**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ДВНЗ «УНІВЕРСИТЕТ БАНКІВСЬКОЇ СПРАВИ»**

**ІНСТИТУТ БАНКІВСЬКИХ ТЕХНОЛОГІЙ ТА БІЗНЕСУ**

**КАФЕДРА КІБЕРБЕЗПЕКИ ТА СОЦІАЛЬНИХ НАУК**

**КУРСОВА РОБОТА**

з дисципліни

**«ОБ’ЄКТНО-ОРІЄНТОВНЕ ПРОГРАМУВАННЯ»**

(назва дисципліни)

на тему:**дОВІДНИК ТУРИСТА**

Студента(ки) 2 курсу 203 групи

спеціальності Кібербезпека»

Шевченко В.С

(прізвище та ініціали)

Керівник

к.т.н, доцент, завідуючий кафедри

(посада, вчене звання, науковий ступінь,

Гордєєв О.О.

прізвище та ініціали)

Національна шкала

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_

Оцінка: ECTS \_\_\_\_

Члени комісії

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

Київ - 2019 рік

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ДВНЗ «УНІВЕРСИТЕТ БАНКІВСЬКОЇ СПРАВИ»**

**ІНСТИТУТ БАНКІВСЬКИХ ТЕХНОЛОГІЙ ТА БІЗНЕСУ**

**КАФЕДРА КІБЕРБЕЗПЕКИ ТА СОЦІАЛЬНИХ НАУК**

Спецiальність: «Кібербезпека»

Курс 2 Група 203 - Кб Семестр 3

**ЗАВДАННЯ**

**на курсову роботу студента/студентки**

Шевченко Вероніки Сергіївни

(прiзвище, iм`я, по батьковi)

1. Тема курсової роботи:

Довідник туриста

2. Термiн здачi студентом закiнченої роботи 19.12.2019

3. Вихiднi данi до (роботи): інформаційно-аналітичні інтернет джерела, зразки

рішень практичних завдань та задач, нормативно-правова база, щодо оформлен-

ня технічної документації програмних продуктів, початкові значення для об-

робки.

4. Змiст пояснювальної записки (перелiк питань, які належить розробити)

Вступ. Формулювання вимог до програми. Проектування та практична

реалізація розв’язку поставленої задачі. Тестування програми і результати

її виконання. Висновки.

5. Перелiк графiчного матерiалу (з точним зазначенням обов`язкових креслень)

Таблиці та схеми представлено в роботі.

6. Дата видачі завдання “ 10 ” жовтня 2019 р

зміст

[Вступ 4](#_Toc28256055)

[Розділ 1. теоретичні аспекти й теорії об’єктно-орієнтованого програмування 6](#_Toc28256056)

[1.1. Аналіз сучасного стану технологій та обґрунтування теми 6](#_Toc28256057)

[1.2. Використання функцій при розробці програм 8](#_Toc28256058)

[1.3. Використання класів при розробці програм 9](#_Toc28256059)

[Розділ 2. Практична реалізація розв’язку поставлених задач. 11](#_Toc28256060)

[2.1. Оцінка коректності алгоритму та аналіз алгоритму на часову та просторову ефективності 11](#_Toc28256064)

[2.2. Опис програми (обґрунтування вибору мови програмування, опис основних процедур та функцій) 12](#_Toc28256065)

[2.3. Опис алгоритмів та кодів. 13](#_Toc28256066)

[Розділ 3. Тестування програми і результати її виконання 15](#_Toc28256067)

[3.1. Опис контрольних прикладів 15](#_Toc28256070)

[Висновки 17](#_Toc28256071)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 19](#_Toc28256072)

[Додатки 20](#_Toc28256073)

# Вступ

Мова програмування Сi була розроблена в 1972 р. Деннисом Рiтчi, який є одним з авторів операцiйноi системи UNIX. Мова програмування Сi пізніше була використана для програмування цiеi системи, а також багатої бібліотеки обслуговуючих програм, оскільки, являлася універсальною мовою загального призначення, мова Сi зручна для програмування системних задач. Поява мікрокомп’ютерів закріпила позицii мови Сi. Було створено близько 30 його нових компіляторів, а після проведення Американським національним інститутом стандартів (American National Standards Institute) робiт по стандартизацii в області програмування почали розроблятися компілятори, що вiдповiдали опублікованому весною 1986 р. проекту стандарту.

Першим компілятором по стандарту ANSI являлась система Турбо Сi версii 1.0 фiрми Borland International. Ця система, що складається з компілятора мови Сi, пов'язаного з ним редактора, компоновщика i бібліотек, забезпечила користувачам зручну інтегровану операційну оболонку, а також суттєво полегшувала професійне програмування, в якому визначальними параметрами є висока швидкість компiляцii, висока якість згенерованого коду та невеликий об’єм оперативної пам’ять. Мова С серед сучасних мов програмування є однією iз найбільш поширених. Вона універсальна, але найбільш ефективне її використання в задачах системного програмування - розробки трансляторів, операційних систем, інструментальних засобів. Мова С добре зарекомендувала себе ефективністю, лаконічністю запису алгоритмів, логічною стрункістю програм. У багатьох випадках програми, написані на мові С, можна порівняти по швидкості з програмами, написаними на мові Асемблера, при цьому вони більш наглядні i прості у супроводі.

Основними перевагами мови С вважаться висока переча написаних на нiй програм мiж комп'ютерами з різною архітектурою, мiж різними операційними середовищами. Транслятори мови С дiять практично на всiх персональних комп'ютерах, які використовуються в наш час. Перечислимо окремі особливості мови С:

-в мові С реалізовані окремі операцii низького рівня (а саме операцii над бітами). Окремі з таких операцій напряму вiдповiдають машинним командам;

- мова С підтримує механізм показчикiв на змiннi i функцii;

-базовi типи даних мови С відображають тi ж об’єкти, з якими потрібно мати справу програмі на мові Асемблера,- байти, машинні слова, символи, строки;

- як в нiякiй iншiй мові програмування високого рівня в мові С накладаються лише незначні обмеження на перетворення типів даних;

- не дивлячись на ефективність і потужність конструкцii мови С, він відносно малий по об'єму, але в системне оточення мови С входить бібліотека стандартних функцій, в який реалізовані встроенi оператори введення - виведення, динамічного розподілу пам’ять, управління процесами i т.д.

# розділ 1. теоретичні аспекти й теорії об’єктно-орієнтованого програмування

## Аналіз сучасного стану технологій та обґрунтування теми

У 80-х роках мова С++ стала одною з найбільш розповсюджених та універсальних мов програмування. З допомогою цієї мови можна створювати програми для широкого класу комп'ютерів. В 70-х роках серед створювачів мов програмування популярною стала концепція об'єкту. Об'єкт є сукупністю коду і даних, що створена для відтворення властивостей фізичних предметів чи абстрактних понять.

Об'єктно-орiєнтоване програмування - результат природної еволюції більш ранніх методологій програмування. Воно виникло з процедурного програмування. В об'єктно-орiєнтованому програмуванні ви відходите від ділення задач на під задачі. Ви намагаєтесь побачити вашу задачу, утворену з взаємодій між абстракціями – ідеалізованими об'єктами реального світу. Таким чином, ціль у тому, щоб використовувати в програмуванні повсякденний досвід і знання про поведінку звичайних об'єктів. Об’єкт – це абстракція, але з чітко визначеними властивостями чи ролями. Об'єктно-орiєнтоване програмування є методом програмування, який імітує виконання людиною якої-небудь роботи. Воно більш структуроване i більш модульне i абстрактне, ніж традиційне програмування.

За винятком другорядних деталей C++ є надмножиною мови програмування C. Крім можливостей, що дає C, C++ надає гнучкі й ефективні засоби визначення нових типів. Використовуючи визначення нових типів, програміст може розділяти розроблювальну програму на піддающіся легкому контролю частини. Такий метод побудови програм часто називають абстракцією даних. Інформація про типи утримується в деяких об'єктах типів, визначених користувачем. Такі об'єкти прості й надійні у використанні в тих ситуаціях, коли їхній тип не можна встановити на стадії компіляції. Програмування з застосуванням таких об'єктів часто називають об'єктно-орієнтованим. При правильному використанні цей метод дає більш короткі й зрозуміліші контрольовані програми.

Ключовим поняттям в С++ є клас. Клас - це тип, визначений користувачем. Класи забезпечують захист (закриття) даних, гарантовану ініціалізацію даних, не явне перетворення типів для типів, визначених користувачем, динамічне завдання типу, контрольоване користувачем керування пам'яттю і механізми перевантаження операцій. C++ надає набагато кращі, чим у C, засоби вираження модульності програми і перевірки типів. У мові є також удосконалення, не пов'язані безпосередньо з класами, що включають у себе символічні константи, inline- підстановку функцій, параметри функції по замовчуванню, перевантажені імена функцій, операції керування вільною пам'яттю. У C++ збережені можливості мови C по роботі з основними об'єктами апаратного забезпечення (біти, байти, слова, адреси і т.п.). Це дозволяє дуже ефективно реалізовувати типи, обумовлені користувачем.

Об’єктно-орієнтовне програмування – методологія, яка концентрується більше на зв'язках між об'єктами, ніж на деталях реалізації. Дані зв'язку зазвичай розвиваються по принципу дерев, при цьому нові типи об’єктів утворюються із вже існуючих.

В С++ деякі об’єкти є частиною самої мови. Але в цілому задача створення нових об’єктів полягає на користувача. В Borland C++ є велика кількість типів об’єктів, але реальне використання мови потребує утворення нових типів. Ефективність ООП проявляється тільки при утворенні і застосуванні груп зв'язаних між собою об'єктів. Такі групи зазвичай називають ієрархіями класів. Розвиток цих ієрархій класів є основою в діяльності ООП.

C++ і його стандартні бібліотеки спроектовані так, щоб забезпечувати мобільність. Наявна на сучасний момент реалізація мови буде йти в більшості систем, що підтримують C. З C++ програм можна використовувати C бібліотеки, і з C++ можна використовувати велику частину інструментальних засобів, що підтримують програмування на C.

Всі мови об’єктна орієнтованого програмування (ООП), включаючи С++, основані на трьох основних концепціях, які називаються: інкапсуляція, поліморфізм і наслідування.

## Використання функцій при розробці програм

До цих пір ми писали програми єдиним, функціонально неподільним, кодом. Алгоритм програми знаходився в головній функції, причому інших функцій в програмі не було. Ми писали маленькі програми, тому не було потреби в оголошенні своїх функцій. Для написання великих програм, досвід показує, що краще користуватися функціями. Програма складатиметься з окремих фрагментів коду, під окремим фрагментом коду розуміється функція. Окремим, тому, що робота окремої функції не залежить від роботи якої-небудь іншої. Тобто алгоритм в кожної функції функціонально достатній і не залежить від інших алгоритмів програми. Одного разу написавши функцію, її можна буде з легкістю переносити в інші програми. Функція (в програмуванні) – це фрагмент коду або алгоритм, реалізований на якійсь мові програмування, з метою виконання певної послідовності операцій. Отже, функції дозволяють зробити програму модульною, тобто розділити програму на кілька маленьких підпрограм (функцій), які в сукупності виконують поставлене завдання. Ще один величезний плюс функцій в тому, що їх можна багаторазово використовувати. Дана можливість дозволяє багаторазово використовувати один раз написаний код, що в свою чергу, набагато скорочує обсяг коду програми!

Крім того, що в С передбачено оголошення своїх функцій, також можна скористатися функціями визначеними в стандартних заголовних файлах мови програмування С++. Щоб скористатися функцією, визначеної в заголовному файлі, потрібно його підключити. Наприклад, щоб скористатися функцією, яка зводить деяке число в степінь, потрібно підключити заголовний файл <cmath> і в запустити функцію pow() в тілі програми. Розробимо програму, в якій запустимо функцію pow().(Рис.1.1)

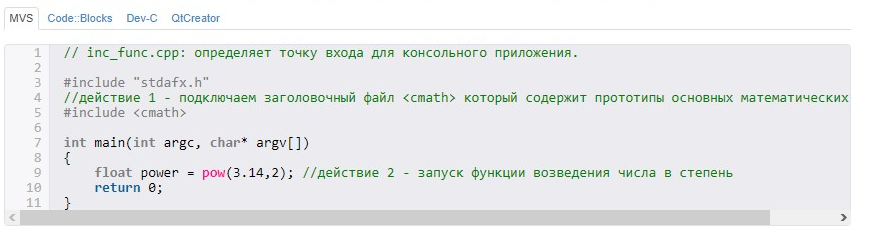


Рис. 1.1 Використання функції та бібліотек

## Використання класів при розробці програм

Класи і об'єкти в С є основними концепціями об'єктно-орієнтованого програмування – ООП. Об'єктно-орієнтоване програмування – розширення структурного програмування, в якому основними концепціями є поняття класів і об'єктів. Основна відмінність мови програмування С++ від С полягає в тому, що в С немає класів, а отже мова С не підтримує ООП, на відміну від С++ .

Щоб зрозуміти, для чого ж насправді потрібні класи, проведемо аналогію з яким-небудь об'єктом з повсякденного життя, наприклад, з велосипедом. Велосипед – це об'єкт, який був побудований згідно з кресленнями. Так ось, ці самі креслення грають роль класів в ООП. Таким чином класи – це деякі описи, схеми, креслення по яких створюються об'єкти. Тепер зрозуміло, що для створення об'єкта в ООП необхідно спочатку скласти креслення, тобто класи. Класи мають свої функції, які називаються методами класу. Пересування велосипеда здійснюється за рахунок обертання педалей, якщо розглядати велосипед з точки зору ООП, то механізм обертання педалей – це метод класу. Кожен велосипед має свій колір, вага, різні складові – все це властивості. Причому у кожного створеного об'єкта властивості можуть відрізнятися. Маючи один клас, можна створити необмежено кількість об'єктів (велосипедів), кожен з яких буде володіти однаковим набором методів, при цьому можна не замислюватися про внутрішню реалізації механізму обертання педалей, коліс, спрацьовування системи гальмування, так як все це вже буде визначено в класі. Розібравшись з призначенням класу, дамо йому грамотне визначення.

Класи в С++ – це абстракція, що описує методи, властивості, ще не існуючих об'єктів. Об'єкти – конкретне уявлення абстракції, що має свої властивості і методи. Створені об'єкти на основі одного класу називаються екземплярами цього класу. Ці об'єкти можуть мати різну поведінку, властивості, але все одно будуть об'єктами одного класу. В ООП існує три основних принципи побудови класів:

Інкапсуляція – це властивість, що дозволяє об'єднати в класі і дані, і методи, які працюють з ними і приховати деталі реалізації від користувача.

Спадкування – це властивість, що дозволяє створити новий клас-нащадок на основі вже існуючого, при цьому всі характеристики класу батька присвоюються класу-нащадка.

Поліморфізм– властивість класів, що дозволяє використовувати об'єкти класів з однаковим інтерфейсом без інформації про тип і внутрішню структуру об'єкта.

# Розділ 2. Практична реалізація розв’язку поставлених задач.



## Оцінка коректності алгоритму та аналіз алгоритму на часову та просторову ефективності

Складність алгоритму — це кількісна оцінка ресурсів, що витрачаються алгоритмом. В якості ресурсів можна розглядати людські (на створення і розуміння алгоритму) і обчислювальні (на виконання алгоритму) ресурси. Тому розрізняють описову (інтелектуальну) і обчислювальну складність алгоритму.

Описова складність визначається виходячи з самого запису (тексту) алгоритму. Єдиного критерію її оцінки не існує. Зазвичай, описова складність характеризується довжиною запису алгоритму

Як обчислювальні ресурси для виконання алгоритму розглядають пам'ять і процесорний час. Тому основними мірами обчислювальної складності алгоритмів є:

ємнісна (просторова) складність, яка характеризує кількість пам'яті, необхідної для виконання алгоритму;

часова складність, яка характеризує кількість часу, необхідного на виконання алгоритму (цей час, як правило, визначається кількістю елементарних операцій, які потрібно виконати для реалізації алгоритму).

Як відомо, комп'ютери мають обмежений обсяг пам'яті. Якщо дві програми реалізують ідентичні функції, то та, яка використовує менший обсяг пам'яті, характеризується меншою ємнісною складністю. Іноді пам'ять є домінуючим чинником в оцінці ефективності програм. Однак в останні роки у зв'язку із швидким її здешевленням ця складова ефективності поступово втрачає своє значення.

Для оцінки часової складності можна просто запустити кожен алгоритм на декількох задачах і порівняти час виконання. Інший спосіб - математично оцінити час виконання підрахунком операцій.

У цьому випадку складність алгоритму (і часова, і ємнісна) описується функцією f(n), де n - розмір вхідних даних (розмірність оброблюваного масиву, кількість слів в тексті, що аналізується, довжина послідовності в алгоритмі, число вершин оброблюваного графа тощо).

Знаходження точної залежності f(n) для конкретного алгоритму - задача досить складна. Найчастіше така докладна оцінка і не потрібна. З цієї причини зазвичай обмежуються лише асимптотичними оцінками швидкості росту цієї функції, які описують граничну поведінку складності алгоритму при збільшенні n (наскільки швидко або повільно зростає ця функція).

Загалом використовується декілька підходів, які виражають так званий порядок складності алгоритму. В їх основі лежить порівняння функції f(n) з якою-небудь функцією, чия поведінка добре досліджена.

## Опис програми (обґрунтування вибору мови програмування, опис основних процедур та функцій)

Для написання програм використовується С++. Це універсальна мова програмування, так як компілятори С++ є у кожній операційній системі. Також перевагою є висока швидкість виконання коду. Будь-який тест з продуктивності мов програмування покаже, що С++ стоїть на лідируючому місці. Java, JavaScript, C#, як і безліч інших популярних мов програмування, містять в основі принципи C++. Тому учбові програми багатьох ВНЗ містять курс С++.

Потрібно розробити довідник туриста, який допоміг би обрати найбільш привабливий для користувача маршрут подорожі. Програма включає в себе вибір агенції та запропонованої пропозиції у вигляді: країни, міста або круїзу, умов проживання, екскурсійного супроводження, сервісу прийнятого боку та вартості поїздки.

## Опис алгоритмів та кодів.

Блок-схема (Рис. 2.1)

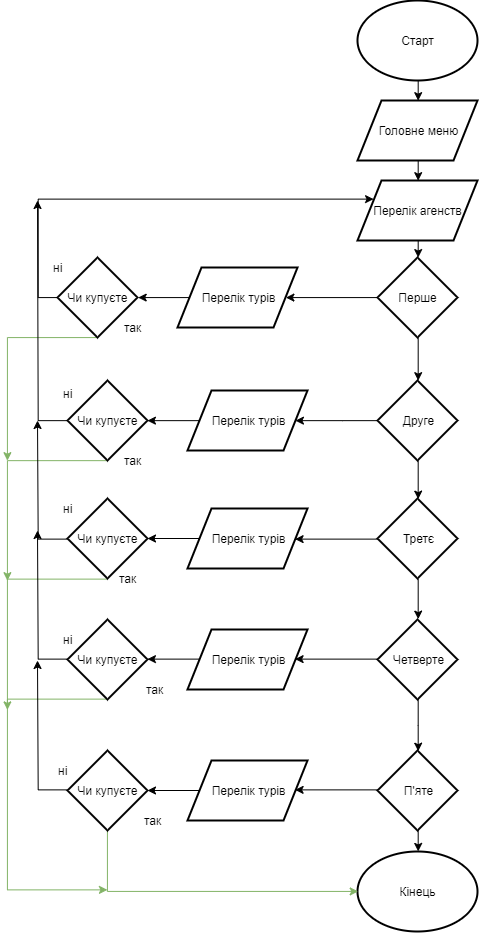


Рис. 2.1 блок-схема розроблена у онлайн сервісі

Для початку ми створюємо абстрактний клас для агенції і в кожному окремому класі кожного агенства присвоюємо їм значення, методи та підкласи у вигляді турів. Наприклад, метод виводу, метод обрання та купівлі білетів. До полів відносяться назви агенцій, країни, ціна та тури. У тілі програми прописано вивід головного меню, де ми можемо спочатку подивитися перелік агенцій, а після обрання однієї з них перелік всіх турів і через switch-case придбання білетів, після чого визивається мето виводу, або повернення у головне меню.

# Розділ 3. Тестування програми і результати її виконання

## Опис контрольних прикладів

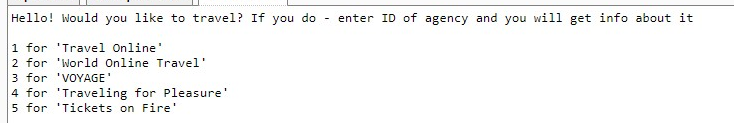
При запуску програми користувач бачить меню з вибором агенств (Рис.3.1).

Рис. 3.1 Перелік агенцій

Після обрання закладу можна побачити путівки, які нам пропонує заклад (Рис.3.2).

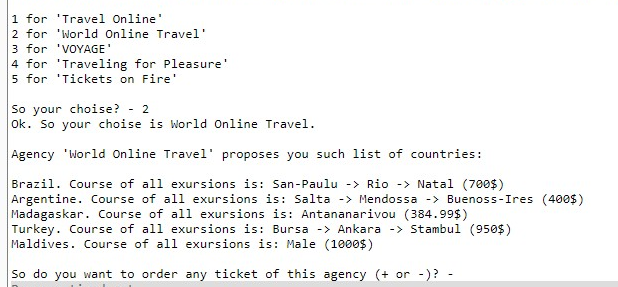


Рис. 3.2 Перелік путівок

Після перегляду усіх пропозицій програма запитує чи хоче користувач придбати один з білетів. Якщо пропозиція задовольняє, то потрібно ввести номер туру, та система автоматично придбає квиток, так виведе ціну покупки (Рис.3.3), якщо ж ні, то користувач повертається у головне меню разом із переліком тур-агенцій.

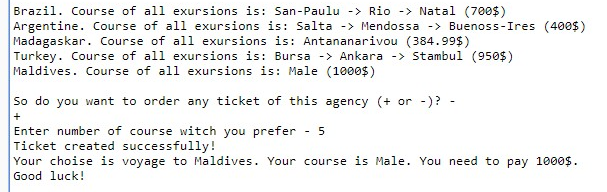


Рис. 3.3 Придбання білету

# Висновки

В даній курсовій роботі була розроблена програма для роботи з класами та структурами, а також різними функціями. Для досягнення поставленої практичної цілі було розглянута такі теми:

* Цикли;
* Функції;
* Класи, робота з класами;
* Покажчики;
* Умови перевірки введених даних;

До недоліків програми можна віднести:

* Робота лише у консольному режимі;
* Малий функціонал;
* Неможливість обрання кількості білетів;

Зазначені зауваження не є вагомими, тому можна вважати, що поставлені задачі є виконаними.

При виконанні курсової роботи, були удосконаленні навички написання програм мовою С++ у програмному середовищі Visual Studio 2017 та онлайн компіляторі C++ SHELL.

Об'єктно-орієнтоване програмування виявилося універсальним видом програмування та проектування.

Об'єктно-орієнтоване програмування - це метод програмування, заснований на поданні програми у вигляді сукупності взаємодіючих об'єктів, кожен з яких є екземпляром певного класу, а класи є членами певної ієрархії наслідування. Програмісти спочатку пишуть клас, а на його основі при виконанні програми створюються конкретні об'єкти (екземпляри класів). На основі класів можна створювати нові, які розширюють базовий клас і таким чином створюється ієрархія класів.

При такому програмуванні все є об'єктами програмування. Всі вони мають деякі свої дані, які мають вигляд посилань на інші об'єкти, ще вони можуть обмінюватися повідомленнями між об'єктами. Коли об'єкт отримує повідомлення, запускається відповідний обробник - метод, для кожного об'єкта є свій асоціативний контейнер, завдяки якому підбирається метод його обробки. Крім цього, у кожного об'єкта є свій предок, якщо метод для обробки повідомлення не знайдено, то це повідомлення буде відправлено предку. Вся така структура виділяється в окремий об'єкт, який називається класом конкретного об'єкта. Вище існує метакласом, який об'єднує класи як об'єкти, певної поведінки.

У наш час вже не можна уявити своє життя без технологій та програм. Саме тому людство винайшло найпростіший та найбільш універсальний спосіб програмування – ООП.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Оцінка ефективності алгоритму [Електронний ресурс] // 2015 – Режим доступу до ресурсу: https://studfile.net/preview/3740818/page:2/;
2. Програмування мовою С++ з використанням об’єктно-орієнтованого програмування [Електронний ресурс] – Режим доступу до ресурсу: https://www.bestreferat.ru/referat-141723.html.
3. Класи в С++ [Електронний ресурс] – Режим доступу до ресурсу: http://cppstudio.com/uk/post/439/.
4. Функціїї в С++ [Електронний ресурс] – Режим доступу до ресурсу: http://cppstudio.com/uk/post/396/.
5. Об'єктно-орієнтоване програмування [Електронний ресурс] – Режим доступу до ресурсу: http://programming.in.ua/programming/basisprogramming/25-oop.html.
6. Використання класів в С++ [Електронний ресурс] – Режим доступу до ресурсу: http://cpp.dp.ua/vykorystannya-klasiv/.
7. C++. Класи. Частина 1. Поняття класу. Оголошення класу. Об’єкт класу. [Електронний ресурс] – Режим доступу до ресурсу: https://www.bestprog.net/uk/2018/03/25/c-classes-part-1-the-concept-of-class-declaring-a-data-type-class-the-object-of-the-class-classes-in-the-clr-encapsulation-of-data-in-a-class\_ua/.
8. Основні поняття об'єктно-орієнтованого програмування: інкапсуляція, успадкування і поліморфізм [Електронний ресурс] – Режим доступу до ресурсу: https://studfile.net/preview/5851327/page:11/.
9. Програмування - основа сучасного світу - SEO [Електронний ресурс] – Режим доступу до ресурсу: https://replace.org.ua/topic/194/.
10. Технологія об'єктно-орієнтованого програмування [Електронний ресурс] – Режим доступу до ресурсу: lib.mdpu.org.ua › e-book › vstup.

# Додатки

**Додаток А.** Код програми

#include <iostream>

#include <string>

#include <string.h>

using namespace std;

// Абстрактный класс агентства

class agencie

{

public:

virtual string get\_name() = 0;

virtual void create\_ticket(int id) = 0;

virtual void print\_list() = 0;

private:

string name;

char countries[5][20];

char course\_exursions[5][50];

float prices\_$[5];

};

// Классы-наследники от абстрактного класса агентств

class travel\_online : public agencie

{

public:

// =========МЕТОДЫ=========

string get\_name() override // Геттер имени агенстсва

{

return this->name;

}

void print\_list() override // Метод вывода предлагаемых агентством услуг

{

cout << "Agency '" << this->name << "' proposes you such list of countries:" << endl << endl;

for (int i = 0; i < 5; i++)

cout << this->countries[i] << ". Course of all exursions is: " << this->course\_exursions[i] << " (" << this->prices\_$[i] << "$)" << endl;

}

void create\_ticket(int id) override // Метод создания билета

{

cout << "Ticket created successfully!" << endl;

cout << "Your choise is voyage to " << this->countries[id - 1] << ". Your course is " << this->course\_exursions[id - 1] << ". You need to pay " << this->prices\_$[id - 1] << "$." << endl;

cout << "Good luck!" << endl;

}

private:

string name = "Travel Online"; // Название агентства

char countries[5][15] = { "Malasia", "New Zeland", "Philipines", "Australia", "Southern Corea" }; // Список предлагаемых агентсвом стран

char course\_exursions[5][44] = // Маршруты

{

"Kuala-Lumpur -> Malakka -> Kuala-Trenghanou",

"Oakland -> Rororoua -> Nelson",

"Sebu -> Davao -> Manila",

"Sydney",

"Degu -> Busan -> Seul"

};

float prices\_$[5] = { 500., 650., 550., 900., 524.99 }; // Цены

};

class world\_online\_travel : public agencie

{

public:

string get\_name() override

{

return this->name;

}

void print\_list() override

{

cout << "Agency '" << this->name << "' proposes you such list of countries:" << endl << endl;

for (int i = 0; i < 5; i++)

cout << this->countries[i] << ". Course of all exursions is: " << this->course\_exursions[i] << " (" << this->prices\_$[i] << "$)" << endl;

}

void create\_ticket(int id) override

{

cout << "Ticket created successfully!" << endl;

cout << "Your choise is voyage to " << this->countries[id - 1] << ". Your course is " << this->course\_exursions[id - 1] << ". You need to pay " << this->prices\_$[id - 1] << "$." << endl;

cout << "Good luck!" << endl;

}

private:

string name = "World Online Travel";

char countries[5][15] = { "Brazil", "Argentine", "Madagaskar", "Turkey", "Maldives" };

char course\_exursions[5][44] =

{

"San-Paulu -> Rio -> Natal",

"Salta -> Mendossa -> Buenoss-Ires",

"Antananarivou",

"Bursa -> Ankara -> Stambul",

"Male"

};

float prices\_$[5] = { 700., 400., 384.99, 950., 1000 };

};

class VOYAGE : public agencie

{

public:

string get\_name() override

{

return this->name;

}

void print\_list() override

{

cout << "Agency '" << this->name << "' proposes you such list of countries:" << endl << endl;

for (int i = 0; i < 5; i++)

cout << this->countries[i] << ". Course of all exursions is: " << this->course\_exursions[i] << " (" << this->prices\_$[i] << "$)" << endl;

}

void create\_ticket(int id) override

{

cout << "Ticket created successfully!" << endl;

cout << "Your choise is voyage to " << this->countries[id - 1] << ". Your course is " << this->course\_exursions[id - 1] << ". You need to pay " << this->prices\_$[id - 1] << "$." << endl;

cout << "Good luck!" << endl;

}

private:

string name = "VOYAGE";

char countries[5][15] = { "Austria", "France", "Poland", "Germany", "England" };

char course\_exursions[5][44] =

{

"Insbrouk -> Vienna -> Baden",

"Paris -> Marcel -> Renn",

"Gdansk -> Krakow -> Warsava",

"Hamburg -> Colougne -> Berlin",

"Liverpool -> Manchester -> London"

};

float prices\_$[5] = { 650., 750., 350., 700., 755. };

};

class travel\_for\_pleasure : public agencie

{

public:

string get\_name() override

{

return this->name;

}

void print\_list() override

{

cout << "Agency '" << this->name << "' proposes you such list of countries:" << endl << endl;

for (int i = 0; i < 5; i++)

cout << this->countries[i] << ". Course of all exursions is: " << this->course\_exursions[i] << " (" << this->prices\_$[i] << "$)" << endl;

}

void create\_ticket(int id) override

{

cout << "Ticket created successfully!" << endl;

cout << "Your choise is voyage to " << this->countries[id - 1] << ". Your course is " << this->course\_exursions[id - 1] << ". You need to pay " << this->prices\_$[id - 1] << "$." << endl;

cout << "Good luck!" << endl;

}

private:

string name = "Travel for Pleasure";

char countries[5][15] = { "USA", "Canada", "Mexico", "Carribes", "Brazil" };

char course\_exursions[5][44] =

{

"New-York -> Houston -> Washington",

"Quebec -> Torronto -> Ottawa",

"Kankunn -> Mexico -> Merrida",

"Antigua -> Great Kaiman -> Ghaitti",

"San-Paulu -> Rio -> Natal"

};

float prices\_$[5] = { 990., 1100., 900., 1000., 890. };

};

class tickets\_on\_fire : public agencie

{

public:

string get\_name() override

{

return this->name;

}

void print\_list() override

{

cout << "Agency '" << this->name << "' proposes you such list of countries:" << endl << endl;

for (int i = 0; i < 5; i++)

cout << this->countries[i] << ". Course of all exursions is: " << this->course\_exursions[i] << " (" << this->prices\_$[i] << "$)" << endl;

}

void create\_ticket(int id) override

{

cout << "Ticket created successfully!" << endl;

cout << "Your choise is voyage to " << this->countries[id - 1] << ". Your course is " << this->course\_exursions[id - 1] << ". You need to pay " << this->prices\_$[id - 1] << "$." << endl;

cout << "Good luck!" << endl;

}

private:

string name = "Tickets on Fire";

char countries[5][15] = { "Turkey", "Germany", "Mexico", "Egypt", "Thailand" };

char course\_exursions[5][44] =

{

"Bursa -> Ankara -> Stambul",

"Hamburg -> Colougne -> Berlin",

"Kankunn -> Mexico -> Merrida",

"Alexandria -> Sharm-el-Sheikh -> Khair",

"Chiangmay -> Pattaia -> Bankgouk"

};

float prices\_$[5] = { 990., 1100., 900., 1000., 890. };

};

int main()

{

int id, id1;

char choise;

point:

// =========ГЛАВНОЕ МЕНЮ=========

cout << "Hello! Would you like to travel? If you do - enter ID of agency and you will get info about it" << endl << endl;

cout << "1 for 'Travel Online'" << endl;

cout << "2 for 'World Online Travel'" << endl;

cout << "3 for 'VOYAGE'" << endl;

cout << "4 for 'Traveling for Pleasure'" << endl;

cout << "5 for 'Tickets on Fire'" << endl << endl;

cout << "So your choise? - ";

cin >> id;

// ==============================

system("cls"); // Функция очищения консоли

point1:

switch (id)

{

case 1:

{

travel\_online ticket; // Объявление переменной типа выбраного агентства (зависит от введенного нами ID)

cout << "Ok. So your choise is " << ticket.get\_name() << "." << endl << endl;

ticket.print\_list();

cout << endl;

cout << "So do you want to order any ticket of this agency (+ or -)? - "; // Желаем ли мы купить у этого агентства билет

cin >> choise;

if (choise == '+')

{

cout << "Enter number of course witch you prefer - "; // Выбор одного из курсов

cin >> id1;

ticket.create\_ticket(id1);

break;

}

else

if (choise == '-') // Если не желаем, то вернуться на главное меню

{

system("cls");

goto point;

break;}

else // Елси же команда введена не корректно

{

system("cls");

cout << "Incorrect command! Try againg!" << endl;

goto point1;

break; }

}

case 2:

{

world\_online\_travel ticket;

cout << "Ok. So your choise is " << ticket.get\_name() << "." << endl << endl;

ticket.print\_list();

cout << endl;

cout << "So do you want to order any ticket of this agency (+ or -)? - ";

cin >> choise;

if (choise == '+')

{

cout << "Enter number of course witch you prefer - ";

cin >> id1;

ticket.create\_ticket(id1);

break;

}

else

if (choise == '-')

{

system("cls");

goto point;

break;

}

else

{

system("cls");

cout << "Incorrect command! Try againg!" << endl;

goto point1;

break;

}

}

case 3:

{

VOYAGE ticket;

cout << "Ok. So your choise is " << ticket.get\_name() << "." << endl << endl;

ticket.print\_list();

cout << endl;

cout << "So do you want to order any ticket of this agency (+ or -)? - ";

cin >> choise;

if (choise == '+')

{

cout << "Enter number of course witch you prefer - ";

cin >> id1;

ticket.create\_ticket(id1);

break;

}

else

if (choise == '-')

{

system("cls");

goto point;

break;

}

else

{

system("cls");

cout << "Incorrect command! Try againg!" << endl;

goto point1;

break;

}

}

case 4:

{

travel\_for\_pleasure ticket;

cout << "Ok. So your choise is " << ticket.get\_name() << "." << endl << endl;

ticket.print\_list();

cout << endl;

cout << "So do you want to order any ticket of this agency (+ or -)? - ";

cin >> choise;

if (choise == '+')

{

cout << "Enter number of course witch you prefer - ";

cin >> id1;

ticket.create\_ticket(id1);

break;

}

else

if (choise == '-')

{

system("cls");

goto point;

break;

}

else

{

system("cls");

cout << "Incorrect command! Try againg!" << endl;

goto point1;

break;

}

}

case 5:

{

tickets\_on\_fire ticket;

cout << "Ok. So your choise is " << ticket.get\_name() << "." << endl << endl;

ticket.print\_list();

cout << endl;

cout << "So do you want to order any ticket of this agency (+ or -)? - ";

cin >> choise;

if (choise == '+')

{

cout << "Enter number of course witch you prefer - ";

cin >> id1;

ticket.create\_ticket(id1);

break;

}

else

if (choise == '-')

{

system("cls");

goto point;

break;

}

else

{

system("cls");

cout << "Incorrect command! Try againg!" << endl;

goto point1;

break }

}

default:

{

system("cls");

cout << "Incorrect number! Try againg!" << endl;

goto point1;

break }

}

return 0;}