**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ДВНЗ «УНІВЕРСИТЕТ БАНКІВСЬКОЇ СПРАВИ»**

**ІНСТИТУТ БАНКІВСЬКИХ ТЕХНОЛОГІЙ ТА БІЗНЕСУ**

**КАФЕДРА КІБЕРБЕЗПЕКИ ТА СОЦІАЛЬНИХ НАУК**

**КУРСОВА РОБОТА**

з дисципліни

**«Програмування (Рівень В- Об’єктно-орієнтовне програмування)»**

(назва дисципліни)

на тему: **Каса авіаквитків**

Студента(ки) 2 курсу 203-Кб групи

спеціальності «Кібербезпека»

. Собчук А.В. .

(прізвище та ініціали)

Керівник

к.т.н., доцент .

(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 БАНКІВСКЬИХ ТЕХНОЛОГІЙ ТА БІЗНЕСУ**

**КАФЕДРА КІБЕРБЕЗПЕКИ ТА СОЦІАЛЬНИХ НАУК**

Спеціальність: «Кібербезпека»

Курс 2 Група 203-Кб Семестр 3

**ЗАВДАННЯ**

**На курсову роботу студента/студентки**

Собчук Анни Вікторівни

1. Тема курсової роботи: Каси авіаквитків
2. Термін здачі курсової роботи студентом: 12.12.2019
3. Вихiднi данi до (роботи): інформаційно-аналітичні інтернет джерела, зразки рішень практичних завдань та задач, нормативно-правова база, щодо оформлення технічної документації програмних продуктів, початкові значення для обробки.
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. Дата видачі завдання «10» жовтня 2019 р.

**Зміст**

[Вступ 4](#_Toc28008688)

[Розділ 1. Загальні відомості 6](#_Toc28008689)

[1.1 Формулювання вимог до програми 6](#_Toc28008690)

[1.2 Визначення та їх пояснення 6](#_Toc28008691)

[Розділ 2. Проектування продукту 11](#_Toc28008692)

[2.1. Опис алгоритму програми 11](#_Toc28008693)

[2.2 Створення об’єктів і розробка програми. Опис змінних та їх ідентифікаторів, що використовуються. 13](#_Toc28008694)

[2.3 Опис програми 14](#_Toc28008695)

[Розділ 3. Тестовий приклад роботи програми 18](#_Toc28008696)

[Висновки 20](#_Toc28008697)

[Список використаних джерел 21](#_Toc28008698)

[Додатки 22](#_Toc28008699)

# Вступ

Мова програмування Сi була розроблена в 1972 р. Деннисом Рiтчi, який є одним з авторів операцiйноi системи UNIX. При створенні С++ прагнули зберегти сумісність з мовою С. Більшість програм на С справно працюватимуть і з компілятором С++. С++ має синтаксис, заснований на синтаксисі С.

Нововведеннями С++ порівняно з С є:

* підтримка об'єктно-орієнтованого програмування через класи;
* підтримка узагальненого програмування через шаблони;
* доповнення до стандартної бібліотеки;
* додаткові типи даних;
* обробка винятків;
* простори імен;
* вбудовані функції;
* перевантаження операторів;
* перевантаження імен функцій;
* посилання і оператори управління вільно розподіленою пам'яттю.

Назва «Сі++» була вигадана Ріком Масситті (Rick Mascitti) і вперше було використана в грудні 1983 року. Раніше, на етапі розробки, нова мова називалася «Сі з класами». Ім'я, що вийшло у результаті, походить від оператора Сі «++» (збільшення значення змінної на одиницю) і поширеному способу присвоєння нових імен комп'ютерним програмам, що полягає в додаванні до імені символу «+» для позначення поліпшень. Згідно Страуструпу, «ця назва указує на еволюційну природу змін Cі».

Переваги мови C++ :

* Продуктивність. Швидкість роботи програм на С++ практично не поступається програмам на С, хоча програмісти отримали в свої руки нові можливості і нові засоби.
* Масштабованість. На мові C++ розробляють програми для самих різних платформ і систем.
* Можливість роботи на низькому рівні з пам'яттю, адресами, портами. (Що, при необережному використанні, може легко перетворитися на недолік.)
* Можливість створення узагальнених алгоритмів для різних типів даних, їх спеціалізація, і обчислення на етапі компіляції, з використанням шаблонів [2].

# **Розділ 1. Загальні відомості**

## 1.1 Формулювання вимог до програми

У нашому випадку є наступні вимоги до програми:

* номер рейсу;
* кількість вільних місць на кожному рейсі;
* бронювання квитків
* відміна бронювання

## 1.2 Визначення та їх пояснення

Клас на C ++ - це графічний об'єкт, який описує правила, атрибути та об’єкти. Причина - ідеальне зображення символічного розчинення та його принципів. Матеріали, виготовлені на основі одного пристрою, називаються китайськими моделями. Вони можуть мати різні характеристики, але вони завжди матимуть однаковий зміст. В ООП є три основні правила побудови класу:

1. Інкапсуляція - функція, яка дозволяє поєднувати аудиторію та дані та методи, що працюють з ними, та приховати від користувача деталі реалізації.
2. Дотримуйтесь - це особливість, яка дозволяє створити новий дочірній клас на основі наявного, з усіма характеристиками батьківського класу, класу, призначеного дитині.
3. Паліморфізм - властивість класу, яка дозволяє використовувати об’єкти класу з тим самим інтерфейсом без будь-якої інформації про тип та внутрішню структуру об’єкта.

Ми пильно придивляємось до кожної вбудованої властивосты класу за потребою, але просто пам’ятаємо ці три. Тепер поверніться до класів, щоб розпочати розробку структури класів. (Рис. 1)

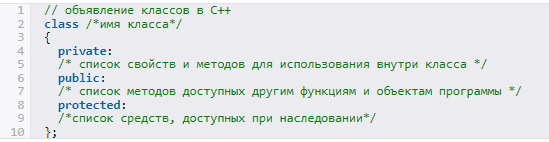
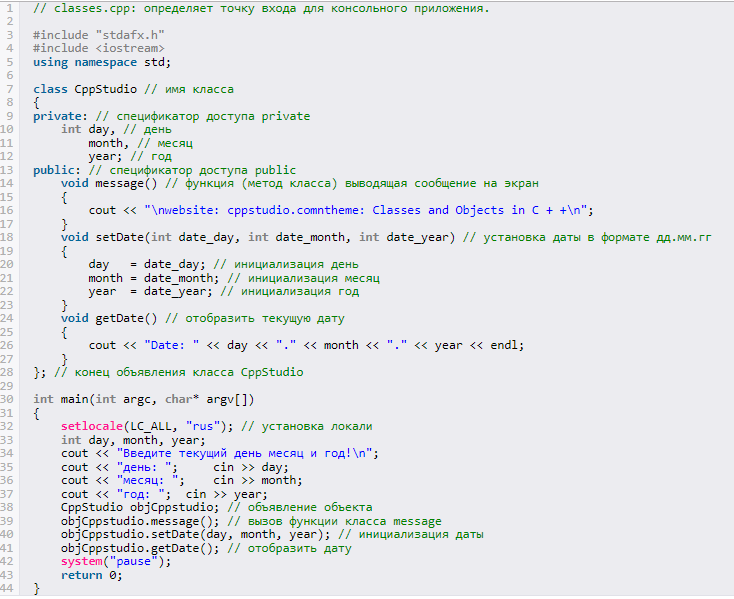


Рис. 1.1. - Структура проголошення занять

Set - функції і get - функції класів

У кожного об'єкта є свої особливості або атрибути, що характеризують його протягом усього життя. Атрибути об'єкта зберігаються в змінних, оголошених в класі, якому належить об'єкт. Крім того, оголошення змінної повинно бути виконано за допомогою приватного доступу. Ці змінні називаються елементами даних. Оскільки елементи даних оголошуються приватними, тільки методи класу можуть отримати доступ до них, зовнішній доступ до елементів даних заборонений. Тому зазвичай оголошують в класах спеціальні методи - так званий набір і отримання функцій, за допомогою яких можна маніпулювати елементами даними (рис. 1.2).

Рис. 1.2 - Реалізація класу з додаванням функцій set та get

Конструктор - це спеціальна функція, яка виконує елементи даних ініціалізації часу, а ім'я конструктора повинно бути обов'язково відповідним імені класу. Важлива відмінність конструктора від інших особливостей полягає в тому, що він не повертає жодних значень, у тому числі недійсних. У будь-якому класі повинен бути конструктор, навіть якщо конструктор не вказано точно (як у попередньому класі), тоді компілятор надає конструктору за замовчуванням без параметрів [3].

Функції - це блоки коду, які виконують певні операції. При необхідності функція може визначити параметри введення, які дозволяють викликає об'єкт передавати йому аргументи. При необхідності функція може також повернути значення в якості вихідного. Функції корисні для інкапсуляції основних операцій в один, багаторазовий прилад. В ідеалі назва цього блоку повинна чітко описувати функцію функції. Наступна функція бере два цілих числа від сміливого об'єкта і повертає їх суму. a і b - параметри типу int.

Параметр функції - це змінна, яка використовується в функції і значення якої надає caller (що викликає об'єкт). Параметри вказуються при оголошенні функції в круглих дужках. Якщо їх багато, то вони перераховуються через кому.

Аргумент функції - це значення, яке передається з caller-а в функцію і яке вказується в дужках при виконанні функції

У функції є список параметрів, в якому через кому перераховано необхідне (можливо, нульовий) число типів. Кожному параметру присвоюється ім'я, за яким до нього можна отримати доступ в тілі функції. У шаблоні функції можуть зазначатися додаткові типи або значення параметрів. Зухвалий об'єкт передає аргументи, що представляють собою конкретні значення, типи яких сумісні зі списком параметрів.

За замовчуванням аргументи передаються функції за значенням, тобто функція отримує копію переданого об'єкта. Копіювання великих об'єктів може бути ресурсозатратності і невиправданим.

Типи функцій повернення значень

Функція не може повертати інші функції і вбудовані масиви; Однак можливе повернення покажчиків на ці типи, або лямбда-вираз, що створює об'єкт функції. За винятком того, в цих випадках функція може повернути значення будь-якого типу, який знаходиться в області, або він не повертати ніякого значення, в цьому випадку повертається тип - void.

Всі типи функцій, крім типу [void](https://uk.wikipedia.org/w/index.php?title=Void&action=edit&redlink=1) повертають значення, визначене виразом в операторі return. Тобто, якщо є вказівка, що функція повертає якесь значення, то воно повинно бути заздалегідь зазначено для оператора return. У цьому випадку виконання функції, яка не належить до типу void, повертає довільне значення.

Також у мові C передбачена можливість повернення функцією вказівника. Щоб функція мала можливість повернути вказівник, вона повинна бути оголошена як функція, що повертає вказівник на потрібний тип. Тобто, тип покажчика, що повертається, повинен декларуватися явно. [4]

**Змінна** — це іменована область пам’яті, у якій зберігаються дані визначеного типу. Змінна має ім’я, розмір та інші атрибути, такі як видимість, час існування тощо. Ім’я змінної служить для звертання до області пам’яті, у якій зберігається її значення. **Перед використанням будь-яка змінна повинна бути описана**, при цьому для неї резервується деяка область пам’яті, розмір якої залежить від конкретного типу змінної. Під час виконання програми змінна може приймати рiзнi значення.

Ініціювання — це присвоювання змінній при описі початкового значення, яке записується зі знаком рівності — = значення або в круглих дужках — (значення). Зазначимо, що константа повинна бути ініційована при описі. Один оператор може містити опис декількох змінних одного типу, розділяючи їх кома­ми [1].

Компілятоp (від англ. Compile - збирати разом, складати) - системна програма, що виконує перетворення програми, написаної на одній з алгоритмічних мов, у програму мовою, близькою до машинної, і в певному сенсі еквівалентною першій.

# **Розділ 2. Проектування продукту**

## 2.1. Опис алгоритму програми

Програма написана за досить простим алгоритмом. Для початку ми вводимо інформацію по першому рейсі, а саме кількість місць та кількість заброньованих. Далі виконується вибір команди за допомогою умов. У нашому випадку їх шість. А саме: «додати», «видалити», «відмінити», «створити», «вивести», «вихід». При виборі команди виконується відповідна дія. Тобто при виборі команди:

* «додати» виконується дія «Вибір рейсу, до якого будуть додані місця»;
* «видалити» виконується дія «Вибір рейсу та введення кількості місць для видалення»;
* «відмінити» виконується дія «Вибір рейсу»;
* «створити» виконується дія «Введення кількості нових рейсів та інформації по кожному з них»;
* «вивести» виконується дія «Вибір рейсу та виведення інформації по ньому»;
* «вихід» виконується завершення програми.

Наступним кроком є реалізація дій для перших трьох команд відповідно:

* додаткові місця до вибраного рейсу;
* Видалення заданої кількості місць;
* Відміна рейсу.

При виборі одного з перших п’яти кроків, після їх реалізації ми повертаємось до моменту, де обираємо команду. Такий алгоритм повторюється поки не буде обрано команду «вихід» після чого програма завершує свою роботу.

Рис. 2.1. - Алгоритм

Якщо команда «вихід»

Кінець

Якщо команда «вивести»

Вибір рейсу та виведення інформації по ньому

Якщо команда «відмінити»

Якщо команда «створити»

Вибір рейсу

Введення кількості нових рейсів та інформації по кожному з них

Відміна рейсу

Якщо команда «видалити»

Вибір рейсу та введення кількості місць для видалення

Видалення заданої кількості місць

Якщо команда «додати»

Вибір рейсу, до якого будуть додані місця

Додаткові місця до вибраного рейсу

Введення інформації по першому рейсу

Вибір команди

Початок

## 2.2 Створення об’єктів і розробка програми. Опис змінних та їх ідентифікаторів, що використовуються.

Вказуємо дві бібліотеки **#include<iostream>** та **# include<string>.**

**Стандартна бібліотека «С++» -** це набір класів і функцій, які написані набазовій мові і є частиною стандарту «С++»ISO**.**

**#include<iostream> -** бібліотека і відповідний заголовний файл, які використовуються для організації введення-виведення в мові «С++».

**# include<string> -** клас з методами і змінними для організації роботи з рядками в мові програмування «С++».

Також прописано **using namespace std**.

**Namespace** – простір імен. Використовується для логічного угрупування класів, функцій і змінних.

**Функції** – це блоки коду, що виконують певні операції.

**Тип** – визначає, які значення може мати змінна, які операції з нею можна робити і скільки байт в пам’яті вона буде займати.

Створюємо функцію типу **void** з назвою *set-reserved(int reserved)*.

Покажчик типу **void –** спеціальний тип покажчика, який може вказувати на об’єкти будь-якого типу даних.

Потім створюємо функції типу **int** *get\_capacity(),get\_reserved(),get\_id().*

**Int** – цілочисельний тип даних.

С**out** – стандартний вихідний потік.

**Endl** – перехід на нову строку.

**Return** – оператор для повернення значен. (Використовується у всіх функціях, крім void).

За допомогою циклу **while** (цикл буде виконуватись до тих пір, поки буде виконуватись умова всередині умови циклу) перевіряємо на вибір команди.

Умова: якщо команда не «quit»

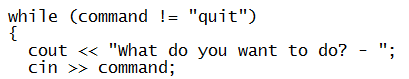


Рис.2.2. – Вибір команди

За допомогою оператора вибору **if**, що знаходиться всередині циклу, виконується дія відповідно команд.

В основній функції **int main()** за допомогою **cout** виводимо запитання про кількість місць на рейсі, та скільки з них зарезервовано.

Виконується цикл **while** поки не буде задано команду *«вихід»*.

У разі введення цієї команди програма завершує роботу.

## 2.3 Опис програми

#include <iostream> - бібліотека для організації вводу-виводу в С++;

Конструктори

FlightBooking() - Порожній конструктор для можливості об'яви заголовку об'єкту класу;

idCount++; - Статичний лічильник об'єктів;

this->id = this->idCount; - генератор індексів;

void set\_reserved(int reserved) - Метод для зміни кількості зарезервоаних місць;

Геттери

int get\_capacity()

{

return this->capacity;

}

int get\_reserved()

{

return this->reserved;

}

int get\_id()

{

return this->id;

}

static int get\_count()

{

return idCount;

}

double get\_percantage()

{

return (double(this->reserved) / double(this->capacity)) \* 100;

}

Геттери - спеціальний метод, що дозволяє отримати дані, доступ до яких безпосередньо обмежений. Це один з методів об'єктно-орієнтованого програмування, який допомагає реалізувати гнучкий механізм інкапсуляції.[5]

void cancel() - Відміна рейсу

{

this->reserved = NULL; - Якщо кільсть зарезервованих місць дорівнює нулю, то рейс відміняється

}

void print() - Виведення статусу рейсу

{

if (this->reserved != NULL) - Якщо рейс не відмінений

cout << "Flight #" << this->id << ".(" << get\_count() << ")\nStatus: reserved\nPlaces reserved: " << this->reserved << "/" << this->capacity << " (" << get\_percantage() << "%)." << endl;

static int idCount; - Статичний лічильник для генератора індексів;

int FlightBooking::idCount = 0; - Початкове значення для статичної змінної;

bool created = false; - Змінна, яка визначає чи були створені нові рейси;

FlightBooking\* booking1; - Заголовок для майбутнього масиву об'єктів розробленого класу (створений для коректності роботи областей видимості);

Введення даних по стартовому рейсу (обов'язкове)

cout << "Enter flight`s capacity: "; - скільки всього місць;

cout << "How many seats are reserved? - "; - скільки зарезервувати;

while (command != "quit") - Цикл, що продовжується поки не буде команди "вийти";

cout << "What do you want to do? - "; - Вибір команди;

if (command == "add" || command == "print" || command == "cancel" || command == "delete" || command == "create" || command == "remove") - Список доступних команд;

if (command == "create") - Команда створення нових рейсів;

cin >> newFlight; - Кількість нових рейсів;

if (newFlight <= 10) - Обмеження по кількості нових рейсів;

booking1 = new FlightBooking[newFlight]; - Створення динамічного масиву об'єктів розробленого класу;

for (int i = 0; i < newFlight; i++) - Заповнення даних по кожному з рейсів;

created = true; - Зміна статусу булевої змінної;

cout << "That`s too much" << endl; - Якщо занадто багато рейсів;

if (command == "print") - Команда виводу;

if (!created) - Якщо не було створено нових рейсів, то вивести інформацію лише по стартовому;

else- Якщо рейси все ж таки були створені;

cout << "Enter id of flight you want to work with - "; - Вибір конкретного рейсу, з яким буде вестись робота;

if (ind == booking.get\_id()) - Якщо вибраний перший (стартовий) рейс;

for (int i = 0; i < newFlight; i++) - Цикл пробігу по масиву рейсів;

if (command == "add") - Резервування додаткових місць в рейсі;

if (!created) - Якщо не були створені нові рейси, то робота ведеться лише зі стартовим;

cout << "How many places you want to add? - "; - Кількість додаткових місць;

if (booking.get\_reserved() + addition <= capacity && booking.get\_reserved() != NULL) - Обмеження, що не дозволяє ввести більше місць ніж всього є в літаку;

booking.set\_reserved(booking.get\_reserved() + addition); - Зміна значення зарезервованих місць через сеттер;

cout << "Flight #" << booking.get\_id() << " is overflowing" << endl; - Якщо рейс стане переповненим;

cout << "Enter id of flight you want to work with - "; - Вибрати рейс, з яким буде вестись робота;

if (command == "cancel") - Команда для відміни рейсу;  
cout << "Enter id of flight you want to work with - "; - Вибір рейсу, з яким буде вестись робота;

if (command == "remove") - Видалення місць;

if (!created) - Якщо не були створені нові рейси, то робота ведеться лише зі стартовим;

if (booking.get\_reserved() - deleting > 0 && booking.get\_reserved() != NULL) - Якщо кількість місць після дії не зайде в мінус, та якщо рейс не відмінили;

cout << "Error" << endl; - Якщо рейс відмінений або після цієї дії кількість зарезервованих місць вийде в мінус;

cout << "Enter id of flight you want to work with - "; - Вибір рейсу, з яким буде вестись робота;

if (command != "quit") - Якщо команди немає в списку доступних і команда не є командою виходу, то вивести помилку;

if (created) - Якщо були створені нові рейси, то вивільнити пам'ять для запобігання (утечки) пам'яті.

# **Розділ 3. Тестовий приклад роботи програми**

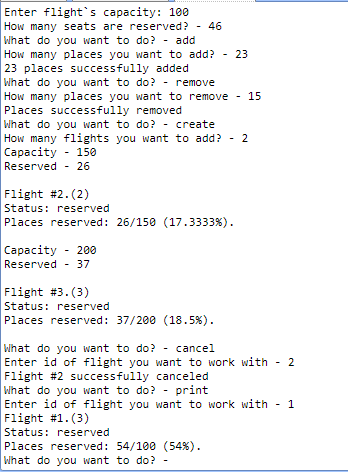


Рис 3.1. - Приклад роботи програми з використанням усіх команд

Як ми можемо бачити на рис.3 самою першою виведеною фразою є Кількість місць на рейсі, яку ми вказуємо з клавіатури. Наступне речення – кількість зарезервованих місць, яку користувач теж вводить самостійно. Далі це вибір команди. Ось список слів, які відповідають за виклик тієї чи іншої команди:

* add – резервування місць
* remove – віднімання зарезервованих місць на рейсі
* create – додавання рейсу
* cancel – відміна рейсу
* print – виведення на екран
* quit – вихід з програми

Якщо ми обираєм «add», то на екрані з’являється запитання скільки місць ми хочем зарезервувати, потім результат, що вказану кількість місць зарезервовано, або, що рейс переповнений у разі помилки.

При обиранні «remove» висвічується запитання зі скількох місць користувач бажає зняти резерв.

Далі «create», при виборі цієї дії запитується скільки рейсів ми бажаємо створити, скільки у кожному з них всього місць, та скільки з них зарезервовані, для кожного рейсу окремо. Також виводить відсоток зарезервованих місць від загальної кількості на кожному рейсі.

«cancel», за допомогою цього ми можемо видалити будь-який рейс, просто вказавши його номер у відповіді на запитання який рейс потрібно видалити.

Наступним є «print» який виводить усю інформацію по одному з рейсів, номер якого користувач вказує самостійно, як відповідь на запитання з яким рейсом ми хочемо працювати.

Ну і на останок «quit», який завершує роботу програми.

# **Висновки**

У цій роботі було наведено теоретичний та практичний варіант реалізації програми «Каса авіаквитків». Код було створено за допомогою класів, методів та функцій. Розроблено у онлайн компіляторі C++Shell. Також додано блок‑схему алгоритму роботи програми та розписано усі деталі покроково.

Користувач може задати кількість вільних місць, заброньованих місць, також є можливість забронювати та відмінити бронювання. Ще однією можливістю є додавання і видалення рейсів.

Стосовно першої можливості, користувач може забронювати кількість, не більше, ніж вказано спочатку, в іншому випадку програма виведе на екран помилку.

# **Список використаних джерел**

1. Змінні [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: http://cpp.dp.ua/zminni/.
2. Огляд і основи мови програмування C++ [Електронний ресурс] – Режим доступу до ресурсу: <http://www.znannya.org/?view=Cpp_basics>.
3. Класи в С++ [Електронний ресурс] – Режим доступу до ресурсу: <http://cppstudio.com/uk/post/439/>.
4. Functions (C++) [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.microsoft.com/ru-ru/cpp/cpp/functions-cpp?view=vs-2019>.
5. Геттер [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D1%82%D1%82%D0%B5%D1%80\_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)#C++.

# **Додатки**

**Додаток А. Лістинг програмного продукту**

#include <iostream>

#include <string>

using namespace std;

class FlightBooking

{

public:

FlightBooking()

{

}

FlightBooking(int capacity, int reserved)

{

idCount++;

this->id = this->idCount;

this->capacity = capacity;

this->reserved = reserved;

}

void set\_reserved(int reserved)

{

this->reserved = reserved;

}

int get\_capacity()

{

return this->capacity;

}

int get\_reserved()

{

return this->reserved;

}

int get\_id()

{

return this->id;

}

static int get\_count()

{

return idCount;

}

void cancel()

{

this->reserved = NULL;

}

double get\_percantage()

{

return (double(this->reserved) / double(this->capacity)) \* 100;

}

void print()

{

if (this->reserved != NULL)

cout << "Flight #" << this->id << ".("<< get\_count() <<")\nStatus: reserved\nPlaces reserved: " << this->reserved << "/" << this->capacity << " (" << get\_percantage() << "%)." << endl;

else

cout << "Flight #" << this->id << " is canceled." << endl;

}

private:

static int idCount;

int id;

int capacity;

int reserved;

};

int FlightBooking::idCount = 0;

int main()

{bool created = false;

FlightBooking\* booking1;

int capacity, reserved, newFlight, ind;

string command;

cout << "Enter flight`s capacity: ";

cin >> capacity;

cout << "How many seats are reserved? - ";

cin >> reserved;

FlightBooking booking(capacity, reserved);

while (command != "quit")

{

cout << "What do you want to do? - ";

cin >> command;

if (command == "add" || command == "print" || command == "cancel" || command == "delete" || command == "create" || command == "remove")

{

if (command == "create")

{

cout << "How many flights you want to add? - ";

cin >> newFlight;

if (newFlight <= 10)

{

int cap, res;

booking1 = new FlightBooking[newFlight];

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

{

cout << "Capacity - ";

cin >> cap;

cout << "Reserved - ";

cin >> res;

booking1[i] = FlightBooking(cap, res);

cout << endl;

booking1[i].print();

cout << endl;

}

created = true;

}

else

cout << "That`s too much" << endl;

}

if (command == "print")

{

if (!created)

booking.print();

else

{

cout << "Enter id of flight you want to work with - ";

cin >> ind;

if (ind == booking.get\_id())

{

booking.print();

break;

}

else

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

if (ind == booking1[i].get\_id())

{

booking1[i].print();

break;

}

}

}

if (command == "add")

{

int addition;

if (!created)

{

cout << "How many places you want to add? - ";

cin >> addition;

if (booking.get\_reserved() + addition <= capacity && booking.get\_reserved() != NULL)

{

booking.set\_reserved(booking.get\_reserved() + addition);

cout << addition << " places successfully added" << endl;

}

else

cout << "Flight #" << booking.get\_id() << " is overflowing" << endl;

}

else

{

cout << "Enter id of flight you want to work with - ";

cin >> ind;

if (ind == booking.get\_id())

{

cout << "How many places you want to add? - ";

cin >> addition;

if (booking.get\_reserved() + addition <= capacity && booking.get\_reserved() != NULL)

{

booking.set\_reserved(booking.get\_reserved() + addition);

cout << addition << " places successfully added" << endl;

}

else

cout << "Flight #" << booking.get\_id() << " is overflowing" << endl;

}

else

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

if (booking1[i].get\_id() == ind)

{

cout << "How many places you want to add? - ";

cin >> addition;

if (booking1[i].get\_reserved() + addition <= booking1[i].get\_capacity() && booking1[i].get\_reserved() != NULL)

{

booking1[i].set\_reserved(booking1[i].get\_reserved() + addition);

cout << addition << " places successfully added" << endl;

}

else

cout << "Flight #" << booking1[i].get\_id() << " is overflowing" << endl;

}

}

}

if (command == "cancel")

{

if (!created)

{

booking.cancel();

cout << "Flight #" << booking.get\_id() << " successfully canceled" << endl;

}

else

{

cout << "Enter id of flight you want to work with - ";

cin >> ind;

if (ind == booking.get\_id())

{

booking.cancel();

cout << "Flight #" << booking.get\_id() << " successfully canceled" << endl;

}

else

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

if (booking1[i].get\_id() == ind)

{

booking1[i].cancel();

cout << "Flight #" << booking1[i].get\_id() << " successfully canceled" << endl;

}

}

}

if (command == "remove")

{

int deleting;

if (!created)

{

cout << "How many places you want to remove - ";

cin >> deleting;

if (booking.get\_reserved() - deleting > 0 && booking.get\_reserved() != NULL)

{

booking.set\_reserved(booking.get\_reserved() - deleting);

cout << "Places successfully removed" << endl;

}

else

cout << "Error" << endl;

}

else

{

cout << "Enter id of flight you want to work with - ";

cin >> ind;

if (ind == booking.get\_id())

{

cout << "How many places you want to remove - ";

cin >> deleting;

if (booking.get\_reserved() - deleting > 0 && booking.get\_reserved() != NULL)

{

booking.set\_reserved(booking.get\_reserved() - deleting);

cout << "Places successfully removed" << endl;

}

else

cout << "Error" << endl;

}

else

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

if (booking1[i].get\_id() == ind)

{

cout << "How many places you want to remove - ";

cin >> deleting;

if (booking1[i].get\_reserved() - deleting > 0 && booking1[i].get\_reserved() != NULL)

{

booking1[i].set\_reserved(booking1[i].get\_reserved() - deleting);

cout << "Places successfully removed" << endl;

}

else

cout << "Error" << endl;

}

}

}

}

else

if (command != "quit")

cout << "You cannot do this" << endl;

}

if (created)

delete[] booking1;

return 0;

}

**Додаток В. Блок-схема**