

Slides #3

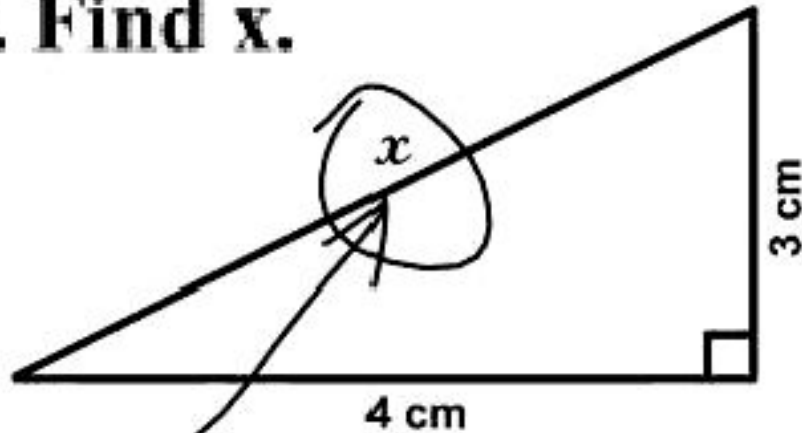
Variables

Chapter 2.1-2.2



"Just a darn minute! — Yesterday you said that X equals two!"

3. Find x.



Here it is

Topics

- 1) How can we store data, such as numbers?
- 2) How can we do calculations like:
10 times 3?

Variables

Variables

- A variable stores a value.
 - It is..
 - C++ is..Each variable is given a type, like “integer” when it is created.

- Example:

- the variable:

```
int numStudents;
```

- the variable:

```
numStudents = 72;
```

Variable declarations tell the compiler the variable's type (int) and name (numStudents).

All variables must be..

"Error: Undeclared identifier"

This assignment statement copies the value (72) into the variable (numStudents).

Example with Variables

```
// Small demonstration of variables.
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    // Create the variable, give it a value, and then display it.
```

```
    int numStudents;
```

```
    numStudents = 5;
```

```
    cout << "The value of numStudents is: " << "numStudents" << endl;
```

```
    cout << "The value of numStudents is: " << numStudents << endl;
```

```
    // Change the value and re-display it.
```

```
    numStudents = 7;
```

```
    cout << "Now the value of numStudents is: "
```

```
        << numStudents << endl;
```

```
    return 0;
```

```
}
```

Output:

```
The value of numStudents is: numStudents
```

```
The value of numStudents is: 5
```

```
Now the value of numStudents is: 7
```

Identifiers

- Identifier: a programmer-defined name which..
 - Ex: Variable names, or function (later...)
- Valid Identifiers:
 - First character: a-z or A-Z or _
 - Any other characters: a-z or A-Z or _ or 0-9
 - Examples:
 - height, i, x1, numStudents, NUM_PEOPLE, cur_weight
- Invalid Identifiers:
 - 2Tall, 11a, test#2, 3dGraphics

Identifiers

- Identifiers cannot be keywords:
 - Keywords are...
 - Ex: int, return, char, for, while, switch, case...
- Tips:
 - Use meaningfully descriptive names:
 - numStudents is better than n
 - boxHeight is better than x
 - Use camel case for variables names:
First word is lower case,
Capitalize first letter of later words.
 - Ex: Students per course: ...

Naming



What's in a name? that which we call a rose
By any other name would smell as sweet;
-- Shakespeare: *Romeo and Juliet*.

- A variable name *is* important:
 - It's what other programmers will read.
 - It tells us..

```
#include <iostream>
using namespace std;
int main()
{
    int s = 90;
    int f = s * 10;
    cout << f << endl;
}
```

- What does this code output?
- Guess what is s? Any better names?
- Guess what is f? Any better names?

Variable Example

```
// Calculate the length and cost of a fence around a rectangular area
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int landWidth = 10;
```

```
    int landLength = 15;
```

```
    int fenceLength = (2 * landWidth) + (2 * landLength);
```

```
    cout << "For some land "
```

```
        << landWidth << "m by "
```

```
        << landLength << "m, the total fence length require is "
```

```
        << fenceLength << "m.\n";
```

```
    double costPerMeter = 3.50;
```

```
    double fenceCost = fenceLength * costPerMeter;
```

```
    cout << "Total fence cost (at $" << costPerMeter
```

```
        << "/m) is $" << fenceCost << ".\n";
```

```
    return 0;
```

For some land 10m by 15m, the total fence length require is 50m.

Total fence cost (at \$3.5/m) is \$175.

double type holds
floating point numbers
like 3.1415

Exercise: Bad names?

- What's wrong with the following variable names?
 - 1) x
 - 2) 3LittlePigs
 - 3) sumofalltestscores
 - 4) numNeuronsPerClusterInLayer2ObjectDetector
 - 5) double

Operations on Numbers

- Most basic math operations work on numbers.

- `int x = 10; int y = 3; int z = 0;`

- Addition $z = x + y;$

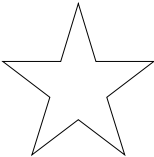
- Subtraction $z = x - y;$

- Multiplication $z = x * y;$

- Division $z = x / y;$

- Modulo $z = x \% y;$

- Negation $z = -x;$



Negation is Unary:
it takes only one argument.

$+$, $-$, $*$, $/$, $\%$ are
binary operators:
they take two arguments.

Get real!

- Give each variable a type based on what it will hold.
 - int for integers

```
int numStudents = 42;
int missionClock = -10;
int numPinkElephants = 0;
```
 - double for real (“floating point”) numbers

```
double treeHeight_m = 42.9;
double averageDogs = 0.35;
double distanceToPluto_m = 7.5E9;
// 7.5*109;
```
- For each of your variables, pick the best type.

For now, all real numbers should be in doubles:
`double dogsInClass = numStudents * averageDogs;`

Out Of Class Review Question

- Write a program which:
 - Create two int variables; hard-code them to be two different values.
 - Calculate their:
 - sum (+)
 - difference (-)
 - product (*)
 - Use good variable names to store each result.
 - Display each result to the screen.
 - Try making the variables double and see what happens; change their values too.

Summary

- C++ variables are strongly typed: int, double, char, string
 - Must declare variables before use.
 - Operators: +, -, *, /, %
 - How to write a program.

Slides #3

Variables - Part 2

Chapter 2.1-2.2

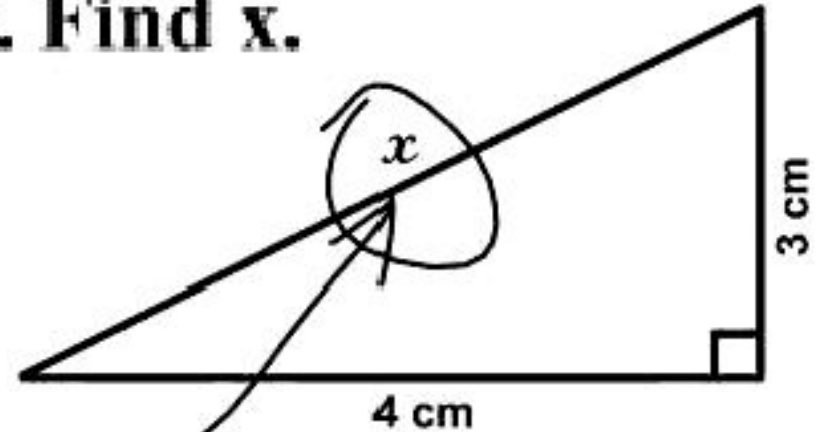
CMPT 130

© Dr. B. Fraser



"Just a darn minutel — Yesterday you said that X equals two!"

3. Find x.



Here it is

char and string

char

- The char type can hold a single character.
 - Pronounced like "charred" not like "car".
- Characters are represented by the computer..
 - 'A' is 65, 'B' is 66, 'C' is 67, etc (ASCII codes)
 - cout shows char's as a character (65 as 'A').

```
char aLetter = 'A';  
cout << "char A:  " << aLetter << endl;  
  
aLetter = 70;  
cout << "char 70: " << aLetter << endl;  
  
aLetter = aLetter + 1;  
cout << "char 71: " << aLetter << endl;
```

Output:

```
char A:  A  
char 70:  F  
char 71:  G
```

string Class

- The string class stores and manipulates strings.
 - string class defined in library: `#include <string>`

```
// Example for string
#include <iostream>
#include <string>
using namespace std;
```

```
int main()
{
    string name;
    cout << "Who are you? ";
    cin >> name;

    cout << "Welcome to the great \"" << name << "\"! :)" << endl;
}
```

Sample Run:

```
Who are you? Me
Welcome to the great "Me"! :)
```

Working with strings

- = String Assignment

```
string name = "Bond";
```

- + String Concatenation

- Use a + to join two strings together.

```
string full = "James" + "Bond" // = "James Bond"
```

- String Length

- Use the "member-function" length on a string:

```
int nameLen = name.length() // = 10 chars long.
```

- [] Get a character in a string

```
char firstChar = name[0]; //
```

```
char secondChar = name[1]; //
```

Keyboard Input and Basic Output Formatting

Input

- Almost every computer program needs input.
- Examples:
 - Calculate # pizzas for a party: input # people.
 - Calculate gas mileage: input distance and fuel used.
- Input with cin:....

```
int people = 0;  
cin >> people;
```

 - >> is the...
 - cin waits for the user to type in...
 - Places the answer in the given variable.

Prompts

- Prompting the User:
 - cout: Display a prompt to user asking for input.
 - cin: Read keyboard input into a variable.

```
#include <iostream>
using namespace std;
```

```
int main() {
    int favNum = 0;

    // Read in user's favourite number:
    cout << "Enter your favourite number: ";
    cin >> favNum;

    if (favNum < 0) {
        cout << "Now that's interesting! " << favNum << " eh?\n";
    } else {
        cout << "Your favourite number is: " << favNum << endl;
    }
}
```

```
Enter your favourite number: 42
Your favourite number is: 42
```

Input Example

```
// Ask the user for their personal information.
#include <iostream>
#include <string>
using namespace std;

int main()
{
    cout << "What is your name? ";
    string name;
    cin >> name;

    cout << "What is your height in cm? ";
    int height = 0;
    cin >> height;

    cout << "What is the airspeed velocity of an unladen swallow? ";
    int speed = 0;
    cin >> speed;

    cout << "Hello Sir " << name << ", whose height is " << height << "cm.\n";
    cout << "A swallow's airspeed is NOT " << speed << "!\n";
}
```

setw()

- setw() is a manipulator:
 -
 - Great for lining up data on the screen.
 - setw() only affects the one next element.

- Example:

```
int age = 12
cout << "[" << age << "];
cout << "[" << setw(5) << age << "];
```

Output

```
[12]
[  12]
```

Pads with spaces when
item is fewer characters
than the setw()'s width.

..
if it's larger than width.

Making a table

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    const int WIDTH1 = 15;
    const int WIDTH2 = 18;
    const int WIDTH3 = 12;

    cout << setw(WIDTH1) << "Name:"
         << setw(WIDTH2) << "Fav Food"
         << setw(WIDTH3) << "Fav Number"<<endl;

    cout << setw(WIDTH1) << "Dr. Evil"
         << setw(WIDTH2) << "Cupcakes"
         << setw(WIDTH3) << "100000000"<<endl;

    cout << setw(WIDTH1) << "I.L.B. Bach"
         << setw(WIDTH2) << "Anchovies"
         << setw(WIDTH3) << "1997"<<endl;
    // . . . . . omitted to fit on slide.
}
```

Name:	Fav Food	Fav Number
Dr. Evil	Cupcakes	100000000
I.L.B. Bach	Anchovies	1997
Me	Pizza and Cake	0

Review

1. What is wrong with each of these?
 - a) `int 1stVar = 10;`
 - b) `int return = 0;`

2. What is the value of each of these variables?
 - a) `int x = 5 / 2;`
 - b) `int y = 21 % 5;`
 - c) `double z = 4 * 1.5;`

3. What statement displays variable `age` using 6 columns?

4. What statement reads in a number to the variable `age`?

Summary

- Formatted output:
 `cout << setw(10) << "Hello";`
- Keyboard Input:
 `cin >> myAge;`

Slides #3

Variables - Part 3

Chapter 2.1-2.2

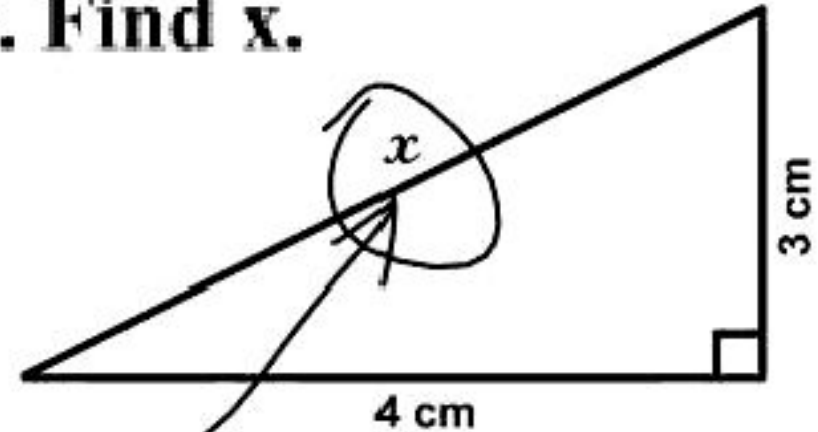
CMPT 130

© Dr. B. Fraser



"Just a darn minute! — Yesterday
you said that X equals two!"

3. Find x.



Here it is

Initialization, Scope, and Comments

Uninitialized Variables

- Variables which are not initialized...
 - That value is garbage (unknown).

```
short g1, g2, g3, g4, g5, g6, g7, g8;  
cout << setw(8) << g1;  
cout << setw(8) << g2;  
cout << setw(8) << g3;  
cout << setw(8) << g4 << "\n";  
cout << setw(8) << g5;  
cout << setw(8) << g6;  
cout << setw(8) << g7;  
cout << setw(8) << g8 << "\n";
```

Output:

2052	-29221	114	8240
51	25765	-16446	2216

Variable Initialization

- Variable Initialization:
 - You should always..
 - Can **initialize** with either:..



- C++ does not require variable initialization; but it is a good safe practice.
- Each variable must be defined exactly once.

```
int height = 1;  
int height = 1;    ..
```

Uniform Initializer Example

```
// Show uniform initializers
#include <iostream>
#include <iomanip>
using namespace std;

int main () {
    const int SIDES_PER_TRIANGLE {3};
    const int WIDTH {5};

    cout << "How many triangles? ";
    int triangles {0};
    cin >> triangles;

    int totalSides = (triangles * SIDES_PER_TRIANGLE);

    cout << "# Triangles: " << setw(WIDTH) << triangles << endl;
    cout << "# Sides:      " << setw(WIDTH) << totalSides << endl;

    return 0;
}
```

```
How many triangles? 8
# Triangles:      8
# Sides:          24
```


Scope

- Scope is the region of the program where..

```
int main() {  
    int height = 10;  
    cout << height;    // OK.  
  
    cout << width;    // ERROR: not defined yet!  
    int width = 10;  
    return 0;        ..  
}
```

More on this later!

Comments

- Good comments tell you..
- Which comment is best?
 - `double rate = 0.12; // Set to 0.12`
 - `double rate = 0.12; // Set to current tax rate.`
- Rule of thumb:
 - Comment the purpose of every 3-4 lines of code.

Comment Style

- Single line comments use double slash:

```
// Insert meaningful comment here.
```

```
int i=2;
```

- Multiple line comments use `/* ... and ... */`

```
/*
```

```
These are good for larger comments.
```

```
For example, describing a function's purpose,  
Arguments, return value, and errors.
```

```
*/
```

- When changing the code...

- An incorrect comment is worse than no comment!

Out Of Class Review Question

- Write a program to help out at a health center:
 - Reads in two numbers from the keyboard:
 - Number of patients waiting
 - Number of nurses working
 - Calculate and display how many patients each nurse sees (and how many left over)
 - Calculate total number of people at the health centre
 - Calculate how long it will be until any nurse has a break from seeing patients (assume 10m / patient)
 - Line up output nicely on screen.
 - Use good variable names to store each result.

Summary

- Importance of variable initialization
- Include meaningful comments!