

Розробка системи класів

Програма складається з 8 класів, кожен з яких має свої змінні та методи.

| class Client |
|--|
| <pre>string fNameC, lNameC, emC, nC;</pre> |
| <pre>void message(){} void getCl(){} void zdClient(){} void write_to_file(){} void read_from_file(){} </pre> |

| class Representative |
|--|
| <pre>string fNameR, lNameR, adR, pR, emR, pdR, nR, sR;</pre> |
| <pre>void message(){} void getRl(){} void zdRepresentative(){} void write_to_file(){} void read_from_file(){} </pre> |

| class table_client |
|--|
| <pre>vector <Client> vec</pre> |
| <pre>void print_all_clients(){} void add_new_client(){} int search_client(){} void poshuk(){} void edit_client(){} void delete_client(){} void write_clients_to_file(){} void read_clients_from_file(){} void menu(){} table_client(){} ~table_client(){} </pre> |

| class table_client |
|---|
| <pre>vector <Representative> vec</pre> |
| <pre>void print_all_representatives(){} void add_new_representative(){} int search_representative(){} void poshuk(){} void edit_representative(){} void delete_representative(){} void write_representatives_to_file(){} void read_representatives_from_file(){} void menu_representatives(){} table_representative (){} ~table_representative(){} </pre> |

| class Services |
|---|
| <pre>string services; int price;</pre> |
| <pre>void message(){} void getSer(){} void zdServices(){} void write_to_file(){} void read_from_file(){} </pre> |

| class table_services |
|--|
| <pre>vector <Services> vec</pre> |
| <pre>void print_all_services(){} void add_new_services(){} int search_services(){} void poshuk(){} void edit_services (){} void delete_services(){} void write_services_to_file(){} void read_services_from_file(){} void menuServices(){} table_services(){} ~table_services(){} </pre> |

| class Order |
|--|
| <pre>string orderData, dataOfIssue; int client_id, representative_id, service_id;</pre> |
| <pre>void message(){} void getOrder(){} void zdOrder(){} void write_to_file(){} void read_from_file(){} </pre> |

| class table_order |
|--|
| <pre>vector <Order> vec</pre> |
| <pre>void print_all_orders(){} void add_new_orders (){} int search_orders(){} void poshuk(){} void edit_orders (){} void delete_orders(){} void write_orders_to_file(){} void read_orders_from_file(){} void menuOrder(){} table_orders(){} ~table_orders(){} </pre> |

Клас Client було створено для зберігання даних про окремого клієнта. У вищевказаному класі було передбачено поля, які зберігають загальні дані, такі як ім'я і прізвище, email та номер телефону. В цьому класі містяться методи, які дозволяють вивести дані про клієнта на екран або зчитати дані з консолі.

Клас `table_client` зберігає дані про всіх клієнтів у векторі, реалізує можливість вивести всіх клієнтів, знайти клієнта, редагувати дані, видалити клієнта, а також організовує меню клієнтів.

Клас `Representatives` було створено для зберігання даних про окремого представника фотостудії. У вищевказаному класі було передбачено змінні, які являють собою загальні дані, такі як ім'я і прізвище, email, паспортні дані, номер телефону, адресу, посаду та зарплату працівника які потрібні для бази даних. В цьому класі містяться методи, які дозволяють вивести дані про представника на екран або зчитати дані з консолі.

Клас `table_representatives` зберігає дані про всіх клієнтів у векторі, реалізує можливість вивести всіх представників салону, знайти представника, редагувати дані, видалити їх, а також організовує меню представників.

Клас `Services` було створено для зберігання даних про окрему послугу студії. У вищевказаному класі було передбачено змінні, які являють собою загальні дані, такі як назву послуги та її ціну. В цьому класі містяться методи, які дозволяють вивести дані про послугу на екран або зчитати дані з консолі.

Клас `table_services` зберігає дані про всі послуги у векторі, реалізує можливість вивести їх, знайти, редагувати та видалити дані, а також організовує меню послуг.

Клас `Order` було створено для зберігання даних про окреме замовлення. У вищевказаному класі було передбачено змінні, які являють собою загальні дані, такі як дату замовлення, дату видачі замовлення, ім'я та прізвище клієнта та представника та назву послуги. В цьому класі містяться методи, які дозволяють вивести дані про замовлення на екран або зчитати дані з консолі.

Клас `table_order` зберігає дані про всі замовлення у векторі, реалізує можливість виведення їх, знаходження, редагування та видалення даних, а також організовує меню замовлень.

Розробка методів

Метод в об'єктно-орієнтованому програмуванні — підпрограма (процедура, функція), що використовується виключно разом із класом (методи класу) або з об'єктом(методи екземпляра).

В даній програмі ми розглянемо приклади методів реалізованих в класі `table_client`.

```
void print_all_clients()
{
    for(int i=0; i<vec.size(); i++)
    {
        vec[i].getCl(); /*cout<<endl;*/
    }
}

void add_new_client()
{
    Client temp;
    temp.zdClient();
    vec.push_back(temp);
}

int search_client(bool mozhna_povernutus=true)
{
    string firstNameC;
    string lastNameC;
    int id=-1;
    while(true)
    {
        cout << "\n\tІм'я: ";
        cin >> firstNameC;
        cout << "\tПрізвище: ";
        cin >> lastNameC;
        for(int i=0; i<vec.size(); i++)
        {
            if((vec[i].getfNameC()==firstNameC)&&(vec[i].getlNameC()==lastNameC))id=i;
        }
        if(id==-1)
        {
            if(!mozhna_povernutus)
            {
                cout << "\tКлієнта не знайдено. Спробуйте ще раз." << endl;
                continue;
            }
            cout<<"\tКлієнта не знайдено. Спробуйте ще раз |1|, або поверніться в попереднє меню |2|" << endl;
            int in;
            cin >> in;
            switch(in)
            {
                case 1:
                    break;
                case 2:
                    return -1;
            }
        }
    }
}
```

```

        break;
    }
}
else return id;
}
}

void poshuk()
{
    int id;
    id=search_client();
    if(id!=-1)vec[id].getCl();
}

void edit_client()
{
    int id;
    id=search_client();
    if(id!=-1)vec[id].zdClient();
}

void delete_client()
{
    int id;
    id=search_client();
    if(id!=-1)
    {
        vec.erase(vec.begin()+id);
        cout << "\tКлієнта видалено" <<endl;
    }
    else cout << "\tКлієнта не видалено" <<endl;
}

void write_clients_to_file()
{
    ofstream fClient("client1.txt");
    for(int i=0; i<vec.size(); i++) vec[i].write_to_file(fClient);//для
кожного елемента викликати функцію яка записує у файл
    fClient.close();
}

void read_clients_from_file()
{
    ifstream fClient("client1.txt");
    if(fClient.is_open())
    {
        Client temp;
        while(!fClient.eof())
        {
            temp.read_from_file(fClient);
            vec.push_back(temp);
        }
        vec.erase(vec.end()); //видалення останнього об'єкта, бо його
дублює
        fClient.close();
    }
}
}

```

```

void menu()
{
    while(1)
    {
        cout <<"\n\tВивести всіх клієнтів----->|1|\n\tЗнайти клієнта-----
----->|2|\n\tДодати клієнта----->|3|\n\tРедагувати дані про
клієнта>|4|\n\tВидалити клієнта----->|5|\n\tВийти-----
>|6|" << endl;
        int inC;
        cin >> inC;
        switch(inC)
        {
            case 1:
                print_all_clients();
                break;
            case 2:
                poshuk();
                break;
            case 3:
                add_new_client();
                break;
            case 4:
                edit_client();
                break;
            case 5:
                delete_client();
                break;
            case 6:
                return;
        }
    }
}

table_client()
{
    read_clients_from_file();    //конструктор
}

~table_client()
{
    write_clients_to_file();    //деструктор
}
};

```

Метод void void print_all_clients за допомогою циклу виводить всі дані про всіх клієнтів;

Метод void void add_new_client створює тимчасовий об'єкт класу Client, викликає для нього метод введення даних з клавіатури;

Метод int search_client знаходить індекс клієнта у векторі за іменем та прізвищем;

Метод void poshuk викликає метод search_client, який знаходить індекс клієнта і для об'єкта з цим індексом викликає метод виведення на екран;

Метод `void edit_client` викликає метод `search_client`, який знаходить індекс клієнта і для об'єкта з цим індексом викликає метод введення з консолі (тобто змінює дані);

Метод `void delete_client` викликає метод `search_client`, який знаходить індекс клієнта і об'єкт з цим індексом видаляє з вектора

Метод `void write_clients_to_file` для запису даних про клієнтів в файл "Client.txt";

Метод `void read_clients_from_file` для зчитування даних про клієнтів з файлу "Client.txt";

Метод `void menu` реалізує підменю умовної таблиці клієнти;

За таким принципом реалізовані методи інших класів.

Інтерфейс програми.

Програма використовує зручний і зрозумілий інтерфейс, організована система меню і підменю.

=====PHOTO STUDIO=====

Клієнти|1|, Послуги|2|, Представники|3|, Замовлення|4|, Вийти|5|

При виборі пункту |1|:

```
Вивести всіх клієнтів----->|1|
Знайти клієнта----->|2|
Додати клієнта----->|3|
Редагувати дані про клієнта>|4|
Видалити клієнта----->|5|
Вийти----->|6|
```

Дані підменю дає змогу вивести всіх клієнтів|1|:

```
Ім'я:      Andrey
Прізвище:   Gnatun
Email:      andrey@gmail.com
Номер телефону: 380997362738
```

```
Ім'я:      Gnat
Прізвище:   Tkachuk
Email:      tkachuk
Номер телефону: 380674829480
```

```
Ім'я:      Anna
Прізвище:   Budz
Email:      budz@gmail.com
Номер телефону: 380674380924
```

Також знайти дані про клієнта за іменем та прізвищем|2|:

```
Ім'я: Anna
Прізвище: Budz
```

```
Ім'я:      Anna
Прізвище:   Budz
Email:      budz@gmail.com
Номер телефону: 380674380924
```


Пункт |3| додає нового клієнта:

```
Введіть дані про клієнта
Ім'я: Inna
Прізвище: Savchuk
Email: innas@gmail.com
Номер: 380994809208
Дані збережено!
```

Пункт |4| дає змогу редагувати дані про клієнта:


```
Ім'я: Gnat
Прізвище: Tkachuk
Введіть дані про клієнта
Ім'я: Vlad
Прізвище: Tkachuk
Email: vlad@gmail.com
Номер: 380994739284
Дані збережено!
```

Пункт |5| знаходить клієнта за іменем та прізвищем і видаляє його:

```
Ім'я: Inna
Прізвище: Savchuk
Клієнта видалено
```

Пункт |6| повертається в головне меню.

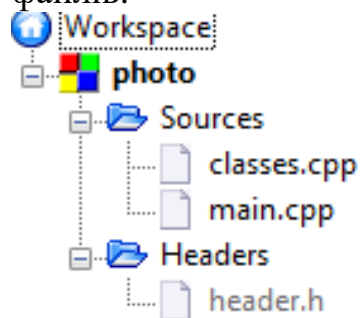
Після завершення роботи з програмою всі дані зберігаються в файловій системі.

 client1 - Блокнот

| Файл | Редагування | Формат | Вигляд | Довідка |
|--------|-------------|------------------|--------------|---------|
| Andrey | Gnatun | andrey@gmail.com | 380997362738 | |
| Vlad | Tkachuk | vlad@gmail.com | 380994739284 | |
| Anna | Budz | budz@gmail.com | 380674380924 | |

За таким принципом працюють інші умовні таблиці.

Дана програма створена як багатофайловий проект, вона складається з 3 файлів:



У файлі main.cpp реалізоване головне меню і створене об'єкти деяких класів;

У файлі header.h оголошуються класи;

У файлі classes.cpp реалізовані методи;