Capitulo 15

This activity contains 33 questions.

1. *Section 15.1 Introduction*

15.1 Q1: Which of the following statements is not true about C++ input/output (I/O) features?

- *C++ has some object-oriented I/O features.*
- *C++ borrowed its type safe I/O capabilities from C.*
- *C++ automatically calls the correct I/O operations for standard types.*
- *C++ allows users to specify I/O for their own data types.*

2. *Section 15.2 Streams*

15.2 Q1: Select the correct statement regarding C++ I/O streams:

- *C++ provides only high-level" I/O capabilities because it is a high-level programming language.*
- *Low-level I/O breaks information down into small, meaningful groups of related bytes.*
- *Programmers generally prefer high-level I/O to low-level I/O.*
- *High-level (formatted) I/O is best for large-volume transfers.*

3. *15.2 Q2: _____ is usually faster than _____.*

- *Low-level I/O, high-level I/O.*
- *High-level I/O, internal data processing.*
- *Low-level I/O, internal data processing.*
- *High-level I/O, low-level I/O.*

4. *Section 15.2.1 Classic Streams vs. Standard Streams*

15.2.1 Q1: Which C++ data type was designed to store Unicode characters?

- *size_t.*
- *char.*

- *wchar_t.*
- *long.*

5. *15.2.2 Q1: Which of the following libraries does not deal with input/output?*

- *fstream.*
- *iomanip.*
- *iostream.*
- *fiostream.*

6. *Section 15.2.3 Stream Input/Output Classes and Objects*

15.2.3 Q1: Which of the following classes is a base class of the other three?

- *basic_iostream.*
- *basic_ostream.*
- *basic_ios.*
- *basic_istream.*

7. *15.2.3 Q2: Which of the following is not a member of the ostream class?*

- *cin.*
- *cerr.*
- *cout.*
- *clog.*

8. *15.2.3 Q3: Which of the following classes is deepest in the inheritance hierarchy?*

- *basic_ofstream.*
- *basic_ifstream.*
- *basic_iostream.*
- *basic_fstream.*

9. *Section 15.3 Stream Output*

15.3 Q1: Which of the following is not a member function of the C++ ostream class?

- *d. write.*
- *Stream-insertion operator (<<).*
- *put.*
- *Stream-extraction operator (>>).*

10. *Section 15.3.1 Output of char * Variables*

15.3.1 Q1: Which of the following prints the address of character string string given the following declaration?

```
char * string = "test";
```

- *cout << static_cast< void * >(string);.*
- *cout << string;.*
- *cout << * string;.*
- *cout << *&string;.*

11. *Section 15.3.2 Character Output using Member Function put*

15.3.2 Q1: Which of the following is an illegal use of function put?

- *cout.put('A').put('\n');.*
- *cout.put("A");.*
- *cout.put(65);.*
- *cout.put('A');.*

12. *Section 15.4 Stream Input*
15.4 Q1: The stream-extraction operator:

- *Sets the stream's failbit if the operation fails.*
- *Sets the stream's badbit if the data is of the wrong type.*
- *Does not normally accept white-space characters.*
- *Returns true when the end-of-file is encountered.*

13. *Section 15.4.1 get and getline Member Functions*

15.4.1 Q1: One difference between the three-argument version of the get function and the getline function is that:

getline stores the characters it reads into its character array argument.

- *Only get adds the delimiter to the array.*
- *The getline function removes the delimiter from the stream.*
- *Only get has a delimiter.*

14. *Section 15.4.2 istream Member Functions peek, putback and ignore*

15.4.2 Q1: The putback member function returns to the input stream the previous character obtained by:

- *A get from the input stream.*
- *Reading input from the keyboard.*
- *Using the stream extraction operator on the input stream.*
- *Reading a file from disk.*

15. *15.4.2 Q2: Upon encountering the designated delimiter character, the ignore member function will:*

- *Replace it with an EOF character.*
- *Read it in and return its value.*
- *Ignore it and continue reading and discarding characters.*
- *Terminate.*

16. *Section 15.4.3 Type-Safe I/O*

15.4.3 Q1: If unexpected data is processed in an I/O operation:

- *The program will terminate execution.*
- *Various error bits will be set.*
- *An error message will automatically be displayed.*
- *An exception will be thrown.*

17. *Section 15.5 Unformatted I/O using read, write and gcount*

15.5 Q1: Which of the following is a difference between the read and write functions?

- *The failbit is set only with read.*
- *One performs formatted I/O and the other does not.*
- *They take different types of parameters.*

- *write and gcount are member functions of the same class, whereas read is not.*

18. *Section 15.6.1 Integral Stream Base: dec, oct, hex and setbase*

15.6.1 Q1: Which of the following is not a difference between hex and setbase?

- *setbase is a parameterized stream manipulator and hex is not.*
- *setbase is provided by a different header file than hex.*
- *setbase takes an argument but hex does not.*
- *setbase(16) and hex have different effects on stream output.*

19. *Section 15.6.2 Floating-Point Precision (precision, setprecision)*

15.6.2 Q1: What will be output by the following statements?

```
double x = 1.23456789;
cout << fixed;
cout << setprecision(5) << x << endl;
cout.precision(3);
cout << x << endl;
cout << x << endl;
```

- *1.2346*
  1.23
  1.23456789
- *1.2346*
  1.23
  1.23
- *1.23457*
  1.235
  1.23456789
- *1.23457*
  1.235
  1.235

20. *Section 15.6.3 Field Width (width, setw)*

15.6.3 Q1: Which of the following is not true about setw and width?

- *They only apply for the next insertion/extraction.*
- *Both of them can perform two tasks, setting the field width and returning the current field width.*
- *They are used to set the field width of output.*
- *If the width set is not sufficient the output prints as wide as it needs.*

21. *Section 15.6.4 User-Defined Output Stream Manipulators*

15.6.4 Q1: Which of the following is a valid user-defined output stream manipulator header?

- *ostream tab(ostream output).*
- *istream& tab(istream output).*
- *ostream& tab(ostream& output).*
- *void tab(ostream& output).*

22. *Section 15.7.1 Trailing Zeros and Decimal Points (showpoint)*

15.7.1 Q1: What will be output by the following statement?

```
cout << showpoint << setprecision(4) << 11.0 << endl;
```

- *11.00.*
- *11.000.*
- *11.*
- *11.0.*

23. *Section 15.7.2 Justification (left, right and internal)*

15.7.2 Q1: Which of the following stream manipulators causes an outputted number's sign to be left justified, its magnitude to be right justified and the center space to be filled with fill characters?

- *left.*
- *showpos.*
- *right.*
- *internal.*

24. *15.7.3 Q1: Which of the following statements restores the default fill character?*

- *cout.fill(0);.*
- *cout.defaultFill();.*
- *cout.fill(' ');.*
- *cout.fill();.*

25. *Section 15.7.4 Integral Stream Base (dec, oct, hex, showbase)*

15.7.4 Q1: When the showbase flag is set:

- *Decimal numbers are not outputted any differently.*
- *Octal numbers can appear in one of two ways.*
- *The base of a number precedes it in brackets.*
- *c. "oct" or "hex" will be displayed in the output stream.*

26. *Section 15.7.5 Floating-Point Numbers; Scientific and Fixed Notation (scientific, fixed)*

15.7.5 Q1: What will be output by the following statements?

```
double x = .0012345;
      cout << fixed << x << endl;
      cout << scientific << x << endl;
```

- *.001235*
  1.234500e-003
- *.00123450*
  1.23450e-003
- *1.234500e-003*
  .001235
- *1.23450e-003*
  .00123450

27. *Section 15.7.6 Uppercase/Lowercase Control (uppercase)*

15.7.6 Q1: Which of the following outputs does not guarantee that the uppercase flag has been set?

- *All hexadecimal numbers appear in the form 0X87.*
- *All text outputs appear in the form SAMPLE OUTPUT.*
- *All hexadecimal numbers appear in the form AF6.*
- *All numbers written in scientific notation appear the form 6.45E+010.*

28. *Section 15.7.7 Specifying Boolean Format (boolalpha)*

15.7.7 Q1: Which of the following is not true about bool values and how they are outputted with the output stream?

- *Both boolalpha and noboolalpha are "sticky" settings.*
- *Stream manipulator boolalpha sets the output stream to display bool values as the strings "true" and "false".*

- *A bool value outputs as 0 or 1 by default.*
- *The old style of representing true/false values used -1 to indicate false and 1 to indicate true.*

29. *Section 15.7.8 Setting and Resetting the Format State via Member Function flags*

15.7.8 Q1: To reset the format state of the output stream:

- *You must manually apply each individual format change member function or stream manipulator to restore the default format state.*
- *Call the flags member function with the ios_base::fmtflags constant as the argument.*
- *Call the reset member function.*
- *Save a copy of the fmtflags value returned by calling member function flags before making any format changes, and then call flags again with that fmtflags value as the argument.*

30. *Section 15.8 Stream Error States*

15.8 Q1: The good member function will return false if:

- *The eof member function would return true.*
- *Any of the above.*
- *b. The bad member function would return true.*
- *The failbit member function would return true.*

31. *15.8 Q2: The difference between the operator! member function and the operator void* member function is that:*

- *d. Of the two member functions, only operator void* checks if eof has been set.*
- *Of the two member functions, only operator! checks if eof has been set.*
- *They occasionally return opposite boolean values.*
- *They always return opposite boolean values.*

32. *Section 15.9 Tying an Output Stream to an Input Stream*

15.9 Q1: Select the false statement. Outputs are:

- *Flushed automatically at the end of a program.*
- *Able to be synchronized with inputs.*
- *Never automatically tied to inputs.*

- ◯ *Flushed when the buffer fills.*

33. *15.9 Q2: Untying an input stream, inputStream, from an output stream, outputStream, is done with the function call:*

- ◯ *inputStream.tie(0).*
- ◯ *inputStream.tie().*
- ◯ *inputStream.untie().*
- ◯ *inputStream.untie(&outputStream).*

Clear Answers / Start Over Submit Answers for Grading

Answer choices in this exercise appear in a different order each time the page is loaded.