Quizlet

Chapter 5 C++

Study online at quizlet.com/_ll81h

1. **Divide a program into manageable tasks.**	What do functions enable you to do?
2. **Need the empty parantheses in both the function heading and the function call.**	What do you need if the function has no parameters?
3. **Any order in the program.**	Where do user-defined functions appear in function prototypes?
4. **Void**	What does the heading of a void function start with?
5. **Variables declared within a function or block.**	What are local variables?
6. **Functions**	What are like miniature programs?
7. **Modules**	What are mini programs called?
8. **Standard predefined functions**	What does the C++ system provide?
9. **1) Know the name of the header file that contains the function's specification. 2) Include that header file in the program. 3) Know the name and type of the function and know the number and types of parameters.**	What are the 3 steps to using a standard function?
10. **1) Value-Returning functions 2) Void Functions**	What are the two types of user-defined functions?
11. **Varaiables defined in a function heading.**	What are formal parameters?
12. **Actual Paramaters**	What are expressions, variables, or constant values used in a function called?
13. **The formal parameters in the order given.**	What musr the number of actual parameters match with in a function call?
14. **Use its name together with the actual parameter list.**	How do you call a function?
15. **Returns a value**	What does a value-returning function do?
16. **As an expression or an out put statement or as a parameter in a function call.**	How is a value-returning function used?
17. **functionType functionName (formal parameter list) { statements }**	What is the general syntax of a user-defined function?
18. **function heading or function header**	What is the line functionType functionName called?
19. **Body of a function**	What are statements enclosed between braces { and } called?
20. **1) Function Heading 2) Body of Function**	What are called the definition of the function?
21. **Via the return statement**	How does a value-returning function return its value?
22. **More than one return statement.**	What can a function have?
23. **Remaining statements are skipped and the function exits.**	What happens when a return statement executes in a function?
24. **Returns only one value.**	What does a return statement do?
25. **Prototype: Function heading without the body of a function Ending: Ends with a semicolon**	What is a function prototype and when does it end?
26. **Function type, type, and number of parameters.**	What does a function prototype announce?
27. **Names of the variables in the formal parameter list are optional.**	What happens in a function prototype?
28. **Correctly translate each function call**	What do the function prototypes help the compiler do?
29. **Before every function definition, including the definition pf the function main.**	Where are function prototypes placed in a program?
30. **1st statement in the main function**	When the program executes where does the execution begin?
31. **When they are called**	When do user-defined functions execute?
32. **Controls from the caller to the function**	What does a call to a function transfer?
33. **Specify only the actual parameters not their data type or the function type.**	What do you do in a function call statement?
34. **Control goes back to the caller**	What happens when a function exists?
35. **Void Function**	What is a function that does not have a data type called?
36. **1) Void Function 2) Exit the function early**	Where can a return statement without any value be used and what is it typically used for?
37. **Reserved Word**	What is void in C++?

38. **Void Function**	What may or may not have parameters?
39. **Stand-alone statement**	What is a call to a void function?
40. **Use the function name together with the actual parameters in a stand-alone statement?**	How do you call a void function?
41. **1) Value Parameters 2) Reference Parameters**	What are the two types of formal parameters?
42. **Corresponding actual parameter**	What does a value parameter receive a copy of?
43. **Address of its corresponding actual parameter**	What does a reference parameter receive?
44. **Expression, variable, or constant value**	What is the corresponding actual parameter of a value parameter?
45. **Constant Value**	What cannot be passed to a reference parameter?
46. **Variable**	What must be the corresponding actual parameter of a reference parameter?
47. **Formal parameter becomes a reference parameter**	What happens when you include the & after the data type of a formal parameter?
48. **Reference**	What should the stream variable be passed by to get a function?
49. **Must declare this formal parameter as a reference parameter in the function header**	What must you do if a formal parameter needs to change the value of an actual parameter?
50. **Those parts of the program where the identifier is accessible**	What does the scope of an identifier refer to?
51. **Variables declared outside of every function definition and block**	What are global variables?
52. **Scope of an identifier name declared outside of any block**	What is the scope of a function the same as?
53. **The nesting of a function definition**	C++ does not allow what?
54. **False; Know the appropriate header file**	T or F: To use a predefined function in a program, you need to know only the name of the function and how to use it.
55. **True**	T or F: A value-returning functon returns only one value.
56. **True**	T or F: Parameters allow you to use different values each tme the function is called.
57. **True**	T or F: When a return statement executes in a user-defined function, the function immediately exits.
58. **False; Returns any values it is programmed to return**	T or F: A value-returning function returns only integer values.
59. **True**	T or F: A function that changes the value of a reference parameter also changes the value of the actual parameter.
60. **False; it can**	T or F: A variable name cannot be passed to a value parameter.
61. **True**	T or F: If a C++ function does not use paramters, parantheses around the empty parameter list are stll required.
62. **False; Don't have to match**	T or F: In C++, the names of the corresponding formal and actual parameters must be the same.
63. **True**	T or F: Whenever the value of a reference parameter changes, the value of the actual parameter changes.
64. **False**	T or F: In C++, function definitions can be nested; that is, the definition of one function can be enclosed in the body of another function.
65. **False; Variables that you put outside of int main**	T or F: Using global variables in a program is a better programming style than using local variables, because extra variables can be avoided.
66. **False**	T or F: In a program, global constants are as dangerous as global variables.
67. **True**	T or F: The memory for a static variable remains allocated between function calls.
68. **Invalid; need a data type**	Is this function heading valid? one (int a, int b)
69. **Valid**	Is this function heading valid? int thisone (char x)
70. **Invalid; list the data type**	Is this function valid? char another (int a, b)
71. **Invalid; double is not a function, it is a parameter type. Has no parameters!**	Is this function valid? double yetanother
72. **A variable declared in the heading of a function definition.**	What is a formal parameter?
73. **A variable or expression used in a function call.**	What is an actual parameter?

74. **Receives address of actual parameter.**	What is a reference parameter?
75. **Receives a copy of the actual parameter's data.**	What is a value parameter?
76. **Calculates the largest whole number that is less than or equal to x**	What does the floor function do?
77. **functions that have a return data type**	What is a value-returning function?
78. **Functions that do not have a return type**	What is a void function?
79. **Causes body of the called function to execute**	What is a function call?
80. **Function heading without body of function.**	What is a function prototype?
81. **Provide a common link between calling function and called function**	What is a parameter?
82. **A variable for which memory is allocated at block entry and deallocated at block exit.**	What is an automatic variable?
83. **A variable for which memory remains allocated as long as the program executes is a static variable.**	What is a static variable?