**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ**

**КАФЕДРА КОМП’ЮТЕРНИХ НАУК**

**СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ**

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

**до курсової роботи**

з дисципліни

**«Основи об’єктно-орієнтованого програмування»**

Виконав: студент групи IT-91

Дєдовський Дмитро

Варіант: 22

Перевірив: Ващенко С. М.

2020

**ЗМІСТ**

[1 ЗАВДАННЯ 1 2](#_Toc59043005)

[1.1 Постановка задачі 3](#_Toc59043006)

[1.1 Опис класу користувача 3](#_Toc59043007)

[1.2 Тексти основних файлів 4](#_Toc59043008)

[2 ЗАВДАННЯ 2 7](#_Toc59043009)

[2.1 Постановка задачі 7](#_Toc59043010)

[2.2 Опис класу користувача 8](#_Toc59043011)

[2.3 Тексти основних файлів 10](#_Toc59043012)

[3 ГОЛОВНА ФУНКЦІЯ 16](#_Toc59043013)

[3.1 Код головної функції 17](#_Toc59043014)

[3.2 Контрольний приклад 17](#_Toc59043015)

1. **ЗАВДАННЯ 1**

**1.1 Постановка задачі**

1. Описати клас відповідно таблиці 1.1. Клас повинен бути максимально функціональним.

2. Створити масив із об’єктів описаного класу.

3. Виконати в масиві пошук інформації згідно варіанту.

Таблиця 1.1 – Завдання 1

|  |  |  |
| --- | --- | --- |
| Варіант | Клас | Пошук |
| 22 | Клас Butter(масло): назва, наявність рослинних добавок, ціна. | Визначити назви та середню ціну масла, які мають або не мають(за вибором користувача). |

* 1. **Опис класу користувача**

Для реалізації завдання потрібно ввести 3 змінні-члени класу.

Пояснення цих змінних наведено в таблиці 1.2

Таблиця 1.2 – Змінні-члени класу Butter

|  |  |  |
| --- | --- | --- |
| **Тип змінної** | **Позначення у програмі** | **Пояснення** |
| string | name | Назва масла |
| bool | dobavka | Наявність рослинних добавок |
| int | price | Ціна масла |

Визначимо геттери і сеттери до цих змінних. Геттер дає змогу швидко отримати значення конкретної змінної. Сеттер дає змогу змінити значення конкретної змінної. Дружня функція перевизначає операцію виведення. Також наявні звичайний і додатковий конструктори.

Детально про склад класу Butter наведено в таблиці 1.3

Таблиця 1.3 – Склад класу Butter

|  |  |
| --- | --- |
| **Метод** | **Пояснення** |
| Butter(); | Звичайний конструктор |
| Butter(string a, bool b, int c); | Додатковий конструктор |
| string get\_name(); | Геттер змінної name |
| bool get\_dobavka(); | Геттер змінної dobavka |
| int get\_price(); | Геттер змінної price |
| void set\_name(string a); | Сеттер змінної name |
| void set\_dobavka(bool b); | Сеттер змінної dobavka |
| void set\_price(int c); | Сеттер змінної price |
| friend ostream& operator<< (ostream& out, const Butter& butter); | Дружня функція |

* 1. **Тексти основних файлів**

Butter.h

#pragma once

#include <iostream>

#include <string>

using namespace std;

class Butter

{

string name;

bool dobavka;

int price;

public:

Butter();

Butter(string a, bool b, int c);

string get\_name();

bool get\_dobavka();

int get\_price();

void set\_name(string a);

void set\_dobavka(bool b);

void set\_price(int c);

friend ostream& operator<< (ostream& out, const Butter& butter) {

out << butter.name << " " << butter.dobavka << " " << butter.price << " ";

return out;

}

};

Butter.cpp

#include "Butter.h"

#include "zav1.h"

#include <iostream>

using namespace std;

Butter::Butter() {

name = "";

dobavka = 1;

price = 0;

}

Butter::Butter(string a, bool b, int c) {

name = a;

dobavka = b;

price = c;

}

int Butter::get\_price() {

return price;

}

bool Butter::get\_dobavka() {

return dobavka;

}

string Butter::get\_name() {

return name;

}

void Butter::set\_name(string a) {

name = a;

}

void Butter::set\_dobavka(bool b) {

dobavka = b;

}

void Butter::set\_price(int c) {

price = c;

}

zav1.h

#pragma once

#include "Butter.h"

double Print\_Res\_zav1();

zav1.cpp

#include "zav1.h"

#include "Butter.h"

#include <iostream>

#include <vector>

#include <fstream>

using namespace std;

double Print\_Res\_zav1() {

int size;

cout << "Size -> ";

cin >> size;

while (!cin.good() || cin.fail()) {

cout << "Size -> ";

cin.clear();

cin.ignore(32767, '\n');

cin >> size;

}

Butter\* arr = new Butter[size];

for (int i = 0; i < size; i++) {

string name;

bool dobavka;

int price;

cout << "Name oil -> ";

cin.ignore(32767, '\n');

getline(cin, name);

cout << "Have additions(1 - yes, 0 - no) -> ";

cin >> dobavka;

while (!cin.good() || cin.fail()) {

cout << "Have additions(1 - yes, 0 - no) -> ";

cin.clear();

cin.ignore(32767, '\n');

cin >> dobavka;

}

cout << "Price -> ";

cin >> price;

while (!cin.good() || cin.fail()) {

cout << "Price -> ";

cin.clear();

cin.ignore(32767, '\n');

cin >> price;

}

arr[i] = Butter(name, dobavka, price);

}

bool SearchNumber;

cout << "1. Oil, which have additions." << endl << "2. Oil, which do not have additions." << endl << "You choose -> ";

int ChooseNumber;

cin >> ChooseNumber;

while (!cin.good() || cin.fail() || (ChooseNumber != 1 && ChooseNumber != 2)) {

cout << "1. Oil, which have additions." << endl << "2. Oil, which do not have additions." << endl << "You choose -> ";

cin.clear();

cin.ignore(32767, '\n');

cin >> ChooseNumber;

}

if (ChooseNumber == 1) {

SearchNumber = 1;

}

else {

SearchNumber = 0;

}

if (ChooseNumber) {

cout << "A list of oils that have an additions: " << endl;

}

else {

cout << "A list of oils that do not have an additions: " << endl;

}

int sum = 0;

int amount = 0;

for (int i = 0; i < size; i++) {

if (SearchNumber) {

if (arr[i].get\_dobavka() == 1) {

cout << amount + 1 << ". " << arr[i].get\_name() << endl;

amount++;

sum = sum + arr[i].get\_price();

}

}

else {

if (arr[i].get\_dobavka() == 0) {

cout << amount + 1 << ". " << arr[i].get\_name() << endl;

amount++;

sum = sum + arr[i].get\_price();

}

}

}

double res;

res = (double)sum / amount;

ofstream myFile;

myFile.open("zav1.txt");

if (myFile.is\_open()) {

myFile << res << endl;

cout << "Successfully recorded in file called zav1.txt" << endl;

}

myFile.close();

return res;

}

1. **ЗАВДАННЯ 2**
   1. **Постановка задачі**

Описати клас. Внести в список атрибутів колір. В класі передбачити перевизначення зазначеної операції. Продемонструвати роботу перевизначеної операції. Продемонструвати роботу класу. Завдання знаходиться в таблиці 2.1

Таблиця 2.1 – Завдання 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Варіант** | **Клас** | **Суть операції** | **Знак** |
| 22 | Клас описує пряму лінію, положення якої задається координатами початку і кінця лінії. | Операція полягає у формуванні лінії, координати конечних точок якої визначаються як середні арифметичні координат кінцевих точок ліній-операндів. | >>= |

**2.2** **Опис класу користувача**

Для реалізації класу потрібно створити 5 змінних-членів класу.

Пояснення цих змінних наведено в таблиці 2.2

Таблиця 2.2 – Опис класу Line

|  |  |  |
| --- | --- | --- |
| **Тип змінної** | **Позначення у програмі** | **Пояснення** |
| double | x1 | Значення координати початку по вісі х |
| double | x2 | Значення координати кінця по вісі х |
| double | y1 | Значення координати початку по вісі у |
| double | y2 | Значення координати кінця по вісі у |
| string | color | Колір |

Визначимо геттери і сеттери до цих змінних. Геттер дає змогу швидко отримати значення конкретної змінної. Сеттер дає змогу змінити значення конкретної змінної. Дружня функція перевизначає операцію виведення. В класі наявне перевизначення операції,яке вказано в завданні. Також наявні звичайний і додатковий конструктори.

Детально про склад класу Line наведено в таблиці 2.3

Таблиця 2.3 – Склад класу Line

|  |  |
| --- | --- |
| **Метод** | **Пояснення** |
| Line(); | Звичайний конструктор |
| Line(double val\_x1, double val\_x2, double val\_y1, double val\_y2); | Додатковий конструктор |
| double get\_x1(); | Геттер змінної x1 |
| void set\_x1(double val\_x1); | Сеттер змінної x1 |
| double get\_x2(); | Геттер змінної x2 |
| void set\_x2(double val\_x2); | Сеттер змінної x2 |
| double get\_y1(); | Геттер змінної y1 |
| void set\_y1(double val\_y1); | Сеттер змінної y1 |
| double get\_y2(); | Геттер змінної y2 |
| void set\_y2(double val\_y2); | Сеттер змінної y2 |
| string get\_color(); | Геттер змінної color |
| void set\_color(string val\_color); | Сеттер змінної color |
| Line operator>>=(Line op2); | Метод, який перевизначає операцію вказану в завданні |
| void show(); | Метод, який виводить інформацію про координати |
| friend ostream& operator<< (ostream& out, const Line& line); | Дружня функція |

* 1. **Тексти основних файлів**

Line.h

#pragma once

#include <iostream>

#include <string>

using namespace std;

class Line {

double x1;

double x2;

double y1;

double y2;

string color;

public:

Line();

Line(double val\_x1, double val\_x2, double val\_y1, double val\_y2);

double get\_x1();

void set\_x1(double val\_x1);

double get\_x2();

void set\_x2(double val\_x2);

double get\_y1();

void set\_y1(double val\_y1);

double get\_y2();

void set\_y2(double val\_y2);

string get\_color();

void set\_color(string val\_color);

Line operator>>=(Line op2);

void show();

friend ostream& operator<< (ostream& out, const Line& line) {

out << "Koordinats new line: " << endl << "Koordinat x1 -> " << line.x1 << endl << "Koordinat x2 -> " << line.x2 << endl << "Koordinat y1 -> " << line.y1 << endl << "Koordinat y2 -> " << line.y2 << endl;

return out;

}

};

Line.cpp

#include "Line.h"

#include "zav2.h"

#include <iostream>

using namespace std;

Line::Line() {

x1 = x2 = y1 = y2 = 0;

color = "";

}

Line::Line(double val\_x1, double val\_x2, double val\_y1, double val\_y2) {

x1 = val\_x1;

x2 = val\_x2;

y1 = val\_y1;

y2 = val\_y2;

}

double Line::get\_x1() {

return x1;

}

double Line::get\_x2() {

return x2;

}

double Line::get\_y1() {

return y1;

}

double Line::get\_y2() {

return y2;

}

string Line::get\_color() {

return color;

}

void Line::set\_x1(double val\_x1) {

x1 = val\_x1;

}

void Line::set\_x2(double val\_x2) {

x2 = val\_x2;

}

void Line::set\_y1(double val\_y1) {

y1 = val\_y1;

}

void Line::set\_y2(double val\_y2) {

y2 = val\_y2;

}

void Line::set\_color(string val\_color) {

color = val\_color;

}

Line Line::operator>>=(Line op2) {

Line temp;

temp.x1 = (x1 + op2.x1) / 2.0;

temp.x2 = (x2 + op2.x2) / 2.0;

temp.y1 = (y1 + op2.y1) / 2.0;

temp.y2 = (y2 + op2.y2) / 2.0;

return temp;

}

void Line::show() {

cout << "Koordinats new line: " << endl;

cout << "Koordinat x1 -> " << x1 << endl;

cout << "Koordinat x2 -> " << x2 << endl;

cout << "Koordinat y1 -> " << y1 << endl;

cout << "Koordinat y2 -> " << y2 << endl;

}

zav2.h

#pragma once

#include "Line.h"

Line Print\_Res\_zav2();

zav2.cpp

#include "zav2.h"

#include "Line.h"

#include <iostream>

#include <fstream>

using namespace std;

Line Print\_Res\_zav2() {

double x1, x2, y1, y2;

cout << "We have two lines, enter koordinats" << endl;

cout << "First line, koordinat x1 -> ";

cin >> x1;

while (!cin.good() || cin.fail()) {

cout << "First line, koordinat x1 -> ";

cin.clear();

cin.ignore(32767, '\n');

cin >> x1;

}

cout << "First line, koordinat x2 -> ";

cin >> x2;

while (!cin.good() || cin.fail()) {

cout << "First line, koordinat x2 -> ";

cin.clear();

cin.ignore(32767, '\n');

cin >> x2;

}

cout << "First line, koordinat y1 -> ";

cin >> y1;

while (!cin.good() || cin.fail()) {

cout << "First line, koordinat y1 -> ";

cin.clear();

cin.ignore(32767, '\n');

cin >> y1;

}

cout << "First line, koordinat y2 -> ";

cin >> y2;

while (!cin.good() || cin.fail()) {

cout << "First line, koordinat y2 -> ";

cin.clear();

cin.ignore(32767, '\n');

cin >> y2;

}

Line first(x1, x2, y1, y2);

cout << "First line, koordinat x1 -> ";

cin >> x1;

while (!cin.good() || cin.fail()) {

cout << "First line, koordinat x1 -> ";

cin.clear();

cin.ignore(32767, '\n');

cin >> x1;

}

cout << "First line, koordinat x2 -> ";

cin >> x2;

while (!cin.good() || cin.fail()) {

cout << "First line, koordinat x2 -> ";

cin.clear();

cin.ignore(32767, '\n');

cin >> x2;

}

cout << "First line, koordinat y1 -> ";

cin >> y1;

while (!cin.good() || cin.fail()) {

cout << "First line, koordinat y1 -> ";

cin.clear();

cin.ignore(32767, '\n');

cin >> y1;

}

cout << "First line, koordinat y2 -> ";

cin >> y2;

while (!cin.good() || cin.fail()) {

cout << "First line, koordinat y2 -> ";

cin.clear();

cin.ignore(32767, '\n');

cin >> y2;

}

cout << endl;

Line second(x1, x2, y1, y2);

Line test;

test = first >>= second;

ofstream myFile;

myFile.open("zav2.txt");

if (myFile.is\_open()) {

myFile << test << endl;

cout << "Succesfully recorded in file called zav2.txt" << endl;

}

myFile.close();

return test;

}

1. **ГОЛОВНА ФУНКЦІЯ**

**3.1 Код головної функції**

#include <iostream>

#include "Butter.h"

#include "zav1.h"

#include "Line.h"

#include "zav2.h"

using namespace std;

int main(){

int ChooseTask;

cout << "The developer of the course work on object-oriented programming is Dedovskiy Dmytro, a student of group IT-91" << endl;

cout << "In the coursework we have 2 tasks.\nIf you want open first task - enter number 1.\nIf you want open second task - enter number 2.\nYou choose ->";

cin >> ChooseTask;

while (!cin.good() || cin.fail() || (ChooseTask != 1 && ChooseTask != 2)) {

cout << "In the coursework we have 2 tasks.\nIf you want open first task - enter number 1.\nIf you want open second task - enter number 2.\nYou choose ->";

cin.clear();

cin.ignore(32767, '\n');

cin >> ChooseTask;

}

system("cls");

if (ChooseTask == 1) {

cout << "\nResult First Task -> " << Print\_Res\_zav1() << endl;

}

else {

cout << "\nResult Second Task \n" << Print\_Res\_zav2() << endl;

}

system("pause");

return 0;

}

**3.2** **Контрольний приклад**

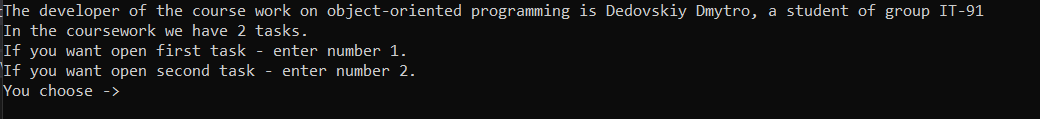


Рисунок 3.1 - Вибір виконання завдання

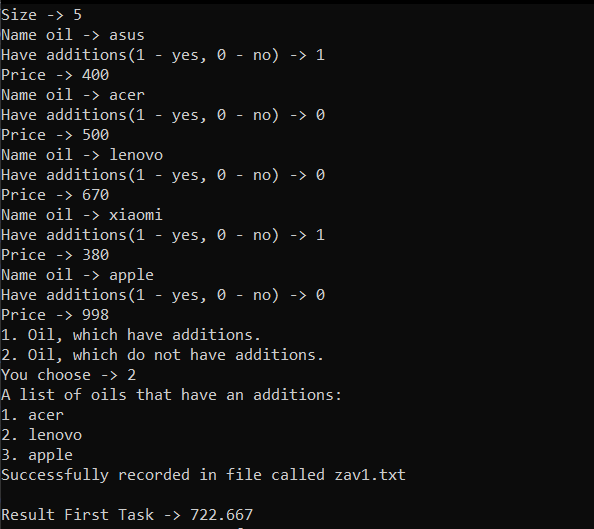


Рисунок 3.2 - Робота завдання 1

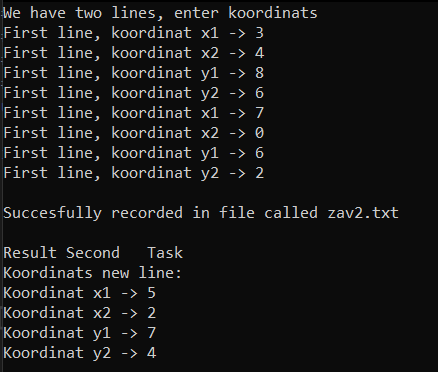


Рисунок 3.3 - Робота завдання 2

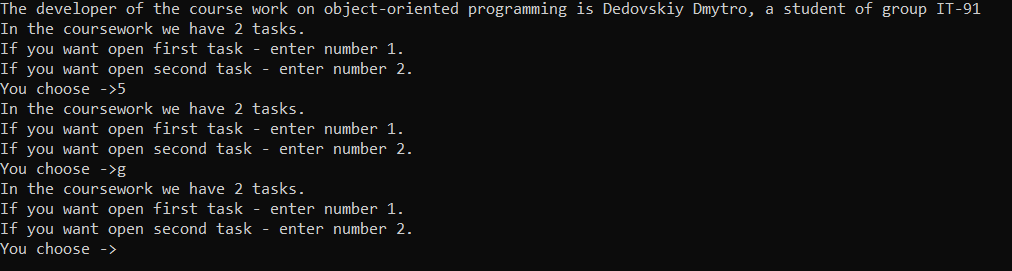


Рисунок 3.4 - Ситуація неправильного вибору номера завдання

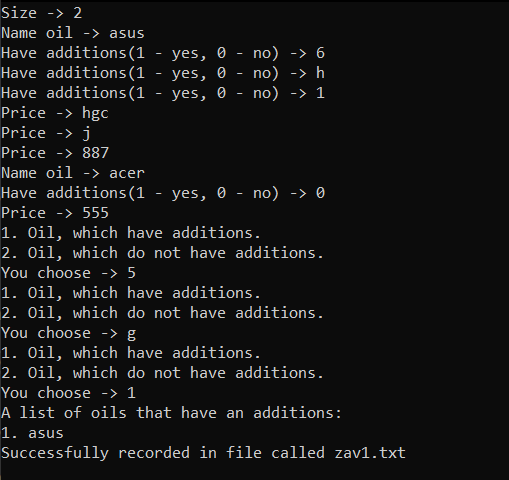


Рисунок 3.5 - Ситуація неправильного введення даних в завданні 1

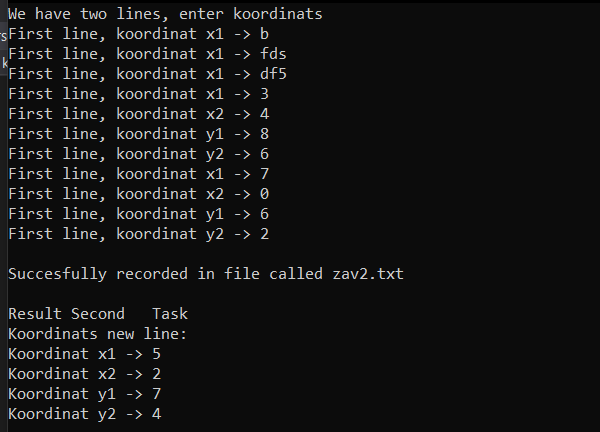


Рисунок 3.6 - Ситуація неправильного введення даних в завданні 2

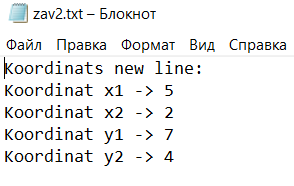


Рисунок 3.7 - Запис у файл завдання 2

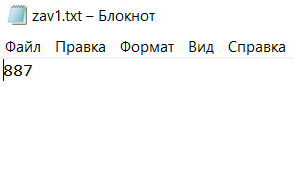


Рисунок 3.8 - Запис у файл завдання 1