1. The **function header** is the part of a function definition that shows the function name, return type, and parameter list.
2. If a function doesn’t return a value, the keyword **void** will appear as its return value.
3. If function showValue has the following header: void showValue(int quantity) you would use the statement **showValue(5)** to call it with the argument 5.
4. Either a function’s **definition** or its **prototype** must precede all calls to the function.
5. Values that are sent into a function are called **arguments**.
6. Special variables that hold copies of function arguments are called **parameters**/
7. When only a copy of an argument is passed to a function, it is said to be passed by **value**.
8. A(n) **function prototype** eliminates the need to place a function definition before all calls to the function.
9. A(n) **local** variable is defined inside a function and is not accessible outside the function.
10. **Global** variables are defined outside all functions and are accessible to any function within their scope.
11. **Global** variables provide an easy way to share large amounts of data among all the functions in a program.
12. Unless you explicitly initialize numeric global variables, they are automatically initialized to **zero**.
13. If a function has a local variable with the same names as a global variable, only the **local** variable can be seen by the function.
14. **Static** local variables retain their value between function calls.
15. The **return** statement causes a function to end immediately.
16. **Default** arguments are passed to parameters automatically if no argument is provided in the function call.
17. When a function uses a mixture of parameters with and without default arguments, the parameters with default arguments must be defined **last**.
18. The value of a default argument must be a(n) **literal or constant**.
19. When used as parameters, **reference** variables allow a function to access the parameter’s original argument.
20. Reference variables are defined like regular variables, except there is a(n) **ampersand(&)** in front of the name.
21. Reference variables allow arguments to be passed by **reference**.
22. The **exit()** function causes a program to terminate.
23. Two or more functions may have the same name, as long as their **parameter lists** are different.

|  |  |
| --- | --- |
| // function example | #include <iostream> | using namespace std; |  int addition (int a, int b)  {  int r;  r=a+b;  return r;  }  int main ()  {  int z;  z = addition (5,3);  cout << "The result is " << z;  } | The result is 8 |
| // function example | #include <iostream> | using namespace std; |  int subtraction (int a, int b)  {  int r;  r=a-b;  return r;  }  int main ()  {  int x=5, y=3, z;  z = subtraction (7,2);  cout << "The first result is " << z << '\n';  cout << "The second result is " << subtraction (7,2) << '\n';  cout << "The third result is " << subtraction (x,y) << '\n';  z= 4 + subtraction (x,y);  cout << "The fourth result is " << z << '\n';  } | The first result is 5  The second result is 5  The third result is 2  The fourth result is 6 |
| // void function example | #include <iostream> |using namespace std; |  void printmessage ()  {  cout << "I'm a function!";  }  int main ()  {  printmessage ();  } | I'm a function! |
| // default values in functions | #include <iostream> | using namespace std;  int divide (int a, int b=2)  {  int r;  r=a/b;  return (r);  }  int main ()  {  cout << divide (12) << '\n';  cout << divide (20,4) << '\n';  return 0;  } | 6  5 |
| // passing parameters by reference | #include <iostream> | using namespace std;  void duplicate (int& a, int& b, int& c)  {  a\*=2;  b\*=2;  c\*=2;  }  int main ()  {  int x=1, y=3, z=7;  duplicate (x, y, z);  cout << "x=" << x << ", y=" << y << ", z=" << z;  return 0;  } | x=2, y=6, z=14 |
| // declaring functions prototypes | #include <iostream> | using namespace std;  void odd (int x);  void even (int x);  int main()  {  int i;  do {  cout << "Please, enter number (0 to exit): ";  cin >> i;  odd (i);  } while (i!=0);  return 0;  }  void odd (int x)  {  if ((x%2)!=0) cout << "It is odd.\n";  else even (x);  }  void even (int x)  {  if ((x%2)==0) cout << "It is even.\n";  else odd (x);  } | Please, enter number (0 to exit): 9  It is odd.  Please, enter number (0 to exit): 6  It is even.  Please, enter number (0 to exit): 1030  It is even.  Please, enter number (0 to exit): 0  It is even. |