Introduction to C++

- C++ Basics
- First example program
- Identifiers
- Data types
- Variables and constants
C++ Language History

- 1967: "BCPL" Language (Martin Richards, Cambridge)
- 1970: "B" Language (Ken Thompson, Bell Labs) used to write first UNIX
- 1972: "C" Language – task oriented (Dennis Ritchie Bell Labs)
- 1985: “extended C” – object oriented (Bjarne Stroustrup Bell Labs) “++” connotes increment
- 1998 ISO/ANSI Standard C++
  - International standard enables portability
  - what we use in this course
The Start of a Program

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

int main (void)
{
    return 0;
}
```
The Start of a Program

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

int main (void)
{
    return 0;
}
```
C++ Libraries

• C++ provides a base set of built-in capabilities
• Many additional features are provided in standard libraries
• Eg: the I/O library iostream defines features for input and output
  – needs to be specifically included to be used
#include <iostream>   //library
using namespace std;

int main (void)
{
    return 0;
}
#include <iostream>     //library
using namespace std;

int main (void) {
    return 0;
}
The Start of a Program

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

int main (void)
{
    return 0;
}
```
The Start of a Program

#include <iostream>     //library
using namespace std;

int main (void)
{
    return 0;
}
The Start of a Program

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

int main (void)
{
  return 0;
}
```
#include <iostream>
using namespace std;

int main (void)
{
    cout << "Hello World!" << endl;
    return 0;
}
Let’s add Output

#include <iostream>
using namespace std;

int main (void)
{
    cout << "Hello World!" << endl;
    return 0;
}
Commenting the Program

// Program to output a message
// Author:  183 Student
// Date:    1.09.2008

#include <iostream>
using namespace std;

int main (void)
{
    cout << "Hello World!" << endl;
    return 0;
}
Advice

- Always have a program that compiles/runs (even if it doesn’t do anything – yet)
- Compile and run your program often – check that it works
- You don’t want to add a lot of lines and then discover something is wrong!
- (why???)
Doing Calculations

- Common “operators” or symbols to do calculations are:

  +    Add 2 numbers
  -    Subtract 2 numbers
  *    Multiply 2 numbers
  /    Divide one number by another
  %    Remainder with INTEGER division (MOD)
Doing Calculations

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

int main (void)
{
    cout << "Hello World!" << endl;
    cout << 3+4*7 << endl;
    return 0;
}
```
#include <iostream>
using namespace std;

int main (void)
{
    cout << "Hello World!" << endl;
    cout << 3+4*7 << endl;
    return 0;
}
#include <iostream>
using namespace std;

int main (void)
{
    cout << "Hello World!" << endl;
    cout << 3+4*7 << endl;
    cout << "3+4*7" << endl;
    return 0;
}

Doing Calculations
Precedence

- Operations are done in a certain order – this is known as precedence

( ) operations in parenthesis are done first
+ - unary operations
* / % Multiplication, division, modulus are done next
+ - addition and subtraction are done last
= assignment (NOT equal)

- If there are multiple operations with the same precedence, the operations are done from left to right.
## Precedence Table

<table>
<thead>
<tr>
<th>OPERATOR</th>
<th>ASSOCIATIVITY</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HIGH</strong> - done first</td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>left to right</td>
</tr>
<tr>
<td>+</td>
<td>unary; right to left</td>
</tr>
<tr>
<td>-</td>
<td>left to right</td>
</tr>
<tr>
<td>*</td>
<td>binary; left to right</td>
</tr>
<tr>
<td>/</td>
<td>right to left</td>
</tr>
<tr>
<td>%</td>
<td></td>
</tr>
<tr>
<td><strong>LOW</strong> - done last</td>
<td></td>
</tr>
</tbody>
</table>
## Precedence Table

<table>
<thead>
<tr>
<th>OPERATOR</th>
<th>ASSOCIATIVITY</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HIGH</strong></td>
<td></td>
</tr>
<tr>
<td>(</td>
<td>done first</td>
</tr>
<tr>
<td>+</td>
<td>left to right</td>
</tr>
<tr>
<td>-</td>
<td>unary; right to left</td>
</tr>
<tr>
<td>*</td>
<td>left to right</td>
</tr>
<tr>
<td>/</td>
<td>binary; left to right</td>
</tr>
<tr>
<td>%</td>
<td>right to left</td>
</tr>
<tr>
<td>=</td>
<td></td>
</tr>
<tr>
<td><strong>LOW</strong></td>
<td>done last</td>
</tr>
</tbody>
</table>

**Example Expression:**

\[ 5 + -(3 - 1) \times 6 \div 3 \]
Division can be Tricky!

• watch out if you have int / int

  – 5 / 2 is
Division can be Tricky!

- watch out if you have int / int
  - 5 / 2 is 2
Division can be Tricky!

- watch out if you have int / int
  - 5 / 2 is 2
  - 2 / 3 is ?
Division can be Tricky!

• watch out if you have int / int

  – 5 / 2  is  2
  – 2 / 3  is  0
Division can be Tricky!

- watch out if you have int / int
  - 5 / 2 is 2
  - 2 / 3 is 0
  - 5.0 / 2.0 is ?
Division can be Tricky!

- watch out if you have int / int
  - 5 / 2 is 2
  - 2 / 3 is 0
  - 5.0 / 2.0 is 2.5
Division can be Tricky!

- watch out if you have int / int

  - 5 / 2 is 2
  - 2 / 3 is 0
  - 5.0 / 2.0 is 2.5
  - 2.0 / 3.0 is ?
Division can be Tricky!

- watch out if you have int / int
  - 5 / 2 is 2
  - 2 / 3 is 0
  - 5.0 / 2.0 is 2.5
  - 2.0 / 3.0 is 0.666...
Division can be Tricky!

• watch out if you have int / int

  – 5 / 2 is 2
  – 2 / 3 is 0
  – 5.0 / 2.0 is 2.5
  – 2.0 / 3.0 is 0.666…
  – 5 % 2 is ?
Division can be Tricky!

- watch out if you have int / int
  - 5 / 2 is 2
  - 2 / 3 is 0
  - 5.0 / 2.0 is 2.5
  - 2.0 / 3.0 is 0.666...
  - 5 % 2 is 1
Division can be Tricky!

- watch out if you have int / int
  - 5 / 2 is 2
  - 2 / 3 is 0

- 5.0 / 2.0 is 2.5
- 2.0 / 3.0 is 0.666...

- 5 % 2 is 1
- 2 % 3 is ?
Division can be Tricky!

- watch out if you have int / int
  - \(5 / 2\) is 2
  - \(2 / 3\) is 0
  - \(5.0 / 2.0\) is 2.5
  - \(2.0 / 3.0\) is 0.666...
  - \(5 \% 2\) is 1
  - \(2 \% 3\) is 2
/ and % are very useful

• e.g., How many hours and minutes are there in 375 minutes?
/ and % are very useful

• e.g., How many hours and minutes are there in 375 minutes?

• \(\frac{375}{60}\)
/ and % are very useful

- e.g., How many hours and minutes are there in 375 minutes?

- $375 / 60$ 6 hours
/ and % are very useful

- e.g., How many hours and minutes are there in 375 minutes?
  - 375 / 60       6 hours
  - 375 % 60
/ and % are very useful

• e.g., How many hours and minutes are there in 375 minutes?

• \( 375 \div 60 \) \hspace{1cm} 6 hours

• \( 375 \mod 60 \) \hspace{1cm} 15 minutes
or how about --- elective or required?

• Suppose courseNum is a 3-digit int variable, with the 2\textsuperscript{nd} digit representing whether a course is required (even) or elective (odd)

• Question: Find the 2nd digit.
  – e.g., say courseNum is 253  find the ‘5’:
or how about --- elective or required?

• Suppose courseNum is a 3-digit int variable, with the 2\textsuperscript{nd} digit representing whether a course is required (even) or elective (odd)
• Question: Find the 2nd digit.
  – e.g., say courseNum is 253 find the ‘5’:

\[
\begin{align*}
(253 / 10) \% 10 \\
25 \% 10 \\
5
\end{align*}
\]
or how about --- elective or required?

- Suppose courseNum is a 3-digit int variable, with the 2\textsuperscript{nd} digit representing whether a course is required (even) or elective (odd).
- Question: Find the 2nd digit.
  - e.g., say courseNum is 253  find the ‘5’:

\[(253 / 10) \% 10\]
\[25 \% 10\]
\[5\]

\[
(courseNum / 10) \% 10
\]
Numbers

- Numbers can be input in a variety of ways but a couple of basic rules
- No commas (1,000 won’t work!)
- No spaces (1 000 000 won’t work!)

- A decimal point implies that the value is a real or floating point number and can have a decimal or fractional portion.
Some Samples of Numbers

1
1.0
1.
1000
3.141592
1.3e12 \ (1.3 \times 10^{12})
2 types of numbers

- integers
  - no fractional part
- real
  - decimal present

We have to tell the compiler what each storage value means!
“I’d like a double mocha, no, make it an int.”

int primitives

small    medium    tall    grande

in C++:
char      short     int      long
Binary Numbers

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
Binary Numbers

- 0 000
- 1 001
- 2 010
- 3 011
- 4 100
- 5 101
- 6 110
- 7 111
- 8 1000
int primitives – cup size – depends on system

char
1 byte
-128 to 127

short
2 bytes
-32768 to 32767

int
4 bytes
-2147483468 to 2147483467

long
4 bytes
-2147483468 to 2147483467
If you want to know integer primitives sizes/ranges

• INT_MIN through INT_MAX

```
#include <climits>

cout << INT_MIN << endl;
cout << INT_MAX << endl;
```
If you want to know integer primitives sizes/ranges

• INT_MIN through INT_MAX

```cpp
#include <climits>

cout << INT_MIN << endl;
cout << INT_MAX << endl;
```
If you want to know integer primitives sizes/ranges

• INT_MIN through INT_MAX

```cpp
#include <climits>
cout << INT_MIN << endl;
cout << INT_MAX << endl;
```

• Same for:
  - CHAR_MIN, CHAR_MAX
  - SHRT_MIN, SHRT_MAX
  - LONG_MIN, LONG_MAX
Floating-Point Types: real #s

**float**
- 4 bytes
- 1.17549e-38
- 3.40282e+38

**double**
- 8 bytes
- 2.22507e-308 to
- 1.79769e+308
Floating-Point Types: real #s

#include <cfloat>
cout << FLT_MIN << " " << FLT_MAX
    << DBL_MIN << " " << DBL_MAX;

float
4 bytes
1.17549e-38
3.40282e+38

double
8 bytes
2.22507e-308 to
1.79769e+308
Floating-Point Types: real #'s

#include <cfloat>

cout << FLT_MIN << " " << FLT_MAX
    << DBL_MIN << " " << DBL_MAX;

float
4 bytes
1.17549e-38
3.40282e+38

double
8 bytes
2.22507e-308 to
1.79769e+308
Character Type

• char: a numeric (integral) type
• Syntax: character enclosed in single quotes
  – e.g. 'A' $ '5' 'e'
• Represented in computer by numeric codes (ASCII codes)
  – e.g., 'A' stored as 65 (01000001)
  – Can be viewed as binary representation of integer 65
  – 8 bits = 1 byte

• Appendix E (page 1028) in DW book
<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>Null</td>
<td>32</td>
<td>20</td>
<td>Space</td>
<td>64</td>
<td>40</td>
<td>U</td>
<td>96</td>
<td>60</td>
<td>`</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>Start of heading</td>
<td>33</td>
<td>21</td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>A</td>
<td>97</td>
<td>61</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>Start of text</td>
<td>34</td>
<td>22</td>
<td>&quot;</td>
<td>66</td>
<td>42</td>
<td>B</td>
<td>98</td>
<td>62</td>
<td>b</td>
</tr>
<tr>
<td>3</td>
<td>03</td>
<td>End of text</td>
<td>35</td>
<td>23</td>
<td>#</td>
<td>67</td>
<td>43</td>
<td>C</td>
<td>99</td>
<td>63</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td>04</td>
<td>End of transmit</td>
<td>36</td>
<td>24</td>
<td>$</td>
<td>68</td>
<td>44</td>
<td>D</td>
<td>100</td>
<td>64</td>
<td>d</td>
</tr>
<tr>
<td>5</td>
<td>05</td>
<td>Enquiry</td>
<td>37</td>
<td>25</td>
<td>%</td>
<td>69</td>
<td>45</td>
<td>E</td>
<td>101</td>
<td>65</td>
<td>e</td>
</tr>
<tr>
<td>6</td>
<td>06</td>
<td>Acknowledge</td>
<td>38</td>
<td>26</td>
<td>&amp;</td>
<td>70</td>
<td>46</td>
<td>F</td>
<td>102</td>
<td>66</td>
<td>f</td>
</tr>
<tr>
<td>7</td>
<td>07</td>
<td>Accept</td>
<td>39</td>
<td>27</td>
<td>'</td>
<td>71</td>
<td>47</td>
<td>G</td>
<td>103</td>
<td>67</td>
<td>g</td>
</tr>
<tr>
<td>8</td>
<td>08</td>
<td>Space</td>
<td>40</td>
<td>28</td>
<td>(</td>
<td>72</td>
<td>48</td>
<td>H</td>
<td>104</td>
<td>68</td>
<td>h</td>
</tr>
<tr>
<td>9</td>
<td>09</td>
<td>Horizontal tab</td>
<td>41</td>
<td>29</td>
<td>)</td>
<td>73</td>
<td>49</td>
<td>I</td>
<td>105</td>
<td>69</td>
<td>i</td>
</tr>
<tr>
<td>10</td>
<td>0A</td>
<td>Vertical tab</td>
<td>42</td>
<td>2A</td>
<td>^</td>
<td>74</td>
<td>4A</td>
<td>J</td>
<td>106</td>
<td>6A</td>
<td>j</td>
</tr>
<tr>
<td>11</td>
<td>0B</td>
<td>Form feed</td>
<td>43</td>
<td>2B</td>
<td>_</td>
<td>75</td>
<td>4B</td>
<td>K</td>
<td>107</td>
<td>6B</td>
<td>k</td>
</tr>
<tr>
<td>12</td>
<td>0C</td>
<td>Carriage return</td>
<td>44</td>
<td>2C</td>
<td>`</td>
<td>76</td>
<td>4C</td>
<td>L</td>
<td>108</td>
<td>6C</td>
<td>l</td>
</tr>
<tr>
<td>13</td>
<td>0D</td>
<td>Shift out</td>
<td>45</td>
<td>2D</td>
<td>-</td>
<td>77</td>
<td>4D</td>
<td>M</td>
<td>109</td>
<td>6D</td>
<td>m</td>
</tr>
<tr>
<td>14</td>
<td>0E</td>
<td>Shift in</td>
<td>46</td>
<td>2E</td>
<td>.</td>
<td>78</td>
<td>4E</td>
<td>N</td>
<td>110</td>
<td>6E</td>
<td>n</td>
</tr>
<tr>
<td>15</td>
<td>0F</td>
<td>Control H</td>
<td>47</td>
<td>2F</td>
<td>/</td>
<td>79</td>
<td>4F</td>
<td>O</td>
<td>111</td>
<td>6F</td>
<td>o</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>Data link escape</td>
<td>48</td>
<td>30</td>
<td>0</td>
<td>80</td>
<td>50</td>
<td>P</td>
<td>112</td>
<td>70</td>
<td>p</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>Device control 1</td>
<td>49</td>
<td>31</td>
<td>1</td>
<td>81</td>
<td>51</td>
<td>Q</td>
<td>113</td>
<td>71</td>
<td>q</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>Device control 2</td>
<td>50</td>
<td>32</td>
<td>2</td>
<td>82</td>
<td>52</td>
<td>R</td>
<td>114</td>
<td>72</td>
<td>r</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>Device control 3</td>
<td>51</td>
<td>33</td>
<td>3</td>
<td>83</td>
<td>53</td>
<td>S</td>
<td>115</td>
<td>73</td>
<td>s</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>Device control 4</td>
<td>52</td>
<td>34</td>
<td>4</td>
<td>84</td>
<td>54</td>
<td>T</td>
<td>116</td>
<td>74</td>
<td>t</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>Neg acknowledge</td>
<td>53</td>
<td>35</td>
<td>5</td>
<td>85</td>
<td>55</td>
<td>U</td>
<td>117</td>
<td>75</td>
<td>u</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>Synchronous idle</td>
<td>54</td>
<td>36</td>
<td>6</td>
<td>86</td>
<td>56</td>
<td>V</td>
<td>118</td>
<td>76</td>
<td>v</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>End trans. back</td>
<td>55</td>
<td>37</td>
<td>7</td>
<td>87</td>
<td>57</td>
<td>W</td>
<td>119</td>
<td>77</td>
<td>w</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>Cancel</td>
<td>56</td>
<td>38</td>
<td>8</td>
<td>88</td>
<td>58</td>
<td>X</td>
<td>120</td>
<td>78</td>
<td>x</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>End of medium</td>
<td>57</td>
<td>39</td>
<td>9</td>
<td>89</td>
<td>59</td>
<td>Y</td>
<td>121</td>
<td>79</td>
<td>y</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>Substitution</td>
<td>58</td>
<td>3A</td>
<td>A</td>
<td>90</td>
<td>5A</td>
<td>Z</td>
<td>122</td>
<td>7A</td>
<td>z</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>Escape</td>
<td>59</td>
<td>3B</td>
<td>B</td>
<td>91</td>
<td>5B</td>
<td>\</td>
<td>123</td>
<td>7B</td>
<td>`</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>File separator</td>
<td>60</td>
<td>3C</td>
<td>C</td>
<td>92</td>
<td>5C</td>
<td>`</td>
<td>124</td>
<td>7C</td>
<td>`</td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>Group separator</td>
<td>61</td>
<td>3D</td>
<td>D</td>
<td>93</td>
<td>5D</td>
<td>]</td>
<td>125</td>
<td>7D</td>
<td>]</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>Record separator</td>
<td>62</td>
<td>3E</td>
<td>E</td>
<td>94</td>
<td>5E</td>
<td>^</td>
<td>126</td>
<td>7E</td>
<td>^</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>Unit separator</td>
<td>63</td>
<td>3F</td>
<td>?</td>
<td>95</td>
<td>5F</td>
<td>@</td>
<td>127</td>
<td>7F</td>
<td>@</td>
</tr>
</tbody>
</table>

ASCII Tables
Other Type – more later

- bool
  - true or false
We now have containers!!
Pick one.
Most coffeehouse only need size
C++ also needs a **name**.

**sum**
Name/Identifier Rules

- Start with a letter or underscore ("_")
- After the first character, you can use numbers as well. Just DON’T start it with a number.
- Can’t be reserved word

- Max size 31 chars (most systems)
- C++ is case-sensitive (int != Int)
- System identifiers usually start with ‘_’
Variables must have a type
Variables must have a name

int count;

- type
- (size)
- name
Variables

• Declared at start of program or module. (Before they are used!!)

• Example:
  – int value;
  – float pi;
  – int a, b, c;
  – char oneChar;
What does a Declaration do?

- Sets up a storage location
- Does not put anything into the storage location (yet!!)

int value;
Get values into variable

```cpp
int sum;
sum = 11;

can also do:
int sum = 11;

or
int sum(11);

or
cout << "Enter a value ", cin >> sum;
```
Declaration + Initialization

int value = 4;
double pi = 3.141592;
char oneChar = ‘G’;

int value ( 4 );
double pi ( 3.141592 );
char oneChar ( ‘G’ );

Initialization by Reading in a value

cout << “Enter an integer value: “;
cin >> value;
cout << “Enter your character choice: “;
cin >> oneChar;
Assignment Statements

value = 4;
pi = 3.141592;
a = pi * (value * value);
b = pi * 2 * value;
oneChar = ‘G’;
i = i+1;

• Done in order
• These are not formulas – They do the calculation on the Right Hand Side and places the result into the variable on the left hand side (sometimes referred to as lvalue).
Assignment Statements

value ← 4;
pi ← 3.141592;
a ← pi * (value * value);
b ← pi * 2 * value;
oneChar ← ‘G’;
i ← i+1;

• Done in order
• These are not formulas – They do the calculation on the Right Hand Side and places the result into the variable on the left hand side (sometimes referred to as lvalue).
Assignment Statements

value = 4;
pi = 3.141592;
a = pi * (value * value);
b = pi * 2 * value;
oneChar = ‘G’;
i = i+1;

• Done in order

• These are not formulas – They do the calculation on the Right Hand Side and places the result into the variable on the left hand side (sometimes referred to as lvalue).
What value is stored?

float a;
float b;

a = 8.5;
b = 9.37;
What value is stored?

```plaintext
a = 8.5;
b = 9.37;
```

```plaintext
a = b;
```

```
8.5
a
```

```
9.37
b
```

```
9.37
```

```
a
```

```
b
```

```
9.37
```

```
a
```

```
b
```

```
9.37
```

```
9.37
```
• int a;
• cout << a << endl;
What happens?

```java
int giraffe = 23;
int rabbit = 3;
rabbit = giraffe;
```
What happens?

double giraffe = 23456.789;
int rabbit = 3;

rabbit = giraffe;
You really don’t want to spill that...

You can’t put a large value into a small cup.

- Well, OK, you can, but you’ll lose some. You’ll get, as we say, spillage. The compiler tries to help prevent this if it can tell from your code that something’s not going to fit in the container (variable/cup) you’re using.
You can’t pour a double-full of stuff into an int-sized container without spillage.

```cpp
#include <iostream>

using namespace std;
int main( void ) {
    double x = 6.4;
    int a = x;
    cout << a << endl;
    return 0;
}
```
Difficulty

- You can’t pour a double-full of stuff into an int-sized container
- without spillage.

```cpp
#include <iostream>

using namespace std;

int main( void )
{
    double x = 6.4;
    int a = x;
    cout << a << endl;  // prints 6
    return 0;            // the decimal portion spills
}
```
Larger will NOT fit into smaller
Something will GO

sizeof:

bool = char \leq \text{short} \leq \text{int} \leq \text{long} \leq \text{float} \leq \text{double}
Smaller into larger
Room to spare
What is stored?

double someDouble;

someDouble = 12;

// causes implicit type conversion

12.0

someDouble

double

int
int someInt;
someInt = 4.8;

// causes implicit type conversion
4

What is stored?
Check it out
Type Conversion in Assignment

```c
float x;
int i;
char ch;

i = 25;  //OK
ch = '$';  //good
x = 0.5;  //float ← double
x = 4;  //float ← int
i = 3.7;  //int ← double
// truncation
```
What to do when have different data types?

• In a mixed type expression,
  – each sub-expression is *promoted* to the "highest" type prior to evaluation
• Type Promotion Guidelines
  bool = char ≤ short ≤ int ≤ long ≤ float ≤ double

• What is the type of
  ‘A’ + 3 * 5.0  ?
double x;

x = 11 / 5;
More Example Assignments

double x;

x = 11 / 5; // x ← 2.0
More Example Assignments

double x;

x = 11 / 5;    // x ← 2.0

x = 11.0 / 5;
More Example Assignments

double x;

x = 11 / 5;  // x ← 2.0

x = 11.0 / 5;  // x ← 2.2
Common Assignment Errors

double x,y;
char ch;
int i,j;

x = x % y;
Common Assignment Errors

double x,y;
char ch;
int i,j;

x = x % y;   // syntax error
Common Assignment Errors

double x,y;
char ch;int i,j;

x = x % y;           // syntax error

ch = $;
Common Assignment Errors

double x,y;
char ch;
int i,j;

x = x % y;  // syntax error

ch = $;    // syntax error
Common Assignment Errors

double x,y;
char ch;
int i,j;

x = x % y;  // syntax error

ch = $;     // syntax error

i = 5;
j = 0;
x = i / j;
Common Assignment Errors

double x,y;
char ch;
int i,j;

x = x % y;    // syntax error

ch = $;    // syntax error

i = 5;
j = 0;
x = i / j;    // runtime error
Type Casting

- We can explicitly convert types by using casting operators

- Syntax -- old form:
  (int) x
  (double) i * 3

- Syntax -- new form:
  static_cast<int> (x)
  static_cast<double>(i * 3)
Cast Examples – old form

int i = 1, j = 3;

double x = i / j;
Cast Examples – old form

int i = 1, j = 3;

double x = i / j;          // x ← 0.0
int i = 1, j = 3;

double x = i / j;  // x ← 0.0

x = (double) i / (double) j;
int i = 1, j = 3;

double x = i / j;  // x ← 0.0

x = (double) i / (double) j;  // x ← 0.33
Cast Examples – old form

int i = 1, j = 3;

double x = i / j;  // x ← 0.0

x = (double) i / (double) j;  // x ← 0.33

x = (double) (i / j);
Cast Examples – old form

int i = 1, j = 3;

double x = i / j; // x ← 0.0

x = (double) i / (double) j; // x ← 0.33

x = (double) (i / j); // x ← 0.0
Cast Examples – new form

```cpp
int i = 1, j = 3;

double x = i / j; // x ← 0.0

x = static_cast<double> (i) / static_cast<double> (j); // x ← 0.33

x = static_cast<double> (i / j); // x ← 0.0
```
Fun with Casting

double x = 3.7;

i = (int) (x);
Fun with Casting

double x = 3.7;

i = (int) (x);    // i ← 3
Fun with Casting

double x = 3.7;

i = (int) (x); // i ← 3

i = (int) (x + 0.5); // i ← 4 (rounds x)
Outputting a Variable’s Value

- to output a variable’s value, you just need to put it in the `cout` statement
- No " " needed

```cpp
cout << a << endl;
cout << value << endl;
```
Sharpen your pencil

• From the following list, circle the statements that would be legal if these lines were in a single main program.

```c
int x = 34.5;
bool boo = x;
int g = 17;
int y = g;
y = y + 10;
short s;
s = y;
int b = 3;
int v = b;
short n = 12;
v = n;
short k = 33000;
float y = 9.5;
int p = 3 * g + y;
```
Basics

Across
3. Can't pin it down
4. Acronym for a chip
7. What's a prompt good for?
8. Just gotta have one
10. RUN
13. You're never going to change!!!
14. Could be called "Father"

Down
1. Quite a crew of characters
2. Not an integer (or ____ your boat)
3. Nothing is there
5. Source code consumer
6. Acronym for your laptop's power
9. Announce a new variable
11. Number variable type
12. Department of LAN jockeys
13. Say something