SRS Setup
Login: student.turningtechnologies.com
Session ID: 20220131<A|D>

Replace <A|D> with this section's letter

Structs

—

CS 2124: Object Oriented Programming
Darryl Reeves, Ph.D.

Agenda

- Structs
- Programming tips
- In-class problem

Structs

Structures for grouping values

- grouping related values often desired
- vectors provide ability with restrictions (type)

```
vector<string> student_names{"Charlene", "John", "Simon", "Julie"};
```

- **structs** allow us to store 2 or more values together
  - values can be different types
  - names given to values for referencing

```
struct Cat {
   string color;
   string name;
   double weight;
};
```

What if we wanted to store attributes of a cat?

name	value
color	brown
name	Whiskers
weight	8

`struct` format

name for `struct`
(traditionally capitalized)

open curly brace

keyword indicates `struct` being defined

```
struct StructName {
    type_1 attribute_1;
    type_2 attribute_2;
    type_3 attribute_3;
        ...
};
```

type must be specified for every attribute

each value in the struct must be given an *attribute* name

attributes aka *fields* or *member variables*

close curly brace

semicolon terminates definition (very important!)

Assigning `struct` values

```
Cat my_cat;

my_cat.name = "Felix";
my_cat.color = "grey";
my_cat.weight = 3.14;

cout << my_cat.name << '\t' << my_cat.color << '\t' << my_cat.weight << endl;
```

Note: *dot-notation* used to access attributes

Initializing `struct` on creation

```
struct Cat {
   string color;
   string name;
   double weight;
};
```

+

```
Cat my_cat;

my_cat.name = "Whiskers";
my_cat.color = "brown";
my_cat.weight = 8;
```

```
struct Cat {
   string color = "brown";
   string name = "Whiskers";
   double weight = 8;
};
```

requires C++11 or later

Assigning `struct` objects

```
struct Cat {
   string color;
   string name;
   double weight;
};
```

```
Cat my_cat;

my_cat.name = "Whiskers";
my_cat.color = "brown";
my_cat.weight = 8;
```

Cat jims_cat;
jims_cat = my_cat;

equivalent to

```
Cat jims_cat;

jims_cat.name = "Whiskers";
jims_cat.color = "brown";
jims_cat.weight = 8;
```

a lot less typing!!

Testing for equality

```
if (my_cat == jims_cat) {
    cout << "cats are the same!" << endl;
}
```

won't work! (results in compilation error)

```
if (

    /* need to test all attributes individually */

){
    cout << "cats are the same!" << endl;
}
```

Testing for equality

will learn implementation later

```
if (my_cat == jims_cat) {
    cout << "cats are the same!" << endl;
}
```

won't work! (results in compilation error)

```
if (
    my_cat.name == jims_cat.name &&
    my_cat.color == jims_cat.color &&
    my_cat.weight == jims_cat.weight
){
    cout << "cats are the same!" << endl;
}
```

Printing structs

will learn implementation later

```
cout << my_cat << endl;
```

won't work! (results in compilation error)

```
void print_cat(const Cat& kitty) {
    cout << kitty.name << ' ' << kitty.color << ' ' << kitty.weight << endl;
}
```

Reading structs from file

```
void fill_cat_vector(ifstream& in_fs, vector<Cat>& in_vec) {
    Cat kitty;

    while (in_fs >> kitty.name >> kitty.color >> kitty.weight) {
        in_vec.push_back(kitty);
    }

}
```

- references used for `ifstream` and `vector` parameters

Whiskers brown 8
Felix grey 6.3
Garfield orange 10.1

Reading structs from file

```
void fill_cat_vector(ifstream& in_fs, vector<Cat>& in_vec) {
    Cat kitty;

    while (in_fs >> kitty.name >> kitty.color >> kitty.weight) {
        in_vec.push_back(kitty);
    }

}
```

Whiskers brown 8
Felix grey 6.3
Garfield orange 10.1

- references used for `ifstream` and `vector` parameters
- assumes a file with one cat per line
  - components separated by space

Programming tips

Globals

- DON'T USE GLOBAL VARIABLES IN CS 2124

one major exception:

- **global constants** are helpful AND encouraged

```
const int CLASS_NUM = 2124;
```

Note: all upper-case letters

Scope

- limit scope of variables as much as possible

```
void update_vector(vector<int>& int_vec, int new_val);

void print_vector(const vector<int>& int_vec) {
    for (int num; int_vec) {
         cout << num << ' ';
    }
    cout << endl;
}
int main() {
    vector<int> nums(1000000000000, 0);

    update_vector(nums, -1);
    print_vector(nums);
}
```

Scope

- limit scope of variables as much as possible

```
vector<int> nums(1000000000000, 0);

void update_vector(vector<int>& int_vec, int new_val);

void print_vector(const vector<int>& int_vec) {
    for (int num; int_vec) {
        cout << num << ' ';
    }
    cout << endl;
}

int main() {
    update_vector(nums, -1);
    print_vector(nums);
}
```

code works but uses global

Scope

- limit scope of variables as much as possible

```
/* i not used to print count above */

int i;
for (i = 0; i < 10; i++) {
     cout << i << ' ';
}
cout << endl;

/* i not used to print count below */
```

Scope

- limit scope of variables as much as possible

```
/* i not used to print count above */

for (int i = 0; i < 10; i++) {
     cout << i << ' ';
}
cout << endl;
/* i not used to print count below */
```

Scope

- limit scope of variables as much as possible

```
int main() {
   int num;

   cout << "Enter a positive integer: ";
   cin >> num;

   int sum = 0;
   for (int i = 0; i <= num; ++i) {
        sum += i;
   }
   cout << "Sum = " << sum << endl;
}
```

`sum` needed outside of `for` loop

Coding guidelines

https://cse.engineering.nyu.edu/jsterling/cs2124/Notes/CodeGuideline.html

Note: I use underscores in variable names.
Choose your style and be consistent.

In-class problem

Automobile dealership program

TurningPoint

SRS Setup
Login: student.turningtechnologies.com
Session ID: 20220131<A|D>

Replace <A|D> with this section's letter

What data about an automobile would be useful for an automobile dealership program?

Which feature can we use to group automobile details together in a C++ program for representing a real-world automobile?

Program design choices

- represent vehicle using `struct`
  - year
  - make (Nissan, Toyota, Ford, etc)
  - model (Altima, Corolla, Mustang, etc)
  - mileage
  - price
- `Vehicle` structs stored in `vector` (inventory)
- simple input interface for populating inventory

```
Welcome to CS 2124 Autos!

Enter new vehicle details.
Add a vehicle (y/n)? y
Enter year: 2007
Enter make: Nissan
Enter model: Altima
Enter price: 11200
Enter mileage: 70000

Add a vehicle (y/n)? y
Enter year: 2019
Enter make: Ford
Enter model: Mustang
Enter price: 28998
Enter mileage: 18230

Add a vehicle (y/n)? n
Added 2 vehicles.
```

Defining a `Vehicle` type

```
struct ___ ___
    // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
___ ___
```

Defining a `Vehicle` type

```
struct _1_ ___
    // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
___ ___
```

Which token/word should replace blank #1 to create a `struct` named `Vehicle`?

```
struct _1_ ___
    // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
___ ___
```

Defining a `Vehicle` type

```
struct Vehicle ___
    // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
--- ---
```

Defining a `Vehicle` type

```
struct Vehicle _2_
    // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
--- ---
```

Which token/symbol should replace blank #2 to properly define the `Vehicle struct`?

```
struct Vehicle _2_
    // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
--- ---
```

Defining a `Vehicle` type

```
struct Vehicle {
    // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
--- ---
```

Defining a `Vehicle` type

```
struct Vehicle {
    // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
_3_ ___
```

Which token/symbol should replace blank #3 to properly define the `Vehicle struct`?

```
struct Vehicle {
    // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
_3_ ___
```

Defining a `Vehicle` type

```
struct Vehicle {
    // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
} ___
```

Defining a `Vehicle` type

```
struct Vehicle {
    // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
}_4_
```

Which token/symbol should replace blank #4 to properly define the `Vehicle struct`?

```
struct Vehicle {
    // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
}_4_
```

Defining a `Vehicle` type

```
struct Vehicle {
    // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
struct Vehicle {
    short ___ ___ // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
struct Vehicle {
    short _5_ ___ // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Which name should replace blank #5 in order to declare the `year` member of the `Vehicle struct`?

```
struct Vehicle {
    short _5_ ___ // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
struct Vehicle {
    short year ___ // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
struct Vehicle {
    short year_6_ // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

What should replace blank #6 in order to finish defining the `year` member of the `Vehicle struct`?

```
struct Vehicle {
    short year _6_ // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Which type should be used for the `make` member (e.g. *Nissan*, *Toyota*, *Ford*, etc)?

```
struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
--- ---

struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

_6_ ___

```
struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Which directive should replace blank #6 in order to use the `string` type in the program?

```
_6_ ___

struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include ___

struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Which token should replace blank #7 in order to use the `string` type in the program?

```
#include _7_

struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>

struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

string type specified with std::string

Defining a `Vehicle` type

```
#include <string>
--- --- --- ---

struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
_8_ ___ ___ ___

struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Which keyword (replacing blank #8) begins the directive allowing strings to be written as `string` instead of `std::string`?

```
#include <string>
_8_ ___ ___ ___

struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using ___ ___ ___


struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using _9_ ___ ___


struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Which keyword (replacing blank #9) follows `using` in the directive allowing strings to be written as `string` instead of `std::string`?

```
#include <string>
using _9_ ___ ___


struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using namespace ___ ___


struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using namespace _10_ ___


struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Which `namespace` should replace blank #10 to allow strings to be written as `string` instead of `std::string`?

```
#include <string>
using namespace _10_ ___


struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using namespace std ___


struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using namespace std _11_


struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

What should replace blank #11 in order to terminate the directive allowing strings to be written as `string` instead of `std::string`?

```
#include <string>
using namespace std _11_


struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    ___ ___ ___ // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    _12_ ___ ___ // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Which type should replace blank #12 to declare the `make` member of the `Vehicle struct`?

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    _12_ ___ ___ // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string ___ ___ // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string _13_ ___ // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Which name should replace blank #13 to declare the `make` member of the `Vehicle struct`?

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string _13_ ___ // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make ___ // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

What should replace blank #14 to declare the `make` member of the `Vehicle struct`?

```
#include <string>
using namespace std;


struct Vehicle {
    short year; //declare year member
    string make _14_ // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    _15_ // declare model member
    // declare mileage member
    // declare price member
};
```

Which declaration should replace blank #15 to declare the `model` member as a `string`?

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    _15_ // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    _16_ // declare mileage member
    // declare price member
};
```

Which declaration should replace blank #16 to declare the `mileage` member as an `int`?

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    _16_ // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    _17_ // declare price member
};
```

Which declaration should replace blank #17 to declare the `price` member as a `double`?

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    _17_ // declare price member
};
```

Defining a `Vehicle` type

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    double price; // declare price member
};
```

Declaring `add_to_inventory` function

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    double price; // declare price member
};

// function prototype for an add_to_inventory function
```

The `add_to_inventory` function will not return a value. Which keyword should be listed for the return type?

Declaring `add_to_inventory` function

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    double price; // declare price member
};

void add_to_inventory(/* function parameter */)_16_
```

What should replace blank #18 in order to complete the function prototype?

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    double price; // declare price member
};

void add_to_inventory(/* function parameter */)_18_
```

vector of Vehicles

Declaring `add_to_inventory` function

```
#include <string>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    double price; // declare price member
};

void add_to_inventory(/* function parameter */);
```

vector of Vehicles

Declaring `add_to_inventory` function

```
#include <string>
// enable using vector type
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    double price; // declare price member
};

void add_to_inventory(/* function parameter */);
```

vector of Vehicles

Declaring `add_to_inventory` function

```
#include <string>
_19_ // enable using vector type
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    double price; // declare price member
};

void add_to_inventory(/* function parameter */);
```

vector of Vehicles

With which directive do we need to replace blank #19 in order to use `vector` objects?

```
#include <string>
_19_ // enable using vector type
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    double price; // declare price member
};

void add_to_inventory(/* function parameter */);
```

Declaring `add_to_inventory` function

```
#include <string>
#include <vector>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    double price; // declare price member
};

void add_to_inventory(_20_);
```

vector of Vehicles

The parameter for add_to_inventory is a **modifiable** `vector` of `Vehicle` structs. What should replace blank #20?

```
#include <string>
#include <vector>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    double price; // declare price member
};

void add_to_inventory(_20_);
```

`vector` of `Vehicle`s

Declaring `add_to_inventory` function

```
#include <string>
#include <vector>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    double price; // declare price member
};

void add_to_inventory(vector<Vehicle>&);
```

& (reference operator) allows function argument to be modified in calling scope

Defining a `main()` function

```
int main() {

    char usr_input = 0;
    vector<Vehicle> auto_lot;

    cout << "Welcome to CS 2124 Autos!" << endl << endl;
    cout << "Enter new vehicle details." << endl;

    while (usr_input != 'n') {

        while (usr_input != 'n' ___ usr_input != 'y') {
            cout << "Add a vehicle (y/n)? ";
            cin >> usr_input;
        }

        if (usr_input == 'y') {
            add_to_inventory(auto_lot);
            usr_input = 0;
        }

    }


    cout << "Added " << auto_lot.size() << " vehicles." << endl;

}
```

```
Welcome to CS 2124 Autos!

Enter new vehicle details.
Add a vehicle (y/n)? y
Enter year: 2007
Enter make: Nissan
Enter model: Altima
Enter price: 11200
Enter mileage: 70000

Add a vehicle (y/n)? y
Enter year: 2019
Enter make: Ford
Enter model: Mustang
Enter price: 28998
Enter mileage: 18230

Add a vehicle (y/n)? n
Added 2 vehicles.
```

Which logical operator should replace blank #21 requiring both conditions to be true?

```
int main() {

    char usr_input = 0;
    vector<Vehicle> auto_lot;

    cout << "Welcome to CS 2124 Autos!" << endl << endl;
    cout << "Enter new vehicle details." << endl;

    while (usr_input != 'n') {

        while (usr_input != 'n' _21_ usr_input != 'y') {
            cout << "Add a vehicle (y/n)? ";
            cin >> usr_input;
        }

        if (usr_input == 'y') {
            add_to_inventory(auto_lot);
            usr_input = 0;
        }

    }


    cout << "Added " << auto_lot.size() << " vehicles." << endl;

}
```

```
Welcome to CS 2124 Autos!

Enter new vehicle details.
Add a vehicle (y/n)? y
Enter year: 2007
Enter make: Nissan
Enter model: Altima
Enter price: 11200
Enter mileage: 70000

Add a vehicle (y/n)? y
Enter year: 2019
Enter make: Ford
Enter model: Mustang
Enter price: 28998
Enter mileage: 18230

Add a vehicle (y/n)? n
Added 2 vehicles.
```

Defining a `main()` function

```
int main() {

    char usr_input = 0;
    vector<Vehicle> auto_lot;

    cout << "Welcome to CS 2124 Autos!" << endl << endl;
    cout << "Enter new vehicle details." << endl;

    while (usr_input != 'n') {

        while (usr_input != 'n' && usr_input != 'y') {
            cout << "Add a vehicle (y/n)? ";
            cin >> usr_input;
        }

        if (usr_input == 'y') {
            add_to_inventory(auto_lot);
            usr_input = 0;
        }

    }

    cout << "Added " << auto_lot.size() << " vehicles." << endl;

}
```

need to enable streams

```
Welcome to CS 2124 Autos!

Enter new vehicle details.
Add a vehicle (y/n)? y
Enter year: 2007
Enter make: Nissan
Enter model: Altima
Enter price: 11200
Enter mileage: 70000

Add a vehicle (y/n)? y
Enter year: 2019
Enter make: Ford
Enter model: Mustang
Enter price: 28998
Enter mileage: 18230

Add a vehicle (y/n)? n
Added 2 vehicles.
```

Accessing stream objects

```
#include <string>
#include <vector>
// enable usage of input/output streams
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    double price; // declare price member
};

void add_to_inventory(vector<Vehicle>&);
```

Accessing stream objects

```
#include <string>
#include <vector>
_22_ // enable usage of input/output streams
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    double price; // declare price member
};

void add_to_inventory(vector<Vehicle>&);
```

Which directive should replace blank #22 to enable usage of input/output streams in the program?

```
#include <string>
#include <vector>
_22_ // enable usage of input/output streams
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    double price; // declare price member
};

void add_to_inventory(vector<Vehicle>&);
```

Accessing stream objects

```
#include <string>
#include <vector>
#include <iostream>
using namespace std;


struct Vehicle {
    short year; // declare year member
    string make; // declare make member
    string model; // declare model member
    int mileage; // declare mileage member
    double price; // declare price member
};

void add_to_inventory(vector<Vehicle>&);
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(___ ___) {
    // body of add_to_inventory
}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(_24_ ___) {
    // body of add_to_inventory
}
```

Which type should be used for the `add_to_inventory()` function's parameter (replacing blank #24)?

```
// function prototype
void add_to_inventory(vector<Vehicle>&);


void add_to_inventory(_24_ ___) {
    // body of add_to_inventory
}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& ___) {
    // body of add_to_inventory
}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& _25_) {
    // body of add_to_inventory
}
```

The name of the `Vehicle` `vector` parameter is **inventory**. Which name should replace blank #25 for completing the parameter declaration of `add_to_inventory()`?

```
// function prototype
void add_to_inventory(vector<Vehicle>&);


void add_to_inventory(vector<Vehicle>& _25_) {
    // body of add_to_inventory
}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    ___ ___ ___ // declare Vehicle named auto_
}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    _26_ ___ ___ // declare Vehicle named auto_
}
```

Which type should replace blank #26 for declaring a `Vehicle` named `auto_`?

```
void add_to_inventory(vector<Vehicle>& inventory) {
    _26_ ___ ___ // declare Vehicle named auto_
}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle ___ ___ // declare Vehicle named auto_
}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle _27_ ___ // declare Vehicle named auto_
}
```

Which name should replace blank #27 for the `Vehicle` declaration?

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle _27_ ___ // declare Vehicle named auto_
}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_ ___ // declare Vehicle named auto_
}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_ _28_ // declare Vehicle named auto_
}
```

What should replace blank #28 to complete the declaration of the *local* variable `auto_`?

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_ _28_ // declare Vehicle named auto_
}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_
}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    ___ ___ ___ ___ // prompt for Vehicle year
}
```

```
Welcome to CS 2124 Autos!

Enter new vehicle details.
Add a vehicle (y/n)? y
Enter year: 2007
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    _29_ ___ ___ ___ // prompt for Vehicle year
}
```

```
Welcome to CS 2124 Autos!

Enter new vehicle details.
Add a vehicle (y/n)? y
Enter year: 2007
```

Which stream object should replace blank #28 to display text to standard output?

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    _29_ ___ ___ ___ // prompt for Vehicle year
}
```

```
Welcome to CS 2124 Autos!

Enter new vehicle details.
Add a vehicle (y/n)? y
Enter year: 2007
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout ___ ___ ___ // prompt for Vehicle year
}
```

```
Welcome to CS 2124 Autos!

Enter new vehicle details.
Add a vehicle (y/n)? y
Enter year: 2007
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout _30_ ___ ___ // prompt for Vehicle year
}
```

```
Welcome to CS 2124 Autos!

Enter new vehicle details.
Add a vehicle (y/n)? y
Enter year: 2007
```

Which operator should replace blank #30 to display text to standard output?

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout _30_ ___ ___ // prompt for Vehicle year
}
```

```
Welcome to CS 2124 Autos!

Enter new vehicle details.
Add a vehicle (y/n)? y
Enter year: 2007
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << ___ ___ // prompt for Vehicle year
}
```

```
Welcome to CS 2124 Autos!

Enter new vehicle details.
Add a vehicle (y/n)? y
Enter year: 2007
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << _31_ ___ // prompt for Vehicle year
}
```

```
Welcome to CS 2124 Autos!

Enter new vehicle details.
Add a vehicle (y/n)? y
Enter year: 2007
```

Which `string` *literal* should replace blank #31 to prompt the user for the `Vehicle`'s year?

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << _31_ ___ // prompt for Vehicle year
}
```

```
Welcome to CS 2124 Autos!

Enter new vehicle details.
Add a vehicle (y/n)? y
Enter year: 2007
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: " _32_ // prompt for Vehicle year
}
```

```
Welcome to CS 2124 Autos!

Enter new vehicle details.
Add a vehicle (y/n)? y
Enter year: 2007
```

What should replace blank #32 to complete the expression prompting the user for `Vehicle year`?

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: " _32_ // prompt for Vehicle year
}
```

```
Welcome to CS 2124 Autos!

Enter new vehicle details.
Add a vehicle (y/n)? y
Enter year: 2007
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
}
```

```
Welcome to CS 2124 Autos!

Enter new vehicle details.
Add a vehicle (y/n)? y
Enter year: 2007
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    // store input value as year member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    ___ ___ ___ ___ // store input value as year member of Vehicle

}
```

Which stream object should replace blank #32 to assign the user's input to the `year` member of the `Vehicle` auto_?

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    _32_ ___ ___ ___ // store input value as year member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin ___ ___ ___ // store input value as year member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin _33_ ___ ___ // store input value as year member of Vehicle

}
```

Which operator should replace blank #33 to extract input entered by the user?

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin _33_ ___ ___ // store input value as year member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin >> ___ ___ // store input value as year member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin >> _34_ ___ // store input value as year member of Vehicle

}
```

Which variable should replace blank #34 to store the user's input of the `Vehicle year`?

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin >> _34_ ___ // store input value as year member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin >> auto_.year ___ // store input value as year member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin >> auto_.year _35_ // store input value as year member of Vehicle

}
```

What should replace blank #35 to complete the expression storing the user's input for `Vehicle year`?

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin >> auto_.year _35_ // store input value as year member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin >> auto_.year; // store input value as year member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin >> auto_.year; // store input value as year member of Vehicle
    // prompt for Vehicle make
    // store input value as make member of Vehicle
    // prompt for Vehicle model
    // store input value as model member of Vehicle
    // prompt for Vehicle mileage
    // store input value as mileage member of Vehicle
    // prompt for Vehicle price
    // store input value as price member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin >> auto_.year; // store input value as year member of Vehicle
    cout << "Enter make: "; // prompt for Vehicle make
    // store input value as make member of Vehicle
    // prompt for Vehicle model
    // store input value as model member of Vehicle
    // prompt for Vehicle mileage
    // store input value as mileage member of Vehicle
    // prompt for Vehicle price
    // store input value as price member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin >> auto_.year; // store input value as year member of Vehicle
    cout << "Enter make: "; // prompt for Vehicle make
    cin >> auto_.make; // store input value as make member of Vehicle
    // prompt for Vehicle model
    // store input value as model member of Vehicle
    // prompt for Vehicle mileage
    // store input value as mileage member of Vehicle
    // prompt for Vehicle price
    // store input value as price member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin >> auto_.year; // store input value as year member of Vehicle
    cout << "Enter make: "; // prompt for Vehicle make
    cin >> auto_.make; // store input value as make member of Vehicle
    cout << "Enter model: "; // prompt for Vehicle model
    // store input value as model member of Vehicle
    // prompt for Vehicle mileage
    // store input value as mileage member of Vehicle
    // prompt for Vehicle price
    // store input value as price member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin >> auto_.year; // store input value as year member of Vehicle
    cout << "Enter make: "; // prompt for Vehicle make
    cin >> auto_.make; // store input value as make member of Vehicle
    cout << "Enter model: "; // prompt for Vehicle model
    cin >> auto_.model; // store input value as model member of Vehicle
    // prompt for Vehicle mileage
    // store input value as mileage member of Vehicle
    // prompt for Vehicle price
    // store input value as price member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin >> auto_.year; // store input value as year member of Vehicle
    cout << "Enter make: "; // prompt for Vehicle make
    cin >> auto_.make; // store input value as make member of Vehicle
    cout << "Enter model: "; // prompt for Vehicle model
    cin >> auto_.model; // store input value as model member of Vehicle
    cout << "Enter mileage: "; // prompt for Vehicle mileage
    // store input value as mileage member of Vehicle
    // prompt for Vehicle price
    // store input value as price member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin >> auto_.year; // store input value as year member of Vehicle
    cout << "Enter make: "; // prompt for Vehicle make
    cin >> auto_.make; // store input value as make member of Vehicle
    cout << "Enter model: "; // prompt for Vehicle model
    cin >> auto_.model; // store input value as model member of Vehicle
    cout << "Enter mileage: "; // prompt for Vehicle mileage
    cin >> auto_.mileage; // store input value as mileage member of Vehicle
    // prompt for Vehicle price
    // store input value as price member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin >> auto_.year; // store input value as year member of Vehicle
    cout << "Enter make: "; // prompt for Vehicle make
    cin >> auto_.make; // store input value as make member of Vehicle
    cout << "Enter model: "; // prompt for Vehicle model
    cin >> auto_.model; // store input value as model member of Vehicle
    cout << "Enter mileage: "; // prompt for Vehicle mileage
    cin >> auto_.mileage; // store input value as mileage member of Vehicle
    cout << "Enter price: "; // prompt for Vehicle price
    // store input value as price member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin >> auto_.year; // store input value as year member of Vehicle
    cout << "Enter make: "; // prompt for Vehicle make
    cin >> auto_.make; // store input value as make member of Vehicle
    cout << "Enter model: "; // prompt for Vehicle model
    cin >> auto_.model; // store input value as model member of Vehicle
    cout << "Enter mileage: "; // prompt for Vehicle mileage
    cin >> auto_.mileage; // store input value as mileage member of Vehicle
    cout << "Enter price: "; // prompt for Vehicle price
    cin >> auto_.price; // store input value as price member of Vehicle

}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_; // declare Vehicle named auto_

    // assign Vehicle members from user input
    cout << "Enter year: "; // prompt for Vehicle year
    cin >> auto_.year; // store input value as year member of Vehicle
    cout << "Enter make: "; // prompt for Vehicle make
    cin >> auto_.make; // store input value as make member of Vehicle
    cout << "Enter model: "; // prompt for Vehicle model
    cin >> auto_.model; // store input value as model member of Vehicle
    cout << "Enter mileage: "; // prompt for Vehicle mileage
    cin >> auto_.mileage; // store input value as mileage member of Vehicle
    cout << "Enter price: "; // prompt for Vehicle price
    cin >> auto_.price; // store input value as price member of Vehicle
    cout << endl;
}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_;

    cout << "Enter year: ";
    cin >> auto_.year;
    cout << "Enter make: ";
    cin >> auto_.make;
    cout << "Enter model: ";
    cin >> auto_.model;
    cout << "Enter mileage: ";
    cin >> auto_.mileage;
    cout << "Enter price: ";
    cin >> auto_.price;
    cout << endl;

    // add vehicle to vector
    _36_
}
```

Which expression should replace blank #36 to add `auto_` to the end of the `vector inventory`?

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_;

    cout << "Enter year: ";
    cin >> auto_.year;
    cout << "Enter make: ";
    cin >> auto_.make;
    cout << "Enter model: ";
    cin >> auto_.model;
    cout << "Enter mileage: ";
    cin >> auto_.mileage;
    cout << "Enter price: ";
    cin >> auto_.price;
    cout << endl;

    // add vehicle to vector
    _36_
}
```

Defining the `add_to_inventory()` function

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_;

    cout << "Enter year: ";
    cin >> auto_.year;
    cout << "Enter make: ";
    cin >> auto_.make;
    cout << "Enter model: ";
    cin >> auto_.model;
    cout << "Enter mileage: ";
    cin >> auto_.mileage;
    cout << "Enter price: ";
    cin >> auto_.price;
    cout << endl;

    inventory.push_back(auto_);
}
```

Full program (includes, `struct`, and function prototype)

```
#include <vector>
#include <string>
#include <iostream>
using namespace std;


struct Vehicle {
    short year;
    string make;
    string model;
    int mileage;
    double price;
};

void add_to_inventory(vector<Vehicle>&);
```

Full program (`main()` function)

```
int main() {
    char usr_input = 0;
    vector<Vehicle> auto_lot;

    cout << "Welcome to CS 2124 Autos!" << endl << endl;
    cout << "Enter new vehicle details." << endl;


    while (usr_input != 'n') {

        while (usr_input != 'n' && usr_input != 'y') {
            cout << "Add a vehicle (y/n)? ";
            cin >> usr_input;
        }

        if (usr_input == 'y') {
            add_to_inventory(auto_lot);
            usr_input = 0;
        }
    }

    cout << "Added " << auto_lot.size() << " vehicles." << endl;

}
```

Full program (`add_to_inventory()` function)

```
void add_to_inventory(vector<Vehicle>& inventory) {
    Vehicle auto_;

    cout << "Enter year: ";
    cin >> auto_.year;
    cout << "Enter make: ";
    cin >> auto_.make;
    cout << "Enter model: ";
    cin >> auto_.model;
    cout << "Enter mileage: ";
    cin >> auto_.mileage;
    cout << "Enter price: ";
    cin >> auto_.price;
    cout << endl;

    inventory.push_back(auto_);
}
```

Input from a file of vehicles

- typing details for each vehicle becomes tedious

vehicles.txt

```
2007 Nissan Altima 11200 70000
2019 Ford Mustang 28998 18230
2012 Toyota Corolla 14349 43819
...
```

```
Welcome to CS 2124 Autos!

Provide a vehicle file
Filename: cars.txt
Failed to open vehicle file

Filename: vehicles.txt

Added 3 vehicles.
```

How can we support file input of vehicles?

Reading vehicles from a file

```
void read_inventory(___ ___, vector<Vehicle>& inventory) {

}
```

Which ***type*** should replace blank #37 to declare a modifiable file input stream?

```
void read_inventory(_37_ ___, vector<Vehicle>& inventory) {

}
```

Reading vehicles from a file

```
void read_inventory(ifstream& ___, vector<Vehicle>& inventory) {

}
```

Reading vehicles from a file

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {

}
```

Reading vehicles from a file

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (___) {
        inventory.push_back(auto_);
    }

}
```

Reading vehicles from a file

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (_39_) {
        inventory.push_back(auto_);
    }

}
```

Which ***object*** should replace blank #39 to extract tokens from the input file?

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (_39_) {
        inventory.push_back(auto_);
    }

}
```

vehicles.txt

```
2007 Nissan Altima 11200 70000
2019 Ford Mustang 28998 18230
2012 Toyota Corolla 14349 43819
...
```

Reading vehicles from a file

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (ifs ) {
        inventory.push_back(auto_);
    }

}
```

Reading vehicles from a file

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (ifs ___) {
        inventory.push_back(auto_);
    }

}
```

Reading vehicles from a file

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (ifs _40_) {
        inventory.push_back(auto_);
    }

}
```

Which ***operator*** should replace blank #40 to extract tokens from the input file?

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (ifs _40_) {
        inventory.push_back(auto_);
    }

}
```

vehicles.txt

```
2007 Nissan Altima 11200 70000
2019 Ford Mustang 28998 18230
2012 Toyota Corolla 14349 43819
...
```

Reading vehicles from a file

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (ifs >> ) {
        inventory.push_back(auto_);
    }

}
```

Reading vehicles from a file

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (ifs >> ___) {
        inventory.push_back(auto_);
    }

}
```

Reading vehicles from a file

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (ifs >> _41_) {
        inventory.push_back(auto_);
    }

}
```

Which ***variable*** should replace blank #41 to assign the text up to the first space character to the `Vehicle`'s `year` member?

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (ifs >> _41_) {
        inventory.push_back(auto_);
    }

}
```

vehicles.txt

```
2007 Nissan Altima 11200 70000
2019 Ford Mustang 28998 18230
2012 Toyota Corolla 14349 43819
...
```

Reading vehicles from a file

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (ifs >> auto_.year) {
        inventory.push_back(auto_);
    }

}
```

Reading vehicles from a file

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (ifs >> auto_.year ___) {
        inventory.push_back(auto_);
    }

}
```

Which ***operator*** should replace blank #42 to extract tokens from the input file?

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (ifs >> auto_.year _42_) {
        inventory.push_back(auto_);
    }

}
```

vehicles.txt

```
2007 Nissan Altima 11200 70000
2019 Ford Mustang 28998 18230
2012 Toyota Corolla 14349 43819
...
```

Reading vehicles from a file

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (ifs >> auto_.year >> ) {
        inventory.push_back(auto_);
    }

}
```

Reading vehicles from a file

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (ifs >> auto_.year >> ___ ) {
        inventory.push_back(auto_);
    }

}
```

Reading vehicles from a file

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (ifs >> auto_.year >> _43_ ) {
        inventory.push_back(auto_);
    }

}
```

Which ***variable*** should replace blank #43 to assign the text from the first space character up to the second space character to the `Vehicle`'s `make` member?

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (ifs >> auto_.year >> _43_ ) {
        inventory.push_back(auto_);
    }

}
```

vehicles.txt

```
2007 Nissan Altima 11200 70000
2019 Ford Mustang 28998 18230
2012 Toyota Corolla 14349 43819
...
```

Reading vehicles from a file

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (ifs >> auto_.year >> auto_.make ) {
        inventory.push_back(auto_);
    }
}
```

Reading vehicles from a file

```
void read_inventory(ifstream& ifs, vector<Vehicle>& inventory) {
    Vehicle auto_;

    while (ifs >> auto_.year >> auto_.make >> auto_.model >> auto_.mileage >> auto_.price) {
        inventory.push_back(auto_);
    }
}
```

Full program source code available on NYU Brightspace

```
Welcome to CS 2124 Autos!

Provide a vehicle file
Filename: cars.txt
Failed to open vehicle file

Filename: vehicles.txt

Added 3 vehicles.
```