Here's what the code does step by step:

```
#include <iostream>
using namespace std;
class Germany {
protected:
   int germanyScore;
public:
   Germany() {
      germanyScore = 0;
   }
   void goal() {
      germanyScore++;
   }
   int getScore() {
      return germanyScore;
   }
};
```

1. The code defines a class called Germany, which has a protected member variable called "germanyScore" and three member functions:

   a. A default constructor that initializes "germanyScore" to 0.

   b. A function called "goal" that increments "germanyScore" by 1.

   c. A function called "getScore" that returns the current value of "germanyScore".

```
class Argentina : public Germany {
private:
    int argentinaScore;
public:
    Argentina() {
        argentinaScore = 0;
    }
    void goal() {
        argentinaScore++;
        Germany::goal();
    }
    int getScore() {
        return argentinaScore;
    }
};
```

2. The code defines a class called Argentina that inherits from Germany (using the "public" keyword), and adds a private member variable called "argentinaScore" and two member functions:

a. A default constructor that initializes "argentinaScore" to 0.

b. A function called "goal" that increments "argentinaScore" by 1 and also calls the "goal" function of the parent class (Germany) using the scope resolution operator (::).

```
class Brasil : public Germany {
private:
    int brasilScore;
public:
    Brasil() {
        brasilScore = 0;
    }
    void goal() {
        brasilScore++;
        Germany::goal();
    }
    int getScore() {
        return brasilScore;
    }
};
```

3. The code defines a class called Brasil that also inherits from Germany (using the "public" keyword), and adds a private member variable called "brasilScore" and two member functions:

a. A default constructor that initializes "brasilScore" to 0.

b. A function called "goal" that increments "brasilScore" by 1 and also calls the "goal" function of the parent class (Germany) using the scope resolution operator (::).

```
int main() {
    Germany* teams[3];
    teams[0] = new Germany();
    teams[1] = new Argentina();
    teams[2] = new Brasil();
    for (int i = 0; i < 3; i++) {
        teams[i]->goal();
        cout << "Team " << i << " score: " << teams[i]->getScore() << endl;
    }
    return 0;
}
```

4. In the main function, an array of pointers to the base class Germany is declared and initialized with instances of the three classes (Germany, Argentina, and Brasil). Since Argentina and Brasil inherit from Germany, they can be stored in the array of pointers to the base class. A for loop is used to simulate the match. In each iteration of the loop, the "goal" function of the current team is called using the arrow operator (->), which accesses the function through the pointer to the base class. Then, the current team's score is printed to the console using the "getScore" function.