First C++ Programs, Continued **(+Functions *(check attached pdf called Functions)*, Scope, Namespaces)**

Scope:

A declaration (*int i=3;* for example) is a statement that introduces the name of a variable into scope.

Scope: a specific region of program text where your names can be used

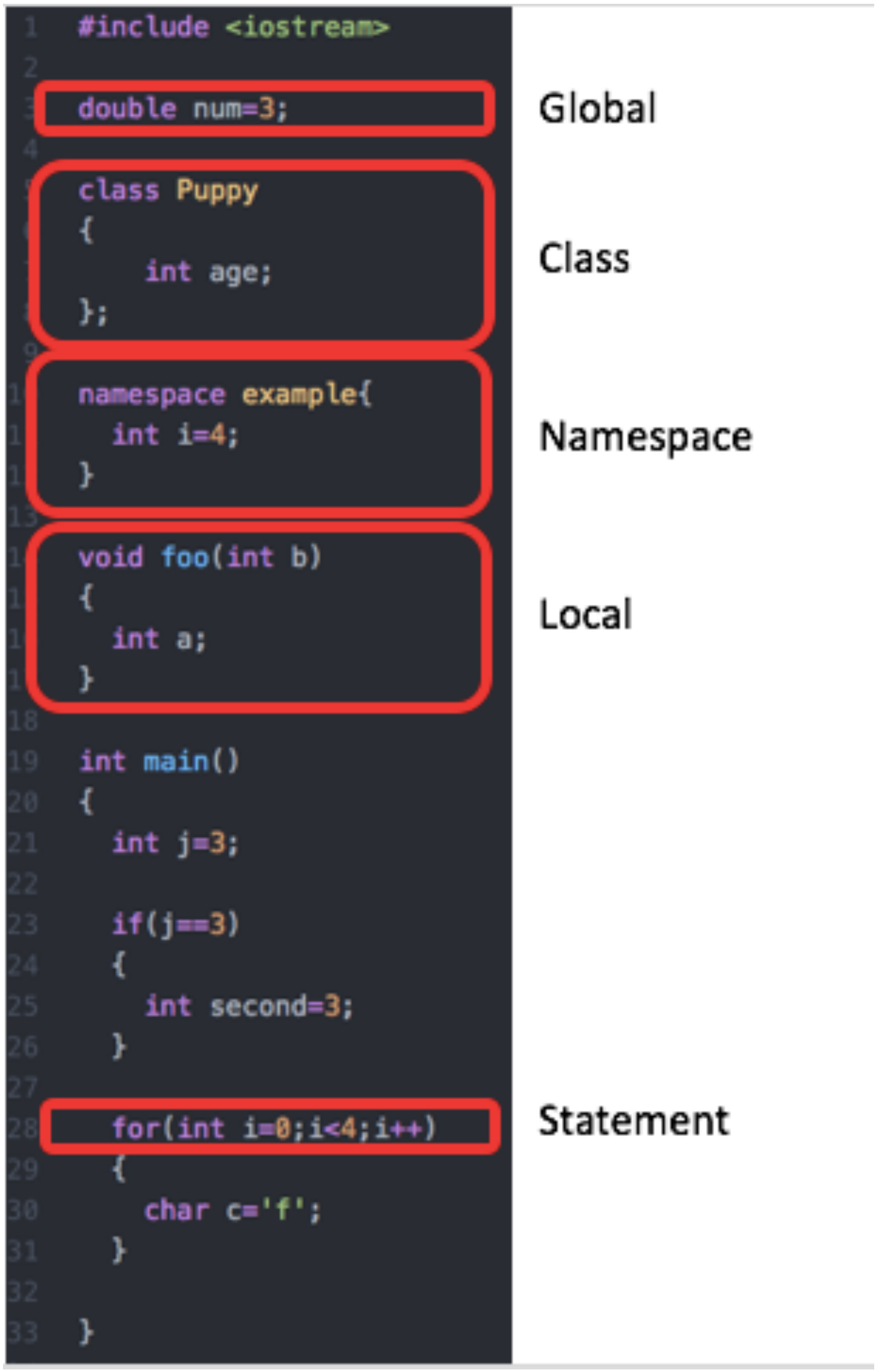
Local Scope: a variable is only visible within the curly brackets (see below)

Class Scope: area of code within a class

Namespace Scope: area of code within a namespace

Global Scope: area of code outside any other scope
Note: If I notice you overusing global variables, I will deduct points

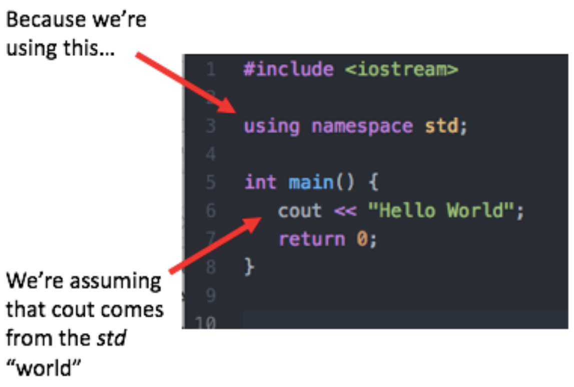
Statement Scope: inside a *for loop* for example



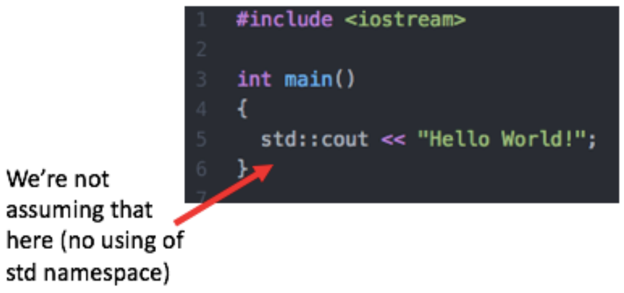
```cpp
#include <iostream>

double num=3;

class Puppy
{
    int age;
};

namespace example{
  int i=4;
}

void foo(int b)
{
  int a;
}

int main()
{
  int j=3;

  if(j==3)
  {
    int second=3;
  }

  for(int i=0;i<4;i++)
  {
    char c='f';
  }

}
```

Namespaces:

- I like to think of namespaces as “worlds” where certain things exist
- *std* is the “world” of the standard C++ library
- When we say we are using a certain namespace (**using namespace std;** for example), we are saying it can be assumed we are working in that world (and all the functions/variables not specified are assumed to be from that world).
- Without this statement, we need to include the “world” before (for example, we would need to say std::cout (:: is called the scope resolution operator) instead of just *cout* alone because it cannot be assumed we are using *cout* from the *std* “world”
    - Maybe we are using the function cout from another library
    - https://docs.microsoft.com/en-us/cpp/standard-library/cpp-standard-library-reference

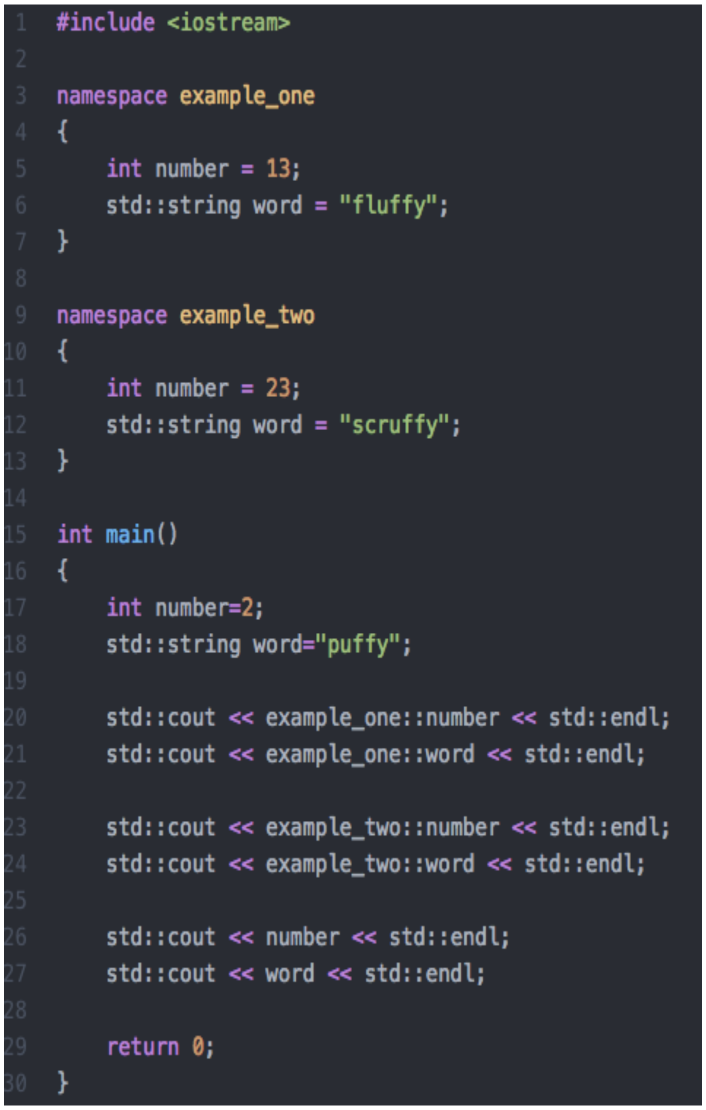

Because we’re using this…
We’re assuming that cout comes from the *std* “world”

```cpp
#include <iostream>

using namespace std;

int main() {
    cout << "Hello World";
    return 0;
}
```

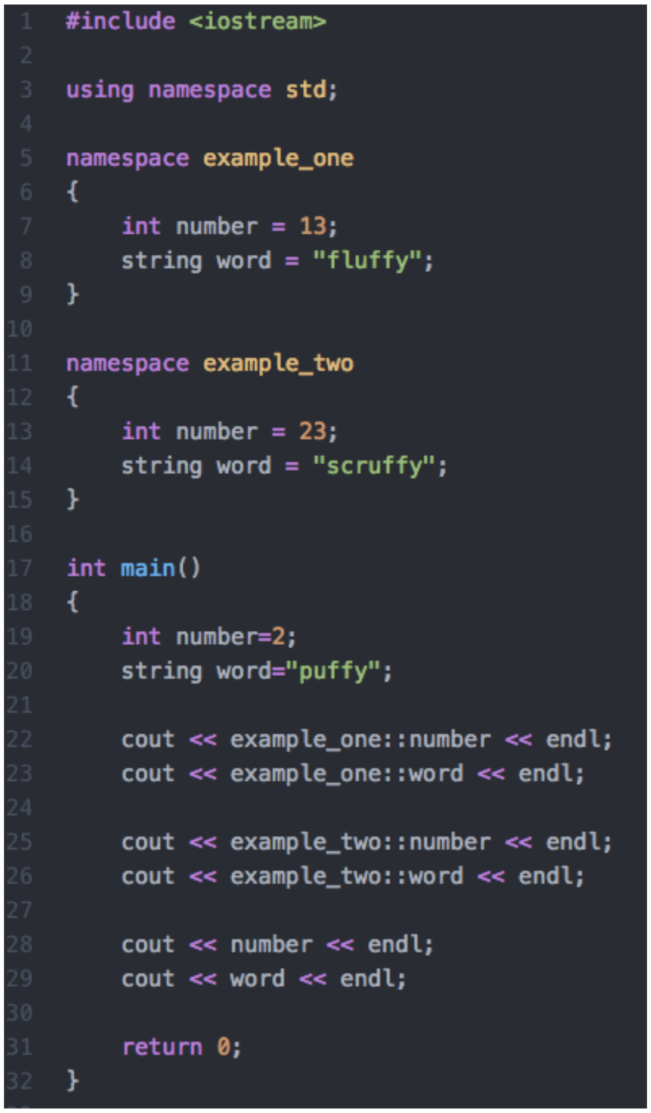


```cpp
#include <iostream>

int main()
{
  std::cout << "Hello World!";
}
```

- We can create our own namespaces:



```cpp
#include <iostream>

namespace example_one
{
    int number = 13;
    std::string word = "fluffy";
}

namespace example_two
{
    int number = 23;
    std::string word = "scruffy";
}

int main()
{
    int number=2;
    std::string word="puffy";

    std::cout << example_one::number << std::endl;
    std::cout << example_one::word << std::endl;

    std::cout << example_two::number << std::endl;
    std::cout << example_two::word << std::endl;

    std::cout << number << std::endl;
    std::cout << word << std::endl;

    return 0;
}
```



```cpp
#include <iostream>

using namespace std;

namespace example_one
{
    int number = 13;
    string word = "fluffy";
}

namespace example_two
{
    int number = 23;
    string word = "scruffy";
}

int main()
{
    int number=2;
    string word="puffy";

    cout << example_one::number << endl;
    cout << example_one::word << endl;

    cout << example_two::number << endl;
    cout << example_two::word << endl;

    cout << number << endl;
    cout << word << endl;

    return 0;
}
```

Output:

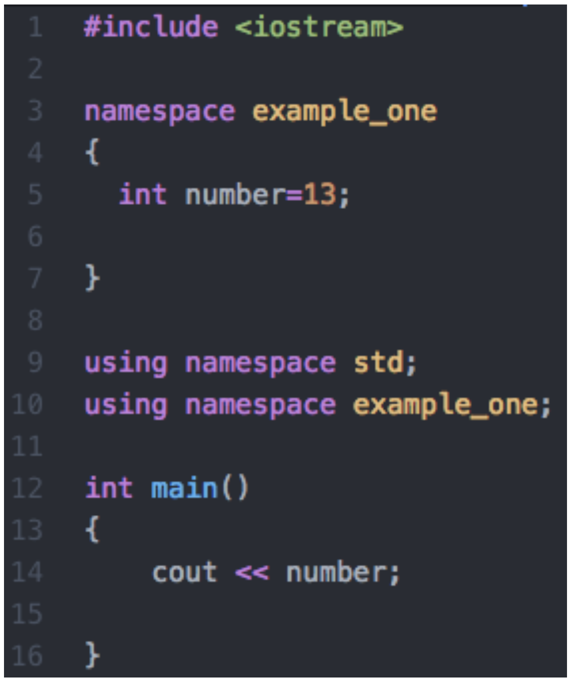
```
13
fluffy
23
scruffy
2
puffy
```

- Notice the picture on the right has **using namespace std**; so we don't precede anything from the std namespace with std. The variables on the left are preceded with std (we cant assume they are from any specific namespace)

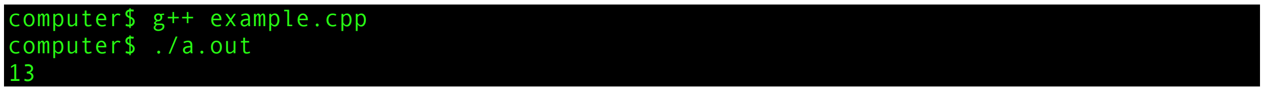
- Notice that our program knows which namespace we are using variables from because we precede with the namespace name. When we don't specify a name space, the local variable is assumed to be used (last output)

- We could also just use one item from a namespace: **using std::cout;** (*namespace using declaration*)

- We can also use the *using* keyword with the namespaces we create:

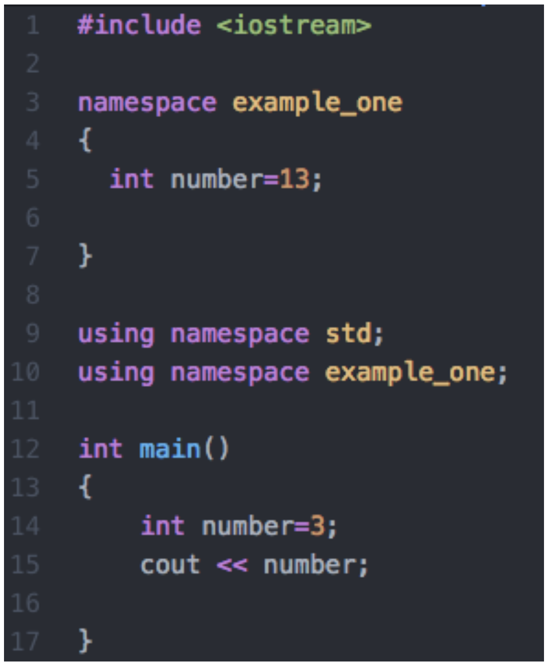


```cpp
#include <iostream>

namespace example_one
{
  int number=13;

}

using namespace std;
using namespace example_one;

int main()
{
    cout << number;

}
```



```
computer$ g++ example.cpp
computer$ ./a.out
13
```

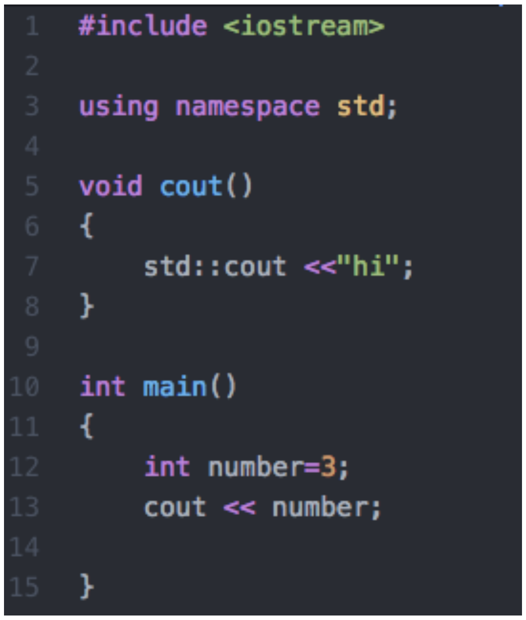
- Note that it will take a local variable named *number* before it would take the one used in the namespace (if not specified):



```cpp
#include <iostream>

namespace example_one
{
  int number=13;

}

using namespace std;
using namespace example_one;

int main()
{
    int number=3;
    cout << number;

}
```

```
computer$ g++ example.cpp
computer$ ./a.out
3
```

- It can get confusing when we use namespaces with the *using* keyword:



```cpp
#include <iostream>

using namespace std;

void cout()
{
    std::cout <<"hi";
}

int main()
{
    int number=3;
    cout << number;

}
```

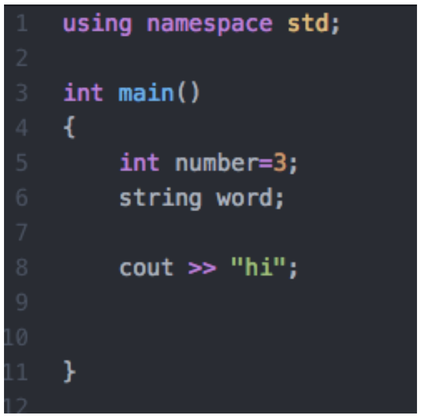
```
computer$ g++ example.cpp
closet.cpp:13:5: error: reference to 'cout' is ambiguous
    cout << number;
    ^
closet.cpp:5:6:       candidate found by name lookup is 'cout'
void cout()
     ^
/Library/Developer/CommandLineTools/usr/include/c++/v1/iostream:54:33:
      candidate found by name lookup is 'std::__1::cout'
extern _LIBCPP_FUNC_VIS ostream cout;
                                ^
1 error generated.
```

▪ It is not clear which *cout* you are referring to

- Notice that even though we are saying **using namespace std;** (meaning we are using things from the std namespace), we still need to include iostream (this is the actual headerfile) if using

- Don't confuse the idea of using a namespace with the idea of a header file

- The header file (like in C) actually contains your function declarations and other information



```
using namespace std;

int main()
{
    int number=3;
    string word;

    cout >> "hi";


}
```

```
computer$ g++ example.cpp
example.cpp:2:17: warning: using directive refers to implicitly-defined
namespace 'std'
using namespace std;
                ^
example.cpp:7:5: error: unknown type name 'string'
    string word;
    ^
example.cpp:9:5: error: use of undeclared identifier 'cout'
    cout >> "hi";
    ^
1 warning and 2 errors generated.
```

- Not including your header files (even though you say using namespace) will cause errors.

Program 1 (Namespace example)

```
computer$ g++ planet.cpp
computer$ ./a.out
Which planet are you interested in?
Jupiter
Jupiter info!
****Marvin is currently on Jupiter.***
Would you like to change the alien's name?
no
Marvin's name is staying the same!
```

#include <iostream>
#include <vector>
#include <string>

```
using namespace std;

namespace mars
{
 string alien_name="ET";

 void who_is_present()
 {
   cout<<"****"<<alien_name<<" is currently on Mars.***"<<endl;
 }

 void change_name()
 {
   string answer;

   cout<<"Enter new name?"<<endl;
   cin>>answer;

   if(answer.compare("yes")==0)
   {
     cout<<"New name: "<<endl;
     cin>>alien_name;
   }

 }

}

namespace jupiter
{
 string alien_name="Marvin";

 void who_is_present()
 {
   cout<<"****"<<alien_name<<" is currently on Jupiter.***"<<endl;
 }

 void change_name()
 {
   string answer;

   cout<<"Enter new name?"<<endl;
   cin>>answer;

   if(answer.compare("yes")==0)
   {
     cout<<"New name: "<<endl;
```

```
      cin>>alien_name;
    }

  }
}

//note: if you include both, it is ambiguous-which function are you calling?
using namespace mars;
//using namespace jupiter; you can switch them to see the diff in code

//0 means Jupiter, 1 means Mars, 2 means exit, 3 means unknown response
int which_planet(string answer)
{
  int ret;
  if(answer.compare("Jupiter")==0||answer.compare("jupiter")==0)
  {
    ret=0;
  }

  else if(answer.compare("Mars")==0||answer.compare("mars")==0)
  {
    ret=1;
  }

  else if(answer.compare("Exit")==0||answer.compare("exit")==0)
  {
    ret=2;
  }

  else //assume unknown response
  {
    ret=3;
  }

  return ret;
}



int main(int argc, char **argv)
{
  string answer;
  int choice;
  cout<<"Which planet are you interested in?"<<endl;
  cin>>answer;

  choice=which_planet(answer);
```

```
if(choice==2) //Exit
{
 cout<<"Bye!"<<endl;
}

else if(choice==3)//unknown response
{
 cout<<"Unknown response"<<endl;
}

else if(choice==1)//Mars
{
 cout<<"Mars info!"<<endl;
 who_is_present();

 cout<<"Would you like to change the alien's name?"<<endl;
 cin>>answer;

 if(answer=="yes")
 {
  change_name();
 }

 else
 {
  cout<<alien_name<<"'s name is staying the same!"<<endl;
 }

}

else //assume Jupiter, not using namespace jupiter so we will prefix
{
 cout<<"Jupiter info!"<<endl;
 jupiter::who_is_present(); //using scope resolution operator ::

 cout<<"Would you like to change the alien's name?"<<endl;
 cin>>answer;

 if(answer=="yes")
 {
  jupiter::change_name();
 }

 else
 {
  cout<<jupiter::alien_name<<"'s name is staying the same!"<<endl;
```

```
    }
  }
}
```

Program 2:

Marko wants to clean out and sell some of the items in his closet. He has asked you to create a program to help him do this. The program should allow him to enter the products he wants to sell (along with the price) and then continuously allow customers to look for a product and buy it. Marko should be able to check at any time the amount of money he has made from sales.

```
g++ -o closet closet.cpp
computer$ ./closet
Hello closet cleaner! Enter your name:
Marko

How many items do you want to sell?
2

Enter item and price: belt 2.99
Enter item and price: pants 3.99

**********
Marko's Closet!
**********
Hello shopper! What item are you looking for?
sunglasses
Sorry item wasn't found!
Hello shopper! What item are you looking for?
pants
We have: pants. Would you like to buy? yes

Ok! Purchase made!

Hello shopper! What item are you looking for?
check balance
Hello Marko! Your current balance is $3.99
Hello shopper! What item are you looking for?
exit
Bye!
```

```
#include <iostream>
#include <vector>
#include <string>
#include <iomanip>

using namespace std;

//This struct holds each item entered
struct item{
   string name;
   float price;
};

//This takes in the number of items for the user to enter and places them into a vector
vector <struct item> get_items(int num){
   vector <struct item> v;
   struct item item_get;

   for(int i=0;i<num;i++){
      cout << "Enter item and price: ";
      cin >> item_get.name >> item_get.price;
      v.push_back(item_get);
   }

   return v;
}


//This checks if the item is in the closet to buy
double buy_item(string n, vector <struct item> closet){
   string input;

   for(int i=0;i<closet.size();i++){
      //Item was found
      if(n.compare(closet[i].name)==0){
         cout << "We have: "<< n<<". Would you like to buy? ";
         getline(cin,input);

         //If user wants to purchase, return value of item
         if(input.compare("yes")==0){
            cout << "\nOk! Purchase made!"<< endl;
            return closet[i].price;
         }

         //Assume otherwise purchase is not made
         else{
            cout << "\nOk!" << endl;
            return 0; //Return 0 if no purchase is made
         }
      }
   }
   //At end of loop, item is not found
   cout << "Sorry item wasn't found!" << endl;
   return 0;
}
```

```
int main(int argc, char **argv) {
   string user_name;
   int num_items;

   //Get user info
   cout << "Hello closet cleaner! Enter your name: "<<endl;
   cin >> user_name;
   cout << endl << "How many items do you want to sell? "<<endl;
   cin >> num_items;
   cout << endl;

   //Get user input
   vector <struct item> closet=get_items(num_items);

   cout << endl << "**********" << endl << user_name <<"'s Closet!" << endl << "**********" << endl;

   bool check=true;
   string input;
   double balance=0;
   getchar();
   while(check){
      cout << "Hello shopper! What item are you looking for?" << endl;

      getline(cin,input);

      //Exit
      if(input.compare("exit")==0){
         cout << "Bye!" << endl;
         check=false;
         break;
      }

      //Let user check balance
      else if(input.compare("check balance")==0){
         cout << "Hello " << user_name << "! Your current balance is $" << fixed << setprecision(2) <<
balance << endl;
      }

      //update balance
      else{
         balance = balance + buy_item(input, closet); //either the price of an item or 0 is returned-adding
//0 to the price won't change anything
      }

   }

   return 0;
}
```