Basic Class Implementation, Part 2

By

Jorge Jurado-Garcia

EE2510 Sec. 021, Spring 2021

Week 3 lab

Milwaukee School of Engineering

Submitted to:

Professor: Joshua D. Carl, Ph.D

EECS Department

Date Report Submitted: 03/18/21

Objective

The objective of this lab is to create two classes, that have at least 3 constructor, 2 data members, and 2 methods. Furthermore, we will have to implement good data hiding principles by having public interface and private data as appropriate.

Description

In my project I created two classes one for milk and the other for hayfield. In my program we are given random values for some of our classes data members and given a Beloits WI linear demand and linear curve for that day. After Farmer Jorge and Carl count the amount of food and water they see if what price they should try to sell there milk at the price equilibrium point. They have to much milk then demanded then they will be told to produce and vice versa for less. If the price equilibrium point was not given from the “MILK DATA” then the program will give then a potential supply target to hit for maximum profitability.

Conclusions

The lab was successful and was able to implement all the required classes and their class definitions. The most difficult part for me for during this lab is how to implement the copy constructor and how they what affect how the data is used and causes to other data. For my copy constructor I implement a function that calls any object an a specific class and outputs the information wanted in the console.

Main code:

```
/*
  main.cpp

    Created on: Mar 24, 2021
        Author: Jorge Jurado-Garcia

    Create a program that demonstrates the complete functionality of both
classes.
    The program must create at least one object of each class, display the
values
    stored in the properties of each object, call all class methods on each
object,
    and display the results of the methods. Your program must also
demonstrate the
    use of the copy constructor. To do that, your program must contain at
least
    one function that has a class object as a parameter,
    or a function that returns a class object.

Extra Credit Opportunity:
To optionally earn extra credit, follow the instructions on Canvas to setup a
and install SFML, and link it to your Code
Blocks project (SFML stands for Simple and Fast
Multimedia Library - you can use it to create graphics).
 */
#include <iostream>
#include <fstream>
#include <sstream>
#include <random>
#include <math.h>
#include <string>
using namespace std;

#include "DairyCow.h"
#include "HayField.h"

float printCowData(DairyCow Cow);
int  printHayFieldData(HayField Hay);
int MarketAnalysis(int demand[], int supply[], int price,float milk);
//In my code ill use some economics so that the farmer
//knows how much he should make it depending on supply and demand curves
int main(){
    int bd,sloped,bs,slopes;
    //infor for classes
    float water, food, fieldsize, balesize;
    int balecount;
    while(1){
    int i;
    cin>>i;

    for(int j=1;j<3;j++){
    cout<<"Day"<<j<<endl;
    fieldsize = ((rand() %120+2)+0.0)/ (rand()%10+1);
    balesize = ((rand() %120+2)+0.0)/(rand()%8+1);
```

```
    balecount = rand() %10+1;
    float drymatter = 0.8;

    water = ((rand() %99+2)+0.0)/ (rand()%10+1);
    food = ((rand() %101+2)+0.0)/(rand()%8+1);
    int price_dairy = 0;

    int *demand = new int[i];
    int *supply = new int[i];
    //generates number from 1 and 10;
    bd = rand() %100+1;
    bs = -(rand() %5+1);
    sloped = -(rand() %5+1);
    slopes = rand() %7+1;

    HayField hay1;
    hay1.set_fieldsize(25.2);
    hay1.set_baleCount(10);
    hay1.set_balesize(5.9);
    hay1.set_drymatter(0.80);
    cout<<"Jorges "<<hay1.calculateYield()<<endl;
    cout<<"Jorge "<<hay1.calculatCarbonCapture()<<endl;

    HayField hay2(fieldsize,balecount,balesize,drymatter);
     printHayFieldData(hay2);
    cout<<"Carl "<<hay2.calculateYield()<<endl;
    cout<<"Carl "<<hay2.calculatCarbonCapture()<<endl;

    cout<<"PRICE FOR DAIRY IN BELOIT,WI TODAY"<<endl;
    for(int price=0; price<i;price++){

    float k = bd+sloped*(price);
    float kk = bs+slopes*(price);
    demand[price]=k;
    supply[price]=kk;
    cout<<"price: $"<<price<<"||"<<"demand: "<<demand[price]<<"||"<<"supply:
"<<supply[price]<<endl;

    if(k==kk){
        price_dairy = price;
        cout<<"price equilibrlum is: "<<price_dairy<<endl;
        }
    }
    price_dairy = ((bd-bs)/(slopes-sloped));
    cout<<"price equilibrium found: "<<price_dairy<<endl;


     cout<<"farmer Jorge"<<endl;
    DairyCow Jorge(food,water);
    float milk1 = printCowData(Jorge);
    if(price_dairy>i){
        int l = bs+slopes*(price_dairy);
        cout<<"cannot compute how much to produce must have more
data."<<endl;
         cout<<"potential supply:"<<l<<"units"<<endl;
    }
    else{
```

```
    MarketAnalysis(demand,supply,price_dairy,milk1);
    }

    cout<<"farmer Carl"<<endl;
    DairyCow Carl;
    Carl.set_food(rand()%10+1);
    Carl.set_water(rand()%10+1);
    float milk2 = printCowData(Carl);
     if(price_dairy>i){
        int l = bs+slopes*(price_dairy);
        cout<<"cannot compute how much to produce must have more
data."<<endl;
        cout<<"potential supply:"<<l<<"units"<<endl;
    }
    else{
    MarketAnalysis(demand,supply,price_dairy,milk2);
    }
    //now we will find farmer has excess supply or excess demand
    //and where or not they should create more milk or produce les
    delete[] demand;
    delete[] supply;
        }
    }

    return 0;
}

float printCowData(DairyCow Cow){
    cout<<Cow.get_water()<<"||"<<Cow.get_food()<<endl;
    cout<<"Dairy Produced: "<<Cow.calculateMilkProduced()<<"L"<<endl;
    cout<<"Waste Produced: "<<Cow.calculateWasteProduced()<<"Kg"<<endl;
    return Cow.calculateMilkProduced();
}
int MarketAnalysis(int demand[], int supply[], int price,float milk){
    if(supply[price] == milk){
        cout<<"the amount of milk produced is the right amount sell at:
$"<<price<<endl;
    }
    else if( supply[price] < milk){
        cout<<"produced too much milk!";
        cout<<"try to produce: "<<supply[price]<<"[L]"<<endl;
        cout<<"or reduce price to: $"<<price<<endl;
    }
    else{
        cout<<"produced not enough milk!";
        cout<<"try to produce: "<<supply[price]<<"[L]"<<endl;
        cout<<"or increase price to: $"<<price<<endl;
    }
    return 0;
}
int  printHayFieldData(HayField Hay){
    cout<<"field size: "<<Hay.get_fieldSize()<<"||"<<"bale size:
"<<Hay.get_balsize()<<endl;
    cout<<"bale count: "<<Hay.get_baleCount()<<"||"<<"dry matter:
"<<Hay.get_drymatter()<<endl;
    return 0;
}
```

Class Definitions

```
/*
 * RandB.cpp
 *
 *  Created on: Mar 17, 2021
 *      Author: Jorge Jurado-Garcia
 *
 * Defines the Rectangle and Box class
 *
 */

 #include "MyBox.h"
 #include "rectangle.h"

 #include <iostream>
 #include <string>
 using namespace std;
//this holds all the functions guts form the two classes that we are using
Rectangle::Rectangle(){
    //called when object is being created
    cout<<"***In default constructor****"<<endl;
    length = 0;
    width = 0;
}

int Rectangle::getArea(){
    cout<<"your Area:"<<endl;
    return length*width;
}
int Rectangle::getPerimeter(){
    cout<<"your Perimeter:"<<endl;
    return (2*length)+(2*width);
}
MyBox:: MyBox(){
     //called when object is being created
    cout<<"***In default constructor****"<<endl;
    length = 0;
    width = 0;
    height =0;
}
int MyBox::getSurfaceArea(){
    cout<<"your SurfaceArea:"<<endl;
    return (2*length*width) + (2*length*height) + (2*height*width);
}
int MyBox::getVolume(){
    cout<<"your Volume:"<<endl;
    return (length*height*width);
}
```

Class Declarations & Defintions

```
/*
 * DairyCow.h
 *
 *  Created on: Mar 24, 2021
 *      Author: Jorge Jurado-Garcia
 *
 *  Declares the cow class
 *  This is the official lab being done
 *
 *  March 24, 2021 Added copy constructor to accept arguments
 */

#ifndef DAIRYCOW_H_INCLUDED
#define DAIRYCOW_H_INCLUDED

#include <string>
#include <stdio.h>
using namespace std;

class DairyCow{
public:
    //constructor generic
    DairyCow();

    //parameter constructor
    DairyCow(float food, float water);

    //copy constructor
    DairyCow(const DairyCow &CowToCopy);

    //milk produced
    float calculateMilkProduced();

    //wasted products
    float calculateWasteProduced();

    float get_food();

    float get_water();

    set_food(float fd);

    set_water(float wr);

private:
    float food;
    float water;
};

#endif // DAIRYCOW_H_INCLUDED



/*
```

```
 * HayField.h
 *
 *  Created on: Mar 24, 2021
 *      Author: Jorge Jurado-Garcia
 *
 *  Declares the cow class
 *  This is the official lab being done
 *
 *  March 24, 2021 Added copy constructor to accept arguments
 */
#ifndef HAYFIELD_H_INCLUDED
#define HAYFIELD_H_INCLUDED

#include <string>
#include <stdio.h>
using namespace std;

class HayField{
public:
    HayField();
    HayField(float fieldsize, int balecount, float balesize, float
drymatter);
    HayField( const HayField &fieldToCopy);

    float calculateYield();
    float calculatCarbonCapture();

    float get_fieldSize();
    int get_baleCount();
    float get_balsize();
    float get_drymatter();

    set_fieldsize(float fldsize);
    set_baleCount(int baleCt);
    set_balesize(float balesz);
    set_drymatter(float drymter);
private:
    float fieldsize;
    int balecount;
    float balesize;
    float drymatter;


};

#endif // HAYFIELD_H_INCLUDED
```

```
/*
 * Cows and Fields.ccp
 *
 *  Created on: Mar 24, 2021
 *      Author: Jorge Jurado-Garcia
 *
 * Defines the functions in DairyCows and HayField
 *
 */

 #include "DairyCow.h"
 #include "HayField.h"

 #include <iostream>
 #include <stdexcept> //std::invalid argument
 #include <string>
 #include <cmath>
 using namespace std;
DairyCow::DairyCow(){
     //called when object is being created
    cout<<"***In default constructor****"<<endl;
    food = 0.0;
    water = 0.0;
}
DairyCow::DairyCow(float food, float water){
    cout<<"***In parameter constructor***"<<endl;
    set_food(food);
    set_water(water);
}
DairyCow::DairyCow(const DairyCow &CowToCopy){
    cout<<"**copy constructor***"<<endl;
    /*According to beginnerbooks.com the "this" pointer holds the address
    of current object.
    so for this problem the pointer for my variable length is equal
    to the dynamic integer of my object myRectangle. Which should have a
length and width
    Reference: https://beginnersbook.com/2017/08/cpp-this-pointer/
    */
    this->food = CowToCopy.food;
    this->water = CowToCopy.water;
}
DairyCow:: set_food(float fd){
    cout<<"***setting food***"<<endl;
    if(fd<=0){
        cout<<"food cannot be negative or zero"<<endl;
        food = 0;
    }
    food = fd;
}
DairyCow:: set_water(float wr){
    cout<<"***setting water***"<<endl;
    if(wr<=0){
        cout<<"water cannot be negative or zero"<<endl;
        wr = 0;
    }
    water = wr;
}
```

```
float DairyCow:: get_food(){
    cout<<"food: ";
    return food;
}
float DairyCow:: get_water(){
    cout<<"water: ";
    return water;
}
float DairyCow::calculateMilkProduced(){
    if(water!=food){
        if(water>food){
            return 0.75*(food*2);
        }
        else{
            return 0.75*(water*2);
        }
    }
    else{
        return 0.75*(water+food);
    }
}
float DairyCow:: calculateWasteProduced(){
    if(water == food){
        return 0.35*(0.25*(food + water));
    }
    else{
         if(water>food){
            return 0.35*( (0.25*(food+food))+(water-food));
        }
        else{
            return 0.35*( (0.25*(water+water))+(food-water));
        }
    }
}
HayField::HayField(){
    cout<<"***Generic Constructor***"<<endl;
    float fieldsize=0;
    int balecount=0;
    float balesize=0;
    float drymatter=0;
}
HayField::HayField(float fieldsize, int balecount, float balesize, float
drymatter){
    cout<<"***Parameter Constructor***"<<endl;
    set_fieldsize(fieldsize);
    set_baleCount(balecount);
    set_balesize(balesize);
    set_drymatter(drymatter);
}
HayField::HayField( const HayField &fieldToCopy){
 cout<<"**copy constructor***"<<endl;

    this->fieldsize = fieldToCopy.fieldsize;
    this->balecount = fieldToCopy.balecount;
    this->balesize = fieldToCopy.balesize;
    this->drymatter = fieldToCopy.drymatter;
}
```

```
float HayField::calculateYield(){
 float top;
    top = (balecount*balesize*drymatter);
    float bottom;
    bottom = fieldsize;
    cout<<"hay [kg/hectare]: ";
    return top/bottom;
}
float HayField::calculatCarbonCapture(){
    cout<<"carbon captured [kg/hectare]: ";
    return 16000/fieldsize;
}
float HayField::get_fieldSize(){
    return fieldsize;
}
int HayField::get_baleCount(){
    return balecount;
}
float HayField::get_balsize(){
    return balesize;
}
float HayField::get_drymatter(){
    return drymatter;
}
HayField::set_fieldsize(float fldsize){
  cout<<"***setting fieldsize***"<<endl;
    if(fldsize<=0){
        cout<<"size cannot be negative or zero"<<endl;
        fieldsize = 0;
    }
    fieldsize = fldsize;
}
HayField::set_baleCount(int baleCt){
cout<<"***setting bale count***"<<endl;
    if(baleCt<=0){
        cout<<"count cannot be negative or zero"<<endl;
        balecount = 0;
    }
    balecount = baleCt;
}
HayField::set_balesize(float balesz){
cout<<"***setting bale size***"<<endl;
    if(balesz<=0){
        cout<<"size cannot be negative or zero"<<endl;
        balesize = 0;
    }
    balesize = balesz;
}
HayField::set_drymatter(float drymter){
cout<<"***setting drymatter***"<<endl;
    if(drymter<=0){
        cout<<"dry matter cannot be negative or zero"<<endl;
        drymatter = 0;
    }
    drymatter = drymter;
}
```

Console Result:

```
20
Day1
***Generic Constructor***
***setting fieldsize***
***setting bale count***
***setting bale size***
***setting drymatter***
Jorges hay [kg/hectare]: 1.87302
Jorge carbon captured [kg/hectare]: 634.921
***Parameter Constructor***
***setting fieldsize***
***setting bale count***
***setting bale size***
***setting drymatter***
**copy constructor***
field size: 5.375||bale size: 19.2
bale count: 10||dry matter: 0.8
Carl hay [kg/hectare]: 28.5767
Carl carbon captured [kg/hectare]: 2976.74
PRICE FOR DAIRY IN BELOIT,WI TODAY
price: $0||demand: 65||supply: -1
price: $1||demand: 64||supply: 6
price: $2||demand: 63||supply: 13
price: $3||demand: 62||supply: 20
price: $4||demand: 61||supply: 27
price: $5||demand: 60||supply: 34
```

price: $6||demand: 59||supply: 41

price: $7||demand: 58||supply: 48

price: $8||demand: 57||supply: 55

price: $9||demand: 56||supply: 62

price: $10||demand: 55||supply: 69

price: $11||demand: 54||supply: 76

price: $12||demand: 53||supply: 83

price: $13||demand: 52||supply: 90

price: $14||demand: 51||supply: 97

price: $15||demand: 50||supply: 104

price: $16||demand: 49||supply: 111

price: $17||demand: 48||supply: 118

price: $18||demand: 47||supply: 125

price: $19||demand: 46||supply: 132

price equilibrium found: 8

farmer Jorge

In parameter constructor

setting food

setting water

copy constructor*

water: 9.33333||food: 23.3333

Dairy Produced: 14L

Waste Produced: 6.53333Kg

produced not enough milk!try to produce: 55[L]

or increase price to: $8

farmer Carl

In default constructor*

setting food

setting water

copy constructor*

water: 2||food: 8

Dairy Produced: 3L

Waste Produced: 2.45Kg

produced not enough milk!try to produce: 55[L]

or increase price to: $8

Day2

Generic Constructor

setting fieldsize

setting bale count

setting bale size

setting drymatter

Jorges hay [kg/hectare]: 1.87302

Jorge carbon captured [kg/hectare]: 634.921

Parameter Constructor

setting fieldsize

setting bale count

setting bale size

setting drymatter

copy constructor*

field size: 2.16667||bale size: 16

bale count: 7||dry matter: 0.8

Carl hay [kg/hectare]: 41.3538

Carl carbon captured [kg/hectare]: 7384.62

PRICE FOR DAIRY IN BELOIT,WI TODAY

price: $0||demand: 93||supply: -3

price: $1||demand: 91||supply: 3

price: $2||demand: 89||supply: 9

price: $3||demand: 87||supply: 15

price: $4||demand: 85||supply: 21

price: $5||demand: 83||supply: 27

price: $6||demand: 81||supply: 33

price: $7||demand: 79||supply: 39

price: $8||demand: 77||supply: 45

price: $9||demand: 75||supply: 51

price: $10||demand: 73||supply: 57

price: $11||demand: 71||supply: 63

price: $12||demand: 69||supply: 69

price equilibrlum is: 12

price: $13||demand: 67||supply: 75

price: $14||demand: 65||supply: 81

price: $15||demand: 63||supply: 87

price: $16||demand: 61||supply: 93

price: $17||demand: 59||supply: 99

price: $18||demand: 57||supply: 105

price: $19||demand: 55||supply: 111

price equilibrium found: 12

farmer Jorge

In parameter constructor

setting food

setting water

copy constructor*

water: 4||food: 33

Dairy Produced: 6L

Waste Produced: 10.85Kg

produced not enough milk!try to produce: 69[L]

or increase price to: $12

farmer Carl

In default constructor*

setting food

setting water

copy constructor*

water: 6||food: 9

Dairy Produced: 9L

Waste Produced: 2.1Kg

produced not enough milk!try to produce: 69[L]

or increase price to: $12