Jim Lambers
ENERGY 211 / CME 211
Autumn Quarter 2007-08
Homework Assignment 1 Solution

Chapter 2

2.1 What is the difference between an `int`, `long` and a `short` value?

Solution The difference is in the number of bytes of storage that a variable of each type occupies. A `short` must occupy at least 2 bytes, and a `long` must be at least 4. An `int` typically occupies 4 bytes as well.

2.2 What is the difference between an `unsigned` and a `signed` type?

Solution An `unsigned` type reserves all of its bits for representing values, all of which must be nonnegative. A `signed` type reserves one of its bits for the sign, so in exchange for being able to represent negative values, the magnitude of values it can represent is less than for an `unsigned` type.

2.3 If a `short` on a given machine has 16 bits then what is the largest number that can be assigned to a `short`? To an `unsigned short`?

Solution Since 15 bytes are used to represent a `short`, with the 16th bit used for the sign, the largest number it can represent is $2^{15} - 1 = 32,767$. For an `unsigned short`, all 16 bits are used to represent the value, so the largest representable number is $2^{16} - 1 = 65,535$.

2.4 What value is assigned if we assign 100,000 to a 16-bit `unsigned short`? What value is assigned if we assign 100,000 to a plain 16-bit `short`?

Solution If we assign 100,000 to an `unsigned short`, then the value that will be stored is 100,000 modulo 2^{16}, or, equivalently, the binary number represented by the least significant 16 bits of 100,000, which is $100,000 - 65,536 = 34,464$. On the other hand, the value that would be assigned to a signed `short` is compiler-dependent. The compiler included with MDS assigns the value $-31,072$. This is the value obtained by taking the least significant 16 bits of 100,000 and interpreting them as a signed `short`. This causes the most significant bit to be interpreted as the sign bit.

2.5 What is the difference between a `float` and a `double`?

Solution A `float` must occupy at least 4 bytes, while a `double` must occupy at least 8. This allows a `double` to represent numbers of much greater magnitude, and with much greater precision.

2.6 To calculate a mortgage payment, what types would you use for the rate, principal and payment? Explain why you selected each type.

Solution A reasonable choice is to use a `double` for all of these values. Certainly an integral type is not appropriate, because none of these values are normally integers. A `float` could be used, but there is little reason to give up the precision of a `double`, considering that floating-point arithmetic is typically performed in double precision anyway.

2.7 Explain the difference between the following sets of literal constants:

(a) `'a'`, `L'a'`, `"a"`, `L"a"`
Solution The types of these literals are a character, a wide character, a string, and a wide-character string.

(b) `10`, `10u`, `10L`, `10uL`, `012`, `0xC`
Solution The types of these literals are `int`, `unsigned int`, `long`, `unsigned long`, `int` (octal), and `int` (hexadecimal).

(c) `3.14`, `3.14f`, `3.14L`
Solution The types of these literals are `double`, `float` and `long double`.

2.8 Determine the type of each of these literal constants:

(a) `-10`
Solution `int`

(b) `-10u`
Solution `unsigned int`

(c) `-10.`
Solution `double`

(d) `-10e-2`
Solution `double`

2.9 Which, if any, of the following are illegal?

(a) `"Who goes with F\145rgus?\12"`
Solution Legal.

(b) `3.14e1L`

Solution Legal.

(c) `"two" L"some"`

Solution Syntactically, legal, but semantically, illegal. The compiler will not report an error, but the value of this literal is undefined, so a concatenation such as this should *never* be used.

(d) `1024f`

Solution Illegal.

(e) `3.14UL`

Solution Illegal.

(f) `"multiple line`
`comment"`

Solution Illegal.

2.11 Write a program that prompts the user to input two numbers, the base and the exponent. Print the result of raising the base to the power of the exponent.

Solution

```
#include <iostream>

int main()
{
    std::cout << "Enter base:  ";
    int base;
    std::cin >> base;
    std::cout << "Enter exponent:  ";
    int expt;
    std::cin >> expt;
    int result = 1;
    // repeat calculation of result until cnt is equal to expt
    for (int cnt = 0; cnt != expt; ++cnt)
        result *= base; // result = result * base;
    std::cout << base
              << " raised to the power of "
              << expt << ":  \t"
              << result << std::endl;
    return 0;
}
```

2.14 Which, if any, of the following names are invalid? Correct each identified invalid name.

(a) `int double = 3.14159;`

Solution Invalid. `double` is a keyword, so it cannot be used as an name. The variable should be renamed so that its name is not a keyword.

(b) `char _;`

Solution Valid.

(c) `bool catch-22;`

Solution Invalid. An name cannot contain a dash. It should be changed to an underscore.

(d) `char 1_or_2 = ’1’;`

Solution Invalid. An name cannot begin with a digit. The `1` should be preceded by a letter or underscore.

(e) `float Float = 3.14f;`

Solution Valid. Names are case-sensitive, so `float` and `Float` are considered distinct.

2.15 What, if any, are the differences between the following definitions:

```
int month = 9, day = 7;
```

```
int month = 09, day = 07;
```

Solution In the first line, the integer literals are interpreted as decimal, while in the second line, they are interpreted as octal. This makes the literal `09` illegal, as 9 is not a valid octal digit.