Unit 4a

Variables and 'cin'
Unit Objectives

• Know how variables are declared and assigned
• Use cin statement to get keyboard input from the user
• Predict how cin will treat input with whitespaces and extract data
VARIABLES AND ASSIGNMENT
The Need For Variables & Input

- Printing out constants is not very useful (nor exciting)
- In fact, we could just as easily compute the value ourselves in many situations
- The real power of computation comes when we introduce variables and user input
  - Variables provide the ability to remember and name a value for use at a later time
  - User input allows us to write general programs that work for "any" input values
  - Thus, a more powerful program would allow us to enter an arbitrary number and perform conversion to dozens

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

// Execution always starts at the main() function
int main()
{
    cout << "3 dozen is " << 3*12 << " items." << endl;
    // the above results in the same output as below
    cout << "3 dozen is 36 items." << endl;
    return 0;
}
```
C/C++ Variables

- Variables allow us to
  - Store a value until it is needed and change its values potentially many times
  - Associate a descriptive name with a value
- Variables are just memory locations that are reserved to store a piece of data of specific size and type
- Programmer indicates what variables they want when they write their code
  - Difference: C requires declaring all variables at the beginning of a function before any operations. C++ relaxes this requirement.
- The computer will allocate memory for those variables when the code starts to run
- We can provide initial values via '=' or leave them uninitialized

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

int main()
{
  // Sample variable declarations
  char c = 'A';
  int x; // uninitialized variables
      // will have a (random) garbage value until we initialize it
  x = 1; // Initialize x's value to 1
  c = 'B'; // Change c's value to 'B'
}
```

Variables are actually allocated in RAM when the program is run.
C/C++ Variables

- Variables have a:
  - **type** [int, char, unsigned int, float, double, etc.]
  - **name/identifier** that the programmer will use to reference the value in that memory location [e.g. x, myVariable, num_dozens, etc.]
    - Identifiers must start with [A-Z, a-z, or an underscore '_'] and can then contain any alphanumeric character [0-9, A-Z, a-z, _] (but no punctuation other than underscores)
    - Use descriptive names (e.g. numStudents, doneFlag)
    - Avoid cryptic names (myvar1, a_thing)
  - **location** [the address in memory where it is allocated]
  - **Value**
- Reminder: You must declare a variable before using it

```c
int quantity = 4;
double cost = 5.75;
cout << quantity*cost << endl;
```

**What's in a name?**
To give descriptive names we often need to use more than 1 word/term. But we can't use spaces in our identifier names. Thus, most programmers use either camel-case or snake-case to write compound names

- **Camel case**: Capitalize the first letter of each word (with the possible exception of the first word)
  - myVariable, isHighEnough
- **Snake case**: Separate each word with an underscore '_'
  - my_variable, is_high_enough
Know Your Common Variable Types

- Variables are declared by listing their type and providing a name.
- They can be given an initial value using the '==' operator.

<table>
<thead>
<tr>
<th>C Type</th>
<th>Usage</th>
<th>Bytes</th>
<th>Bits</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>Text character</td>
<td>1</td>
<td>8</td>
<td>ASCII characters -128 to +127</td>
</tr>
<tr>
<td></td>
<td>Small integral value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>True/False value</td>
<td>1</td>
<td>8</td>
<td>true / false</td>
</tr>
<tr>
<td>int</td>
<td>Integer values</td>
<td>4</td>
<td>32</td>
<td>-2 billion to +2 billion</td>
</tr>
<tr>
<td>unsigned int</td>
<td></td>
<td></td>
<td></td>
<td>0 to +4 billion</td>
</tr>
<tr>
<td>double</td>
<td>Rational/real values</td>
<td>8</td>
<td>64</td>
<td>±16 significant digits * 10^-308</td>
</tr>
<tr>
<td>string</td>
<td>Arbitrary text</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
When Do We Need Variables?

• When a value will be supplied and/or change at run-time (as the program executes)

• When a value is computed/updated at one time and used (many times) later

• To make the code more readable by another human

double area = (56+34) * (81*6.25);
// readability of above vs. below
double height = 56 + 34;
double width = 81 * 6.25;
double area = height * width;
What Variables Might Be Needed

• Calculator App
  – Current number input, current result

• Video playback (YouTube player)
  – Current URL, full screen, volume level
Assignment (=) Operator

• To update or change a value in a variable we use the assignment operator (=)

• Syntax:
  – `variable = expression;`  
    (Left-Side) (Right-side)

• Semantics:
  – Place the resulting value of 'expression' in the memory location associated with 'variable'
  – Does not mean "compare for equality" (e.g. is w equal to 300?)
    • That is performed by the `==` operator

```cpp
// iostream allows access to 'cout'
#include <iostream>
using namespace std;

// Execution always starts at the main() function
int main()
{
    int w;  // variables don't have to be initialized when declared
    char x; // be initialized when declared

    w = 300;
    x = 'a';
    cout << w << " " << x << endl;

    w = -75;
    x = '!';
    cout << w << " " << x << endl;
    return 0;
}
```

Output: 300 a -75 !

Order of evaluation: right to left

Assignment is one of the most common operations in programs
Assignment & Expressions

- Variables can be used in expressions and be operands for arithmetic and logic
- See inset below on how to interpret a variable's usage based on which side of the assignment operator it is used

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

// Execution always starts at the main() function
int main()
{
    int dozens = 3;
    double gpa = 2.0;

    int num = 12 * dozens;
    gpa = (2 * 4.0) + (4 * 3.7); // gpa updated to 22.8
    gpa = gpa / 6; // integer or double division?

    cout << dozens << " dozen is " << num << " items.\" << endl;
    cout << "Your gpa is " << gpa << endl;
    return 0;
}
```

Order of evaluation: right to left

- `int x = 0;`
- `x = x + 3;`

Semantics of variable usage:
- Right-side of assignment: Substitute/use the current value stored in the variable
- Left-side of assignment: variable is the destination location where the result of the right side will be stored
Exercises

• What is printed by the following two programs?

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

int main()
{
    int value = 1;
    value = (value + 5) * (value - 3);
    cout << value << endl;

    double amount = 2.5;
    value = 7;
    amount = value + 6 / amount;
    cout << amount << endl;
    cout << value % 3 << endl;
    return 0;
}
```

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

int main()
{
    int x = 5;
    int y = 3;
    double z = x % y * 6 + x / y;
    cout << z << endl;
    z = 1.0 / 4 * (z - x) + y;
    cout << z << endl;
    return 0;
}
```
RECEIVING INPUT WITH CIN
Keyboard Input

- In C++, the 'cin' object is in charge of receiving input from the keyboard.
- Keyboard input is captured and stored by the OS (in an "input stream") until cin is called upon to "extract" info into a variable.
- 'cin' converts text input to desired format (e.g. integer, double, etc.)

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

int main()
{
    int dozens;
    cout << "Enter number of dozen: " << endl;
    cin >> dozens;
    cout << 12 * dozens << " eggs" << endl;
    return 0;
}
```
Dealing With Whitespace

- **Whitespace (def.):**
  - Characters that represent horizontal or vertical blank space. Examples: newline ('\n'), TAB ('\t'), spacebar (' ')

- cin sequentially scans the input stream for actual characters, discarding leading whitespace characters

- Once cin finds data to convert it will STOP at the first trailing whitespace and await the next cin command

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

int main()
{
    int dozens;
    cout << "Enter number of dozen: " << endl;
    cin >> dozens;
    cout << dozens << " dozen " << " is " << 12*dozens << "items." << endl;
    return 0;
}
```

Main Take-away:
- cin SKIPS leading whitespace
- cin STOPS on the first trailing whitespace

Suppose at the prompt the user types:
```
    \t15 \n
```

```cpp
    cin
```

dozens
```
    \n
```cpp
```
Timing of Execution

- When execution hits a 'cin' statement it will:
  - Wait for input if nothing is available in the input stream
    - OS will capture what is typed until the next 'Enter' key is hit
    - User can type as little or much as desired until Enter (\n)
  - Immediately extract input from the input stream if some text is available and convert it to the desired type of data

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

int main()
{
    int dozens;

    cout << "Enter number of dozen: "
         << endl;
    cin >> dozens; // input stream empty
    // so wait for input
    cout << 12*dozens << " eggs" << endl;

    double gpa;
    cout << "What is your gpa?" << endl;
    cin >> gpa; // input stream has text
    // so do not wait...
    // just use next text
    cout << "GPA = " << gpa << endl;
    return 0;
}
```
Excercises

- cpp/cin/building_floor
SOLUTIONS
Exercises

• What is printed by the following two programs?

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

int main()
{
    int value = 1;
    value = (value + 5) * (value - 3);
    cout << value << endl;

double amount = 2.5;
    value = 7;
    amount = value + 6 / amount;
    cout << amount << endl;
    cout << value % 3 << endl;
    return 0;
}
```

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

int main()
{
    int x = 5;
    int y = 3;
    double z = x % y * 6 + x / y;
    cout << z << endl;
    z = 1.0 / 4 * (z - x) + y;
    cout << z << endl;
    return 0;
}
```

-12
9.4
1

13 // or 13.0
5 // or 5.0