

Екзаменаційне завдання

**ООП з використанням мови C++**

**Версія 1.0.1**

Практична частина іспиту складається з двох завдань. Студент має вибрати і виконати два завдання із запропонованих нижче.

При виконанні завдань необхідно в повній мірі використовувати апарат ООП та STL.

Код має бути читабельним, добре прокоментованим, і необхідно використовувати єди- ну систему іменування ідентифікаторів.

# Перше завдання

## Реалізувати найпростіший файловий менеджер з використанням ООП (класи, спадкування тощо).

**Файловий менеджер мусить мати такі можливості:**

* показувати вміст дисків;
* створювати папки/файли;
* видаляти папки/файли;
* перейменовувати папки/файли;
* копіювати/переносити папки/файли;
* обчислювати розмір папки/файлу;
* виконувати пошук по масці (з пошуком у підпапках) і так далі.

# Друге завдання

## Реалізувати повнофункціональну систему тестування.

**У системі має бути реалізовано 2 режими:**

* Адміністратор;
* Тестований.

## Опис режиму робота для Тестованого (надалі гість)

Для входу в систему гість мусить зареєструватися. Ця процедура виконується один раз, при подальших входах в систему доступ надається за логіном і паролем.

При реєстрації потрібно вказувати ПІБ, домашню адресу, телефон. Важливо, щоб логіни для користувачів були унікальними.

Після входу гість має змогу переглянути свої попередні результати тестування, пройти нове тестування.

Тестування може бути за різними категоріями знань. Наприклад:

Математика (розділ) –> Дискретна математика (конкретний тест);

–> Математичний аналіз (конкретний тест).

Фізика (розділ) –> Квантова фізика (конкретний тест);

–> Механіка (конкретний тест).

Після здачі тесту гість бачить результат тестування, кількість питань, на які він від- повів правильно, відсоток правильних відповідей і отриману оцінку.

Студент має змогу зупинити тестування і продовжити його тоді, коли йому це буде зручно.

Оцінювання має будуватися на 12-бальній системі, що прив’язана до кількості пи- тань тесту.

Паролі та логіни гостей зберігаються в зашифрованому вигляді.

## Опис режиму робота для Адміністратора (надалі адмін)

У системі може бути тільки один адмін, логін і пароль адміна задається при першо- му вході в програму.

Надалі пароль і логін можна змінити (але цю змогу має тільки адмін). Пароль і логін необхідно зберігати тільки в зашифрованому вигляді. При роботі з системою адмін має такі можливості:

* Управління користувачами – створення, видалення, модифікація користувачів.
* Перегляд статистики – перегляд результатів тестування загалом за катего- ріями, за конкретними тестами, за конкретними користувачами. Результати перегляду статистики можна вивести у файл.
* Управління тестуванням – адмін має змогу додавати категорії, тести, питання до тестів, задавати правильні й неправильні відповіді, імпортувати й експорту- вати категорії і тести з питаннями з файлу (та у файл).

# Третє завдання

## Створіть систему управління персональними фінансами.

**Система мусить мати такі можливості:**

* Наявність різних гаманців і карт (дебетових/кредитних);
* Поповнення гаманців і карт;
* Внесення витрат. Кожна витрата належить певній категорії;
* Формування звітів за витратами та категоріями:
  + день;
  + тиждень;
  + місяць.
* Формування рейтингів за максимальними сумами:
  + ТОП-3 витрат:
    - тиждень;
    - місяць.
  + ТОП-3 категорій:
    - тиждень;
    - місяць.
* Збереження звітів і рейтингів у файл.

# Четверте завдання

## Створіть консольну версію гри «Шибениця».

Правила гри можна переглянути за [**цим посиланням**](https://uk.wikipedia.org/wiki/Шибениця_(гра)).

## Важливо:

* Слово вибирається комп’ютером зі списку слів.
* Список слів є у файлі в зашифрованому вигляді.
* Після завершення гри на екран виводиться статистика гри:
  + кількість часу;
  + кількість спроб;
  + шукане слово;
  + літери гравця.

Завдання 3

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <Windows.h>

#include <string>

#include <vector>

#include <fstream>

#include <algorithm>

using namespace std;

class Cost

{

string name;

string category;

float money;

int day;

int mis;

int year;

public:

void PrintInfo()

{

cout << "------------------------------------------\n";

cout << name << endl;

cout << category << endl;

cout << money << "грн"<<endl;

cout << day << "." << mis << "." << year << endl;

cout << "------------------------------------------\n";

}

string Get\_Name()

{

return name;

}

string Get\_Category()

{

return category;

}

float& Get\_Money()

{

return money;

}

int Get\_Day()

{

return day;

}

int Get\_Mis()

{

return mis;

}

int Get\_Year()

{

return year;

}

Cost()

{

}

Cost(string name, string category, float money, int day, int mis, int year)

{

this->name = name;

this->category = category;

this->money = money;

this->day = day;

this->mis = mis;

this->year = year;

}

void AddCost(string name, string category, float money, int day, int mis, int year)

{

this->name = name;

this->category = category;

this->money = money;

this->day = day;

this->mis = mis;

this->year = year;

}

};

class SortByPrice

{

public:

bool operator()(Cost\* left, Cost\* right)

{

return left->Get\_Money() > right->Get\_Money();

}

};

class Card

{

string name;

string account;

float balance;

vector<Cost\*> costs;

public:

float& Get\_Balance()

{

return balance;

}

Card(string name, string account, float balance)

{

this->name = name;

this->account = account;

this->balance = balance;

}

~Card()

{

for (auto it = costs.begin(); it != costs.end(); it++)

{

delete \*it;

}

}

void PrintInfo()

{

cout << name << " рахунок: " << account;

}

void SetCard(string name, string account, float balance)

{

this->name = name;

this->account = account;

this->balance = balance;

}

void AddCost(Cost\* cost)

{

this->costs.push\_back(cost);

}

void ShowZvitDay(int day, int mis, int year)

{

string txt;

txt += "Витрати за конкретний день\n";

txt += "---------------------------------------\n";

int k;

for (int i = 0; i < costs.size(); i++)

{

if (costs[i]->Get\_Day() == day && costs[i]->Get\_Mis() == mis && costs[i]->Get\_Year() == year)

{

costs[i]->PrintInfo();

txt += costs[i]->Get\_Name()+"\n";

txt += costs[i]->Get\_Category()+"\n";

txt += to\_string(costs[i]->Get\_Money())+"грн\n";

txt += to\_string(costs[i]->Get\_Day()) + ".";

txt += to\_string(costs[i]->Get\_Mis()) + ".";

txt += to\_string(costs[i]->Get\_Year()) + "\n";

txt += "---------------------------------------\n";

}

}

txt += "---------------------------------------\n";

cout << "Зберегти у файл?(1-Так, 0-ні)->";

cin >> k;

if (k == 1)

{

fstream f("data.txt", ios::app|ios::out);

f << txt;

f.close();

}

}

void ShowZvitWeek(int day, int mis, int year)

{

string txt;

txt += "Витрати за тиждень\n";

txt += "---------------------------------------\n";

int k;

for (int i = 0; i < costs.size(); i++)

{

if ( costs[i]->Get\_Day()>= day && costs[i]->Get\_Day()<=day+7 && costs[i]->Get\_Mis() == mis && costs[i]->Get\_Year() == year)

{

costs[i]->PrintInfo();

txt += costs[i]->Get\_Name() + "\n";

txt += costs[i]->Get\_Category() + "\n";

txt += to\_string(costs[i]->Get\_Money()) + "грн\n";

txt += to\_string(costs[i]->Get\_Day()) + ".";

txt += to\_string(costs[i]->Get\_Mis()) + ".";

txt += to\_string(costs[i]->Get\_Year()) + "\n";

txt += "---------------------------------------\n";

}

}

txt += "---------------------------------------\n";

cout << "Зберегти у файл?(1-Так, 0-ні)->";

cin >> k;

if (k == 1)

{

fstream f("data.txt", ios::app | ios::out);

f << txt;

f.close();

}

}

void ShowZvitMonth(int mis, int year)

{

string txt;

txt += "Витрати за місяць\n";

txt += "---------------------------------------\n";

int k;

for (int i = 0; i < costs.size(); i++)

{

if (costs[i]->Get\_Mis() == mis && costs[i]->Get\_Year() == year)

{

costs[i]->PrintInfo();

txt += costs[i]->Get\_Name() + "\n";

txt += costs[i]->Get\_Category() + "\n";

txt += to\_string(costs[i]->Get\_Money()) + "грн\n";

txt += to\_string(costs[i]->Get\_Day()) + ".";

txt += to\_string(costs[i]->Get\_Mis()) + ".";

txt += to\_string(costs[i]->Get\_Year()) + "\n";

txt += "---------------------------------------\n";

}

}

txt += "---------------------------------------\n";

cout << "Зберегти у файл?(1-Так, 0-ні)->";

cin >> k;

if (k == 1)

{

fstream f("data.txt", ios::app | ios::out);

f << txt;

f.close();

}

}

void ShowTop3Week(int day, int mis, int year)

{

sort(this->costs.begin(), this->costs.end(), SortByPrice());

string txt;

txt += "Топ-3 витрати за тиждень\n";

txt += "---------------------------------------\n";

int k;

int count = 0;

for (int i = 0; i < costs.size(); i++)

{

if (costs[i]->Get\_Day() >= day && costs[i]->Get\_Day() <= day + 7 && costs[i]->Get\_Mis() == mis && costs[i]->Get\_Year() == year)

{

costs[i]->PrintInfo();

txt += costs[i]->Get\_Name() + "\n";

txt += costs[i]->Get\_Category() + "\n";

txt += to\_string(costs[i]->Get\_Money()) + "грн\n";

txt += to\_string(costs[i]->Get\_Day()) + ".";

txt += to\_string(costs[i]->Get\_Mis()) + ".";

txt += to\_string(costs[i]->Get\_Year()) + "\n";

txt += "---------------------------------------\n";

count++;

if (count == 3)

break;

}

}

txt += "---------------------------------------\n";

cout << "Зберегти у файл?(1-Так, 0-ні)->";

cin >> k;

if (k == 1)

{

fstream f("data.txt", ios::app | ios::out);

f << txt;

f.close();

}

}

void ShowTop3Month( int mis, int year)

{

sort(this->costs.begin(), this->costs.end(), SortByPrice());

string txt;

txt += "Топ-3 витрати за місяць\n";

txt += "---------------------------------------\n";

int k;

int count = 0;

for (int i = 0; i < costs.size(); i++)

{

if (costs[i]->Get\_Mis() == mis && costs[i]->Get\_Year() == year)

{

costs[i]->PrintInfo();

txt += costs[i]->Get\_Name() + "\n";

txt += costs[i]->Get\_Category() + "\n";

txt += to\_string(costs[i]->Get\_Money()) + "грн\n";

txt += to\_string(costs[i]->Get\_Day()) + ".";

txt += to\_string(costs[i]->Get\_Mis()) + ".";

txt += to\_string(costs[i]->Get\_Year()) + "\n";

txt += "---------------------------------------\n";

count++;

if (count == 3)

break;

}

}

txt += "---------------------------------------\n";

cout << "Зберегти у файл?(1-Так, 0-ні)->";

cin >> k;

if (k == 1)

{

fstream f("data.txt", ios::app | ios::out);

f << txt;

f.close();

}

}

int IsCategoryNeIs(vector<Cost\*>& tmp,string category\_)

{

for (int i = 0; i < tmp.size(); i++)

{

if (category\_ == tmp[i]->Get\_Category())

return i;

}

return -1;

}

void ShowTop3CategoryWeek(int day, int mis, int year)

{

int l;

vector<Cost\*> tmp;

for (int i = 0; i < costs.size(); i++)

{

if (costs[i]->Get\_Day() >= day && costs[i]->Get\_Day() <= day + 7 && costs[i]->Get\_Mis() == mis && costs[i]->Get\_Year() == year)

{

l = IsCategoryNeIs(tmp, costs[i]->Get\_Category());

if (l == -1)

{

tmp.push\_back(new Cost(costs[i]->Get\_Name(), costs[i]->Get\_Category(), costs[i]->Get\_Money(), costs[i]->Get\_Day(), costs[i]->Get\_Mis(), costs[i]->Get\_Year()));

}

else

tmp[l]->Get\_Money() += costs[i]->Get\_Money();

}

}

sort(tmp.begin(), tmp.end(), SortByPrice());

string txt;

txt += "Топ-3 категорії за тиждень\n";

txt += "---------------------------------------\n";

int k;

int count = 0;

for (int i = 0; i < tmp.size(); i++)

{

if (tmp[i]->Get\_Day() >= day && tmp[i]->Get\_Day() <= day + 7 && tmp[i]->Get\_Mis() == mis && tmp[i]->Get\_Year() == year)

{

cout << tmp[i]->Get\_Category() << endl;

txt += tmp[i]->Get\_Category() + "\n";

txt += to\_string(tmp[i]->Get\_Day()) + ".";

txt += to\_string(tmp[i]->Get\_Mis()) + ".";

txt += to\_string(tmp[i]->Get\_Year()) + "\n";

txt += "---------------------------------------\n";

count++;

if (count == 3)

break;

}

}

txt += "---------------------------------------\n";

cout << "Зберегти у файл?(1-Так, 0-ні)->";

cin >> k;

if (k == 1)

{

fstream f("data.txt", ios::app | ios::out);

f << txt;

f.close();

}

for (int i = 0; i < tmp.size(); i++)

{

delete tmp[i];

}

}

void ShowTop3CategoryMonth( int mis, int year)

{

int l;

vector<Cost\*> tmp;

for (int i = 0; i < costs.size(); i++)

{

if ( costs[i]->Get\_Mis() == mis && costs[i]->Get\_Year() == year)

{

l = IsCategoryNeIs(tmp, costs[i]->Get\_Category());

if (l == -1)

{

tmp.push\_back(new Cost(costs[i]->Get\_Name(), costs[i]->Get\_Category(), costs[i]->Get\_Money(), costs[i]->Get\_Day(), costs[i]->Get\_Mis(), costs[i]->Get\_Year()));

}

else

tmp[l]->Get\_Money() += costs[i]->Get\_Money();

}

}

sort(tmp.begin(), tmp.end(), SortByPrice());

string txt;

txt += "Топ-3 категорії за місяць\n";

txt += "---------------------------------------\n";

int k;

int count = 0;

for (int i = 0; i < tmp.size(); i++)

{

if (tmp[i]->Get\_Mis() == mis && tmp[i]->Get\_Year() == year)

{

cout << tmp[i]->Get\_Category() << endl;

txt += tmp[i]->Get\_Category() + "\n";

txt += to\_string(tmp[i]->Get\_Day()) + ".";

txt += to\_string(tmp[i]->Get\_Mis()) + ".";

txt += to\_string(tmp[i]->Get\_Year()) + "\n";

txt += "---------------------------------------\n";

count++;

if (count == 3)

break;

}

}

txt += "---------------------------------------\n";

cout << "Зберегти у файл?(1-Так, 0-ні)->";

cin >> k;

if (k == 1)

{

fstream f("data.txt", ios::app | ios::out);

f << txt;

f.close();

}

for (int i = 0; i < tmp.size(); i++)

{

delete tmp[i];

}

}

};

class Wallet

{

string name;

vector<Card\*> cards;

void AddCost(int i)

{

string name;

string category;

float money;

int day;

int mis;

int year;

cout << "Введіть суму витрати->";

cin >> money;

if (money < cards[i]->Get\_Balance())

{

cards[i]->Get\_Balance() -= money;

cout << "Введіть назву витрати->";

cin >> name;

cout << "Введіть назву категорії витрати->";

cin >> category;

cout << "Введіть число(дата) витрати->";

cin >> day;

cout << "Введіть місяць(дата) витрати->";

cin >> mis;

cout << "Введіть рік(дата) витрати->";

cin >> year;

cards[i]->AddCost(new Cost(name, category, money, day, mis, year));

system("cls");

}

else

cout << "Недостатньо коштів на карті\n";

}

public:

int Get\_Size()

{

return cards.size();

}

string Get\_Name()

{

return name;

}

Wallet(string name)

{

this->name = name;

}

~Wallet()

{

for (auto it = this->cards.begin();it!=cards.end(); it++)

{

delete \*it;

}

}

void AddCard(Card\* card)

{

this->cards.push\_back(card);

}

void PerformOperation()

{

int day;

int mis;

int year;

int m;

int increase;

int k=1;

while (k)

{

int j = 0;

cout << "Оберіть карту:\n";

for (int i = 0; i < cards.size(); i++)

{

cards[i]->PrintInfo();

cout << "->" << i + 1 << endl;

}

cout << "->";

cin >> j;

while (j < 0 || j > cards.size())

{

cout << "Не коректний ввід->";

cin >> j;

}

int a;

system("cls");

cards[j - 1]->PrintInfo();

cout << endl;

cout << cards[j - 1]->Get\_Balance() <<"грн"<< endl;

cout << "Виберіть операцію:\nПоповнити карту->1\nЗробити витрату->2\nСформувати звіт->3\nСформувати рейтинг->4\nВихід->0\n";

cout << "->";

cin >> a;

system("cls");

while (a)

{

cards[j - 1]->PrintInfo();

cout << endl;

cout << cards[j - 1]->Get\_Balance() << "грн" << endl;

switch (a)

{

case 1:  
 cout << "Поповнити на ->";  
 cin >> increase;  
 cards[j-1]->Get\_Balance() += increase;  
 system("cls");  
 cards[j - 1]->PrintInfo();

cout << endl;

cout << cards[j - 1]->Get\_Balance() << "грн" << endl;  
 break;

case 2:

AddCost(j-1);

break;

case 3:

cout << "За конкретний день->1\nЗа тиждень->2\nЗа місяць->3\n";

cout << "->";

cin >> m;

system("cls");

switch (m)

{

case 1:  
 cout << "Введіть число(дата) витрати->";

cin >> day;

cout << "Введіть місяць(дата) витрати->";

cin >> mis;

cout << "Введіть рік(дата) витрати->";

cin >> year;  
 system("cls");  
 cards[j - 1]->ShowZvitDay(day,mis,year);  
 break;

case 2:

cout << "Введіть число(дата від якої почати тиждень) витрати->";

cin >> day;

cout << "Введіть місяць(дата від якої почати тиждень) витрати->";

cin >> mis;

cout << "Введіть рік(дата від якої почати тиждень) витрати->";

cin >> year;

system("cls");

cards[j - 1]->ShowZvitWeek(day, mis, year);

break;

case 3:

cout << "Введіть місяць(дата) витрати->";

cin >> mis;

cout << "Введіть рік(дата) витрати->";

cin >> year;  
 system("cls");  
 cards[j - 1]->ShowZvitMonth(mis, year);

break;

}

break;

case 4:

cout << "Топ 3-витрати(за тиждень->1,за місяць->2), Топ 3-категорії(за тиждень->3, за місяць->4)\n";

cout << "->";

cin >> m;

switch (m)

{

case 1:  
 cout << "Введіть число(дата від якої почати тиждень) витрати->";

cin >> day;

cout << "Введіть місяць(дата від якої почати тиждень) витрати->";

cin >> mis;

cout << "Введіть рік(дата від якої почати тиждень) витрати->";

cin >> year;

cards[j - 1]->ShowTop3Week(day, mis, year);  
 break;

case 2:

cout << "Введіть місяць(дата) витрати->";

cin >> mis;

cout << "Введіть рік(дата) витрати->";

cin >> year;

cards[j - 1]->ShowTop3Month(mis, year);

break;

case 3:

cout << "Введіть число(дата від якої почати тиждень) витрати->";

cin >> day;

cout << "Введіть місяць(дата від якої почати тиждень) витрати->";

cin >> mis;

cout << "Введіть рік(дата від якої почати тиждень) витрати->";

cin >> year;

cards[j - 1]->ShowTop3CategoryWeek(day, mis, year);

break;

case 4:

cout << "Введіть місяць(дата) витрати->";

cin >> mis;

cout << "Введіть рік(дата) витрати->";

cin >> year;

cards[j - 1]->ShowTop3CategoryMonth(mis, year);

break;

}

break;

}

cout << "Виберіть операцію:\nПоповнити карту->1\nЗробити витрату->2\nСформувати звіт->3\nСформувати рейтинг->4\nВихід->0\n";

cout << "->";

cin >> a;

system("cls");

}

system("cls");

cout << "Обрати іншу карту ->1\nВихід->0\n";

cin >> k;

system("cls");

}

}

};

class FinanceManager

{

vector<Wallet\*>wallets;

public:

~FinanceManager()

{

for (int i = 0; i < wallets.size(); i++)

{

delete wallets[i];

}

}

void StartWork()

{

string name;

string account;

float balance;

int k,k2,k3;

cout << "Додати гаманця->1,Вихід->0\n";

cout << "->";

cin >> k;

system("cls");

while (k)

{

switch (k)

{

case 1:  
 cout << "Введіть назву гаманця->";  
 cin >> name;  
 system("cls");  
 this->wallets.push\_back(new Wallet(name));  
 break;

case 2:

cout << "Оберіть гаманець\n";

for (int i = 0; i < wallets.size(); i++)

{

cout << wallets[i]->Get\_Name() << " ->" << i + 1<<endl;

}

cout << "->";

cin >> k2;

system("cls");

cout << "Додати карту->1\nЗайти у гаманець->2\nвихід ->0\n";

cout << "->";

cin >> k3;

while (k3)

{

switch (k3)

{

case 1:  
 system("cls");  
 cout << "Введіть назву карти->";  
 cin >> name;  
 cout << "Введіть рахунок карти->";  
 cin >> account;  
 cout << "Введіть баланс на карті->";  
 cin >> balance;  
 wallets[k2-1]->AddCard(new Card(name, account, balance));  
 system("cls");  
 break;

case 2:

system("cls");

if (wallets[k2 - 1]->Get\_Size() > 0)

wallets[k2 - 1]->PerformOperation();

else

cout << "Карт не знайдено додайте хоча б одну\n";

break;

}

cout << "Додати карту->1\nЗайти у гаманець->2\nвихід ->0\n";

cout << "->";

cin >> k3;

system("cls");

}

}

cout << "Додати гаманця->1,обрати гаманець->2,Вихід->0\n";

cout << "->";

cin >> k;

system("cls");

}

}

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

FinanceManager financemanager;

financemanager.StartWork();

system("pause");

return 0;

}