Memory

- Computers get their powerful flexibility from the ability to store and retrieve data
- Data is stored in *main memory*, also known as *Random Access Memory (RAM)*

Exercise: Inventing Language

- Get a separate (not your notes!) paper
  - Discuss question 1 with your neighbor
  - Write out possible answers to questions 2 & 3
  - (you're going to hand this paper in)
- Given that memory is a big place to store pieces of information:
  1. How might you organize it?
  2. What is a reasonable statement to tell the computer to store the number 68 somewhere?
  3. What is a reasonable statement to get that number back out of storage and print it to the screen?

Memory

- Addressing
  - Sequential locations where data can be stored
  - Each location has an *address* (an integer)
  - A computer with 1 Gigabyte of RAM has a billion 1-byte locations to store data
    - The first location is at address 0
    - The last location is around address 999,999,999
- Data is stored and retrieved by address
  - Could use the actual location number
  - Easier if we are able to name locations
    - "store this data at location *stan*"
    - "get whatever data is a location *stan*"

Allocating Space

- Before a program can store a piece of data, it has to *allocate* space in main memory
  - This involves reserving memory at some location and giving that location a name
  - That name is called a *variable*
  - The program uses the variable to store and retrieve data
- This is done with a *variable declaration* statement
  - Made up of a *type* and a *name*
  - For example:

    ```
    int x;
    ```

  - Where `int` is the type and `x` is the name

Data Types

- Every variable has a type
  - Just like every piece of data (integer, real number, character, string)
  - A variable can only store data of the same type
- Some C++ data types:

int	Integer (whole number, positive or negative)
float	Real number (includes decimal, positive or negative)
char	Character
string	String of characters

Identifiers (Variable Names)

- Consist of letters, digits, and the underscore character (_)
- Must begin with a letter or underscore
- C++ is case sensitive
  - `NUMBER` is not the same as `number`

Identifiers (continued)

- The following are legal identifiers in C++:
  - `first`
  - `conversion`
  - `payRate`

TABLE 2-1 Examples of Illegal Identifiers

Illegal Identifier	Description
`employee Salary`	There can be no space between `employee` and `Salary`.
`Hello!`	The exclamation mark cannot be used in an identifier.
`one+two`	The symbol + cannot be used in an identifier.
`2nd`	An identifier cannot begin with a digit.

Assignment Operator

- The assignment operator (=) stores a piece of data in a memory location
  - LHS argument: the variable where you want to store it
  - RHS argument: the data to store
- For example, storing the number 4 (an integer) is a two-step process:
  - First, allocate memory by declaring a variable of type integer

```
int myVariable;
```

  - Then, assign it the piece of data

```
myVariable = 4;
```

  - We say that `myVariable` has the *value* 4

Assignment and Expressions

- Remember that an operand can be any expression that evaluates to the right type of data
- So the RHS of an assignment can be an expression:

```
int x;
x = 5 + 6 – 7;
```

- The RHS of an assignment is always evaluated first, then the resulting value is stored

Using Stored Data

- Variable are used to store and retrieve data
- Up to this point we have used *literal* data in our expressions:

```
cout << 41;
```

- Instead, we can use a stored piece of data:

```
int x;
x = 41;
cout << x;
```

- A variable can go in any expression where a literal piece of data could go

Using Stored Data

- Variables are reusable
  - Each assignment stores a new value and over writes the old

```
int x;
x = 5;
cout << x;
x = 6;
cout << x;
x = x + 10;
cout << x;
```

Exercise

- (Still on that same sheet of paper)
- What gets printed on the screen when this code runs?

```
int x;
int y;

cout << "Hello";
cout << endl;
x = 1 + 4 - 3;
cout << x;
cout << endl;
y = x - 1;
y = y + 5;
cout << y;
cout << endl;
```

Exercise

- (yeah, same paper)
- Write C++ statements to do the following:
  - Declare an integer variable called banana
  - Declare an integer variable called hamster
  - Assign banana the value 56
  - Assign hamster the value 22 added to the value of banana
  - Print out the value of hamster

Function scope

- Variables are declared *inside* functions
  - We say they have function *scope*
- Those variables are only valid inside the function they are declared in
  - The name is meaningless outside
- Think of memory as being divided up between the functions
  - Each function gets its own chunk of memory
  - Variables declared in a function allocate memory in that chunk
  - When the function is done, that chunk is erased