Activity #4: C++ Types and Typecasting

Recorder's Report

Manager:

Reader:

Recorder:

Driver:

Date:

Score: Satisfactory / Not Satisfactory

Record your team's answers to the key questions (marked with 🔑) below.

(a) Model 1, Question #5

(b) Model 2, Question #8.b

(c) Model 3, Question #15

Activity #4: C++ Types and Typecasting

In this course, you will work in teams of 3–4 students to learn new concepts. This activity will introduce you to the process of analyzing an algorithm complexity.

Content Learning Objectives

After completing this activity, students should be able to:

- Explain how the size of a variable type affects the range of values it can store.
- Explain how a `char` variable encodes non-numeric values as numbers.
- Recognize an implicit type cast in a C++ program
- Predict the side effects of an explicit type cast.

Process Skill Goals

During the activity, students should make progress toward:

- Write C++ code for an explicit type cast.
- Evaluate C++ code as a computer would execute it.

Model 1 Type Sizes

Primitive data types in C++ have predefined sizes, which determines the range of values that can be stored in each data type. Several examples are given below.

type	`bool`	`char`	`int`	`double`
size	1 byte	1 byte	4 bytes	8 bytes

Refer to Model 1 above as your team develops consensus answers to the questions below.

1. A single **bit** has only two values (0 or 1), whereas two bits can store four values (00, 01, 10, and 11).

(a) How many different values can be stored in three bits, and what are they?

(b) What is the smallest binary number in your list in part (a) and what is its base 10 value?

(c) What is the largest binary number in your list in part (a) and what is its base 10 value?

2. A **byte** is equivalent to 8 bits. How many values can be stored in a single byte, and what is the largest possible value? You may wish to use Google as a calculator to solve this problem.

3. Complete the following table to determine how many values can be stored in a different number of bytes. The first one is done for you.

# bytes	# bits	number of values	largest unsigned number possible
1 byte	8	$2^8 = 256$	255
2 bytes	16	$2^{16} = 65,536$	65,535
4 bytes	32	$2^{32} = 4,294,967,296$	4,294,967,295
8 bytes	64	2^{64}	$2^{64} - 1$
x bytes	$8x$	2^{8x}	$2^{8x} - 1$

4. What is the largest value that a `unsigned short int` variable (of size 2 bytes) can hold?

5. What happens when you write a C++ assignment statement to set a `unsigned short int` variable equal to one more than this value? Try it out using the file ‘activity04a.cpp‘. Why?

Model 2   ASCII Character Codes

The American Standard Code for Information Interchange is a system that assigns a numeric value to various characters. A selection of assignments from that system is shown below.

Code	Symbol	Code	Symbol	Code	Symbol	Code	Symbol	Code	Symbol	Code	Symbol
32	(space)	48	0	64	@	80	P	96	‘	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	′	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	ˆ	110	b	126	~
47	/	63	?	79	O	95	_	110	o	127	

Refer to Model 2 above as your team develops consensus answers to the questions below.

6. Determine the numeric value associated with each character below in the ASCII system.

(a) capital A

(b) lowercase f

(c) capital Z

(d) number 0

(e) open square bracket

(f) plus sign

7. Based on your work in model 1, how many distinct characters can be stored in a C++ `char` variable?

8. In the file `activity04b.cpp` you will find two character variables and a statement that outputs the "sum" of the two variables.

(a) What is the output for each pair of variable values?

i.
```
char charOne = '(';
char charTwo = ')';
```

ii.
```
char charOne = '!';
char charTwo = ':';
```

iii.
```
char charOne = '1';
char charTwo = '4';
```

iv.
```
char charOne = '%';
char charTwo = '=';
```

(b) Explain how C++ calculated these answers. Refer to the ASCII table given in the model.

9. An *implicit type cast* happens when C++ treats a variable of one type as if it were a variable of another type. This happened in the program above.

(a) How can you tell that an implicit type cast happen in the program above?

(b) Do you think the type of the variables `charOne` and `charTwo` actually change?

Model 3 A C++ Program

[frame=lines, framesep=2mm, bgcolor=gray!15, baselinestretch=1.2, linenos]cpp include <iostream> using namespace std;
int main() // declare variables double floatOne,floatTwo; int integer = 35; // part I floatOne = integer; cout « "original number: " « integer « endl; cout « "after conversion: " « floatOne « endl « endl; // part II floatOne = 3.9; floatTwo = static$_{c}ast < int > (floatOne); cout << ''secondnumber : '' << floatOne << endl; cout << ''afterconversion : '' << floatTwo << endl;$

5pt

Refer to Model 3 above as your group develops consensus answers to the questions below.

10. This program can be found in `activity02c.cpp`. Run it and then answer the following questions.

(a) What is the printed value of the variable `integer` in part I?

(b) What is the printed value of the variable `floatOne` in part I?

(c) What is the printed value of the variable `floatOne` in part II?

(d) What is the printed value of the variable `floatTwo` in part II?

11. Lines 9 and 14 are both examples of *type casting*. What is a type cast?

12. In the previous model we saw the term *implicit type casting*. Another type of type casting is *explicit type casting*. Which statement (line 9 or 14) is an example of an explicit type cast?

13. What is the difference between an implicit type cast and an explicit type cast?

14. What is the C++ syntax for an explicit type cast? Give an example other than the one in the model.

15. Write a snippet of code that will output the string WWU without using the "W" or 🔑 "U" characters. Hint: use explicit type casting and the ASCII table seen in model 2. Feel free to test your code in the file `activity04c.cpp`.