Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра ЕОМ



**Звіт**

з лабораторної роботи №4

з дисципліни: “Програмування, ч.2 (ООП)”

на тему: “ Класи та об’єкти”

Варіант 19

Виконав: ст.гр. КІ-15

Мармура В.І.

Прийняла: Викладач

Козак Н.Б.

Львів

2020

Мета: познайомитися із класами та об‘єктами.

Теоретичні відомості :

Основна відмінність будь-якої об‘єктно-орієнтованої мови програмування від інших не об‘єктно-орієнтованих мов програмування є можливість створення нових визначених користувачем типів, що називаються класами. Клас – це визначений користувачем тип з даними-елементами (властивостями) та функціями-елементами (методами), що являються членами класу. Він зазвичай описує певну абстракцію якоїсь сутності реального світу з її властивостями і можливими діями. Змінна типу клас називається об‘єктом. Об‘єкт – це вже не абстракція, а елемент реального світу, який може володіти певними характеристиками, які задаються властивостями в класі, та діяти згідно операцій заданих методами у класі. Оголошення класу в мові С++ має наступний синтаксис (не обов‘язково щоб клас мав всі секції чи спадкував базовий клас)

Оголошення класу містить оголошення даних-елементів (властивостей) та функцій-елементів (методів) класу. Одна з фундаментальних основ ООП передбачає інкапсуляцію даних, тобто дані мають бути недоступними ззовні, а 49 лише за посередництвом загальнодоступних методів класу. Оголошення методів має вигляд оголошення прототипу функції в середині однієї з секцій класу, зазвичай у секції public. Якщо методи є службовими і не мають бути доступні ззовні, тоді їх розміщують в секції private або protected. Сама ж реалізація методу може знаходитись як усередині класу (тоді оголошення методу в класі має вигляд оголошення функції з тілом), так і поза оголошенням класу (тоді у класі оголошується прототип функції, а її тіло визначається за межами класу). Але один з найфундаментальніших принципів розробки програмного забезпечення мовою С++ полягає у розмежуванні інтерфейсу класу від його реалізації. Тому при побудові програми мовою C++ кожне оголошення класу зазвичай розміщують у заголовочний файл \*.h назва якого співпадає з назвою класу, а реалізацію методів цього класу – у файл \*.cpp із тим ж іменем, що й \*.h файл. Заголовочні файли включаються (за допомогою директиви #іnclude) у кожен файл, у якому використовується клас, а файли з вихідними кодами компілюються і компонуються з файлом, що містить головну програму (main-функцію).

Завдання (Варіант 19) -

Клас CShip (Корабель). Клас зберігає назву корабля, кількість палива, кількість пасажирів, вартість корабля, вартість пробігу 1 милі, на яку зменшується вартість корабля. Конструктор має ініціалізувати ці властивості. Методи дозволяють модифікувати і читати властивості, здійснювати рух корабля зменшуючи при цьому запас палива і його вартість, садити і висаджувати пасажирів, виводити на екран поточний стан об‘єкта.

Код для вирішення задачі даного варіанту -

#include <iostream>

#include <String>

using namespace std;

void clear() {

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

class Ship {

public:

bool setShipName(string Name) {

sName = Name;

}

bool setShipPass(int passengers) {

sPass = passengers;

}

bool setShipFuel(int fuel) {

sFuel = fuel;

}

bool setShipValue(int value) {

sValue = value;

}

bool setShipMilesValue (int cost) {

sMileValue = cost;

}

bool setShipFuelPerMile(int fuel) {

sFuelPerMile = fuel;

}

string getShipName() {

return sName;

}

int getShipPass() {

return sPass;

}

int getShipFuel() {

return sFuel;

}

int getShipValue() {

return sValue;

}

int getShipMilesValue() {

return sMileValue;

}

int getShipFuelPerMile() {

return sFuelPerMile;

}

bool Move(int miles){

sValue -= sMileValue \* miles;

sMiles += miles;

sFuel -= sFuelPerMile \* miles;

if (sFuel > 0) {

return true;

}

else if (sFuel <= 0) {

cout << "Fuel tank empty, ship had stucked in ocean..." << endl;

return false;

}

}

void PortOut(int passengers) {

sPass -= passengers;

}

void PortIn(int passengers) {

sPass += passengers;

}

void State() {

clear();

cout << "Ship Name - " << sName << endl;

cout << "Ship Passengers - " << sPass << endl;

cout << "Ship Fuel - " << sFuel << endl;

cout << "Ship Value - " << sValue << endl;

cout << "Ship Miles - " << sMiles << endl;

clear();

}

Ship(string name, int passengers, int fuel, int value, int valuepermile , int fuelpermile) {

sName = name;

sPass = passengers;

sFuel = fuel;

sValue = value;

sMileValue = valuepermile;

sMiles = 0;

sFuelPerMile = fuelpermile;

}

private:

string sName;

int sPass, sFuel, sValue, sMileValue, sMiles, sFuelPerMile;

};

int main() {

string sName;

int sPassengers, sFuel, sValue, sMileValue, sFuelPerMile, sMilesSw, sPassIn, sPassOut;

cout << "Enter a ship's name - ";

cin >> sName;

cout << endl << "Enter a ship's passengers on board - ";

cin >> sPassengers;

cout << endl << "Enter a ship's fuel on board - ";

cin >> sFuel;

cout << endl << "Enter a ship's value - ";

cin >> sValue;

cout << endl << "Enter a ship's usage value per 1 mile - ";

cin >> sMileValue;

cout << endl << "Enter a ship's fuel usage per 1 mile - ";

cin >> sFuelPerMile;

Ship myShip(sName, sPassengers, sFuel, sValue, sMileValue, sFuelPerMile);

myShip.State();

cout << endl << "Enter a distance to the next stop - " << endl;

cin >> sMilesSw;

bool isSuccess1 = myShip.Move(sMilesSw);

if (isSuccess1==false) {

return 0;

}

cout << "OK, ship has arrived. Enter a number of passengers that are going off the board - " << endl;

cin >> sPassOut;

myShip.PortOut(sPassOut);

cout << "OK, enter a number of passengers that are going on the board - " << endl;

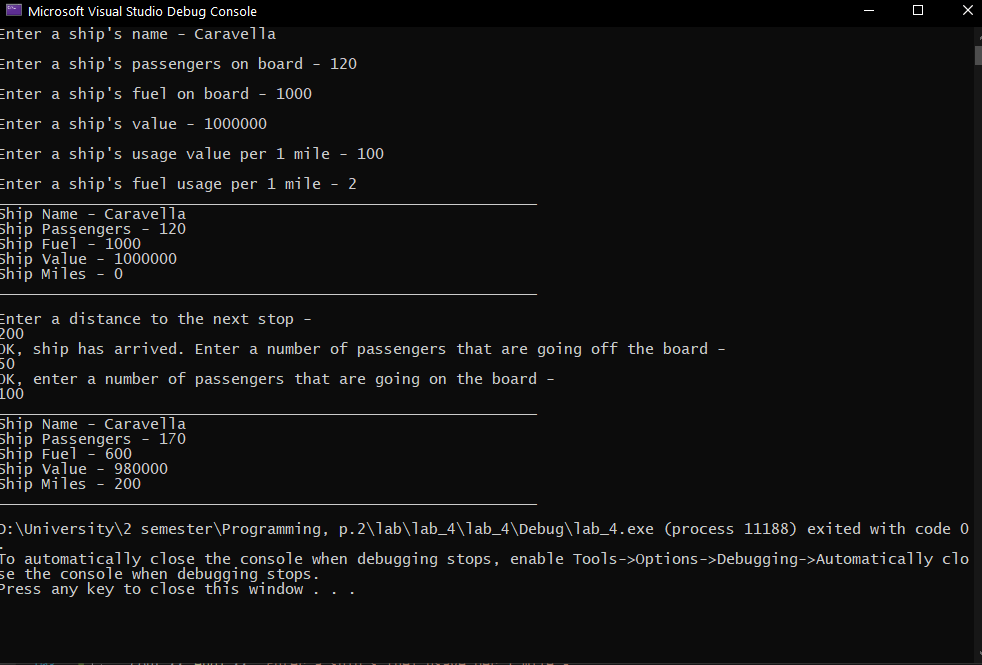
cin >> sPassIn;

myShip.PortIn(sPassIn);

myShip.State();

}

Скріншот виконання програми –



Висновок : в даній лабораторній роботі я опрацював роботу з классами, об’єктами классів, застосуванням їх і використанням геттерів і сеттерів.