Basic C++
(What you should already know)
Chapters 1-5
CS 2308/CS5301
Spring 2020
Jill Seaman

Structure of a C++ Program

• Hello world:

```cpp
// This program outputs a message to the screen
#include <iostream>
using namespace std;

int main() {
    cout << "Hello world!" << endl;
}
```

• In general:

```cpp
// This is a comment
#include <includefile> ...
using namespace std;

int main() {
    statements ...
}
```

Variables, Data Types

• **Variable**: portion of memory that stores a value
• **Identifier**: name of a program element
• Fundamental data types
  
<table>
<thead>
<tr>
<th>Short</th>
<th>Float</th>
<th>Bool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int</td>
<td>Double</td>
<td>Char</td>
</tr>
<tr>
<td>Long</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• **Variable Declaration** statement

```cpp```
datatype identifier;
```

• **Variable Initialization** statement:

```cpp```
datatype identifier = constant;
```

Constants

• **Literals** (specific value of a given type)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12.45</td>
<td>true</td>
<td>'A'</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>-3.8</td>
<td>false</td>
<td>'2'</td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>6.25e-5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• **Named Constants**: variable whose value cannot be changed

```cpp```
const datatype identifier = constant;
```

```cpp```
const double TAX_RATE = 0.0675;
```
Assignment statement, expressions

- To change the value of a variable:
  ```
  variable = expression;  count = 10;
  ```
- **The lefthand side must be a variable**
- **The righthand side is an expression of the right type**
- What is an expression?
  - an expression has a type and evaluates to a value
    - literal
    - named constant
    - variable
    - arithmetic expression
    - etc.

Arithmetic and Relational Operations

- **arithmetic operators:**
  ```
  x + 10  
  7 % 2  
  8 + 5 * 10
  ```

- **relational operators (result is bool):**
  ```
  7 < 25  
  89 == x
  x % 2 != 0  
  8 + 5 * 10 <= 100 * n
  ```

Logical Operations, precedence

- **logical operators (values and results are bool):**
  ```
  x < 10 && x > 0  
  y == 10 || y == 20
  !(a == b)
  ```
- **operator precedence (which happens first?):**
  ```
  !(y == 10) || y == 20 && x > 3 * z
  ```

More assignment statements

- **Compound assignment**

<table>
<thead>
<tr>
<th>operator</th>
<th>usage</th>
<th>equivalent syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>+=</td>
<td>x += e;</td>
<td>x = x + e;</td>
</tr>
<tr>
<td>-=</td>
<td>x -= e;</td>
<td>x = x - e;</td>
</tr>
<tr>
<td>*=</td>
<td>x *= e;</td>
<td>x = x * e;</td>
</tr>
<tr>
<td>/=</td>
<td>x /= e;</td>
<td>x = x / e;</td>
</tr>
</tbody>
</table>

- **increment, decrement**

<table>
<thead>
<tr>
<th>operator</th>
<th>usage</th>
<th>equivalent syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>++</td>
<td>x++;</td>
<td>x = x + 1;</td>
</tr>
<tr>
<td>--</td>
<td>x--;</td>
<td>x = x - 1;</td>
</tr>
</tbody>
</table>
Type conversions

- **Implicit**
  - **assignment:**
    ```c
    int x;
    double d = 3.1415;
    cout << x << endl;
    ```
  - **binary operations:**
    ```c
    int x = 10;
    double d = 2.3;
    cout << x + d << endl;
    ```
  - The type of expression on the right will be converted to type of variable on left, possibly losing information.

- **Explicit**
  ```c
  int x, y;
  ...
  float avg = static_cast<float>(x)/y;
  or
  float avg = x/(float)y; //c-style notation
  ```
  
  Order of types:
  ```
  double
double
float
long
int
char
  ```

Basic Input/Output

- **Output (cout and <<)**
  ```c
  cout << expression;
  cout << expr1 << expr2;
  ```

- **Input (cin and >>)**
  ```c
  cin >> variable;
  cin >> var1 >> var2;
  ```
  ```c
  cin >> x;
  cout << "Enter the height and width: ";
  cin >> height >> width;
  ```
  
  Right hand side must be a variable!

Control structures: if else

- **if and else**
  ```c
  if (expression)
  statement1
  else
  statement2
  ```
  - If expression is true, statement1 is executed
  - If expression is false, statement2 is executed
  - The else is optional.
  - Nested if else

Control structures: loops

- **while**
  ```c
  while (expression)
  statement
  ```
  - If expression is true, statement is executed, repeat

- **for:**
  ```c
  for (expr1; expr2; expr3)
  statement
  ```
  ```c
  equivalent to:
  ```
  ```c
  expr1;
  while (expr2) {
  statement
  expr3;
  }
  ```
  - If expression is true, statement is executed, repeat

- **do while:**
  ```c
  do 
  statement
  while (expression);
  ```
  - Statement is executed, if expression is true, then repeat
Control structures: switch

- switch stmt:
  switch (expression) {
    case constant: statements
    ...
    case constant: statements
    default: statements
  }

  - execution starts at the case labeled with the value of the expression.
  - if no match, start at default
  - use break to exit switch (usually at end of statements)

  example:
  ```
  switch (ch) {
    case 'a':
      case 'A': cout << "Option A";
      break;
    case 'b':
      case 'B': cout << "Option B";
      break;
    default: cout << "Invalid choice";
  }
  ```

The string class

- string literals: represent sequences of chars:
  ```
  cout << "Hello";
  ```

  - To define string variables:
    ```
    string firstName, lastName;
    ```

  - Operations include:
    - = for assignment
    - .size() member function for length
    - ==, <, ... relational operators (alphabetical order)
    - [n] to access one character

File Input/Output

- #include <fstream>

- Output (ofstream)
  ```
  ofstream fout;
  fout.open("filename.txt");
  fout << "hello"
  fout << "Count is: " << count << endl;
  fout.close();
  ```

- Input (ifstream)
  ```
  ifstream fin;
  fin.open("data.txt");
  if (!fin) {
    cout << "error opening file" << endl;
    return (0);
  }
  int x;
  fin >> x;  // right hand side must be a variable!
  cout << "x is " << x << endl;
  fin.close();
  ```