C++ Variables
Instructor: Chi Tse (Ricky)
Variables in C++

- Allows storage of data internally for the program
- Allows storage of information from the user
- There are different types of variables which services different needs

Examples:
- Variables that store texts
- Variables that store integers (positive & negative numbers)
- Variables that store decimals (floating point numbers)
In order to use a variable in C++, we must first declare (i.e. create) it.

Model:

```
variable_type variable_name ;
```

– variable_type: The type of a variable, depends on the type of data we want to store.
– variable_name: The name of a variable, how we like to call it in the rest of the program.

A variable has a **name**, stores a **value** of the declared **type**.
• C++ rules for legal variable names:
  1. Cannot start with a number. Should start with a letter.
  2. The rest of the name can be a letter, a number, or an underscore “_”. (i.e. no space or special characters)
  3. Cannot be a C++ keyword.
• Suggestions for variable names
  • Should be meaningful
  • Should be easy to read
• Check p.26 (3rd ed) or p.29 (2nd ed) of Brian Overland’s textbook for more information
Illegal Variable Names

Example of illegal variable names

- `int number of students;`
- Has spaces
- `int 1number;`
- Begins with a number
- `int discount%;`
- Contains a symbol
- `double int;`
- Contains a keyword, `int`
• C++ is case sensitive!
• Examples:
  int hello; //declares a variable hello
  int Hello; //declares another variable Hello
  Int hello; //error, Int is not a C++ type
  Double amount; //error, Double is not a type
Are the following variable names valid?

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>a)</strong> student name</td>
<td><strong>b)</strong> int</td>
</tr>
<tr>
<td><strong>c)</strong> student_name</td>
<td><strong>d)</strong> 111students</td>
</tr>
<tr>
<td><strong>e)</strong> Fall2014</td>
<td><strong>f)</strong> john@cuny</td>
</tr>
<tr>
<td><strong>g)</strong> &quot;variable_name&quot;</td>
<td><strong>h)</strong> return</td>
</tr>
<tr>
<td><strong>i)</strong> return0</td>
<td><strong>j)</strong> _111students</td>
</tr>
<tr>
<td>a) student name</td>
<td>b) int</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>N (no spaces allowed)</td>
<td>N (C++ keyword)</td>
</tr>
<tr>
<td>c) student_name</td>
<td>d) 111 students</td>
</tr>
<tr>
<td>Y</td>
<td>N (can’t start with a number)</td>
</tr>
<tr>
<td>e) Fall2014</td>
<td>f) john@cuny</td>
</tr>
<tr>
<td>Y</td>
<td>N (no special symbols)</td>
</tr>
<tr>
<td>g) &quot;variable_name&quot;</td>
<td>h) return</td>
</tr>
<tr>
<td>N (no special symbols)</td>
<td>N (C++ keyword)</td>
</tr>
<tr>
<td>i) return0</td>
<td>j) _111students</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
In CS111, we will focus on the following variable types:
int, double, bool, char, string

(from p.16 of Schaum’s Outlines - Programming with C++)
# Data Types Range

## Table B.1: C++ Intrinsic Data Types

<table>
<thead>
<tr>
<th>TYPE</th>
<th>DESCRIPTION (FOR 32-BIT SYSTEMS)</th>
<th>RANGE (ON 32-BIT SYSTEMS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>1-byte integer (used to hold ASCII character value)</td>
<td>0 to 255</td>
</tr>
<tr>
<td>unsigned char</td>
<td>1-byte unsigned integer</td>
<td>0 to 255</td>
</tr>
<tr>
<td>signed char</td>
<td>1-byte signed integer</td>
<td>-128 to 127</td>
</tr>
<tr>
<td>short</td>
<td>2-byte integer</td>
<td>-32,768 to 32,767</td>
</tr>
<tr>
<td>unsigned short</td>
<td>2-byte unsigned integer</td>
<td>0 to 65,535</td>
</tr>
<tr>
<td>int</td>
<td>4-byte integer (but same as short on 16-bit systems)</td>
<td>Approx. ± 2 billion</td>
</tr>
<tr>
<td>unsigned int</td>
<td>4-byte unsigned integer (but same as unsigned short on 16-bit systems)</td>
<td>Approx. 4 billion</td>
</tr>
<tr>
<td>long</td>
<td>4-byte integer</td>
<td>Approx. ± 2 billion</td>
</tr>
<tr>
<td>unsigned long</td>
<td>4-byte unsigned integer</td>
<td>Approx. 4 billion</td>
</tr>
<tr>
<td>bool</td>
<td>Integer in which all nonzero values are converted to true (1); also holds false (0) (ANSI)</td>
<td>true or false</td>
</tr>
<tr>
<td>wchar_t</td>
<td>Wide character, for holding Unicode characters (ANSI)</td>
<td>Same as unsigned int</td>
</tr>
<tr>
<td>long long</td>
<td>64-bit signed integer (C++0x)</td>
<td>Approx. ±9 x 10 to the 18th</td>
</tr>
<tr>
<td>unsigned long long</td>
<td>64-bit unsigned integer (C++0x)</td>
<td>Approx. 1.8 x 10 to the 19th</td>
</tr>
<tr>
<td>float</td>
<td>Single-precision floating point</td>
<td>3.4 x 10 to the 38th</td>
</tr>
<tr>
<td>double</td>
<td>Double-precision floating point</td>
<td>1.8 x 10 to the 308th</td>
</tr>
<tr>
<td>long double</td>
<td>Extra-wide double-precision (ANSI)</td>
<td>At least as great as double</td>
</tr>
</tbody>
</table>

(from Appendix B of Brian Overland’s textbook)
Variable declaration
- int number;
- int year;
- int age;

Examples of VALID integer value assignment
- number = 3;
- year = 2016;
- age = 20;

Examples of INVALID integer value assignment
- number = "3";
- year = '2016';
- age = "thirty";
- age = 20.5;
double (decimals, high precision)

• Variable declaration
  – double pi;
  – double e;

• Examples of VALID double value assignment
  – pi = 3.1415926535;
  – e = 2.71828;

• Examples of INVALID double value assignment
  – pi = "3.141"
  – pi = ' 3.141 ';

bool (boolean: true or false)

• Variable declaration
  – bool reply;
  – bool answer;

• Examples of VALID bool values
  – answer = true;
  – answer = false;
  – reply = 0; // (i.e. false)
  – reply = 1;

• Examples of INVALID bool values
  – answer = “true”;
  – reply = ‘0’; // value becomes true
Variable declaration
- char c;
- char newline;
- char code;

Examples of VALID char values
- c = 'c';
- newline = '\n';
- code = 165; // Yen symbol (¥) in Unicode

Examples of INVALID char values
- code = 456;
- newline = "\n";
• Variable declaration
  – string name;
  – string address;
  – string day;

• Examples of VALID string values
  – name = "Vincent";
  – address = "65-30 Kissena Blvd.";
  – day = "2";

• Examples of INVALID string values
  – name = 'Vincent';
  – address = 65-30 Kissena Blvd.;
  – day = 2;