This is the very first workshop from oop345, and frankly speaking it is not that difficult yet to complete. However, the course materials are actually loaded and my prof covered a lot of new knowledges which I did not really know about.

1. The first thing I would like to mention is the internal and external linkage in my workshop coding part. I used the key word firstly declared a prototype like variables in the header file as g\_taxrate and g\_dailydiscount and then initialized it in the .cpp file. In this way, these two variables finally became to the global variables.

However, for the static variables, I realized that I can not declared in the header file then latter on use it in the same module. I had to put the static int counter in the foodorder.cpp.

Basically, the biggest difference is, internal linkage the variable name can be referred anywhere inside a module, so the value can be used or changed anywhere in that module. But external linkage means the variable could be referred anywhere throughout the program.

Code snippet required:

Foodorder.h:

// the two globals goes here//

#ifndef SDDS\_FOODORDERS\_H

#define SDDS\_FOODORDERS\_H

#include <iostream>

#include <string>

extern double g\_taxrate;

extern double g\_dailydiscount;

// declared inside one of the function, but static int would not go out of the scope, after the function display executed. And it can be reached anywhere in the foodorder module//

Foodorder.cpp:

void FoodOrder::display() const{

// static int counter used for display()//

static int counter = 0;

……

}

w1\_p2.cpp

// Rates change from day 1 to day 2

if (day == 1) {

g\_taxrate = 0.13;

g\_dailydiscount = 1.15;

}

else {

g\_taxrate = 0.14;

g\_dailydiscount = 1.20;

}

I can access these two globals anywhere, but the static variable counter I can only access from the foodorder module.

1. Key word static can be used to make the variable’s life last throughout the program.

void FoodOrder::display() const{

// static int counter used for display()//

static int counter = 0;

I declared the counter inside the function display as a static variable.

ostr << std::left << std::setw(2) << ++counter << ". ";

I used the cout to print out the variable and pre-increment at the same time. And the counter would not go out of the scope and gets declared & initialized again and again.

1. Let me briefly talk about this.

Firstly I added the rule of three as you can see from the code snippet below. And also, I added a deallocate for the deallocating the dynamic memory. So, in the read function, the memory can be deallocated first, then read the new data into the variable. Then in the header file again, I change the one of the member attribute to char\* m\_foodDesc{}. So it can be used for dynamic memory allocation.

Code snnipet:

In foodorder.h:

Public methods:

~FoodOrder();

FoodOrder(const FoodOrder& src);

FoodOrder& operator = (const FoodOrder& src);

Private methods:

void deallocate();

private member attribute:

char\* m\_foodDesc{};

Then in the .cpp file:

I added more dynamic memory allocation in line number in the assignment operator overloading and the read function. Also, I implemented destructor.

In this way, the memory leak is able to be avoided.

Code snippet:

In the foodorder.cpp:

// deallocate function//

void FoodOrder::deallocate(){

delete[]this->m\_foodDesc;

m\_foodDesc = nullptr;

}

//deallocate before read new data//

FoodOrder& FoodOrder::operator=(const FoodOrder& src){

if (this!=&src){

if (src.validateName()){

this->deallocate();

strcpy(this->m\_custName, src.m\_custName);

this->m\_foodDesc = new char[strlen(src.m\_foodDesc) + 1];

strcpy(this->m\_foodDesc, src.m\_foodDesc);

this->m\_dailySpecial = src.m\_dailySpecial;

this->m\_foodPrice = src.m\_foodPrice;

}

}

return \*this;

}

Further feedback, I think code snippet will be almost like a spam to my prof. Because I need to use cite my code to answer the questions, that total words will be like I wrote for this feedback almost 600 words. I am pretty sure my prof would have a very hard time to read through my whole reflect. I would say cite by using the line number and the file name will be more reasonable than this.