Instructions

- Complete the following instructions
  - Take out a piece of paper
  - Take out a pen or pencil
  - Draw a circle
  - Draw a square
  - Draw a triangle

Instructions

- Compare your piece of paper with your neighbor
  - How are they the same and different?

Operations

- Things that you are capable of doing
  - Take out
  - Draw
  - Compare

Arguments

- What additional information did each operation require?

Arguments

- What additional information did each operation require?
  - What to take out, where to get it from, which one
  - Where to draw, what size, with what
  - Compare?

Arguments

- In programming languages, these pieces of information are called *arguments*
  - A single operation does different things with different arguments
    - Write…your name…on the paper
    - Write…your email address…on the board

- Humans can deal with ambiguous, underspecified instructions
  - Computers cannot, need precise, complete, unambiguous
  - Computers produce the same output every time (*deterministic*)

Programs

- Writing a computer program is simply giving instructions that a computer can follow
  - In this course, we'll be giving instructions in the C++ language
- A program is a sequence of *statements*
  - Each statement tells the computer to do one (or more) *operations*

Language, Syntax and Semantics

- C++ is an *artificial language*
  - Similar to a natural language, only the rules are a lot more strict
  - To use a language, you have to know the syntax (what goes where) and the semantics (what does it mean)
  - In natural language, you can bend things quite a bit
    - i are teach u good!!!1!!11!
  - In a programming language, you have to follow the rules
  - Unlike in natural language, statements in C++ have only one meaning: they tell the computer to do a specific thing

Language, Syntax and Semantics

- Every *statement* in C++ is like a sentence in English
  - Both are made up of words and symbols separated by spaces
  - English syntax rule: sentences end with punctuation
  - C++ syntax rule: statements end with a semi-colon (`;`)
    - By convention, each statement gets its own line
- Based on this rule, here is the simplest C++ statement:

```
;
```

  - It is a valid statement (follows the rules)
  - It tells the computer to do nothing

Exercises

- These exercises count towards your in-class participation grade
  - Grading is simple, either complete, incomplete or not turned in
- Don’t take notes on your exercise sheet!
  - You won’t be taking it with you

Exercise: Inventing a Language

- Write down 2 different reasonable statements to tell a computer to:
  1. Print the text “Hi there” to the screen
  2. Add the numbers 17 and 32 and print the result to the screen
  3. Add the numbers 17, 32 and 31 and print the result to the screen
- (This isn’t a “right answer” question. It’s considering the possibilities.)

Operators

- To tell the computer to do something, a statement can use an *operator*
- Each operator specifies a particular operation
  - Insertion operator (<<): print something, somewhere
  - Addition operator (+): add two things
  - Subtraction operator (−): subtract one thing from another
  - Assignment operator (=): store something, somewhere
- Each of these operators is *binary*
  - They take two *arguments*

Operators

- Syntax rule: a binary operator needs a *left-hand side (LHS)* argument and a *right-hand side (RHS)* argument
  - Insertion operator: cout << “Hello there”;
    - LHS: where to print
    - RHS: what to print
  - Addition operator: 5 + 6;
    - LHS/RHS: the two numbers to add
  - Subtraction operator: 19 – 3;
    - LHS: the number to subtract from
    - RHS: the number to subtract
- In general, a statement with a binary operator looks like:

  LHS operator RHS ;

The Print Statement

- How does the computer understand this statement?
  - The *insertion operator* (`<<`) instructs the computer to print something
    - To do so, it has to be told what to print and where
  - To the right of the operator is the *value* that we want it to print
    - `4` is the literal number 4
    - `"Hello there"` is a literal *string* of characters
  - To the left of the operator is the place we want to print to
    - `cout` specifies that black box on the screen
  - All statements in C++ end with a semi-colon (`;`)

Data

- What are valid arguments?
  - Depends on the operator
  - Most of the time, a piece of *data*
- Four *primitive* data types
  - An integer (whole number)

    ```
    12
    ```
  - A real number

    ```
    3.14
    ```
  - A character (surrounded by single quotes)

    ```
    'A'
    ```
  - A string (surrounded by double quotes)

    ```
    "Hello"
    ```

Data

- Addition and subtraction only work with numbers (integers or reals)
- Printing (insertion) works with numbers, characters or strings on the RHS
  - LHS is the place to print, we'll get back to that