Data Types – Declarations and Initializations

Larry Caretto
Computer Science 106

Computing in Engineering and Science

February 7, 2005

California State University Northridge

Outline

- Review last week
- Meaning of data types
- Integer data types have no decimal point and integer division truncates
- Floating point data types: approximate representation of decimal numbers
- Character, string and boolean (logical) data types

Review of Last Week

- Basic elements of C++ programs
  - Template #include, using namespace std; int main(), return EXIT_SUCCESS;
  - C++ is case sensitive
  - White space does not matter except in string constants
  - End statements with a semicolon (;)
  - Braces { and } frame logical code blocks
  - Will use in lab this week

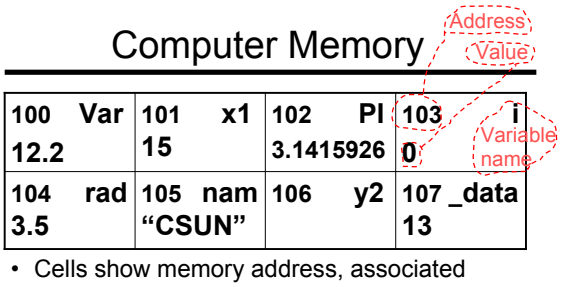
Review of Last Week II

- Screen output using cout and <<
- Keyboard input using cin and >>
- Variables refer to memory locations
  - Use letters, numbers and _
  - Start with letter or _
  - 31 characters maximum
  - Use meaningful names
  - Variables are case sensitive (x2 is not X2)

Review Variables and Memory

- Program variables refer to computer memory (RAM) locations
- When we use a variable (say x) we get the value in the memory location the compiler associated with this variable
  - x = 3; // assigns the value 3 to x
  - cout << x; // writes value of x to the screen
  - y = x; // assigns value of x to y
  - x = x + 3; // replaces x by x + 3

Computer Memory



- Cells show memory address, associated variable name and value stored (if any)
- What is effect of y2 = PI * rad * rad?
  - What happens to cell 106? to cells 102 and 104?
  - Cell 106 gets the new value of $\pi(3.5)^2$

Cells 102 and 104 are not changed

Computer Memory

100 Var 12.2	101 x1 15	102 PI 3.1415926	103 i 0
104 rad 3.5	105 nam “CSUN”	106 y2	107 _data 13

- What is effect of _data = x1?
  - The value of 15 in cell 101 is stored in cell 107
  - What happens to the value of 13 in cell 107?
  - It is lost forever
  - What happens to the value of 15 in cell 101?

Nothing

Computer Memory

100 Var 12.2	101 x1 15	102 PI 3.1415926	103 i 0
104 rad 3.5	105 nam “CSUN”	106 y2	107 _data 13

- What is effect of var = var + 3.5?
  - What happens to cell 100?
  - The value of 12.2 is replaced by 12.2 + 3.5 = 15.7
  - Are any other cells affected?
  - No

Computer Memory

100 Var 12.2	101 x1 15	102 PI 3.1415926	103 i 0
104 rad 3.5	105 nam “CSUN”	106 y2	107 _data 13

- What is effect of var = var + rad on cell 104? on cell 100?
  - The value of 3.5 in cell 104 does not change
  - The value of 12.2 in cell 100 is replaced by the result 12.2 + 3.5 = 15.7

Computer Memory

100 Var 12.2	101 x1 15	102 PI 3.1415926	103 i 0
104 rad 3.5	105 nam “CSUN”	106 y2	107 _data 13

- Can you do y2 = nam?
  - This is a trick question. You have no basis for answering it yet. It depends on the type for y2.
  - If y2 can hold strings the operation is okay
  - If y2 cannot hold strings this is a syntax error

Data Types

- Computer memory stores binary information – a string of ones and zeros
- How are these ones and zeros interpreted? Depends on data type
- Analogy – How do you interpret the following string of characters?

chair

- Depends on language

Integer Data Types

- There are several of these including **int**, the only one we will use in this course
- Integer data types have no decimal part
- When we divide two integers the decimal part is truncated (lost)
  - Not rounded, but truncated
  - What is 12/7? 1
  - What is 4/5? 0

Integer Data Types II

- Range of values for different types
  - Based on number of bits in computer member allocated for type
  - **short** -32,768 to 32,767
  - **int** -2,147,483,648 to 2,147,483,647
  - **long** -2,147,483,648 to 2,147,483,647
  - **unsigned short** 0 to 65,535
  - **unsigned int** 0 to 4,294,967,295
  - **unsigned long** 0 to 4,294,967,295

Integer Data Types III

- Range depends on compiler vendor
  - Ranges shown on previous page are common and are used in Visual C++
  - Only requirement for standard is that digits for long ≥ digits for int ≥ digits for short
  - Range comes from binary storage
  - 16 bits stores a number from 0 to $2^{16} - 1$
  - This is actual range for unsigned short
  - Short range is -2^{15} to $2^{15} - 1$

Integer Data Types IV

- What is result of following code?

```
short result, maxShort = 32767;
result = maxShort + 1;
cout << "Result = " << result;
```

- Result is -32768, the smallest short integer
- What is result of code where you subtract 1 from minShort = -32768?
- Result, 34767, maximum short integer

Floating Point Data Types

- Numbers with decimal points
- Stored as characteristic plus mantissa
- Numbers not stored exactly
- Smallest type, **float**, occupies four bytes
- Range for float is $-3.402823466 \times 10^{38}$ to $-1.175494351 \times 10^{-38}$, 0, and $1.175494351 \times 10^{-38}$ to $3.402823466 \times 10^{38}$

Approximate Representation

- How do you represent 1/3 as decimal?
  - .3, .33, .333, .3333333333333 or ?
  - Representation should be accurate enough for calculations but will never be exact
  - Similar problem on computers
  - Fractional numbers like 1/10 = 0.1 are represented as binary fractions
    - 1/10 is $.0001100110011001100110011001100\ldots_2$
    - 10 * 0.1 will not be exactly one

Floating Point Data Types II

- Type float has about seven significant figures of accuracy
- When does $1 + \varepsilon = 1$?
  - When ε is so small that adding it to 1 does not change the significant figures available
  - For float, $\varepsilon = 1.19 \times 10^{-7}$
- Other floating point types are **double** and **long double**
  - Both same in Visual C++

Floating Point Data Types III

- Type double has about 15–16 significant figures of accuracy
  - Range is -1.7976931348623158x10^{308} to -2.2250738585072014x10^{-308}, 0, and 2.2250738585072014x10^{-308} to 1.7976931348623158x10^{308}
  - For double, $\varepsilon = 2.2 \times 10^{-16}$
- Generally use type double for modern engineering/science/mathematics codes

Other Data Types

- **char** data type represents single characters (constant example: 'a')
- **string** data type represents a string of characters (constant example: "string")
  - Requires #include<string> declaration
- **bool** data type used in logical operations
  - Only two possible values for type **bool** variables: **true** and **false**

Constant Types

- Program constants have types
  - integer 37 is type int
  - decimal 123.4 or 1.234e2 is type double
  - Character (**char**) 'c',
  - string "This is a string constant"
    - The only place where spaces matter
  - logical (**bool**ean) true false
- Can force constants to other types by adding a letter (e.g. 37L is a long int)

Declaring Variables

- All variables must be declared as belonging to a specific data type the first time they appear in the code
  - Failure to do so is a syntax error
- Can initialize a variable with declaration (recommended practice) or subsequent to declaration
- Can declare several variables in a single statement

Examples

int x; // length
int y; // width
int z; // height
	int x = 2, // length
	y = 4, // width
	z = 7; // height

double x, y; char YES = 'Y', NO = 'N';	double radius = 10; char c = '9'; string name = "CSUN";
bool converged = false; int maxIterations = 100;	double maxAllowedError = 1e-13;

Summary

- Variables represent memory locations
- Must declare variables before use
- Data types for variables
  - Integer types (e.g., int) for counting
  - Floating point types (e.g., double) for calculations
    - Do not represent numbers exactly
  - Other types are char (individual characters), string, and bool (true and false)

True-False Review Quiz

- You do not have to declare a variable before using it in a program ***False***
- Declaring a variable is the same as assigning it an initial value ***False***
- You can declare a variable and assign it a value at the same time ***True***
- Subtracting one from the minimum short integer gives the maximum short ***True***

Assignments

- Reading pages in text
  - Today: 45-61, 100-106
  - Thursday: 64-72, 93-98
  - Next Tuesday, February 14:  117-137 117-137
- Recommended homework for this week: pages 75, problems 7 and 8; page 78, problem 27
- Exercise one may be submitted late
- Exercise two due Thursday (9 Feb)