Data Types

Class 4

The char Data Type

- unfortunately, a fairly confusing type
- used for single characters, e.g., 'A' or '7';
- also an integer type that can be used for numbers

- we will never use char in this way, only for characters

- in a C++ program, you can use a char variable when you need a very small range of values
- however, when you print the value of a char to the screen, it prints as a character, not as a number

look at program using_char.cpp

- look at ASCII chart on page 1287

The char Data Type

- most programmers do not use char for integers like this:
  ```
  char a_character = 81;
  ```
- instead, they are used for character data
  ```
  char a_character = 'Q';
  ```
- literal character values must appear in single quotes

Strings

- another confusing topic
- C++ has old-fashioned built-in strings, inherited from C, called C-strings
- C++ has the new-fashioned string class, unique to C++
- we will study both in depth later
- for now, we will encounter C-strings only as string literals
- here is a C-string literal, denoted by double quotes:
  ```
  cout << "Hello, world";
  ```

The string Class

- if you declare a variable of type string, you get a new-fangled string object:
  `string message = "Hello, world";`
- to use features of the string class, you must include the string library:
  `#include <string>`

Floating-Point Data Types

- in a computer, integer values and floating-point values are stored and manipulated in completely different ways
- different circuitry in the CPU is used for each
- internally, floating point numbers are stored in scientific notation, consisting of a mantissa and an exponent
- typically written in e-notation $2.34 \times 10^5 = 2.34\mathrm{E}5$

Type Name	ice	Min Value	Max Value	Epsilon
float	4 bytes	$\pm 1.17 \times 10^{-38}$	$\pm 3.40 \times 10^{38}$	1.19209×10^{-07}
double	8 bytes	$\pm 2.22 \times 10^{-308}$	$\pm 1.80 \times 10^{308}$	2.22045×10^{-16}
long double	16 bytes	$\pm 3.36 \times 10^{-4932}$	$\pm 1.19 \times 10^{4932}$	1.0842×10^{-19}

- note that there are about 10^{80} atoms in the known universe

Floating Point Literals

- almost no one uses float
- the compiler assumes that any floating-point literal is a double
- you can force a literal to be a long double by appending L: `long double my_value = 23.5L;`
- if you try to store a double value in a float variable, or a long double value in a float or double variable, the compiler issues a warning: `float my_value = 23.5;`

- you can use a variety of formats for floating point literals
- all these are equivalent:
  - 1.4959E11
  - 1.4959e11
  - 1.4959E+11
  - 149590000000.0

Conversion

- in algebra, we typically don't distinguish between integers and floating-point value: $x = 2 + \frac{1}{2}$
- however, in a computer program, these are fundamentally different things, and normally you should rarely mix them
- it is not legal to try to store a floating-point value in an integer variable: `int my_value = 12.34;`
- it is legal to do the opposite: `double my_value = 5;` but this is sloppy and you should never do it (even though Gaddis does)
- later we will see how to explicitly convert between integers and floating point values, but for now, don't mix them

Inexact Values

WARNING WARNING WARNING WARNING WARNING WARNING!

- in a computer, integer values are stored exactly, with no error
- floating point values in general cannot be stored exactly
- every time you use floating point numbers, you will encounter rounding errors

look at program float_inexact.cpp

The bool Data Type

- a char variable can have any one of 256 different values, $-128, -127, \ldots, 126, 127$
- an int variable can have any one of 2^{32} different values
- a bool variable can have either of two values, true or false
- true and false are not strings, they are literal values
- we will study Boolean variables and values in chapter 4

Our Data Types

- in spite of what Gaddis says, we will use just these primitive data types:

unsigned for counting and whole number quantities that cannot be negative

int for whole number quantities that might be negative

double for measured values and quantities with fractional parts

char for characters

bool for truth values

Our Data Types

- in spite of what Gaddis says, we will use just these primitive data types:

unsigned for counting and whole number quantities that cannot be negative

int for whole number quantities that might be negative

double for measured values and quantities with fractional parts

char for characters

bool for truth values

- later, we will add another type, size_t, for working with arrays
- and complex non-primitive types such as strings, structs, and classes