C++ Programming

Ch. 3 Dealing with Data

Spring 2014

Myung-Il Roh

Department of Naval Architecture and Ocean Engineering
Seoul National University

Ch. 3 Dealing with Data

Contents

- ☑ Review
- ☑ Variables
- ☑ C++ Built-in Types
  - ■ Integer Variables
  - ■ Character Variables
  - ■ Floating-Point Variables
- ☑ C++ Arithmetic Operators
- ☑ Type Conversions
- ☑ Summary
- ☑ Practice

Review

- ☑ Structure of the C++ Program
- ☑ Programming Practice
  - ■ A program that gets 2 variables, adds variables, and shows its result

Variables (1/3)

☑ All variables must be declared before it is used.

☑ Ex.

```
int x, y;          // declare integer variables x and y
x = 3;             // assign a value to the variable x
y = x + 5;         // assign a value to the variable y
```

Variables (2/3)

☑ Names for Variables

- The only characters you can use in names are
  .
- The first character in a name .
- Uppercase characters are considered distinct from lowercase characters.
- You can't use a C++ keyword for a name.
- Names beginning with two underscore characters or with an underscore character followed by an uppercase letter are reserved for use by the implementation, that is, the compiler and the resources it uses. Names beginning with a single underscore character are reserved for use as global identifiers by the implementation.
  - Caution! It doesn't produce compile error. However, it can lead to undefined behavior.
  - Ex. __time_stop, _Donut
- C++ places , and all characters in a name are significant.

Variables (3/3)

☑ Quiz about Names for Variables

- Put O if it's possible to use as a variable name, or put X if it's impossible.
  - Poodle ( )
  - I_am_going_to_use_this_as_my_fisrt_variable ( )
  - X5 ( )
  - 3x ( )
  - __AB ( )
  - _Com ( )
  - R2D2 ( )
  - int ( )
  - double ( )
  - hello-there ( )
  - GoodMorning ( )
  - My variable ( )

C++ Built-in Types

- bool, char, short, int, long, float, double, long double
  - There are signed type and unsigned type for the integer variables.
- Array: Data form that can hold several values, all of one type
- Structure
  - Data form that can hold several values of differing types
  - Collection of variables under a single name with various types
- String: Array of char type variables

Integer Variables

☑ Integer ('int') Types

- Types of integer variable: (ascending order in data size)

- There are two kind of integers; signed and unsigned integer.
  - short, unsigned short, int, unsigned int, long, unsigned long (ascending order in data size)

- Width of short, int, and long type
  - short: at least 16 bits wide
    - Range of 16 bit signed integer: $-2^{15} \sim 2^{15}-1$
  - int: at least as big as short
  - long: at least 32 bits wide and at least as big as int
    - Range of 32 bit signed integer: $-2^{31} \sim 2^{31}-1$

- Overflow error
  - Runtime error that the value exceeds its limit.

Size of Data Types

☑ Calculation of the Size of Data Types

- ‘sizeof()’ operator: Return the size of a value or data type in byte.
- To use this operator, the ‘climits’ header file should be included.
- Ex.

```
……..

int main(void)
{
    int n_int = INT_MAX;
    cout<<“The size of int is ”<<                <<“bits. \ n”;
    cout<<“int: “<< n_int <<“ \ n”;
    return 0;
}
```

Symbolic Constants in ‘climits’ Header File

Symbolic constant	Represents
CHAR_BIT	Number of bits in a char
CHAR_MAX	Maximum char value
CHAR_MIN	Minimum char value
SCHAR_MAX	Maximum signed char value
SCHAR_MIN	Minimum signed char value
UCHAR_MAX	Maximum unsigned char value
SHRT_MAX	Maximum short value
SHRT_MIN	Minimum short value
USHRT_MAX	Maximum unsigned short value
INT_MAX	Maximum int value
INT_MIN	Minimum int value
UINT_MAX	Maximum unsigned int value
LONG_MAX	Maximum long value
LONG_MIN	Minimum long value
ULONG_MAX	Maximum unsigned long value

Character Variables

☑ Character ('char') Types

- 1 byte (= 8 bits) is enough to represent all symbols because the number of symbols are less than 2^8 (= 256).
- The char type could be used as an integer type that is typically smaller than short type.
- The way to write a character constant
  - Enclose the character within two single quotation mark like 'a'.
  - 'a' ≠ "a" // 'a' is a char, "a" is an array of char (string)
  - Escape sequence codes

Character Name	ASCII Symbol	C++ code	ASCII Decimal Code	ASCII Hex Code
Newline	NL(LF)	\n	10	0xA
Horizontal Tab	HT	\t	9	0x9
Backspace	BS	\b	8	0x8
Double quote	"	\"	34	0x22

The const Qualifier

- ☑ The keyword for declaration and initialization of a symbolic constant
- ☑ After setting with the const qualifier,

  .
- ☑ Similar with ‘ ’ keyword in C language
- ☑ Purpose of use
  - ■ Specify the type explicitly.
  - ■ We can use C++ scoping rules to limit the definition to particular functions or files.
- ☑ Expression

  name of value = initial value;
  - ■ Ex.

    MONTHS = 12; // MONTHS becomes a constant as 12

Floating-Point Variables (Real Numbers)

☑ Float-Point Types

- Types of float-point variable: (order in data size ascending)
- Purpose of use
  - To represent real numbers
  - To represent very small or large number
- With of significant figures (in general)
  - float: at least 32 bits (4 bytes) wide
  - double: at least 48 bits, typically 64 bits (8 bytes) wide
  - long double: at least larger than double. 80, 96, 128 bits wide
- Range of exponents
  - At least -37 to +37
- The way to find the limits for the system
  - the ‘cfloat’ or ‘cloat.h’ header file should be included.

* Significant figures: the meaningful digits in a number

Floating-Point Constants

☑ Example of Float-Point Constants

- 1.234f // float constant
- 2.34E20f // float constant
- 2.3455235E28 // double constant
- 2.2L // long double constant

☑ Caution of Declaration

- A warning occurred when declaring a real constant with ‘const float’.
- Ex.

```
const      PI = 3.1415    // Warning
const      PI = 3.1415f   // OK
const         PI = 3.1415 // OK
```

C++ Arithmetic Operators

☑ Basic Arithmetic Operator

- Operator between 2 values (operands)

☑ Order of Operation

- Same as normal arithmetic operation
- '( )' is the first.
- Ex. x = 3 + 4 * 2 – 3 / 3; // What is the value of x?

Type Conversions (1/3)

☑ Type Conversions

- Automatic type conversion vs. Forced type conversion (type casts)

- C++ makes many type conversions automatically to match types to the need.
- Cases of automatic type conversion

: When you assign a value of one arithmetic type to a variable of another arithmetic type

  - Ex. int y = 3.5; // The variable y stores 3 instead of 3.5.

: When you combine mixed types in expressions

  - Integral promotions
    - » bool, char, unsigned char, signed char, and short ➡ converted to int
  - When an operation involves two types, the smaller is converted to the larger.

: When you pass arguments to function

  - C++ function prototyping controls type conversions for the passing of arguments.

Type Conversions (2/3)

☑ Automatic Type Conversion (continued)

- **Ex.**

```
float x, z;
int y;
x = y;              // assign a value to the different type variable
z = 2.0 / 3 * 3;    // varies of types in one statement
cal(2, 3);          // if the function prototype is declared as
                       void cal(float, float);
```

Conversion Type	Potential Problems
Bigger floating-point type to smaller floating-point type such as **double** to **float**	Loss of precision (significant figures); value might be out of range for target type, in which case result is undefined.
Floating-point type to integer type	Loss of fractional part; original value might be out of range for target type, in which case result is undefined.
Bigger integer type to smaller integer type, such as **long** to **short**	Original value might be out of range for target type; typically just the low-order bytes are copied.

Type Conversions (3/3)

- C++ empowers you to force type conversions explicitly via type cast mechanism.
- Type cast operator
  - Expression

```
                                                    // old C syntax
                                                    // new C++ syntax
```

- Ex. These statements add two values into int type.

```
coots = int(19.99) + int(11.99);                    // new C++ syntax
cout << "coots = "<< coots << ' \ n';
char ch = 'Z';
cout << "The code for " << ch << " is ";            // print 'ch' with char type
cout << int(ch) << ' \ n';                          // print 'ch' with int type
```

Summary (1/2)

- ☑ C++'s basic types fall into two groups. One group consists of values that are stored as . The second group consists of values that are stored in .
- ☑ The integer types differ from each other in the amount of memory used to store values and in whether they are signed or unsigned. From smallest to largest, the integer types are

.

- ☑ The floating-point types can represent fractional values and values much larger than integers can represent. The three floating-point types are .

Summary (2/2)

- ☑ By providing a , C++ lets you match the type to particular data requirements.
- ☑ C++ uses : addition, subtraction, multiplication, division, and taking the modulus.
- ☑ C++ when you assign values to a variable, mix types in arithmetic, and use type casts to force type conversions.

Practice

☑ **Make a program that get a radius of a circle from the user and calculate the area and the circumference of the circle.**

- **Define π (Pi) as a constant 3.14159 using the const qualifier.**

```
Preprocessor directives
Define global variable PI.
int main(void)
{
    declare float type variables r,a,p.
    get the radius from user and store the value at r.
    calculate the circumference and store the value at p.
    calculate the area and store the value at a.
    print out the radius, circumference, and area.
    return 0;
}
```

- **What will happen if you declare a variable a and p with int type?**

Reference Slides

[Reference] Representation of Negative Integers

☑ 2's Complement Method

Ex.

Assume that a computer represents a integer with 4 bits. Then, the range of the number that can be represented is between $-2^3 \sim 2^3-1$.

Dec 5 → 0101

Dec -5 → 1101 ? (X) 1011 (O)

The way to represent a negative integer.

Dec 5 ➡ Bin 0101

1010 (1's complement)

+ 1

(2's complement)

The computer performs addition through XOR operation and subtraction through adding the negative number. For example, .

```
  0111
+ 1011
-------
 0100   (= 2)
```

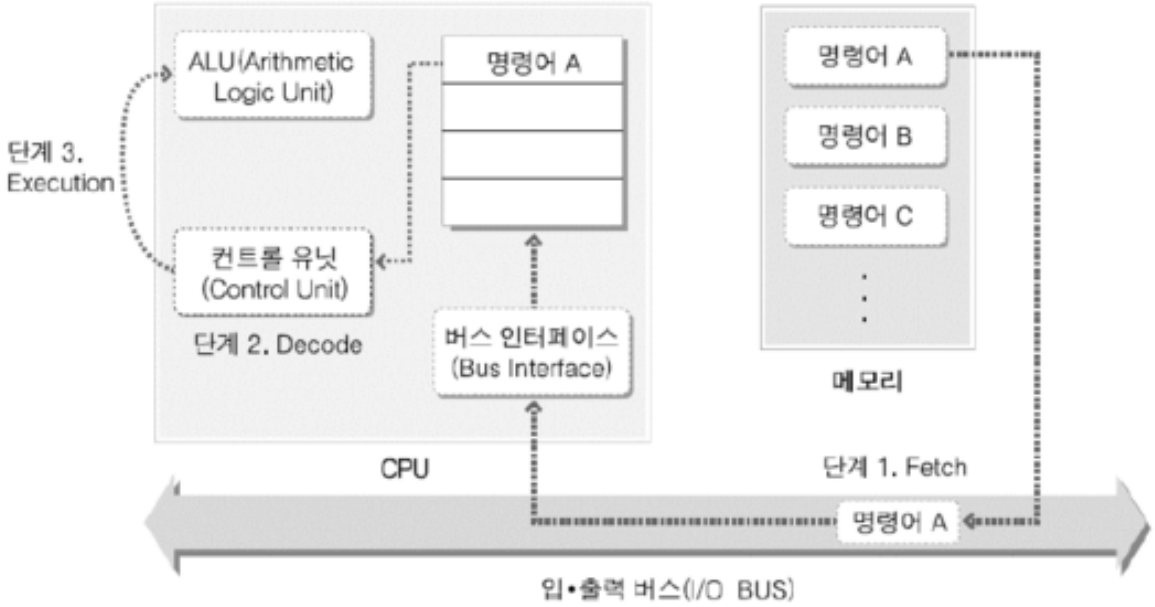
discard carry-out bit.

[Reference] ASCII Code Table

Char	Hex	Oct	Dec	Char	Hex	Oct	Dec	Char	Hex	Oct	Dec	Char	Hex	Oct	Dec
Ctrl-@ NUL	00	000	0	Space	20	040	32	@	40	100	64	`	60	140	96
Ctrl-A SOH	01	001	1	!	21	041	33	A	41	101	65	a	61	141	97
Ctrl-B STX	02	002	2	"	22	042	34	B	42	102	66	b	62	142	98
Ctrl-C ETX	03	003	3	#	23	043	35	C	43	103	67	c	63	143	99
Ctrl-D EOT	04	004	4	$	24	044	36	D	44	104	68	d	64	144	100
Ctrl-E ENQ	05	005	5	%	25	045	37	E	45	105	69	e	65	145	101
Ctrl-F ACK	06	006	6	&	26	046	38	F	46	106	70	f	66	146	102
Ctrl-G BEL	07	007	7	'	27	047	39	G	47	107	71	g	67	147	103
Ctrl-H BS	08	010	8	(	28	050	40	H	48	110	72	h	68	150	104
Ctrl-I HT	09	011	9	)	29	051	41	I	49	111	73	i	69	151	105
Ctrl-J LF	0A	012	10	*	2A	052	42	J	4A	112	74	j	6A	152	106
Ctrl-K VT	0B	013	11	+	2B	053	43	K	4B	113	75	k	6B	153	107
Ctrl-L FF	0C	014	12	,	2C	054	44	L	4C	114	76	l	6C	154	108
Ctrl-M CR	0D	015	13	-	2D	055	45	M	4D	115	77	m	6D	155	109
Ctrl-N SO	0E	016	14	.	2E	056	46	N	4E	116	78	n	6E	156	110
Ctrl-O SI	0F	017	15	/	2F	057	47	O	4F	117	79	o	6F	157	111
Ctrl-P DLE	10	020	16	0	30	060	48	P	50	120	80	p	70	160	112
Ctrl-Q DCI	11	021	17	1	31	061	49	Q	51	121	81	q	71	161	113
Ctrl-R DC2	12	022	18	2	32	062	50	R	52	122	82	r	72	162	114
Ctrl-S DC3	13	023	19	3	33	063	51	S	53	123	83	s	73	163	115
Ctrl-T DC4	14	024	20	4	34	064	52	T	54	124	84	t	74	164	116
Ctrl-U NAK	15	025	21	5	35	065	53	U	55	125	85	u	75	165	117
Ctrl-V SYN	16	026	22	6	36	066	54	V	56	126	86	v	76	166	118
Ctrl-W ETB	17	027	23	7	37	067	55	W	57	127	87	w	77	167	119
Ctrl-X CAN	18	030	24	8	38	070	56	X	58	130	88	x	78	170	120
Ctrl-Y EM	19	031	25	9	39	071	57	Y	59	131	89	y	79	171	121
Ctrl-Z SUB	1A	032	26	:	3A	072	58	Z	5A	132	90	z	7A	172	122
Ctrl-[ESC	1B	033	27	;	3B	073	59	[	5B	133	91	{	7B	173	123
Ctrl-₩ FS	1C	034	28	<	3C	074	60	₩	5C	134	92	|	7C	174	124
Ctrl-] GS	1D	035	29	=	3D	075	61	]	5D	135	93	}	7D	175	125
Ctrl-^ RS	1E	036	30	>	3E	076	62	^	5E	136	94	~	7E	176	126
Ctrl_ US	1F	037	31	?	3F	077	63	_	5F	137	95	DEL	7F	177	127

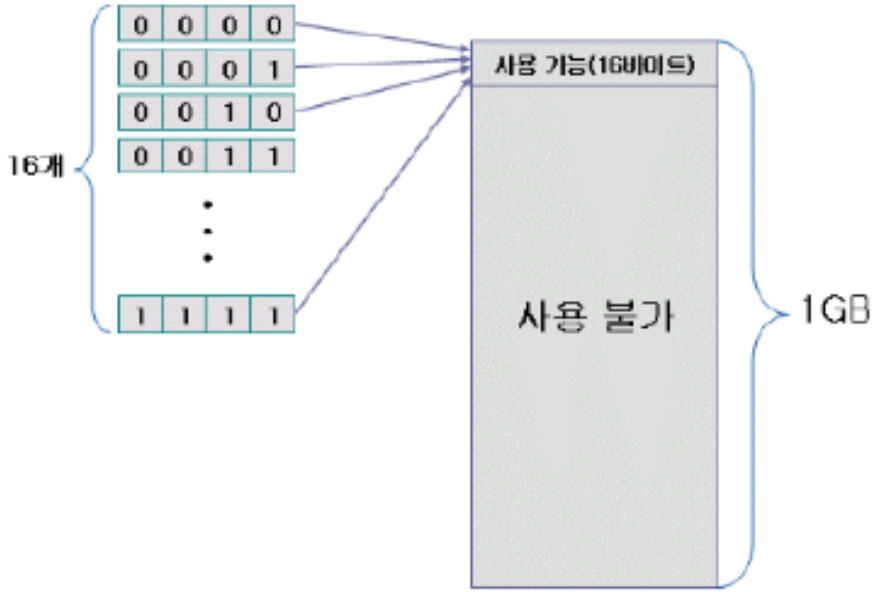
[Reference] 32-bit vs. 64-bit Computer (1/2)

- ☑ The size of the data that is transmitted and received at once
  - ■ If the size of data that is transmitted and received is 32 bits, then it is 32-bit computer, if the size of the data is 64 bits, then 64-bit computer.
- ☑ Capacity of the CPU Processing Power
  - ■ 32-bit and 64-bit refer to the way a computer's processor (also called a CPU) handles information at once.
  - ■ The 64-bit computer handles large amounts of random access memory (RAM) more effectively than a 32-bit system.



[Reference] 32-bit vs. 64-bit Computer (2/2)

☑ If there is a 4-bit computer (which uses 4-bit memory address) with 1GB memory, the size of usable memory is only 16 bytes. Thus, the other memories are all losses.



☑ 32-bit computer uses 4-byte memory address (size of the pointer), and 64-bit computer uses 8-byte memory address. Thus, there is the maximum capacity of the memory depend on the system.

- 4-bit: 2^4 = 16 bytes memory is the maximum.
- 32-bit: 2^{32} = 4GB memory is the maximum.
- 64-bit: 2^{64} = 16GB memory is the maximum.