Fundamental Data Types

Lecture 4
Sections 2.7 - 2.10

Robb T. Koether

Hampden-Sydney College

Mon, Sep 3, 2018

Outline

Integer Types

Integer Type

```
cout << 123 << endl;
```

- Integers are stored as binary numbers.
- An n-bit integer can hold any of 2^n different values.
- Integer types are either signed or unsigned.
- Integer literals must not have a decimal point.

Integer Types

Integer Types

```
int a = 123;
int b = -456;
unsigned int c = 789;
```

- The integer types.
  - **short** – 2 bytes
  - **unsigned short** – 2 bytes
  - **int** – 4 bytes
  - **unsigned int** – 4 bytes
  - **long** – 4 bytes
  - **unsigned long** – 4 bytes
- Each type can be either signed or unsigned.
- On some computers, a **long** can be 8 bytes.

Signed vs. Unsigned Integers

- Unsigned integers.
  - An unsigned integer cannot be negative.
  - Values range from 0 to $2^n - 1$.
- Signed integers
  - A signed integer can be positive or negative.
  - The high-order bit (sign bit) indicates the sign.
  - Values range from -2^{n-1} to $2^{n-1} - 1$.
  - The *unsigned* values 2^{n-1} to $2^n - 1$ represent the *signed* values -2^n to -1. The signed value is the unsigned value minus 2^n.
    - For example, unsigned $100011111 = 159$, so signed $100011111 = 159 - 256 = -97$.
- By default, integers are signed.

Example of Signed and Unsigned Integers

Binary	Unsigned	Signed
00000000	0	0
00000001	1	1
00000010	2	2
00000011	3	3
⋮	⋮	⋮
01111111	127	127
10000000	128	-128
10000001	129	-127
10000010	130	-126
10000011	131	-125
⋮	⋮	⋮
11111111	255	-1

8-bit integers, unsigned and signed

Ranges of Values of Integer Type

Type	Range	
	From	To
`unsigned short`	0	65535
`short`	$-32,768$	32,767
`unsigned int`	0	4,294,967,295
`int`	$-2,147,483,648$	2,147,483,647

Integer Overflow

- What happens when an integer value becomes too large?
- That is, what if we assign to an integer the largest legal value, and then add 1?
- Example
  - `IntLimitTest.cpp`

Outline

Floating-Point Types

Floating-Point Type

```
cout << 12.34 << endl;
```

- Floating-point numbers are stored in two parts.
- The mantissa contains the significant digits.
- The exponent locates the decimal point.
- Floating-point literals must have a decimal point.

Floating-Point Types

Floating-Point

```
float a = 123.456;
double b = 123.456789012345;
```

- The floating-point types.
  - **`float`** – 4 bytes
  - **`double`** – 8 bytes
  - **`long double`** – 8 bytes

Single-Precision Numbers

`float` Type

```
float x = 12.34567;
```

- Single-precision floating point numbers (**`floats`**) occupy 4 bytes of memory.
  - 8-bit exponent, including the sign.
  - 24-bit mantissa, including the sign.
- The positive values range from a minimum of $\pm 1.17549 \times 10^{-38}$ to a maximum of $\pm 3.40282 \times 10^{38}$.
- Approximately 7 decimal-digit precision.

Double-Precision Numbers

`double` Type

```
double pi = 3.141592653589793;
```

- Double-precision floating point numbers (**`doubles`**) occupy 8 bytes of memory.
  - 11-bit exponent, including the sign.
  - 53-bit mantissa, including the sign.
- The values range from a minimum of $\pm 2.22507 \times 10^{-308}$ to a maximum of $\pm 1.79769 \times 10^{308}$.
- Approximately 16 decimal-digit precision.

Floating-Point Overflow and Underflow

- What happens if we begin with the largest possible **`float`** and then double it?
- What happens if we begin with the smallest possible positive **`float`** and divide it by 2?
- Example
  - `FloatLimitTest.cpp`

Outline

The Character Type

`char` Type

```
char letter = 'a';
```

- Characters (**chars**) are stored as one-byte integers using the ASCII values (see p. 1155).
- A character object can have any of 256 different values.
- Character literals must use single quotation marks, e.g., `'A'`.

ASCII Table

ASCII	Char
0	NULL
1	SOTT
2	STX
3	ETY
4	EOT
5	ENQ
6	ACK
7	BELL
8	BKSPC
9	HZTAB
10	NEWLN
11	VTAB
12	FF
13	CR
14	SO
15	SI

ASCII	Char
16	DLE
17	DC1
18	DC2
19	DC3
20	DC4
21	NAK
22	SYN
23	ETB
24	CAN
25	EM
26	SUB
27	ESC
28	FS
29	GS
30	RS
31	US

ASCII	Char
32	(space)
33	!
34	"
35	#
36	$
37	%
38	&
39	‘
40	(
41	)
42	*
43	+
44	’
45	_
46	.
47	/

ASCII	Char
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?

Characters as Integers

`char` Type

```
char letter = 'a';
int value = letter;
letter = value + 1;
```

- Characters are interchangeable with integers in the range 0 to 255.
- The numerical value of a character is its ASCII value.
  - Blank space (ASCII 32).
  - Digits 0 - 9 (ASCII 48 - 57).
  - Uppercase letters A - Z (ASCII 65 - 90).
  - Lowercase letters a - z (ASCII 97 - 122).
- Characters are ordered according to their ASCII values.

Outline

The `string` Type

`string` Type

```
string message = "Hello, World";
```

- A string is stored as a sequence of characters.
- A string may hold any number of characters, including none.
- String literals must use double quotation marks, e.g., `"Hello"`.

The `string` Type

`string` Type

```
string msg = "Hello, World";
cout << msg[9] << msg[1] << msg[11] << endl;
```

- The characters in the string are indexed, beginning with index 0.
- They can be accessed individually by writing the index within square brackets `[...]`, starting with index `0`.

Floating-Point Overflow and Underflow

- Example
  - `CharCalcs.cpp`

Outline

Assignment

Assignment

- Read Sections 2.7 - 2.10.