

Chapter 1 - Test Questions

These test questions are fill in the blank, multiple choice, and true-false. The multiple choice questions may have more than one correct answer. There is one matching question. Mark *all* of the correct answers for full credit.

True False questions require an explanation in addition to the *true/false* response, and, if *false*, also require a correction.

True False:

Comment required.

1. OOP is an acronym that means Object Oriented Programming.

Answer: True.

Explanation: OOP is currently popular and is a powerful programming technique for a large class of problems.

2. C++ not only supports OOP but also supports other programming styles.

Answer: True.

Explanation: C++ supports OOP and traditional procedure oriented programming.

3. The namespace facility is a tool used that assists in the study of genealogy.

Answer: False.

Explanation: The namespace facility helps prevent the libraries from “preempting all the good names,” and allows us to use names we want whether the library has used them.

4. In C++ the variables Alpha, ALPHA and AlphA are the same identifier.

Answer: False.

Explanation: C++ is case sensitive, these are different identifiers.

5. In C++ the compiler will infer the type intended for a variable from the context in which the variable occurs.

Answer: False.

Explanation: C++ requires every identifier to be declared prior to use. The type is specified in the declaration.

6. A C++ declaration introduces only an identifier's spelling and specifies its type.

Answer: True.

Explanation: A declaration introduces the spelling and type, whereas a definition is a

declaration that also allocates memory.

7. A C++ declaration is a definition that also allocates storage for an identifier's value (or function's body etc.).

Answer: True.

Explanation: A declaration introduces the spelling and type, whereas a definition is a declaration that also allocates memory.

8. The range of values for an `int` variable is from about 0 to +2 billion.

Answer: False:

Explanation: The correct range is about -2 Billion to +2 Billion.

9. The names `x`, `y`, and `z` are satisfactory variable names for the lengths of the legs and hypotenuse of a triangle.

Answer: False.

Explanation: Names should communicate to the human reader the meaning of the value. These identifiers do not communicate the meaning of their values..

10. In C++ you can assign an expression of type `int` to a variable of type `double` with no problem.

Answer: True.

Explanation: Assignment from an integer type to a floating point type can lose information and should be avoided. Some compilers will warn, others may give an error message, but you should avoid this.

11. In C++ you can assign an expression of type `double` to a variable of type `int` with no problem.

Answer: False.

Explanation: In general assigning a floating point value to an integer variable mostly loses information. A warning message (or an error message) is issued by C++ compilers.

12. To put a character into a `cstring` constant that causes the output to continue on the next line, insert the escape sequence `\t` into the string constant.

Answer: False.

Explanation: `\t` is the tab escape sequence. Use `\n` instead.

13. If we execute this code in an otherwise correct and complete program:

```
n = 1;
```

```
n = (n++) + (n++);
```

the value of `n` is guaranteed to be 3 after the second line executes.

Answer: False.

Explanation: Some compilers may give the value 3. The result of such code is dependent on the compiler implementation. The C++ Standard says such code is illegal, and the Standard says the results are undefined, meaning the compiler can do anything with it that suits the compiler writer. One of my compilers gives the value 4 to `n`.

14. If we execute the code fragment in an otherwise complete, correct program:

```
n = 1;
cout << n++ << " " << n++ << " " << n++ << endl;
```

the output is guaranteed to be 1 2 3.

Answer: False.

Explanation: The code is illegal because its execution depends on the order of evaluation of arguments. Various compilers give different answers. One compiler this writer uses gives 3 2 1.

15. C++ uses only `/* */` for comments.

Answer: False.

Explanation: C++ uses `/* */` comments *and* `//` “here to the end of the line” comments.

16. A program’s comments should connect the program code to the problem being solved.

Answer: True.

Explanation: The purpose of comments in a program is to help the human reader of the code to connect the program to the problem being solved. Comments are so important there is an aphorism, often quoted by expert programmers: “If the comments and the code disagree, then *both* are probably wrong.

17. A program should have a comment on every line.

Answer: False.

Explanation: This would be satisfactory only on a very complicated assembly language program for an inexperienced programmer.

18. Comments have no value whatsoever and do not belong in a program.

Answer: False.

Explanation: The purpose of comments in a program is to help the human reader of the code to connect the program to the problem being solved. Comments are valuable.

19. The most difficult programming language construct to learn to use *properly* is the comment.

Answer: True.

Explanation: How many comments and what should be in the comment is very hard to decide because these depend on who is the intended reader of the code (and comments).

20. A computer program is a set of instructions intended for only the computer to follow.

Answer: False.

Explanation: A computer program is also intended for human beings to read, and the comments are intended to make this job easier.

Fill in the blank

1. In C++, a legal identifiers may contain these kinds of characters _____, _____, _____.

Answer: Letters, digits, underscore

Explanation: Some implementations allow a very few other characters such as \$.

2. The C++ very nearly contains the _____ programming language as proper subset.

Answer: (the) C (programming language)

3. In C++, a variable that has been defined but not initialized may not be use as an _____. (*l-value* or *r-value*).

Answer: r-value

Explanation: The value of an uninitialized variable may not be fetched. C++ does not enforce this. If an uninitialized variable's value is fetched you get whatever value was left in the memory location by a previous user. Note that the value is garbage in the dictionary sense, It is *not a random* value.

4. Identifiers should at least give a hint to the human reader of the _____ of the identifier in the context of the problem being solved.

Answer: Meaning

Explanation: Any identifier in a program that represents a quantity in a problem should

be *named* in the program with the real life name for that quantity, or at least the should hint at the real life thing it represents.

Multiple Choice

1. The person responsible for the initial development of C++ was

- a) Dennis Ritchie
- b) James Gosling
- c) Brian Kernighan
- d) Bjarne Stroustrup
- e) Bill Joy

Answer: d) Bjarne Stroustrup, at AT&T Bell Labs in the 1980s

2. In C++, some of the following are legal identifiers. Which? Why?

- a) 9xyz
- b) Xyz
- c) X+yz
- d) xy_z
- e) xyz!

Answer:

- a) 9xyz no – An identifier cannot start with a digit
- b) Xyz yes. An identifier may have upper and lower case letters
- c) X+yz no – This has an illegal ‘+’.
- d) xy_z yes.
- e) xyz! no – This has an illegal ‘!’.

3. Which of the following are likely to be poor choices for an identifier in a program?

Why?

- a) x
- b) RATE
- c) data
- d) _abc
- e) A

Answer:

- a) `x` is too short to adequately describe a real world value.
- b) `RATE` suggests a rate of speed, or interest perhaps.
- c) `data` is perhaps a bit generic, nevertheless this writer would allow it.
- d) `_abc` is not a word. It does not bring anything to this writer's mind.
- e) `A` is too short to adequately describe anything in the real world.

Explanation: The answers to this question are subjective.

4. Pick the C++ keywords out of the following list.

- a) `while`
- b) `total_weight`
- c) `double`
- d) `if`
- e) `number_ofBars`

Answer: Keywords are a) `while`, c) `double` and d) `if`.

5. Which of the following does the C++ language *not* support?

- a) classes
- b) global functions
- c) automatic garbage collection
- d) support for object oriented programming
- e) traditional programming techniques

Answers: C++ does not provide automatic garbage collection.

Explanation: There are free and commercial garbage collection libraries for C++. "In C++, you do not pay for what you do not use."

6. Before a variable in C++ is used, it *must* be

- a) defined
- b) initialized
- c) used in some expression
- d) begin with a capital letter
- e) contain only letters, digits and underscores.

Answer: a) defined.

7. Which of the following types are *not* built into the C++ language:

- a) `bool`

- b) `real`
- c) `short`
- d) `int`
- e) `long`
- f) `double`

Answer: Item b) `real` is incorrect. There is no “real” type in C++. C++ does provide items a)-and c) through f) and more.

8. A l-value is

- a) an expression that can be only be placed on the left of any operator such as `+`, `*`, `/`, etc.
- b) assigned a value
- c) can never have a value fetched from it
- d) is designed for use by a left-handed person

Answer: b) an l-value.

Typo should be

Answer b) assigned a value

Explanation: An l-value may be assigned a value.

9. An r-value is

- a) an expression that can be only placed on the right of any operator such as `+`, `*`, `/` etc.
- b) can never be assigned a value
- c) can have a value fetched from it
- d) is designed for use by a right-handed person.

Answer: c) an r-value

Explanation: An r-value can have a value fetched from it.

10. In C++, a variable that has been defined but not initialized may

- a) be used in any way any identifier can be used.
- b) not be used at all
- c) not be used as an l-value
- d) not be used as an r-value
- e) have its value fetched prior to assignment.

Answer: c) is correct.

Explanation: The value of an uninitialized variable may not be fetched, but a value may be assigned to an uninitialized variable.

11. Which of the following are legal definitions with initializations? (Consider each line to be in a different scope so there is no multiple definition of identifiers.)

- a) `int count = 0, limit = 19;`
- b) `int count(0), limit(19);`
- c) `int count = 0, limit(19);`
- d) `int limit = 19;`
- e) `int namespace(0);`

Answer: All are legal except part e) `int namespace(0);` because `namespace` is a keyword. Keywords are reserved so may not be used as an identifier.

Explanation: Note parts a), b), and c). Some authors advocate declaring each identifier in separate definitions on separate lines, though this takes up more lines. Advocates justify this by asserting increased readability. The instructor must decide this issue.

12. Which of the following names might give the human reader some hint about the data that is stored in them?

- a) `aa, bb, cc`
- b) `speed, distance, time`
- c) `hypotenuse, leg1, leg2`
- d) `v1, v2, v3`
- e) `principal, interest, payment`

Answer: Parts b), c), and e) are OK. Parts a) and d) are poor names.

13. The value of the expression `20.0 * (9/5) + 32.0` is

- a) 68.0
- b) 52.0
- c) incorrect expression so there is no value
- d) 32.0
- e) incorrect expression, the `/` should be `%`

Answer: Part b) is correct

Explanation: $9/5$ evaluates to 1 using truncating `int` division.

14. The value of the expression `20.0 * (9.0/5) + 32.0` is

- a) 68.0
- b) 52.0
- c) expression has a syntax error so there is no value
- d) 32.0
- e) an incorrect expression, the `/` should be `%`

Answer: Part a).

Explanation: The expression $9.0 / 5$ evaluates to 1.8 and the computation proceeds correctly.

15. The following contain several `#include` directives that may have problems. Which have one or more problems, and what problems?

- a) `#include <iostream>`
- b) `#include < iostream>`
- c) `#include "MyStream.h"`
- d) `#include "cctype "`
- e) `#include "This file does not exist on the system"`

Answer: All Except a) and c)

Explanation: b) The space between `<` and `iostream` makes preprocessor think the file name has a space at the front of the name. In part c), the system looks in the default directory then in the system directories as it does for `<files>`. d) has space at the end of the file name. e) The file name is unlikely. At the least, the situation should be examined.

16. Which of the following will properly encode the string `"\w\\\t"` into variable `s`?

- a) `s = "\w\\\t";`
- b) `s = R"(\w\\\t)";`
- c) `s = "\\w\\\\\\t";`
- d) `s = "/w//t";`

Answer: b and c (b requires C++11)

17. What is the data type of variable `x` in the following C++11 code:

```
int y= 2;  
double z = 3.5;  
auto x = z * y;
```

- a) int
- b) double
- c) auto
- d) syntax error

Answer: b

Free Form Questions:

18. In the history of the C++ language, what was the immediate predecessor of the C++ language? How is this older language related to C++?

Answer: The immediate predecessor of C++ is the C programming language. C is very nearly a proper subset of C++.

19. Write a C++ program that outputs "My first C++ program" and then outputs a carriage return.

Answer:

```
#include <iostream>  
using namespace std;  
int main()  
{  
    cout << "My first C++ program" << endl;  
}
```

Explanation: An alternative is to place `using namespace std;` after the `#include` directive, or to place this inside the block of the function `main`. This depends on the local rules for placing `namespace` directives and definitions.

20. Given the C++ output statements. What is the output of these lines of code? Explain.

```
cout << "If you have ";  
cout << "a number of pods ";
```

```
cout << "you can quit.";  
cout << "\n";
```

Answer: If you have a number of pods you can quit.

Explanation: Note that there is a `<cr>` emitted by the code, which puts the next output position on the next line at the left end.

21. What does the line

```
#include <iostream>
```

do for your program?

Answer: The `#include` directive provides declarations and definitions of the `iostream` components so that your program can use them in a way that allows the system linker to connect your program to the corresponding library.

22. What is the difference between a warning from the compiler and an error message from the compiler?

Answer: An error message usually causes code generation to stop. A warning does not. Unless you understand what the warning is all about you should not ignore warnings.

23. Give the declaration for two variables, `feet` and `inches`. Declare each to be of type `int`, and both initialized to zero in the declaration. Give both initialization alternatives.

Answer:

```
int feet = 0;  
int inches = 0;  
//Alternate initialization.  
int feet (0);  
int inches(0);
```

24. Write a short program that contains statements to output the values of a variable that you define but neither initialize nor assign. Discuss the output you get.

Answer:

```
#include <iostream>  
using namespace std;  
int main()  
{
```

```

    int uninitialized;
    cout << uninitialized << endl;
    return 0;
}

```

25. When you use a `double`, what is doubled? When you use a `long int`, how much longer is the `long int` than an `int`? Comment.

Answer: In a `double`, the precision part of the value is about twice the precision of a `float`. The size of the exponent is also increased. In most implementations, `long` is the same size as `int`.

26. What is the value assigned to the variable in each of these cases? Explain curious results. Be careful!

```

    int x, y;
    a) x = 1/2;
    b) y = 3.0/2.0;
    double z, w, t;
    c) z = 1/2;
    d) w = 3/2;
    e) t = 3.0/2.0;

```

Answer:

- a) 0 is assigned. $1/2$ is computed using truncating integer divide which gives 0.
- b) 1.5 is calculated and assigned. The fractional part is discarded; 1 is stored.
- c) 0 is calculated because $1/2$ is done with truncating integer divide, gives 0. The floating point value 0.0 is stored.
- d) 1 is calculated, because truncating `int` divide, 1.0 is stored
- e) 1.5 is calculated and stored.

27. With C++11 and higher what data type can you use to guarantee that 32 bits will be used to store a signed integer?

Answer: The type `int32_t` will allocate 32 bits for an integer. The size of a regular `int` may vary (e.g. 16, 32, or 64 bits) depending upon the underlying machine architecture.