

This worksheet is for your use during and after lecture. It will not be collected or graded, but I think you will find it a useful tool as you learn C++ and study for the exams. Explain all false answers for the “True or False” questions; in general, show enough work and provide enough explanation so that this sheet is a useful pre-exam review. I will be happy to review your answers with you during office-hours, via Email, or instant messaging.

1. State whether each variable declaration is valid C++; if it is not, explain why.

(a) `double o2pct;`

Solution: OK

(f) `double oxygen%;`

Solution: Invalid: percent signs not allowed.

(b) `integer X;`

Solution: Invalid: integer should be int.

(g) `const int ROWS(1024);`

Solution: OK

(c) `let a_crowd = 3;`

Solution: Invalid: let is not a variable type.

(h) `bool register;`

Solution: Invalid: register is a reserved word.

(d) `bool switch = false;`

Solution: Invalid: switch is a reserved word.

(i) `int 2TooMany = 3;`

Solution: Invalid: cannot begin with numeral.

(e) `const E = exp(1);`

Solution: Invalid: missing variable type.

(j) `PI const double = 3.14;`

Solution: Invalid: const double comes first.

2. Show three different ways to declare an integer variable named `theInt` and set its value to 724.

Solution:

```
int theInt( 724 );      int theInt = 724;      int theInt;
                        theInt = 724;
```

3. Consider the variable declaration:

```
int X;
```

Does `X` have a value immediately after this line of code is executed by the CPU? What term do programmers use to describe the value of `X`? Can you predict the value of `X`?

Solution: Yes, `X` has a value: `X` is simply the name for 32 bits of memory, those 32 bits can *always* be interpreted as an integer value. Since `X` wasn't initialized with a particular initial value, we say that it holds "garbage" or has a "garbage value."

I don't expect people to get the last question correct, but it is important to think about it. In some instances you cannot predict the value of `X`, but in many cases it is at least theoretically possible to do so. It takes a lot of work to do so, but programmers have done it before. In fact, being able to predict the value of `X` is the fundamental building block of stack smashing and buffer overrun exploits that allow malicious programmers to attack and gain control of other users' personal computers or Internet servers.

4. What does the term `const` mean when it precedes a variable declaration, for instance:

```
const double PI( acos(-1) );
```

Solution: The programmer cannot assign a new value to PI.

5. How many different types of integer variables are there in C++? Name them in order of increasing memory footprint.

Solution: Three: bool, short, int.

6. True or False: Integer variables represent only the counting numbers 1, 2, 3, ...

Solution: False. Negative integers, 0, and the positive integers.

7. True or False: double variables represent all possible numbers from their smallest minimum value to their largest positive value.

Solution: False, there is a small “hole” around (but not containing) zero.

8. The ASCII code for a letter is:

- A. Its number in the English alphabet (A=1, B=2, ...).
- B. A unique number representing the lower and upper case English letter.
- C. An index into the ASCII Character Code Table (or *codepage*) describing many different symbols.

Solution: C

9. True or False: The char variable type stores only an upper or lower case letter of the English alphabet.

Solution: False. It stores upper and lower case English letters, but also numerals, punctuation, tab characters, and other formatting specific character codes.

10. Explain the difference between binary operators and unary operators.

Solution: Binary operators take two arguments, usually one on either side of the operator symbol. Unary operators take one argument.

11. What is the C++ value for $3 + 12 \% 5 * 5$? Explain your reasoning.

Solution: Modulus has equal precedence as multiplication, so the expression simplifies to $3 + 2 * 5 = 3 + 10 = 13$.

12. In the context of C++ operators, what does the word *precedence* mean? Which operator has “first” precedence, which has “last” precedence?

Solution: Precedence refers to the order in which operators are evaluated. Lower precedence operators are evaluated **before** higher precedence operators. `()` has the lowest precedence (evaluated first), while assignment operators (`=`, `+=`, ...) have the highest precedence and are evaluated last.

13. Calculate the final values for the variables x, y, and z given their initial values and the C++ statements operating on them.

Pre Execution				Post		
x	y	z	C++ statement	x	y	z
1	2	3	<code>x = y++ - z;</code>	-1	3	3
1	2	3	<code>y = --z - x++;</code>	2	1	2
1	2	3	<code>y = z-- - ++x;</code>	2	1	2
1	2	3	<code>x = y = z;</code>	3	3	3
1	2	3	<code>z = y = x;</code>	1	1	1
1	2	3	<code>z *= y - ++x;</code>	2	2	0
1	2	3	<code>z += y - x++;</code>	2	2	4

14. Consider the snippet of code at the right. What are the values of x, y, and z after the variables have been initialized?

```
1 double x( 1/10 );
2 double y( 1%10 );
3 double z( 3.0/10 );
```

(a) x 0.0

(b) y 1.0

(c) z 0.3

15. Of the following mathematical functions, which are considered operators in C++?

- (a) addition
- (b) subtraction
- (c) multiplication
- (d) division
- (e) negation
- (f) modulo arithmetic
- (g) exponentiation
- (h) logarithms
- (i) square roots
- (j) parenthetical evaluation

Solution: All **except for g, h, and i**. There are *functions* in the `<cmath>` library for these, but they are not *operators* in C++.