

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ  
И РАДИОЭЛЕКТРОНИКИ (ТУСУР)  
Кафедра компьютерных систем в управлении и проектировании (КСУП)

## **КЛАССЫ И ИНКАПСУЛЯЦИЯ**

Отчёт по лабораторной работе №3 по дисциплине  
«Объектно-ориентированное программирование»

Студент группы 588-1

\_\_\_\_\_ / Чан Хыу Тхай

«\_» \_\_\_\_\_ 2022 г.

Руководитель старший научный  
сотрудник, доцент каф. КСУП

\_\_\_\_\_ / Горяинов А.Е.

«\_\_» \_\_\_\_\_ 2022 г.

Томск 2022

## 1 Цель работы

Цель этой лабораторной работы — ознакомиться с классами и инкапсуляцией, а также получить практические навыки решения связанных с ними задач объектно-ориентированного программирования.

## 2 Теоретические основы

**Класс** – это просто независимая подпрограмма, в которой есть свой набор переменных и функций (обязательно изучи урок про функции). В основной программе мы можем создать экземпляр класса (он же называется объект) и пользоваться теми инструментами, которые имеются в классе. Использование классов позволяет:

- Разделить сложную программу на отдельные независимые части
- Создавать удобные библиотеки
- Использовать свои “наработки” в другом проекте, не переписывая один и тот же код
- Облегчить и упростить программу, если в ней используются повторяющиеся конструкции и алгоритмы

Классы очень похожи на структуры (читай урок по структурам), как по объявлению, так и по использованию, но класс является гораздо более мощной единицей языка благодаря механизмам наследования и прочим ООП-штукам.

Класс объявляется при помощи ключевого слова `class` и содержит внутри себя члены класса – переменные и функции:

```
class Имя_класса {  
  
    член1;  
  
    член2;  
  
};
```

Важное отличие от структуры: содержимое класса делится на области: публичные и приватные. Они определяются при помощи ключевых слов `public` и `private`, область действует до начала следующей области или до закрывающей фигурной скобки класса:

- `public` — члены класса в этой области доступны для взаимодействия из основной программы (скетча), в которой будет создан объект. Например `write()` у серво.
- `private` — члены класса в этой области доступны только **внутри класса**, то есть из программы к ним обратиться нельзя.

```
class Имя_класса {  
  
    public:  
  
    // список членов, доступных в программе  
  
    private:  
  
    // список членов для использования внутри класса  
  
};
```

**Инкапсуляция** это набор инструментов для управления доступом к данным или методам которые управляют этими данными. С детальным определением термина “инкапсуляция” можно ознакомиться в моей предыдущей публикации на Хабре по этой ссылке. Эта статья сфокусирована на примерах инкапсуляции в Си++ и Си.

По умолчанию, в классе (`class`) данные и методы приватные (`private`); они могут быть прочитаны и изменены только классом к которому принадлежат. Уровень доступа может быть изменен при помощи соответствующих ключевых слов которые предоставляет Си++.

В Си++ доступно несколько спецификаторов, и они изменяют доступ к данным следующим образом:

- публичные ( `public` ) данные—доступны всем;
- защищенные ( `protected` )—доступны только классу и дочерним классам;
- приватные ( `private` ) —доступны только классу которому они принадлежат.

## Пример инкапсуляции

В классе `Contact`, публичные переменные и методы доступны из основной программы (`main`). Приватные переменные и методы могут прочитаны, вызваны или изменены только самим классом.

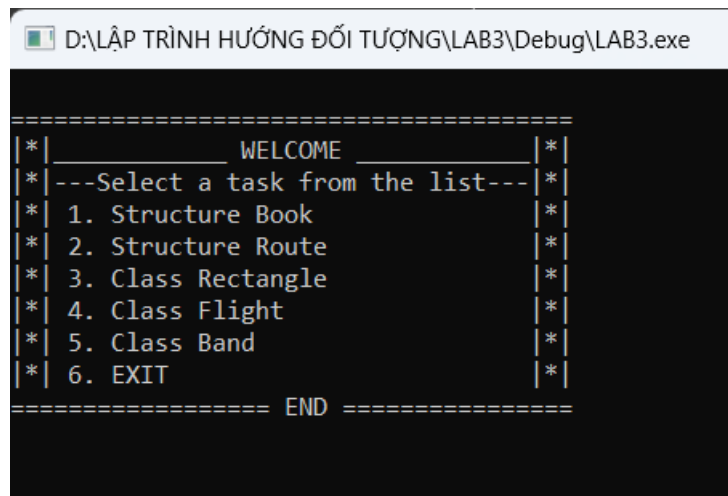
```
class Contact
{
    private:
        int mobile_number;           // private variable
        int home_number;             // private variable
    public:
        Contact()                    // constructor
        {
            mobile_number = 12345678;
            home_number = 87654321;
        }
        void print_numbers()
        {
            cout << "Mobile number: " << mobile_number;
            cout << ", home number: " << home_number << endl;
        }
};

int main()
{
    Contact Tony;
    Tony.print_numbers();
    // cout << Tony.mobile_number << endl;
    // will cause compile time error
    return 0;
}
```

Попытка напечатать или изменить приватную переменную `mobile_number` из основной программы (`main`) вызовет ошибку при компиляции потому как доступ к приватным данным в классе ограничен.

### 3 Ход работы

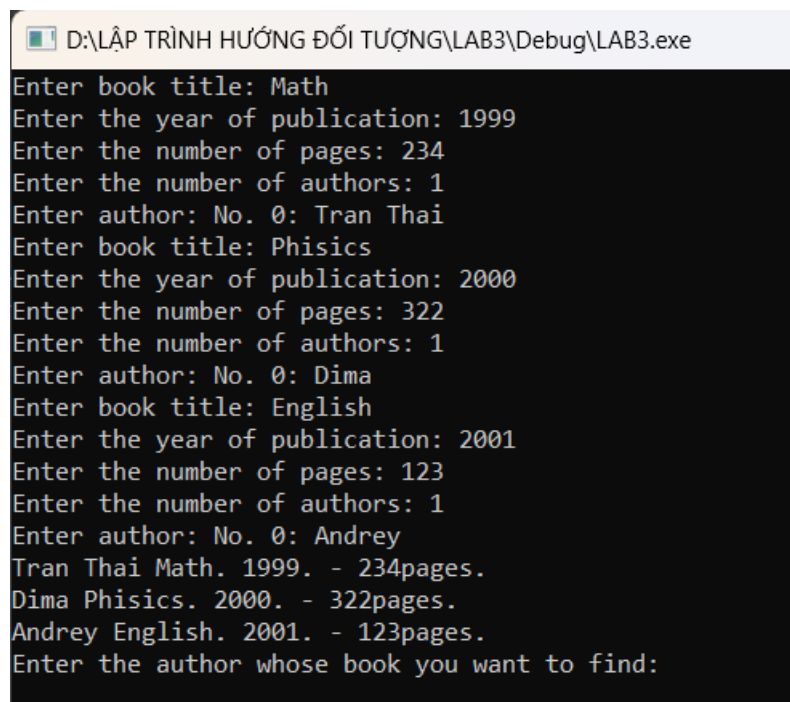
Ниже представлен интерфейс лабораторной работы:



```
D:\LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG\LAB3\Debug\LAB3.exe

=====
*|_____ WELCOME _____|*|
*|---Select a task from the list---|*|
*| 1. Structure Book              |*|
*| 2. Structure Route            |*|
*| 3. Class Rectangle            |*|
*| 4. Class Flight               |*|
*| 5. Class Band                 |*|
*| 6. EXIT                      |*|
===== END =====
```

В первой задаче нужно создать структуру книги Book. В структуре должны быть поля "Название", "Год издания", "Количество страниц", массив строк "Авторы" (не более десяти), поле "Количество авторов" (рис. 1).



```
D:\LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG\LAB3\Debug\LAB3.exe
Enter book title: Math
Enter the year of publication: 1999
Enter the number of pages: 234
Enter the number of authors: 1
Enter author: No. 0: Tran Thai
Enter book title: Phisics
Enter the year of publication: 2000
Enter the number of pages: 322
Enter the number of authors: 1
Enter author: No. 0: Dima
Enter book title: English
Enter the year of publication: 2001
Enter the number of pages: 123
Enter the number of authors: 1
Enter author: No. 0: Andrey
Tran Thai Math. 1999. - 234pages.
Dima Phisics. 2000. - 322pages.
Andrey English. 2001. - 123pages.
Enter the author whose book you want to find:
```

Рис. 1 – Создать структуру книги

Оттуда мы можем ввести информацию об имени автора для поиска и вывода всей информации о книге этого автора. (рис. 2).

```
Enter author: No. 0: Huu
Thai Math. 1999. - 234pages.
Tran English. 2000. - 222pages.
Huu Information. 2002. - 333pages.
Enter the author whose book you want to find: Huu
Huu Information. 2002. - 333pages.
```

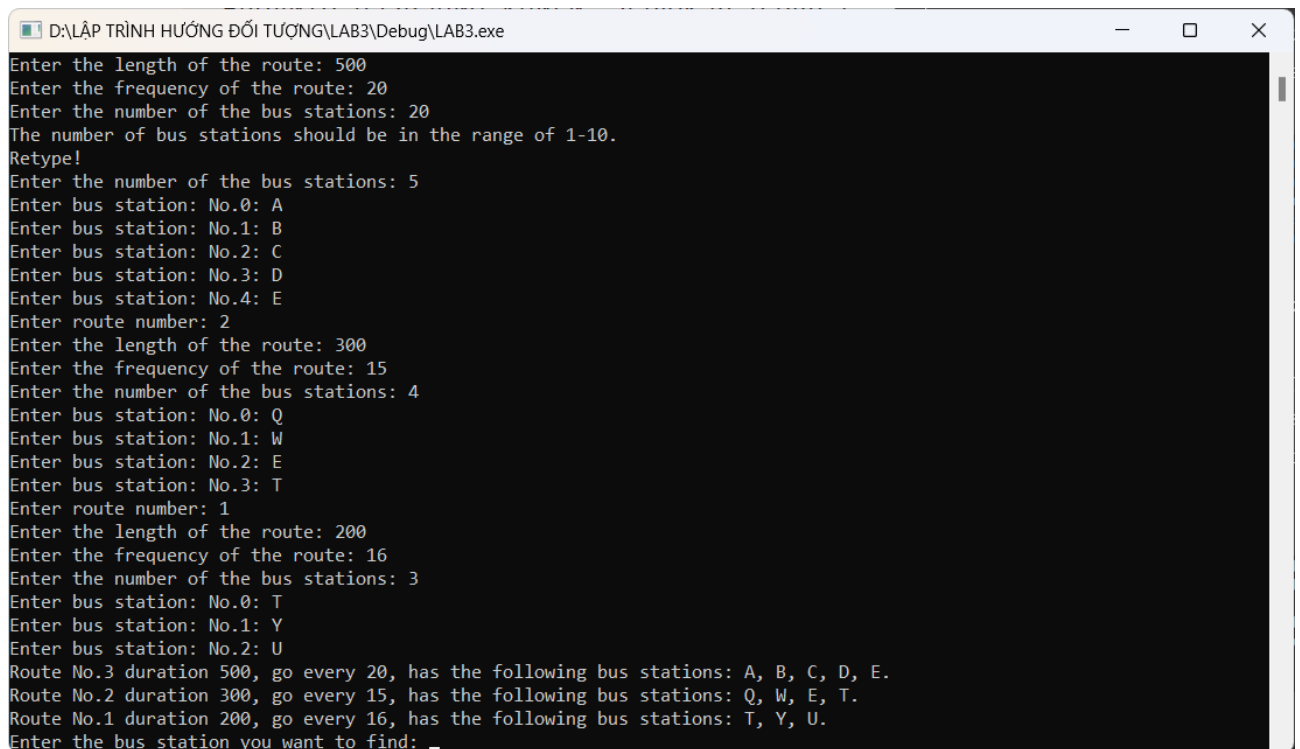
Рис. 2 – Информация о книге разыскиваемого автора

Условие проверки на количество авторов менее 10 проверено и выполнено (рис. 3).

```
Enter the year of publication: 1999
Enter the number of pages: 234
Enter the number of authors: 15
The number of authors should be in the range of 1-10.
Please retype the year!
Enter the number of authors:
```

Рис. 3 – Условия проверки количества авторов

В второй задаче нужно создать функцию WriteBookToConsole(Book& book), которая выводит данные книги на экран. Аналогично вводу, вам будет нужно организовать вывод правильного количества авторов на экран. Пример работы функции (рис. 4).



```
D:\LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG\LAB3\Debug\LAB3.exe
Enter the length of the route: 500
Enter the frequency of the route: 20
Enter the number of the bus stations: 20
The number of bus stations should be in the range of 1-10.
Retype!
Enter the number of the bus stations: 5
Enter bus station: No.0: A
Enter bus station: No.1: B
Enter bus station: No.2: C
Enter bus station: No.3: D
Enter bus station: No.4: E
Enter route number: 2
Enter the length of the route: 300
Enter the frequency of the route: 15
Enter the number of the bus stations: 4
Enter bus station: No.0: Q
Enter bus station: No.1: W
Enter bus station: No.2: E
Enter bus station: No.3: T
Enter route number: 1
Enter the length of the route: 200
Enter the frequency of the route: 16
Enter the number of the bus stations: 3
Enter bus station: No.0: T
Enter bus station: No.1: Y
Enter bus station: No.2: U
Route No.3 duration 500, go every 20, has the following bus stations: A, B, C, D, E.
Route No.2 duration 300, go every 15, has the following bus stations: Q, W, E, T.
Route No.1 duration 200, go every 16, has the following bus stations: T, Y, U.
Enter the bus station you want to find: 
```

Рис. 4 – Информация о маршрутах и направлениях

Условие проверки на количество остановка менее 10 проверено и выполнено (рис. 5).

```
Enter the number of the bus stations: 20
The number of bus stations should be in the range of 1-10.
Retype!
Enter the number of the bus stations: 5
Enter bus station: No.0:
```

Рис. 5 – Условия проверки количества остановка

Оттуда мы можем ввести информацию о маршруте для поиска и отображения всей информации об этом маршруте. (рис. 6).

```
Route No.3 duration 500, go every 20, has the following bus stations: A, B, C, D, E.
Route No.2 duration 300, go every 15, has the following bus stations: Q, W, E, T.
Route No.1 duration 200, go every 16, has the following bus stations: T, Y, U.
Enter the bus station you want to find: E
Route No.2 duration 300, go every 15, has the following bus stations: Q, W, E, T.
```

Рис. 6 – Информация о маршруте для поиска

Следующая задача — продемонстрировать функцию DemoRectangleWithPoint(). Нужно добавить код, который будет отображать данные о прямоугольниках массива (рис. 7).

```
X = 3; Y = 4; Length = 3; Width = 4.
X = 4; Y = 5; Length = 4; Width = 5.
X = 5; Y = 6; Length = 5; Width = 6.
X = 6; Y = 7; Length = 6; Width = 7.
X = 7; Y = 8; Length = 7; Width = 8.
X = 8; Y = 9; Length = 8; Width = 9.
X = 9; Y = 10; Length = 9; Width = 10.
```

Рис. 7 – Демонстрационная функция прямоугольника

Следующая задача — продемонстрировать функцию DemoRectangleWithPoint(). Нужно добавить код, который будет вычислять среднее значение всех центров прямоугольника и отображать его на экране. Пример вывода (расчетные значения для показанных выше координат) (рис. 8).

```

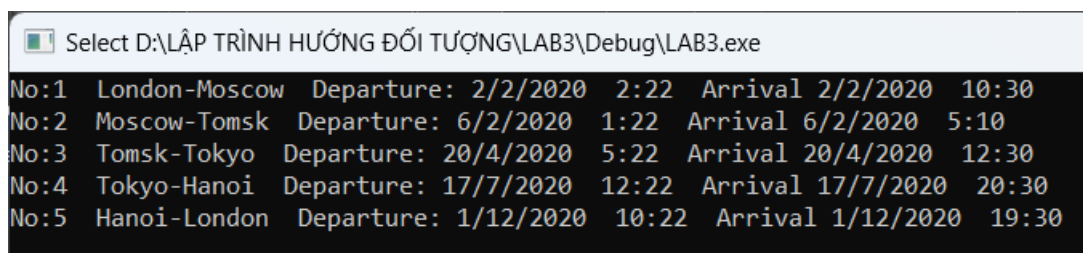
X = 3; Y = 4; Length = 3; Width = 4.
X = 4; Y = 5; Length = 4; Width = 5.
X = 5; Y = 6; Length = 5; Width = 6.
X = 6; Y = 7; Length = 6; Width = 7.
X = 7; Y = 8; Length = 7; Width = 8.
X = 8; Y = 9; Length = 8; Width = 9.
X = 9; Y = 10; Length = 9; Width = 10.

X = 6; Y = 7

```

Рис. 8 – Результат вычисления среднего значения всех центров  
прямоугольников

Для следующей задачи необходимо создать структуру Flight, в которой хранятся поля номера рейса, пункт отправления, пункт назначения, время отправления и время прибытия (компоненты структуры Time). Для структур необходимо создать функции конструктора и сеттера с соответствующими валидациями данных (время прихода не может быть раньше времени отправления) (рис. 9).



```

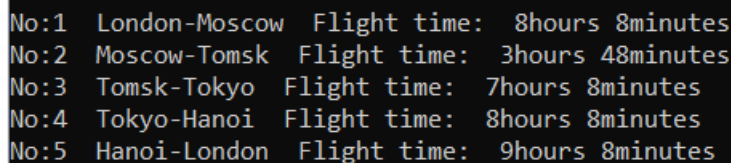
Select D:\LẬP TRÌNH HƯỚNG DẪN TỰ HỌC\LAB3\Debug\LAB3.exe
No:1 London-Moscow Departure: 2/2/2020 2:22 Arrival 2/2/2020 10:30
No:2 Moscow-Tomsk Departure: 6/2/2020 1:22 Arrival 6/2/2020 5:10
No:3 Tomsk-Tokyo Departure: 20/4/2020 5:22 Arrival 20/4/2020 12:30
No:4 Tokyo-Hanoi Departure: 17/7/2020 12:22 Arrival 17/7/2020 20:30
No:5 Hanoi-London Departure: 1/12/2020 10:22 Arrival 1/12/2020 19:30

```

Рис. 9 – Информация о полете

Необходимо написать функцию GetFlightTimeMinutes(Flights & Flights), которая принимает на вход объект полета и возвращает время полета в минутах (или в новом объекте Time).

И нужно продемонстрировать, как работает функция, вызвав функцию для каждого элемента массива в функции DemoFlightWithTime() (рис. 10).



```

No:1 London-Moscow Flight time: 8hours 8minutes
No:2 Moscow-Tomsk Flight time: 3hours 48minutes
No:3 Tomsk-Tokyo Flight time: 7hours 8minutes
No:4 Tokyo-Hanoi Flight time: 8hours 8minutes
No:5 Hanoi-London Flight time: 9hours 8minutes

```

Рис. 10 – Информация о время полете



Следующей задачей необходимо создать функцию Demoband() для вывода информации о музыкальных продуктах. (рис. 11).

```
void DemoBand()
{
    Song* first = new Song[SONGS_COUNT];
    Song* second = new Song[SONGS_COUNT];
    Song* third = new Song[SONGS_COUNT];

    first[0] = Song("Twenty one pilots", 3.52, Rock);
    first[1] = Song("No love for the middle child", 3.08, Rock);
    first[2] = Song("The Rush", 3.08, Rock);
    first[3] = Song("World on Fire", 4.57, Rock);

    second[0] = Song("Weird", 3.18, Rock);
    second[1] = Song("Shutdown", 3.16, Rock);
    second[2] = Song("Never looking back the giving moon", 2.59, Rock);
    second[3] = Song("Not OK!", 3.16, Chanson);

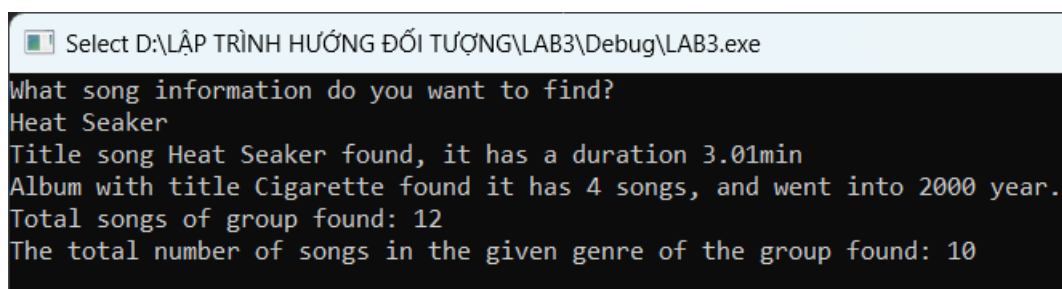
    third[0] = Song("Heat Seaker", 3.01, Rock);
    third[1] = Song("Guys", 4.38, Rock);
    third[2] = Song("Living In a Ghost Town", 4.00, Chanson);
    third[3] = Song("If you are too Shy(Let me know)", 4.10, Rock);

    Album* albums = new Album[ALBUMS_COUNT];
    albums[0] = Album("Dark night", 1998, first, 4);
    albums[1] = Album("Roses", 1989, second, 4);
    albums[2] = Album("Cigarette", 2000, third, 4);

    Band band = Band("Dark night",
        "One of the most popular rock bands of the 199X year.",
        albums, ALBUMS_COUNT);
}
```

Рис. 11 – Функция Demoband() для вывода информации о музыкальных продуктах

И при поиске возвращаются результаты для искомой песни (рис. 12).



```
Select D:\LẬP TRÌNH HƯỚNG DẪN TƯỢNG\LAB3\Debug\LAB3.exe
What song information do you want to find?
Heat Seaker
Title song Heat Seaker found, it has a duration 3.01min
Album with title Cigarette found it has 4 songs, and went into 2000 year.
Total songs of group found: 12
The total number of songs in the given genre of the group found: 10
```

Рис. 12 – Результаты возвращены для искомой песни

#### **4 Вывод**

В ходе работы в лаборатории были выполнены все задачи, разработана природа «Классы и инкапсуляция» в объектно-ориентированном программировании и набор функций для работы с ней.