

Slides #3

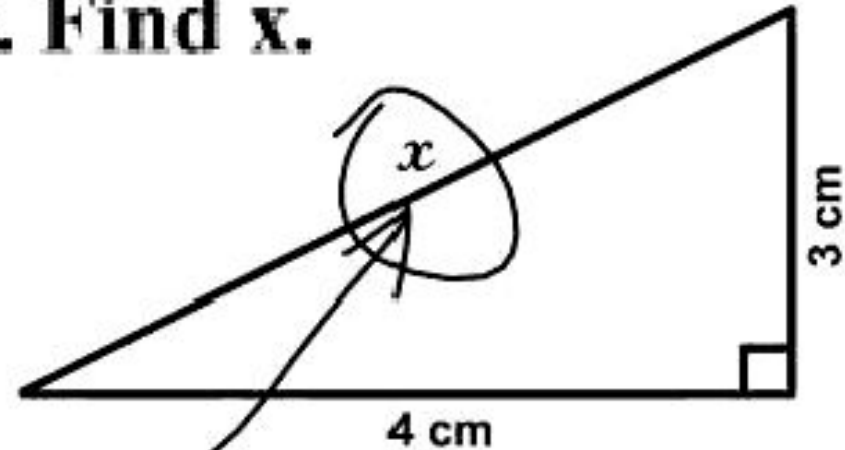
Variables

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 has 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 <print>
using namespace std;

int main() {
    // Create the variable, give it a value, and then display it.
    int numStudents;
    numStudents = 5;
    println("The value of numStudents is: {}", "numStudents");
    println("The value of numStudents is: {}", numStudents);

    // Change the value and re-display it.
    numStudents = 7;
    println("Now the value of numStudents is: {}", numStudents);
    return 0;
}
```

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 <print>
using namespace std;
int main() {
    int s = 90;
    int f = s * 10;
    println(f);
}
```

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

Variable Example

```
// Demonstrate variables
```

```
#include <print>
```

```
using namespace std;
```

```
int main() {
```

```
    int landWidth = 10;
```

```
    int landDepth = 15;
```

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

```
    println("For some land {}m by {}m, the total fence length is {}m.",  
            landWidth, landDepth, fenceLength);
```

```
    double costPerMeter = 3.50;
```

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

```
    println("Total fence cost (at ${} /m) is ${}.",  
            costPerMeter, fenceCost);
```

```
    return 0;
```

```
}
```

double type holds
floating point numbers
like 3.1415

For some land 10m by 15m, the total fence length is 50m.
Total fence cost (at \$3.5/m) is \$175.

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 $z = x * y;$

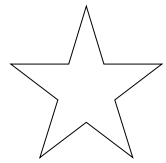
- Division $z = x / y;$

- Modulo $z = x \% y;$

- Negation $z = -x;$

Negation is Unary:
it takes only on 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

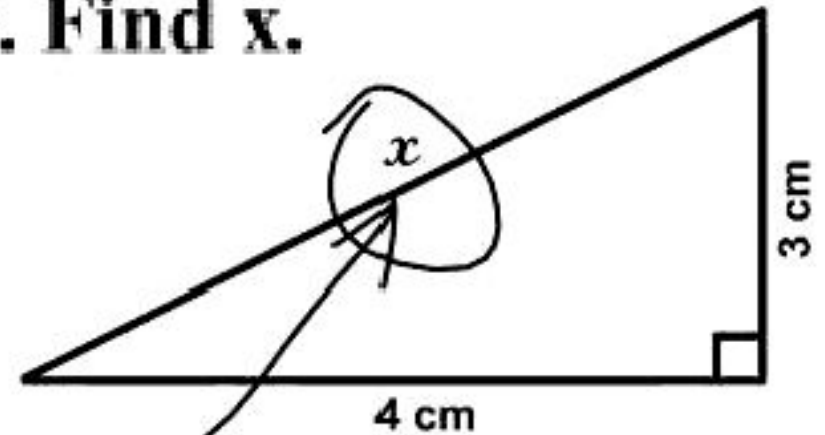
CMPT 130

© Dr. B. Fraser



"Just a darn minute! — 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)
 - println() shows char's as a character (65 as 'A').

```
// Show equivalence of a number and a character
int main() {
    char aLetter = 'A';
    println("char A:  {} ", aLetter);

    aLetter = 70;
    println("char 70: {} ", aLetter);

    aLetter = aLetter + 1;
    println("char 71: {} ", aLetter);
}
```

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 <print>
#include <string>
using namespace std;

int main() {
    string name;
    print("Who are you? ");
    cin >> name;

    println("Welcome to the great \"{}\"! :)", name);
}
```

```
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.at(0); //
```

```
char secondChar = name.at(1) //
```

- Convert something to a string:

- `string with7 = full + to_string(7);` // = "James Bond7"

Keyboard Input and Basic Output Formatting

Input

- Almost every 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:
 - print: Display a prompt to user asking for input.
 - cin: Read keyboard input into a variable.

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

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

```
    // Read in user's favourite number:
    print("Enter your favourite number: ");
    cin >> favNum;
```

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

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

Input Example

// Ask the user for their personal information.

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

```
int main(){
    print("What is your name? ");
    string name;
    cin >> name;
```

```
    print("What is your height in cm? ");
    int height = 0;
    cin >> height;
```

```
    print("What is the airspeed velocity of an unladen swallow? ");
    int speed = 0;
    cin >> speed;
```

```
    println("Hello Sir {}, whose height is {}cm.", name, height);
    println("A swallow's airspeed is NOT {}!", speed);
```

```
}
```

What is your name? Bert

What is your height in cm? 33

What is the airspeed velocity of an unladen swallow? 2

Hello Sir Bert, whose height is 33cm.

A swallow's airspeed is NOT 2!

println() and width

- Lineup output:...

```
int main() {  
    int age = 0;  
    print("Enter your age: ");  
    cin >> age;  
    int days = age * 365;  
    int hours = days * 24;  
  
    println("Age:           {:5}", age);  
    println("Days old:      {:5}", days);  
    println("Hours old:     {:5}", hours);  
}
```

```
Enter your age: 2  
Age:           2  
Days old:      730  
Hours old:    17520
```

- It won't clip if value is too many digits;
just breaks output alignment.

```
Enter your age: 1234  
Age:           1234  
Days old:      450410  
Hours old:    10809840
```

- Hint: Hard-code spaces into the println() string to line up the fields (e.g., after "age" above).

Making a table

```
// Set width of output
#include <print>
using namespace std;

int main() {
    println("{:15}{:18}{:12}", "Name:", "Fav Food", "Fav Number");
    println("{:15}{:18}{:12}", "-----", "-----", "-----");
    println("{:15}{:18}{:12}", "Dr. Evil", "Cupcakes", 1000000000);
    println("{:15}{:18}{:12}", "I.L.B. Bach", "Anchovies", 1997);
    println("{:15}{:18}{:12}", "Me", "Pizza and Cake", 0);
}
```

Name:	Fav Food	Fav Number
-----	-----	-----
Dr. Evil	Cupcakes	1000000000
I.L.B. Bach	Anchovies	1997
Me	Pizza and Cake	0

Alignment: Left, Right, Centre

- {5} uses a default alignment for the data type
- {:<5} for left {:^5} for centred {:>5} for right

```
// Set width and alignment of output
#include <print>
using namespace std;
```

```
int main() {
    println("{:<15}{:^18}{:>12}", "Name:", "Fav Food", "Fav Number");
    println("{:<15}{:^18}{:>12}", "-----", "-----", "-----");
    println("{:<15}{:^18}{:>12}", "Dr. Evil", "Cupcakes", 1000000000);
    println("{:<15}{:^18}{:>12}", "I.L.B. Bach", "Anchovies", 1997);
    println("{:<15}{:^18}{:>12}", "Me", "Pizza and Cake", 0);
}
```

Name:	Fav Food	Fav Number
-----	-----	-----
Dr. Evil	Cupcakes	1000000000
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:
 `int numLights = 4;`
 `println("I see {:>3} lights!", numLights);`
- Keyboard Input:
 `int age = 0;`
 `cin >> myAge;`

Slides #3

Variables - Part 3

Chapter 2.1-2.2

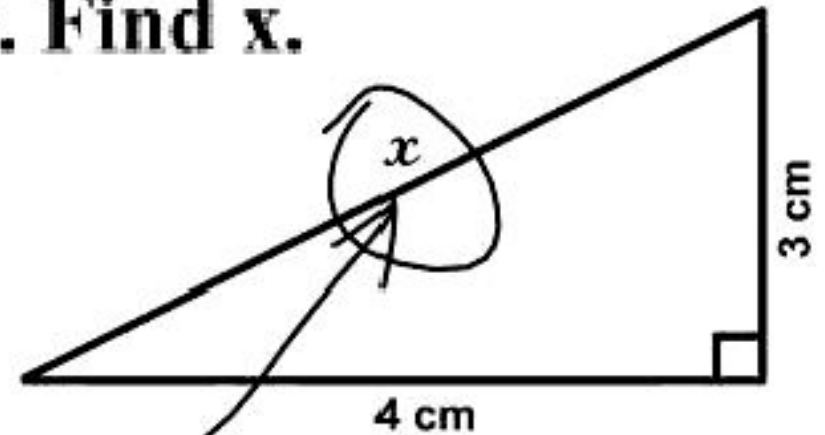
CMPT 130

© 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).

```
// Demonstration of uninitialized variables hold a value.  
#include <print>  
using namespace std;  
  
int main (){  
    short g1, g2, g3, g4, g5, g6, g7, g8;  
    println("{:8}{:8}{:8}{:8}", g1, g2, g3, g4);  
    println("{:8}{:8}{:8}{:8}", g5, g6, g7, g8);  
}
```

0	32767	-2130	15762
0	0	0	0

Variable Initialization

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



Uniform Initializer

- 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 <print>
using namespace std;

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

    print("How many triangles? ");
    int triangles {0};
    cin >> triangles;

    int totalSides = (triangles * SIDES_TRIANGLE);

    println("# Triangles: {:>{}}", triangles, WIDTH);
    println("Total sides: {:>{}}", totalSides, WIDTH);
}
```

Can use an argument for the width,
rather than hard coding:
“hello {:{}}” - nests a {} inside {}

How many triangles? 8
Triangles: 8
Sides: 24

Scope

- Scope is the region of the program where..

```
// Must declare variables before using them.
#include <print>
using namespace std;

int main() {
    int height = 10;
    println("Height: {}", height);    // OK.

    println("Width: {}", 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 centre:
 - 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!