МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КУРСОВА РОБОТА

з дисципліни “Об’єктно-орієнтоване програмування”

за темою “Створення ієрархії класів для салону краси”

|  |  |
| --- | --- |
|  | Керівник роботи: |
|  | Любченко Н.Ю. |
|  |  |
|  | Виконавець:  Лісова Д.В.,  група КН-321-В |
|  |  |

Харків, 2022

ЗАВДАННЯ

на курсову роботу

з курсу «Об’єктно-орієнтоване програмування»

Тема: “Створення ієрархії класів для салону краси”

Постановка задачі: спроектувати та реалізувати програму графічного інтерфейсу користувача для салону краси

Необхідно створити ієрархію із 3-ти і більше класів. Виконання необхідного завдання здійснюється на основі об’єктів класів.

Інтерфейс користувача передбачає створення вікна з головним меню, необхідними кнопками, таблицями для введення даних.

Реалізація програми повинна здійснюватися з використанням об’єктно-орієнтованих технологій. Слід представити необхідну інформацію про програму та способи її застосування.

Короткий зміст роботи:

А) реферативна частина

Дослідження поставленої задачі. Вибір та обґрунтування необхідних програмних методів розв’язання задачі.

Б) теоретична частина

Визначення вимог щодо програмного забезпечення. Проектування інтерфейсу користувача. Розробка необхідних UML-діаграм.

В) програмна частина

Реалізація та тестування програми графічного інтерфейсу користувача.

РЕФЕРАТ

Ключові слова: CPP, наслідування класів, UML-діаграми, графічний інтерфейс, об’єкти класу, ПІБ клієнта, ПІБ спеціаліста, послуга, вартість послуги, таблиця, форми, кнопки button, radiobutton.

Курсова робота виконується з метою закріплення навиків отриманих при вивченні курсів Програмування, ООП.

Об'єктивним дослідженням була розробка прикладної програми графічного інтерфейсу користувача для салону краси.

При виконанні робіт було використано:

1) наслідування класів;

2) роздільна компіляція;

3) створення UML-діаграми в MS Visual Studio.

Зміст

[Вступ 5](#_Toc119527145)

[1.Інтегроване середовище розробки 6](#_Toc119527146)

[1.1 IntelliJ IDEA 7](#_Toc119527147)

[1.2MS Visual Studio 7](#_Toc119527148)

[1.3 Обгрунтування вибору середовища 8](#_Toc119527149)

[2.Проектування програмного забезпечення. 9](#_Toc119527150)

[2.1Опис розробленого програмного забезпечення 9](#_Toc119527151)

[2.2Створення діаграми класів в середовищі VS 10](#_Toc119527152)

[3.Опис програмного продукту 12](#_Toc119527153)

[3.1 Опис інтерфейсу розробленого продукту 12](#_Toc119527154)

[3.2 Тестування розробленого продукту 12](#_Toc119527155)

[Висновок 18](#_Toc119527156)

[Список використаних джерел 19](#_Toc119527157)

[ДОДАТОК А 20](#_Toc119527158)

# Вступ

База даних – це засіб збирання та впорядкування інформації. Бази даних можуть зберігати відомості про людей, продукти, замовлення або будь-що інше. Багато баз даних починаються зі списку в текстовому редакторі або електронній таблиці. Оскільки список збільшується, то в даних починають з'являтися неузгоджені та невідповідні значення. У формі списку дані буде важко розібратися, а також є обмежені способи пошуку або витягування підмножини даних для перевірки. Після того як ці проблеми почнуть з'являтися, можна перенести дані до бази даних, створеної системою керування базами даних (СУБД), наприклад Access.

Але також можна створити власну базу даних на зразок тих, які вже є придумані. Звісно, це цікава ідея й вона вимагає певних зусиль, тому що потрібно мати багато знань й певну мову програмування.

# Інтегроване середовище розробки

Інтегроване середовище розробки (ІСР, англ. Integrated development environment або англ. IDE) — комплексне програмне рішення для розробки програмного забезпечення. Зазвичай, складається з редактора початкового коду, інструментів для автоматизації складання та відлагодження програм. Більшість сучасних середовищ розробки мають можливість автодоповнення коду.

Зазвичай, середовище розробки включає в себе:

* текстовий редактор,
* компілятор і / або інтерпретатор,
* засоби автоматизації збирання,
* відладчик.

Деякі середовища розробки містять компілятор, інтерпретатор або ж обидва (наприклад NetBeans та Eclipse), інші не містять жодного з них (SharpDevelop та Lazarus). Деякі інтегровані середовища розробки містять систему керування версіями або інструменти для полегшення розробки графічного інтерфейсу користувача (GUI) (XCode, Embarcadero Delphi). Багато сучасних ІСР містять інспектор класів, інспектор об'єктів, схему ієрархії класів для полегшення об'єктно-орієнтованої розробки програмного забезпечення.

Основні середовища розробки:

* Eclipse;
* ActiveState;
* Komodo;
* IntelliJ IDEA;
* MyEclipse;
* Oracle JDeveloper;
* NetBeans;
* Codenvy;
* Microsoft Visual Studio.

1.1 IntelliJ IDEA

IntelliJ IDEA – комерційне інтегроване середовище розробки для різних мов програмування (Java, Python, Scala, PHP) від компанії JetBrains. Система поставляється у вигляді урізаної по функціональності безкоштовної версії “Community Edition” і повнофункціональної комерційної версії “Ultimate Edition”, для якої активні розробники відкритих проектів мають можливість отримати безкоштовну ліцензію. Сирцеві тексти Community-версії поширюються рамках ліцензії Apache 2.0. Бінарні збірки підготовлені для Linux, Mac OS X і Windows.

Community версія середовища IntelliJ IDEA підтримує інструменти (у вигляді плагінів) для проведення тестування TestNG і JUnit, системи контролю версій CVS, Subversion, Mercurial і Git, засоби складання Maven, Ant, Gradle, мови програмування Java, Scala, Clojure, Groovy і Dart. Підтримується розробка застосунків для мобільної платформи Android. До складу входить модуль візуального проектування GUI-інтерфейсу Swing UI Designer, XML-редактор, редактор регулярних виразів, система перевірки коректності коду, система контролю за виконанням завдань і доповнення для імпорту та експорту проектів з Eclipse. Доступні засоби інтеграції з системами відстеження помилок JIRA, Trac, Redmine, Pivotal Tracker, GitHub, YouTrack, Lighthouse.

Комерційна версія “Ultimate Edition” відрізняється наявністю підтримки додаткових мов програмування (наприклад, PHP, Ruby, Python, JavaScript, CoffeeScript, HTML, CSS, SQL), підтримкою технологій Java EE, UML-діаграм, підрахунок покриття коду, можливістю роботи з фреймворками (Rails, Grails, Google Web Toolkit, Spring, Play Framework і Hibernate), засобами інтеграції з Perforce, Microsoft Team Foundation Server і Rational ClearCase.

## 1.2MS Visual Studio

MS Visual Studio – серія продуктів фірми Майкрософт, які містять інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів. Ці продукти дають змогу розробляти як консольні програми, так і програми з графічним інтерфейсом, включно з підтримкою технології Windows Forms, а також вебсайти, вебзастосунки, вебслужби як у рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

## 1.3 Обґрунтування вибору середовища

Для розробки бази даних було вибрано MS Visual Studio. На цей вибір вплинули певні причини. По-перше додаток писався на С++. А MS Visual Studio його підтримує, також багато хто знайомий з цією програмою, тому що зазвичай її використовують для консольного програмування. Більш того MS Visual Studio має багато функціональних можливостей. Таких, як автоматичне додавання й видалення коментарів, поетапного висліджування програми, це досить корисно, коли не можна найти помилку. Також за допомогою певних клавіш можна швидко писати команди. Тобто розробнику, не потрібно пам’ятати структуру певних елементів. Як допускається помилка, то код підсвічується й зазвичай показується спосіб вирішення цієї помилки, за допомогою підключень певних бібліотек.

# Проектування програмного забезпечення.

## Опис розробленого програмного забезпечення

Для створення даної бази даних було створено класи, й використано основні принципі ООП, а саме наслідування. В кожному класі було створено конструктор, й методи встановлення й отримання значення певної змінної. Й використано метод інкапсуляції, тобто всі поля класу є private. Основні класи: Age, Persone, Price, Spesialist, Stag, Posluga. Основні функції класів це гетери й сетери. Наприклад: void set\_age(int age) чи int get\_age().

Також функція для заповнення бази даних: private:void create\_mass(Client cl[], int count.

Й наступні функції для роботи з формою:

* private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e);-для виведення таблиці
* private:void create\_mass(Client cl[], int count);
* private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e); для видалення в базі даних
* private: System::Void button4\_Click(System::Object^ sender, System::EventArgs^ e); для пошуку в базі даних
* private: System::Void button5\_Click(System::Object^ sender, System::EventArgs^ e) для створення бази даних
* private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e); для додавання елемента до таблиці

Також в даному проекті було використано таблицю. Вона створюється ось так: dataGridView1->ColumnCount = 5;//задаємо кількість колонок

dataGridView1->RowCount = count;//задаємо кількість рядків

dataGridView1->Columns[0]->Name = "ПІБ клієнта";//створюємо шапку

dataGridView1->Columns[1]->Name = "Вік клієнта";

dataGridView1->Columns[2]->Name = "ПІБ спеціаліста";

dataGridView1->Columns[3]->Name = "Назва послуги";

dataGridView1->Columns[4]->Name = "Ціна, грн";

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

for (int j = 0; j < 1; j++) {

//dataGridView1->Rows[i]->Cells[j]->Value = "5";

//заповнюємо таблицю

dataGridView1->Rows[i]->Cells[j]->Value = gcnew System::String(cl[i].get\_name().c\_str());// Convert::ToString(cl[i].());

dataGridView1->Rows[i]->Cells[j + 1]->Value = Convert::ToString(cl[i].get\_age());

dataGridView1->Rows[i]->Cells[j + 2]->Value = gcnew System::String(cl[i].spesialist.get\_name().c\_str());

dataGridView1->Rows[i]->Cells[j + 3]->Value = gcnew System::String(cl[i].spesialist.get\_name\_posluga().c\_str());

dataGridView1->Rows[i]->Cells[j + 4]->Value = Convert::ToString(cl[i].spesialist.get\_price());

}

}

dataGridView1->AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders);//змінюємо розмір колоно

dataGridView1->AutoResizeColumns(); //робимо колонки по ширині текста

## Створення діаграми класів в середовищі VS

UML (англ. Unified Modeling Language – уніфікована мова моделювання) – мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення, для моделювання бізнес-процесів, системного проектування та відображення організаційних структур.

Діаграма класів (Class diagram) – статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм. Діаграма класів служить для представлення статичної структури моделі системи в термінології класів об'єктноорієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

Процес створення UML-діаграми в середовищі Microsoft Visual Studio на мові програмування С++.

1. Спершу потрібно перевірити, чи в нас встановлений певний пакет (рис.2.1), якщо немає то скачуємо.

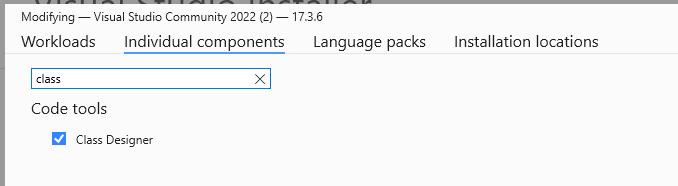


Рисунок 2.1 – Перевірка пакету

2. Натиснути правою кнопкою на рішення проекту.

3. Вибрати представлення, перейти до діаграми класів.

Або можна виконати наступні дії.

1. Додати елемент діаграма класів.

2. Відкрити представлення класів.

1. Перетягнути клас на робочу область.

На рис 2.1 зображена діаграма класів для програми графічного інтерфейсу користувача для салону краси.

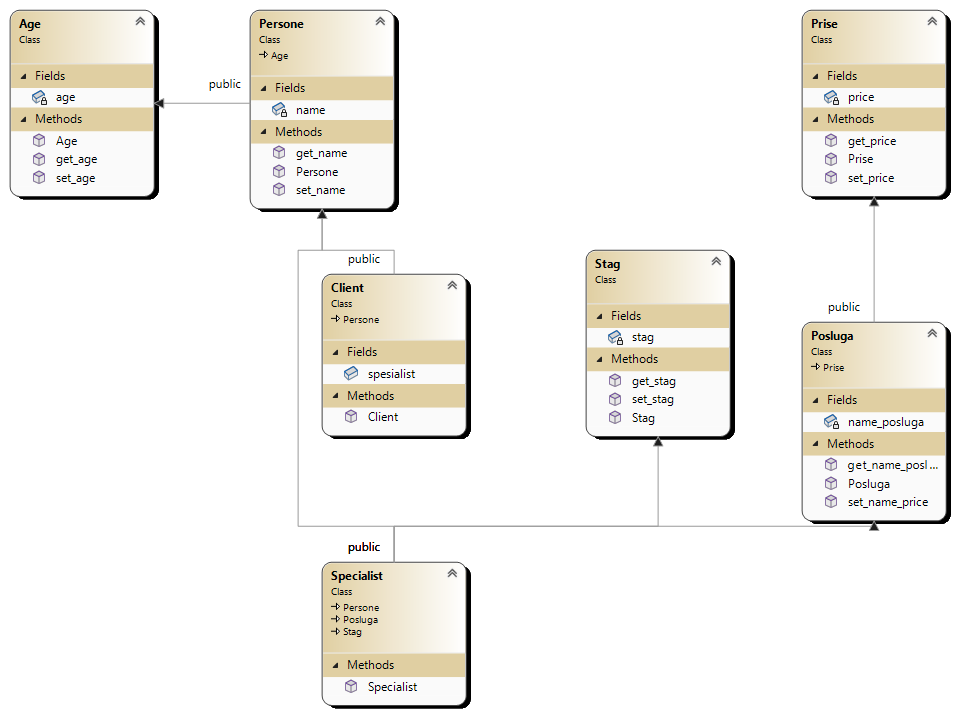


Рисунок 2.1 – Діаграма класів

# Опис програмного продукту

## 3.1 Опис інтерфейсу розробленого продукту

На рисунку 3.1 представлений інтерфейс розробленого продукту.

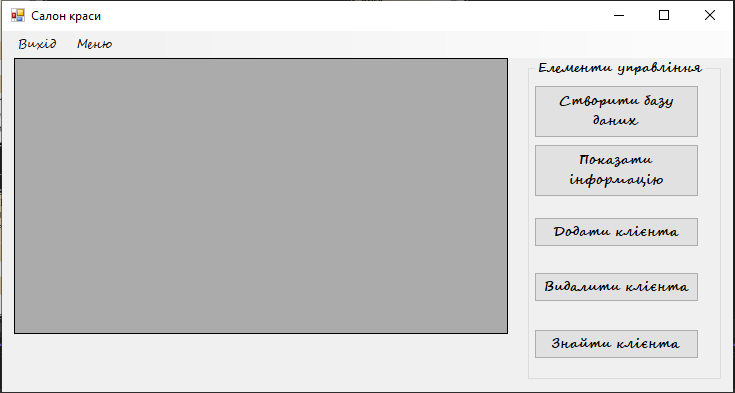


Рисунок 3.1 – Інтерфейс розробленого продукту

Інтерфейс додатку складається з наступних компонентів.

* DataGridView використовується для відображення бази даних.
* Кнопки “Створити базу даних” та “Показати інформацію” використовується для відображення відповідних баз даних по натисканню.
* Кнопки “Додати клієнта”, “Видалити клієнта”, “Знайти клієнта” відповідають за роботу з базою даних.
* Також ToolStripMenuItem відповідає за те щоб вийти з бази даних, або подивитися певну інформацію про базу даних, або про салон краси.

## 

## 3.2 Тестування розробленого продукту

Створимо базу даних нажавши на кнопку створити базу даних (рис.3.2)

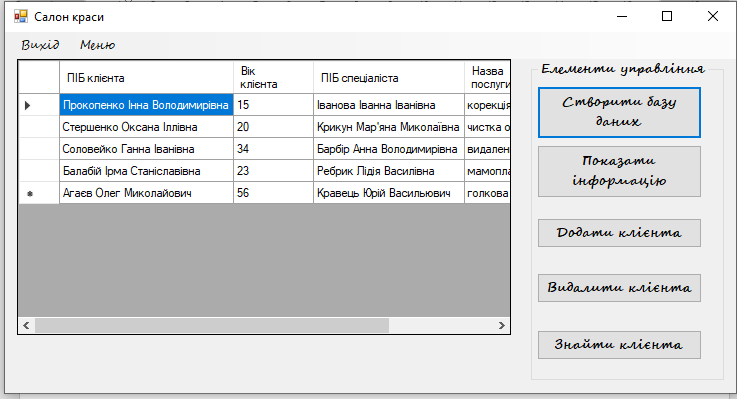


Рисунок 3.2 – Створення бази даних

Натиснемо кнопку “Додати клієнта” (рис. 3.3). й зробимо певні заповнення полів й натиснемо “Додати” (рис. 3.4). Й програма повідомить про успішне додавання клієнта (рис. 3.5). Й до даної бази даних добавиться новий клієнт (рис. 3.6).

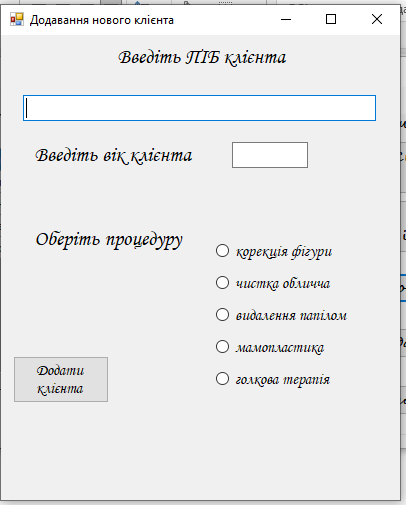


Рисунок 3.3 – Додавання клієнта

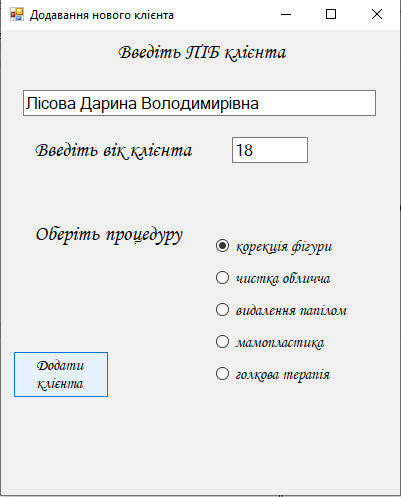


Рисунок 3.4 – Заповнення даних про клієнта

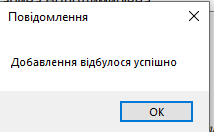


Рисунок 3.5 – Успішне додавання клієнта

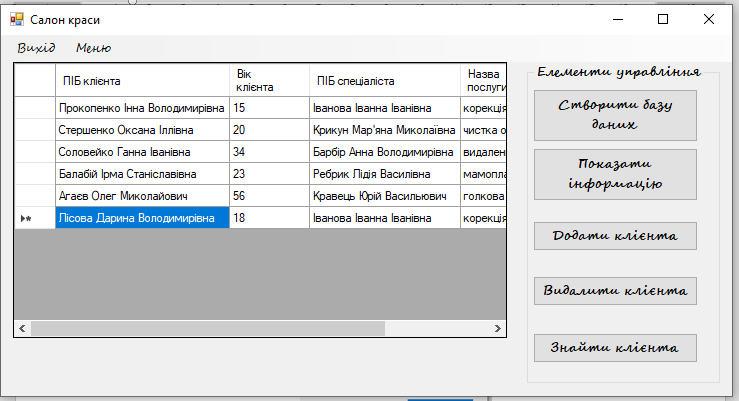


Рисунок 3.6 – Змінена база даних

Також можна видалити клієнта натиснувши кнопку “Видалити клієнта” й появиться вікно в якому потрібно ввести ПІБ клієнта (рис. 3.7). Якщо клієнт знайдений то видалення відбудеться успішно й з’явиться повідомлення (рис. 3.8), якщо клієнта немає – теж повідомлення про те, що даний клієнт відсутній в базі даних (рис. 3.9). Подивитися, що даний клієнт видалився, можна, на (рис. 3.10).

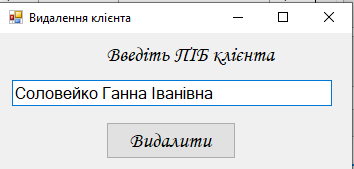


Рисунок 3.7 – Видалення клієнта

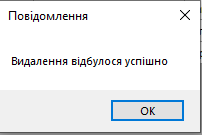


Рисунок 3.8 – Успішний результат видалення

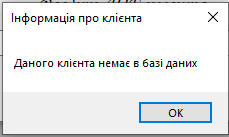


Рисунок 3.9 – Неуспішний результат видалення

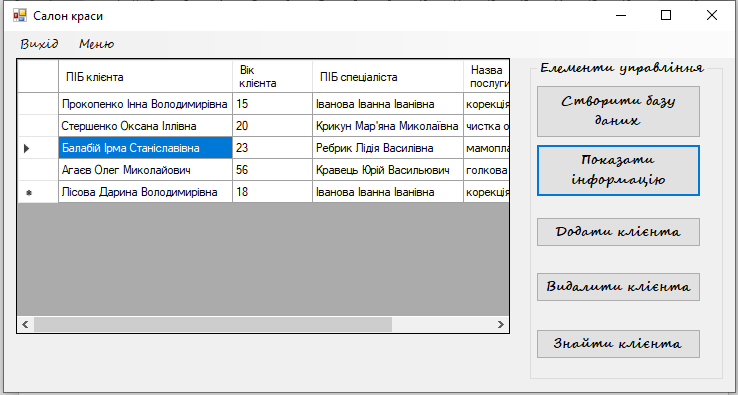


Рисунок 3.10 – Видалення клієнта

Можемо знайти інформацію про клієнта натиснувши кнопку “Знайти клієнта”. Знову потрібно буде ввести ПІБ й якщо даний клієнт існує, то виведеться про нього інформація (рис.3.11, 3.12).

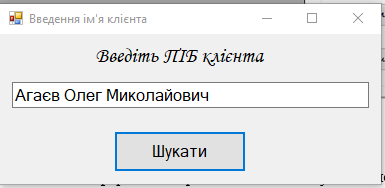


Рисунок 3.11 – Пошук клієнта

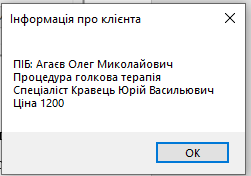


Рисунок 3.12 – Результат пошуку

# Висновок

Під час виконання курсової роботи були отримані навички створення програмного забезпечення на мові С++ з використанням наслідування класів. У MS Visual Studio була створена діаграма класів із точки зору реалізації.

Розроблений програмний продукт дозволяє створити та редагувати базу даних.

Програмний продукт реалізує такі функції: введення даних в базу даних клієнтів, друк бази даних клієнтів салону, введення даних в базу даних клієнтів, друк бази даних лікарів, видалення клієнтів й пошук.

Розроблений застосунок може використовуватись у цілях зберігання інформації клієнтів й для їх запису.

# Список використаних джерел

1) Інформаціїні технологіїї в суспільстві ХХІ століття [Електронний ресурс]. URL: [https://stud.com.ua/86670/informatika/informatsiyni\_tehnologiyi\_ suspilstvi\_stolittya](https://stud.com.ua/86670/informatika/informatsiyni_tehnologiyi_suspilstvi_stolittya)

2) Топ 7 IDE для програмування на С++ [Електронний ресурс]. URL: <https://itvdn.com/ru/blog/article/cplspls-top7(lfnf>

3) Інтегроване середовище розробки [Електронний ресурс]: URL: <https://ru.wikipedia.org/wiki/Інтегроване_середовище_розробки>

4) Діаграма класів [Електронний ресурс]: URL: <https://ru.wikipedia.org/wiki/UML#Діаграма_класів>

5) Ліберті Дж. Опануй самостійно С++ за 21 день. Видавничий дім «Вільямс», 2001. 832 с.

6) Страуструп Б. Мова програмування С++. Радіо та зв’язок ,1991. 352 с.

7) Лукас П. С ++ під рукою. - Київ: «ДиаСофт», 1993. 176 с.

8) Коллінз У. Структури даних і стандартна бібліотека шаблонів. «Біном-Пресс», 2004. 624 с

9) Додавання діаграми класів у проекти [Електронний ресурс]: <https://docs.microsoft.com/ru-ru/visualstudio/ide/class-designer/how-to-add-class-diagrams-to-projects?view=vs-2022>

10) Робота з конструктором класів [Електронний ресурс]: https://docs.microsoft.com/ru-ru/visualstudio/ide/class-designer/designing-and-viewing-classes-and-types?view=vs-2022

# ДОДАТОК А

Header\_for\_class.h:

class Age {

int age;

public:

Age();

void set\_age(int age);

int get\_age();

};

class Persone: public Age {

std::string name;

public:

Persone();

void set\_name(std::string name);

std::string get\_name();

};

class Prise {

int price;

public:

Prise();

void set\_price(int price);

int get\_price();

};

class Stag {

int stag;

public:

Stag();

void set\_stag(int stag);

int get\_stag();

};

class Posluga:public Prise {

std::string name\_posluga;

public:

Posluga();

void set\_name\_price(std::string name\_posluga);

std::string get\_name\_posluga();

};

class Specialist:public Persone,public Posluga,public Stag {

public:

Specialist();

};

class Client:public Persone {

public:

Specialist spesialist;

Client();

};

Source\_for\_class.cpp

#include"Header\_for\_class.h"

Age::Age()

{

age = 0;

}

void Age::set\_age(int age)

{

this->age = age;

}

int Age::get\_age()

{

return age;

}

Persone::Persone():Age()

{

name = " ";

}

void Persone::set\_name(std::string name)

{

this->name = name;

}

std::string Persone::get\_name()

{

return name;

}

Prise::Prise()

{

price = 0;

}

void Prise::set\_price(int price)

{

this->price = price;

}

int Prise::get\_price()

{

return price;

}

Stag::Stag()

{

stag = 0;

}

void Stag::set\_stag(int stag)

{

this->stag = stag;

}

int Stag::get\_stag()

{

return stag;

}

Posluga::Posluga(): Prise()

{

name\_posluga = " ";

}

void Posluga::set\_name\_price(std::string name\_posluga)

{

this->name\_posluga = name\_posluga;

}

std::string Posluga::get\_name\_posluga()

{

return name\_posluga;

}

Specialist::Specialist():Persone(),Posluga(),Stag()

{

}

Client::Client():Persone()

{

}

Функції:

System::Void rgz::MyForm::проБазуДанихToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

MessageBox::Show("За допомогою цієї бази дани, можна легко дізнатися інформацію про салон краси.Також добавити інформацію про спеціаліста, чи зробити запис.", "Інформація про програму");

return System::Void();

}

System::Void rgz::MyForm::проАвтораToolStripMenuItem1\_Click(System::Object^ sender, System::EventArgs^ e)

{

MessageBox::Show("Виконала студентка 2 курсу\n Група: 321-В \n Лісова Дарина", "Інформація про автора");

return System::Void();

}

System::Void rgz::MyForm::button1\_Click(System::Object^ sender, System::EventArgs^ e)

{

/\*int count = 5;\*/

create\_mass(cl, count);

dataGridView1->ColumnCount = count;

dataGridView1->RowCount = count;

dataGridView1->Columns[0]->Name = "ПІБ клієнта";

dataGridView1->Columns[1]->Name = "Вік клієнта";

dataGridView1->Columns[2]->Name = "ПІБ спеціаліста";

dataGridView1->Columns[3]->Name = "Назва послуги";

dataGridView1->Columns[4]->Name = "Ціна, грн";

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

for (int j = 0; j < 1; j++) {

//dataGridView1->Rows[i]->Cells[j]->Value = "5";

dataGridView1->Rows[i]->Cells[j]->Value = gcnew System::String(cl[i].get\_name().c\_str());// Convert::ToString(cl[i].());

dataGridView1->Rows[i]->Cells[j+1]->Value =Convert::ToString(cl[i].get\_age());

dataGridView1->Rows[i]->Cells[j+2]->Value = gcnew System::String(cl[i].spesialist.get\_name().c\_str());

dataGridView1->Rows[i]->Cells[j+3]->Value = gcnew System::String(cl[i].spesialist.get\_name\_posluga().c\_str());

dataGridView1->Rows[i]->Cells[j+4]->Value = Convert::ToString(cl[i].spesialist.get\_price());

}

}

dataGridView1->AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders);

dataGridView1->AutoResizeColumns();

return System::Void();

}

void rgz::MyForm::create\_mass(Client cl[], int count)

{

int age\_cl[5] = { 15,20,34,23,56 };

std::string name\_sp[5] = { "Іванова Іванна Іванівна","Крикун Мар'яна Миколаївна","Барбір Анна Володимирівна","Ребрик Лідія Василівна","Кравець Юрій Васильювич" };

std::string name\_cl[5] = { "Прокопенко Інна Володимирівна","Стершенко Оксана Іллівна","Соловейко Ганна Іванівна","Балабій Ірма Станіславівна","Агаєв Олег Миколайович" };

std::string spe[5] = { "масажист","косметолог","дерматолог","хірург","терапевт-мануальщик" };

std::string posluga[5] = { "корекція фігури","чистка обличча","видалення папілом","мамопластика","голкова терапія" };

int stag[5] = { 10,5,3,2,1 };

int prise[5] = { 2500,650,550,4500,1200 };

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

cl[i].set\_name(name\_cl[i]);

cl[i].set\_age(age\_cl[i]);

cl[i].spesialist.set\_name(name\_sp[i]);

cl[i].spesialist.set\_stag(stag[i]);

cl[i].spesialist.set\_price(prise[i]);

cl[i].spesialist.set\_name\_price(posluga[i]);

cl[i].spesialist.set\_stag(stag[i]);

}

//throw gcnew System::NotImplementedException();

}

System::Void rgz::MyForm::button3\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (count > 0) {

MyForm2^ form2 = gcnew MyForm2();

form2->Show();

}

else {

MessageBox::Show("Неможливе видалення", "Увага");

}

return System::Void();

}

System::Void rgz::MyForm::button4\_Click(System::Object^ sender, System::EventArgs^ e)

{

MyForm1^ form1 = gcnew MyForm1();

form1->Show();

return System::Void();

}

System::Void rgz::MyForm1::button1\_Click(System::Object^ sender, System::EventArgs^ e)//створення бази даних

{

if (textBox1->Text == "") {

MessageBox::Show("Введіть ПІБ клієнта", "Помилка");

}

else {

System::String^ name\_persone = textBox1->Text;

std::string name\_per = msclr::interop::marshal\_as<std::string>(name\_persone);

int k;

bool is = false;

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

if (cl[i].get\_name() == name\_per) {

is = true;

k = i;

}

}

if (is) {

MessageBox::Show("ПІБ: " + gcnew System::String(cl[k].get\_name().c\_str()) + "\n" + "Процедура " + gcnew System::String(cl[k].spesialist.get\_name\_posluga().c\_str()) + "\n" +

"Спеціаліст " + gcnew System::String(cl[k].spesialist.get\_name().c\_str()) + "\n" + "Ціна " + Convert::ToString(cl[k].spesialist.get\_price()), "Інформація про клієнта");

}

else {

MessageBox::Show("Даного клієнта немає в базі даних", "Інформація про клієнта");

}

/\*std::wstring name1 = marshal\_as<std::wstring>(this->textBox1->Text);\*/

//MyForm1::Hide();

}

return System::Void();

}

System::Void rgz::MyForm2::button1\_Click(System::Object^ sender, System::EventArgs^ e)//для видалення

{

if (textBox1->Text == "") {

MessageBox::Show("Введіть ПІБ клієнта", "Помилка");

}

else {

if (count > 1) {

System::String^ name\_persone = textBox1->Text;

std::string name\_per = msclr::interop::marshal\_as<std::string>(name\_persone);

int k;

bool is = false;

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

if (cl[i].get\_name() == name\_per) {

is = true;

k = i;

}

}

if (is) {

Client\* cl1 = new Client[count - 1];

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

cl1[i].set\_age(cl[i].get\_age());

cl1[i].set\_name(cl[i].get\_name());

cl1[i].spesialist.set\_age(cl[i].spesialist.get\_age());

cl1[i].spesialist.set\_name(cl[i].spesialist.get\_name());

cl1[i].spesialist.set\_name\_price(cl[i].spesialist.get\_name\_posluga());

cl1[i].spesialist.set\_price(cl[i].spesialist.get\_price());

cl1[i].spesialist.set\_stag(cl[i].spesialist.get\_stag());

}

for (int i = k + 1; i < count; i++) {

cl1[i - 1].set\_age(cl[i].get\_age());

cl1[i - 1].set\_name(cl[i].get\_name());

cl1[i - 1].spesialist.set\_age(cl[i].spesialist.get\_age());

cl1[i - 1].spesialist.set\_name(cl[i].spesialist.get\_name());

cl1[i - 1].spesialist.set\_name\_price(cl[i].spesialist.get\_name\_posluga());

cl1[i - 1].spesialist.set\_price(cl[i].spesialist.get\_price());

cl1[i - 1].spesialist.set\_stag(cl[i].spesialist.get\_stag());

}

count--;

delete[] cl;

//cl = nullptr;

cl = new Client[count];

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

cl[i].set\_age(cl1[i].get\_age());

cl[i].set\_name(cl1[i].get\_name());

cl[i].spesialist.set\_age(cl1[i].spesialist.get\_age());

cl[i].spesialist.set\_name(cl1[i].spesialist.get\_name());

cl[i].spesialist.set\_name\_price(cl1[i].spesialist.get\_name\_posluga());

cl[i].spesialist.set\_price(cl1[i].spesialist.get\_price());

}

delete[]cl1;

MyForm2::Hide();

MessageBox::Show("Видалення відбулося успішно", "Повідомлення");

}

else {

MessageBox::Show("Даного клієнта немає в базі даних", "Інформація про клієнта");

}

}

else {

count--;

delete[] cl;

MyForm2::Hide();

MessageBox::Show("Видалення відбулося успішно", "Повідомлення");

}

}

return System::Void();

}

System::Void rgz::MyForm::button5\_Click(System::Object^ sender, System::EventArgs^ e)//показати інформацію

{

if (count > 0) {

dataGridView1->ColumnCount = 5;

dataGridView1->RowCount = count;

dataGridView1->Columns[0]->Name = "ПІБ клієнта";

dataGridView1->Columns[1]->Name = "Вік клієнта";

dataGridView1->Columns[2]->Name = "ПІБ спеціаліста";

dataGridView1->Columns[3]->Name = "Назва послуги";

dataGridView1->Columns[4]->Name = "Ціна, грн";

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

for (int j = 0; j < 1; j++) {

//dataGridView1->Rows[i]->Cells[j]->Value = "5";

dataGridView1->Rows[i]->Cells[j]->Value = gcnew System::String(cl[i].get\_name().c\_str());// Convert::ToString(cl[i].());

dataGridView1->Rows[i]->Cells[j + 1]->Value = Convert::ToString(cl[i].get\_age());

dataGridView1->Rows[i]->Cells[j + 2]->Value = gcnew System::String(cl[i].spesialist.get\_name().c\_str());

dataGridView1->Rows[i]->Cells[j + 3]->Value = gcnew System::String(cl[i].spesialist.get\_name\_posluga().c\_str());

dataGridView1->Rows[i]->Cells[j + 4]->Value = Convert::ToString(cl[i].spesialist.get\_price());

}

}

dataGridView1->AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders);

dataGridView1->AutoResizeColumns();

}

else {

dataGridView1->ColumnCount = 5;

dataGridView1->RowCount = 1;

for (int i = 0; i < 1; i++) {

for (int j = 0; j < 5; j++) {

//dataGridView1->Rows[i]->Cells[j]->Value = "5";

dataGridView1->Rows[i]->Cells[j]->Value = " ";

}

}

}

return System::Void();

}

System::Void rgz::MyForm::button2\_Click(System::Object^ sender, System::EventArgs^ e)//додати клієнта

{

MyForm3^ form3 = gcnew MyForm3();

form3->Show();

return System::Void();

}

System::Void rgz::MyForm3::button1\_Click(System::Object^ sender, System::EventArgs^ e)

{

std::string name\_sp[5] = { "Іванова Іванна Іванівна","Крикун Мар'яна Миколаївна","Барбір Анна Володимирівна","Ребрик Лідія Василівна","Кравець Юрій Васильювич" };

//std::string name\_cl[5] = { "Прокопенко Інна Володимирівна","Стершенко Оксана Іллівна","Соловейко Ганна Іванівна","Балабій Ірма Станіславівна","Агаєв Олег Миколайович" };

std::string spe[5] = { "масажист","косметолог","дерматолог","хірург","терапевт-мануальщик" };

std::string posluga[5] = { "корекція фігури","чистка обличча","видалення папілом","мамопластика","голкова терапія" };

int stag[5] = { 10,5,3,2,1 };

int prise[5] = { 2500,650,550,4500,1200 };

System::String^ name\_persone = textBox1->Text;

std::string name\_per = msclr::interop::marshal\_as<std::string>(name\_persone);

int number = System::Convert::ToInt32(textBox2->Text);

count++;

Client\* cl1 = new Client[count];

for (int i = 0; i < count-1; i++) {

cl1[i].set\_age(cl[i].get\_age());

cl1[i].set\_name(cl[i].get\_name());

cl1[i].spesialist.set\_age(cl[i].spesialist.get\_age());

cl1[i].spesialist.set\_name(cl[i].spesialist.get\_name());

cl1[i].spesialist.set\_name\_price(cl[i].spesialist.get\_name\_posluga());

cl1[i].spesialist.set\_price(cl[i].spesialist.get\_price());

cl1[i].spesialist.set\_stag(cl[i].spesialist.get\_stag());

}

cl1[count - 1].set\_name(name\_per);

cl1[count-1].set\_age(number);

if (radioButton1->Checked == true) {

cl1[count-1].spesialist.set\_name(name\_sp[0]);

cl1[count-1].spesialist.set\_stag(stag[0]);

cl1[count-1].spesialist.set\_price(prise[0]);

cl1[count-1].spesialist.set\_name\_price(posluga[0]);

cl1[count-1].spesialist.set\_stag(stag[0]);

}

else if (radioButton2->Checked == true) {

cl1[count - 1].spesialist.set\_name(name\_sp[1]);

cl1[count - 1].spesialist.set\_stag(stag[1]);

cl1[count - 1].spesialist.set\_price(prise[1]);

cl1[count - 1].spesialist.set\_name\_price(posluga[1]);

cl1[count - 1].spesialist.set\_stag(stag[1]);

}

else if (radioButton3->Checked == true) {

cl1[count - 1].spesialist.set\_name(name\_sp[2]);

cl1[count - 1].spesialist.set\_stag(stag[2]);

cl1[count - 1].spesialist.set\_price(prise[2]);

cl1[count - 1].spesialist.set\_name\_price(posluga[2]);

cl1[count - 1].spesialist.set\_stag(stag[2]);

}

else if (radioButton4->Checked == true) {

cl1[count - 1].spesialist.set\_name(name\_sp[3]);

cl1[count - 1].spesialist.set\_stag(stag[3]);

cl1[count - 1].spesialist.set\_price(prise[3]);

cl1[count - 1].spesialist.set\_name\_price(posluga[3]);

cl1[count - 1].spesialist.set\_stag(stag[3]);

}

else {

cl1[count - 1].spesialist.set\_name(name\_sp[4]);

cl1[count - 1].spesialist.set\_stag(stag[4]);

cl1[count - 1].spesialist.set\_price(prise[4]);

cl1[count - 1].spesialist.set\_name\_price(posluga[4]);

cl1[count - 1].spesialist.set\_stag(stag[4]);

}

delete[] cl;

//cl = nullptr;

cl = new Client[count];

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

cl[i].set\_age(cl1[i].get\_age());

cl[i].set\_name(cl1[i].get\_name());

cl[i].spesialist.set\_age(cl1[i].spesialist.get\_age());

cl[i].spesialist.set\_name(cl1[i].spesialist.get\_name());

cl[i].spesialist.set\_name\_price(cl1[i].spesialist.get\_name\_posluga());

cl[i].spesialist.set\_price(cl1[i].spesialist.get\_price());

}

delete[]cl1;

MyForm3::Hide();

MessageBox::Show("Добавлення відбулося успішно", "Повідомлення");

return System::Void();

}