("Trgle w legs %lf and %lf has hyp of %lf\n",x,y,h);
}
double hyp(double a, double b)
{
  double sum, result;
  sum = a*a + b*b;
  result = sqrt(sum);
  return result;
}
```



x 3.0

y 4.0

h ?



a 3.0

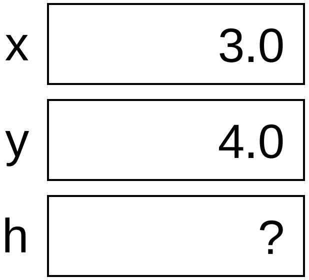
b 4.0

sum 25.0

result ?

Functions - scope

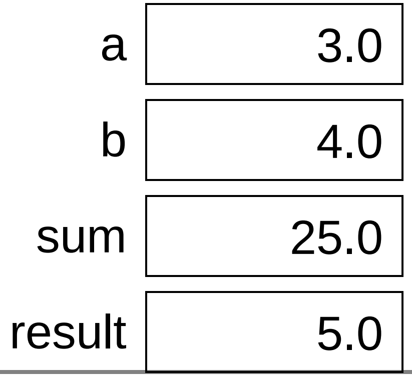
```
#include <stdio.h>
#include <math.h>
double hyp(double a, double b);
void main()
{
  double x,y,h;
  printf("Enter two legs of triangle: ");
  scanf("%lf %lf",&x,&y);
  h=hyp(x,y);
  printf("Trgle w legs %lf and %lf has hyp of %lf\n",x,y,h);
}
double hyp(double a, double b)
{
  double sum, result;
  sum = a*a + b*b;
  result = sqrt(sum);
  return result;
}
```



x 3.0

y 4.0

h ?



a 3.0

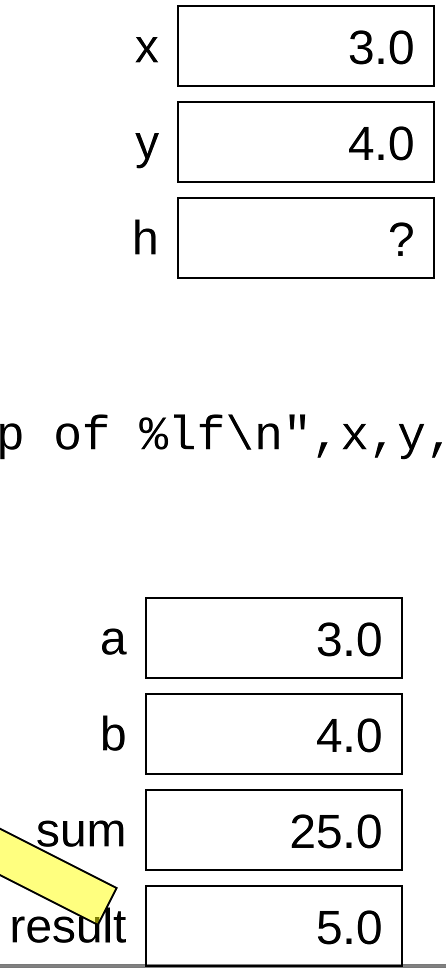
b 4.0

sum 25.0

result 5.0

Functions - scope

```
#include <stdio.h>
#include <math.h>
double hyp(double a, double b);
void main()
{
  double x,y,h;
  printf("Enter two legs of triangle: ");
  scanf("%lf %lf",&x,&y);
  h=hyp(x,y);
  printf("Trgle w legs %lf and %lf has hyp of %lf\n",x,y,h);
}
double hyp(double a, double b)
{
  double sum, result;
  sum = a*a + b*b;
  result = sqrt(sum);
  return result;
}
```



x 3.0

y 4.0

h ?

a 3.0

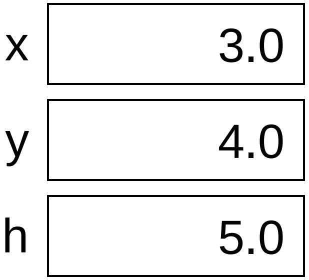
b 4.0

sum 25.0

result 5.0

Functions - scope

```
#include <stdio.h>
#include <math.h>
double hyp(double a, double b);
void main()
{
  double x,y,h;
  printf("Enter two legs of triangle: ");
  scanf("%lf %lf",&x,&y);
  h=hyp(x,y);
  printf("Trgle w legs %lf and %lf has hyp of %lf\n",x,y,h);
}
double hyp(double a, double b)
{
  double sum, result;
  sum = a*a + b*b;
  result = sqrt(sum);
  return result;
}
```



x 3.0

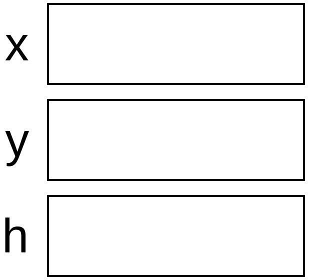
y 4.0

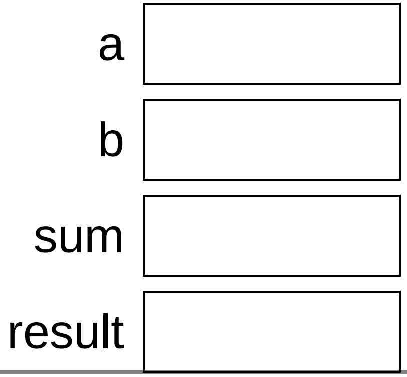
h 5.0

NOTE - a and b are NOT copied back to x and y

Exercise - What prints (if 5, 12 entered)

```
#include <stdio.h>
#include <math.h>
double hyp(double a, double b);
void main()
{
  double x,y,h;
  printf("Enter two legs of triangle: ");
  scanf("%lf %lf",&x,&y);
  h=hyp(x,y);
  printf("Trgle w legs %lf and %lf has hyp of %lf\n",x,y,h);
}
double hyp(double a, double b)
{
  double sum, result;
  a = 3;
  b = 4;
  sum = a*a + b*b;
  result = sqrt(sum);
  return result;
}
```





Answer

```
Trgle w legs 5.000000 and 12.000000 has hyp of 5.00000
```

Next time

- Function examples
- Pointers
- Pointer arguments