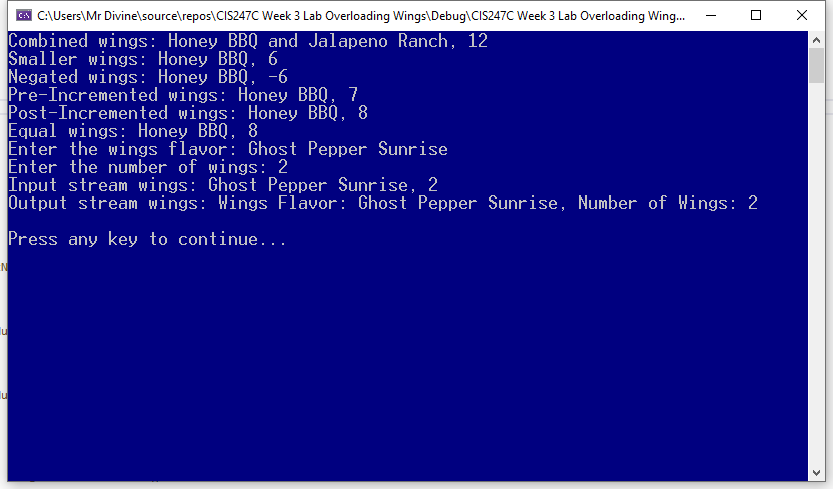
Screenshots:



**Code:**

**Source:**

/\*

Edward Alvarado

Class: CIS247C

Week 3 Lab

Date: 3/17/2021

\*/

#include <iostream>

#include <string>

#include <conio.h>

#include "Wings.h"

using namespace std;

/// Entry point to the applicatrion

int main()

{

// create two wings objects

Wings w1("Honey BBQ", 6);

Wings w2("Jalapeno Ranch", 6);

// test the overloaded operators

Wings w3 = w1 + w2;

cout << "Combined wings: " << w3.getFlavor() << ", " << w3.getNumberOfPieces() << endl;

Wings w4 = w3 - w2;

cout << "Smaller wings: " << w4.getFlavor() << ", " << w4.getNumberOfPieces() << endl;

Wings w5 = -w4;

cout << "Negated wings: " << w4.getFlavor() << ", " << w4.getNumberOfPieces() << endl;

w5 = -w5;

++w5;

cout << "Pre-Incremented wings: " << w5.getFlavor() << ", " << w5.getNumberOfPieces() << endl;

w5++;

cout << "Post-Incremented wings: " << w5.getFlavor() << ", " << w5.getNumberOfPieces() << endl;

Wings w6 = w5;

cout << "Equal wings: " << w6.getFlavor() << ", " << w6.getNumberOfPieces() << endl;

Wings w7;

cin >> w7;

cout << "Input stream wings: " << w7.getFlavor() << ", " << w7.getNumberOfPieces() << endl;

cout << "Output stream wings: " << w7 << endl;

// Pause

cout << "\nPress any key to continue...";

\_getch();

return 0;

}

Wings.h

#pragma once

#include <string>

#include <iostream>

using namespace std;

class Wings

{

private:

string flavor;

short numPieces;

public:

// contructors

Wings(string flavor="unknown", short numberofPieces = 0);

~Wings(void);

//behaviors

Wings operator+(const Wings secondWings) const;

Wings operator-(const Wings secondWings) const; // wings 3 = wings 1 - wings 2;

Wings operator-(void); // wings 2 = -wings1; (negative)

Wings& operator++(void); // ++wings;

Wings operator++(int); // wings++;

Wings& operator=(const Wings& equalWings); // Wings w2 = w1;

// overload the input stream operarator

// cin >> wings;

friend istream& operator>>(istream& is, Wings& obj);

//overload the output stream operator

// cout << wings << endl;

friend ostream& operator<<(ostream& os, Wings& obj);

// accessors and mutators

string getFlavor(void);

void setFlavor(string flavor);

short getNumberOfPieces(void);

void setNumberOfPieces(short numberOfPieces);

};

Wings.cpp

#include "Wings.h"

#include <iostream>

// contructors

Wings::Wings(string flavor, short numberofPieces)

{

setFlavor(flavor);

setNumberOfPieces(numberofPieces);

}

Wings::~Wings(void)

{

// no dynamic objects to destroy

}

//behaviors

Wings Wings::operator+(const Wings secondWings) const

{

//create the wings object to return

Wings thirdWings;

// tell c== how to combine the Wings attributes

thirdWings.numPieces = this->numPieces + secondWings.numPieces;

if (this->flavor != secondWings.flavor)

thirdWings.flavor = this->flavor + " and " + secondWings.flavor;

else

thirdWings.flavor = this->flavor;

// return the new, combined wings

return thirdWings;

}

Wings Wings::operator-(const Wings secondWings) const

{

//create the wings object to return

Wings thirdWings;

// tell c++ how to subtract the Wings attributes

thirdWings.numPieces = this->numPieces - secondWings.numPieces;

string full = this->flavor; // current flavor

short position = full.find("and", 0);

if (position > 0)

thirdWings.flavor = full.substr(0, position - 1);

else

thirdWings.flavor = this->flavor;

//return the new, changed Wings object

return thirdWings;

}

Wings Wings::operator-(void)

{

numPieces = -1 \* numPieces; // numPieces = -numPieces;

//flavor = ?? // typically dont negate strings

return \*this;

}

Wings& Wings::operator++(void)

{

++numPieces;

// return the incremented wings object

return \*this;

}

Wings Wings::operator++(int)

{

//save the old state (old values)

Wings temp = \*this; // save the old state

// tell c++ how to post-increment the current state

++(\*this);

//return the OLD state (old values)

return temp;

}

Wings& Wings::operator=(const Wings& equalWings)

{

if (this != &equalWings) //confirm that this NOT the same object

{

this->flavor = equalWings.flavor;

this->numPieces = equalWings.numPieces;

}

//return this new object

return \*this;

}

istream& operator>>(istream& is, Wings& obj)

{

// append to the input stream

cout << "Enter the wings flavor: ";

getline(is, obj.flavor);

cout << "Enter the number of wings: ";

is >> obj.numPieces;

// return the input stream so more can be appended

return is;

}

ostream& operator<<(ostream& os, Wings& obj)

{

//append our information to the current output stream

os << "Wings Flavor: " << obj.flavor << ", Number of Wings: " << obj.numPieces;

//return the output stream so more can be appended

return os;

}

// accessors and mutators

string Wings::getFlavor(void)

{

return flavor;

}

void Wings::setFlavor(string flavor)

{

if (flavor.length() > 0)

this->flavor = flavor;

else

this->flavor = "unknown";

}

short Wings::getNumberOfPieces(void)

{

return numPieces;

}

void Wings::setNumberOfPieces(short numberOfPieces)

{

if (numberOfPieces > 0)

numPieces = numberOfPieces;

else

numPieces = 0;

}