Elements of a Program

Class 2

```
 1 // This program calculates the user's pay.
 2  #include <iostream>
 3  using namespace std;
 4
 5  int main()
 6  {
 7    double hours, rate, pay;
 8
 9    // Get the number of hours worked.
10    cout << "How many hours did you work? ";
11    cin >> hours;
12
13    // Get the hourly pay rate.
14    cout << "How much do you get paid per hour? ";
15    cin >> rate;
16
17    // Calculate the pay.
18    pay = hours * rate;
19
20    // Display the pay.
21    cout << "You have earned $" << pay << endl;
22    return 0;
23  }
```

The Components of a Program

- common elements in programming languages
  - reserved words
  - programmer-defined identifiers
  - operators
  - punctuation
  - syntax
  - style and format

Reserved Words

- aka key words
- cannot be used for any other purpose
- examples: using, namespace, int, double, and return

Programmer-Defined Identifiers

- names made up by the programmer
- not part of the C++ language
- used to represent various things, especially
  - variables (labels for memory locations)
  - functions
- examples: hours, rate, pay, main

Operators

- usually (but not always) symbols
- used to perform operations on values
- C++ has a large number of operators, e.g.,
  - arithmetic: $+$ $-$ * /
  - assignment: $=$
  - IO: << >>

Punctuation

- single characters that help parse the program
- mark the end of a statement: ;
- separate items: , :
- begin and end blocks: { }
- grouping: ( )

Syntax

- like human language, but much stricter
- the rules of grammar that must be followed when writing a program
- controls the use of reserved words, operators, programmer-defined identifiers, and punctuation

Style

- incredibly important
- program source code must be human-readable
- you can lose lots of points on style even if your program "works"
- look at style guide

Variables

Variable

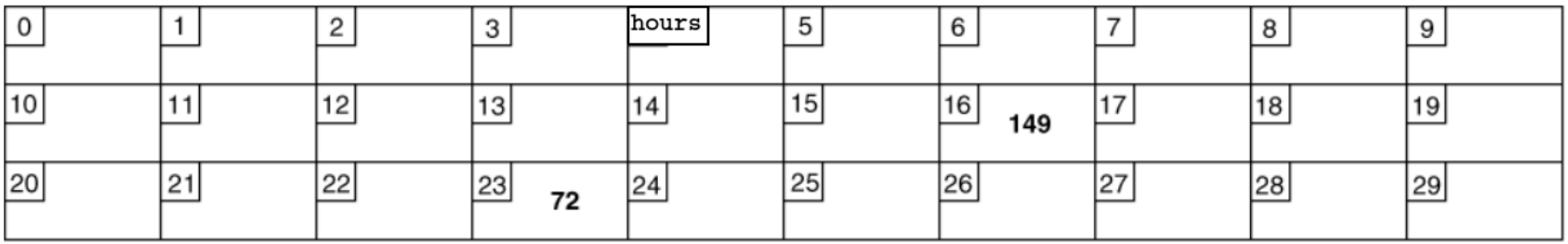
a programmer-defined **named storage location** in memory for holding a piece of data

- must be declared
- must be of a specific type

```
double hours;
double rate;
double pay;
```

Variables

remember the picture of memory from Monday



declaring a variable causes a label to be attached to a memory cell

```
double hours;
```

Batch Processing

- many programs follow this main flow:
  1. gather input data
     - from user via keyboard
     - from file on disk
     - from a scientific device
     - from the network
  2. process the data
  3. output the results
     - display on screen
     - write to files
     - send over the network

Purpose

Computers are used for:

- information jobs too boring for humans
- information jobs too big for humans

But never for:

- jobs too intellectual for humans

Purpose

A computer is used to automate an information job that:

- is repetitive, doing the same thing over and over
- has too many steps to perform in the available time
- encompasses too much data to encompass "by hand"

Programming

a computer program is a model of a real (or virtual) physical process

software development has three main parts:

1. analyze and understand the problem or situation
2. design a solution
3. implement the solution in software

Do not neglect the first two!

Analysis

What do you want the software to do?

Result: a human text description of the requirements
a statement of the goal

- this is the most-neglected part
- it is impossible for you to program a computer to do something you don't understand how to do

Design

How do you want the software to do it?

Result: a framework of the proposed software system in human language

- this is the hardest part
- it is impossible for you to program a computer to do something you can't explain how to do
- you also must know what computers can and can't do

Implementation

Make the program do it!

incrementally, write, test, and use the actual software

(with some training), this is the easy part!

Programming Process

- incremental development
  - build a very small piece
  - make it work
  - extend functionality
  - make it work
  - extend functionality
  - make it work
  - ...

Incremental Programming