Class Documentation

Cargo Class Reference

This class is meant to represent a cargo container.
`#include <Cargo.h>`

Public Member Functions

- **Cargo** ()
- **Cargo** (string **label**, int **height**, int **width**, int **length**, double **weight**)
- int **getHeight** ()
- int **getWidth** ()
- int **getLength** ()
- int **getVolume** ()
- double **getWeight** ()
- string **getLabel** ()
- void **set** (string **label**, int **height**, int **width**, int **length**, double **weight**)
- void **print** ()
- **~Cargo** ()

Private Attributes

- string **label**
  The name/label of the cargo.
- int **height**
  The height of the cargo.
- int **width**
  The width of the cargo.
- int **length**
  The length of the cargo.
- double **weight**
  The weight of the cargo.

Detailed Description

This class is meant to represent a cargo container.

Constructor & Destructor Documentation

Cargo::Cargo ()

Default constructer: **Cargo()**.

This constructer will initialize the member varibles using a member initialization list. All member variables will be initialized to zero or empty string.

Cargo::Cargo (string *label*, int *height*, int *width*, int *length*, double *weight*)

Second constructer: **Cargo(string label, int height, int width, int length, double weight)**

This constructer will initialize the member varibles using a member initialization list. Variables will be initialized to the values that is passed in.

Parameters:

label	A string, The name/label of the cargo.
height	A integer, The height of the cargo.
width	A integer, The width of the cargo.
length	A integer, The length of the cargo.
weight	A double, The weight of the cargo.

Cargo::~Cargo ()

Function: **~Cargo()**

This Function does nothing.

Member Function Documentation

int Cargo::getHeight ()

Function: **getHeight()**

This Function will return the height of the cargo.

Returns:

int The height of the cargo.

string Cargo::getLabel ()

Function: **getLabel()**

This Function will return the Label of the cargo.

Returns:

string The label of the cargo

int Cargo::getLength ()

Function: **getLength()**

This Function will return the Length of the cargo.

Returns:

int The length of the cargo.

int Cargo::getVolume ()

Function: **getVolume()**

This Function will return the Volume of the cargo.

Returns:

int The Volume of the cargo, width * height * length.

double Cargo::getWeight ()

Function: **getWeight()**

This Function will return the Weight of the cargo.

Returns:

double The weight of the cargo

int Cargo::getWidth ()

Function: **getWidth()**

This Function will return the width of the cargo.

Returns:

int The width of the cargo.

void Cargo::print ()

Function: **print()**

This Function will print out the label, weight, volume and dimensions of the cargo. The number outputs will be set to left justified, standard numerical Weight output will be set to two decimal place.

Sets the output to left justified, standard numerical and two decimal place.

void Cargo::set (string *label*, int *height*, int *width*, int *length*, double *weight*)

Function: **set(string label, int height, int width, int length, double weight)**

This Function will set the label, height, width, length and weight varibles of the cargo class.

Parameters:

label	A string, The name/label of the cargo.
height	A integer, The height of the cargo.
width	A integer, The width of the cargo.
length	A integer, The length of the cargo.
weight	A double, The weight of the cargo.

Returns:

void

CargoPlane Class Reference

This class is meant to represent and simulate a cargo plane. Requires the cargo class.
`#include <CargoPlane.h>`

Public Member Functions

- **CargoPlane** ()
- **CargoPlane** (double **maxWeight**, int **maxVolume**, int **fuelCapacity**, int **fuelRate**, const string &city)
- int **loadCargo** (**Cargo** cargo)
- void **unLoadCargo** (string label)
- bool **fly** (string city, int hours, int miles)
- void **print** ()
- void **startCargoPlane_simulation** ()
- int **getRemainingFuel** ()
- int **getRemainingVolume** ()
- double **getRemainingWeight** ()
- **~CargoPlane** ()

Private Attributes

- string **currentCity**
  The current city the plane is at.
- int **milesFlown**
  The number miles the plane has flown.
- int **hoursFlown**
  The number of hours the plane has flown.
- int **fuelRate**
  The fuel consumption rate of the plane, in gallons per hour.
- int **fuelCapacity**
  The maximum amount of fuel the plane can hold, also the starting amnount of fuel of the plane.
- int **fuelConsumed**
  The amount of fuel the plane has used up so far.
- int **maxVolume**
  The maximum volume of cargos the plane can carry.
- int **usedVolume**
  The amount of volume that is used up so far.
- double **maxWeight**
  The maximum weight of cargos the plane can carry.
- double **usedWeight**
  The amount of weight that is used up so far.
- **Cargo** * **loadedCargo**
  *A pointer of type **Cargo**, points to a array of cargos, stores the cargos on the plane.*
- int **maxCargoCount**
  The maximum size of the cargo array, Is set to 200 by default in the construtor.

Detailed Description

This class is meant to represent and simulate a cargo plane. Requires the cargo class.

Constructor & Destructor Documentation

CargoPlane::CargoPlane ()

Funtion: **CargoPlane()**

This is the default constructor, will initialize all member variables of class **CargoPlane** to zero, NULL, empty string.

CargoPlane::CargoPlane (double *maxWeight*, int *maxVolume*, int *fuelCapacity*, int *fuelRate*, const string & *city*)

Funtion: **CargoPlane(double maxWeight, int maxVolume, int fuelCapacity, int fuelRate, const string& city)**;

This is the second constructor.

Parameters:

maxWeight	A double, The maximum weight the cargo plane can carry.
maxVolume	A integer, The maximum volume the plane can carry.
fuelCapacity	A integer, The maximum amount of fuel the plane can store, is also the starting amnount of fuel of the plane.
fuelRate	A integer, The fuel consumption rate of the plane, in gallons per hour.
city	A string, The current or starting city of the plane.

CargoPlane::~CargoPlane ()

Funtion: **~CargoPlane()**()

This is the destructor, will delete the array of cargo objects that the cargo pointer is pointing to, Then the cargo pointer will be set to NULL.

Member Function Documentation

bool CargoPlane::fly (string *city*, int *hours*, int *miles*)

Funtion: **fly(string city, int hours, int miles)**

This funtion will attempt to fly the plane to the city that is passed in.

Parameters:

city	The name of the city that the plane would fly to, the name can be multi-worded.
hours	The number of hours it would take to fly to the location.
miles	The number of miles it takes to fly to the location.

Returns:

bool Returns true if flight was successful, false otherwise.

int CargoPlane::getRemainingFuel ()

Funtion: **getRemainingFuel()**

Returns:

int Returns the remaining amount of fuel.

int CargoPlane::getRemainingVolume ()

Funtion: **getRemainingVolume()**

Returns:

int Returns the remaining amount of volume the plane can store.

double CargoPlane::getRemainingWeight ()

Funtion: **getRemainingWeight()**

Returns:

double Returns the remaining amount of weight the plane can carry.

int CargoPlane::loadCargo (Cargo *cargo*)

Funtion: **loadCargo(Cargo cargo)**

This funtion will attempt to load the cargo that is passed in, into the loadedCargo array. usedVolume and usedWeight are updated if loading was successful.

Parameters:

cargo	A cargo that will be loaded onto the plane.

Returns:

int Returns -1 if weight limit is exceeded, -2 is volume limit is exceeded, -3 if both are exceeded or returns 1 if loading was successful.

void CargoPlane::print ()

Funtion: **print()**

This funtion will print out the status, varaibles and the cargos that are loaded on the plane in a decent format.

void CargoPlane::startCargoPlane_simulation ()

Funtion: **startCargoPlane_simulation()**

This funtion controls the flow of the cargo plane simulation. Starts the cargo plane simulation.

void CargoPlane::unLoadCargo (string *label*)

Funtion: **unLoadCargo(string label)**

This funtion will attempt to unload a cargo with the label that is passed in. Will output a message if unloading is successful or if the unloading was unsuccessful. Variables usedWeight and usedVolume is updated if unloading is successful Unloads by setting all variables of the cargo to zero and the label to empty.

Parameters:

label	The name/label of the cargo that is to be unloaded.