**Declaring functions.**

In C++, identifiers can only be used in expressions once they have been declared. For example, some variable *x* cannot be used before being declared with a statement, such as:

|  |  |  |
| --- | --- | --- |
|  | int x; |  |

The same applies to functions. Functions cannot be called before they are declared. That is why, in all the previous examples of *functions*, the functions were always defined before the main function, which is the function from where the other functions were called. If main were defined before the other functions, this would break the rule that functions shall be declared before being used, and thus would not compile.

The prototype of a function can be declared without actually defining the function completely, giving just enough details to allow the types involved in a function call to be known. Naturally, the function shall be defined somewhere else, like later in the code. But at least, once declared like this, it can already be called.

The declaration shall include all types involved (the return type and the type of its arguments), using the same syntax as used in the definition of the function, but replacing the body of the function (the block of statements) with an ending semicolon.

The parameter list does not need to include the parameter names, but only their types. Parameter names can nevertheless be specified, but they are optional, and do not need to necessarily match those in the function definition. For example, a function called *protofunction* with two int parameters can be declared with either of these statements:

|  |  |  |
| --- | --- | --- |
| 1 2 | int protofunction (int first, int second);  int protofunction (int, int); |  |

Anyway, including a name for each parameter always improves legibility of the declaration.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 | // 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. | [Edit & Run](https://cplusplus.com/doc/tutorial/functions/) |

This example is indeed not an example of efficiency. You can probably write yourself a version of this program with half the lines of code. Anyway, this example illustrates how functions can be declared before its definition:  
The following lines:

|  |  |  |
| --- | --- | --- |
| 1 2 | void odd (int a);  void even (int a); |  |

Declare the prototype of the functions. They already contain all what is necessary to call them, their name, the types of their argument, and their return type (*void* in this case). With these prototype declarations in place, they can be called before they are entirely defined, allowing for example, to place the function from where they are called (*main*) before the actual definition of these functions.

But declaring functions before being defined is not only useful to reorganize the order of functions within the code. In some cases, such as in this particular case, at least one of the declarations is required, because *odd* and *even* are mutually called; there is a call to *even* in *odd* and a call to *odd* in *even*. And, therefore, there is no way to structure the code so that *odd* is defined before *even*, and *even* before *odd*.