

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)

СТРУКТУРЫ И ПЕРЕЧИСЛЕНИЯ

Отчёт по лабораторной работе №2 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 со структурами и типами `enum` и получить практические навыки решения связанных с ними задач в объектно-ориентированном программировании.

2. Теоретические основы

Структура `struct` – очень интересный тип данных: это набор переменных разного типа, объединённых одним именем. В некоторых случаях структуры позволяют сильно упростить написание кода и сделать его более понятным, а также упростить придумывание новых имён для переменных. А ещё структура – это практически класс (урок про классы)! Но без механизмов наследования и приватных-публичных членов.

Структура объявляется по следующей схеме:

```
struct ярлык {  
  
    тип_данных имя_переменной_1;  
  
    тип_данных имя_переменной_2;  
  
    тип_данных имя_переменной_3;  
  
};
```

Ярлык будет являться новым типом данных. Используя этот ярлык, можно объявлять уже непосредственно саму структуру как переменную:

```
ярлык имя_структуры; // объявить одну структуру  
  
ярлык имя_структуры_1, имя_структуры_2; // объявить две  
структуры  
  
ярлык имя_структуры[5]; // объявить массив структур
```

Также есть вариант объявления структуры без ярлыка, т.е. создаём структуру и сразу указываем её имя в конце.

```
struct {  
  
    тип_данных имя_переменной_1;  
  
    тип_данных имя_переменной_2;  
  
    тип_данных имя_переменной_3;  
  
} имя_структуры;
```

Для чего нужны структуры? В большинстве примеров в интернете приводится использование структур для создания базы данных: имя, фамилия, номер телефона, адрес и т.д. Структуры очень удобно передавать по радио и другим интерфейсам связи, на моей практике – в структуре удобнее всего хранить настройки, потому что её можно целиком записать в EEPROM одной строчкой и одной строчкой оттуда же её прочитать, не заморачиваясь с адресами.

Перечисления (enum – enumeration) – тип данных, представляющий собой набор именованных констант, нужен для удобства написания программы. Допустим у нас есть переменная mode, отвечающая за номер режима работы устройства. Мы для себя запоминаем, какому значению переменной какой режим будет соответствовать, и где-нибудь себе записываем, например 0 – обычный режим, 1 – режим ожидания, 2 – режим настройки_1, 3 – режим настройки_2, 4 – калибровка, 5 – аварийный режим, ошибка. При написании или чтении программы часто придётся обращаться к этому списку, чтобы не запутаться. Можно сделать первый шаг по оптимизации: обозвать каждый режим при помощи константы (дефайна):

```
#define NORMAL 0  
  
#define WAITING 1  
  
#define SETTINGS_1 2  
  
#define SETTINGS_2 3
```

```
#define CALIBRATION 4
```

```
#define ERROR_MODE 5
```

Использование `enum` ещё немного упрощает эту задачу: перечисление позволяет создать переменную (по умолчанию имеет тип `int`), которая может принимать только те “названия”, которые для неё указаны. Это удобно также тем, что компилятор выдаст ошибку при создании перечислений с одинаковыми значениями. В случае с дефайнами программа просто будет работать неправильно! Объявление перечисления чем-то похоже на объявление структуры:

```
enum ярлык {имя1, имя2, имя3, имя4, имя5};
```

Таким образом мы объявили ярлык. Теперь, используя этот ярлык, можно объявить само перечисление:

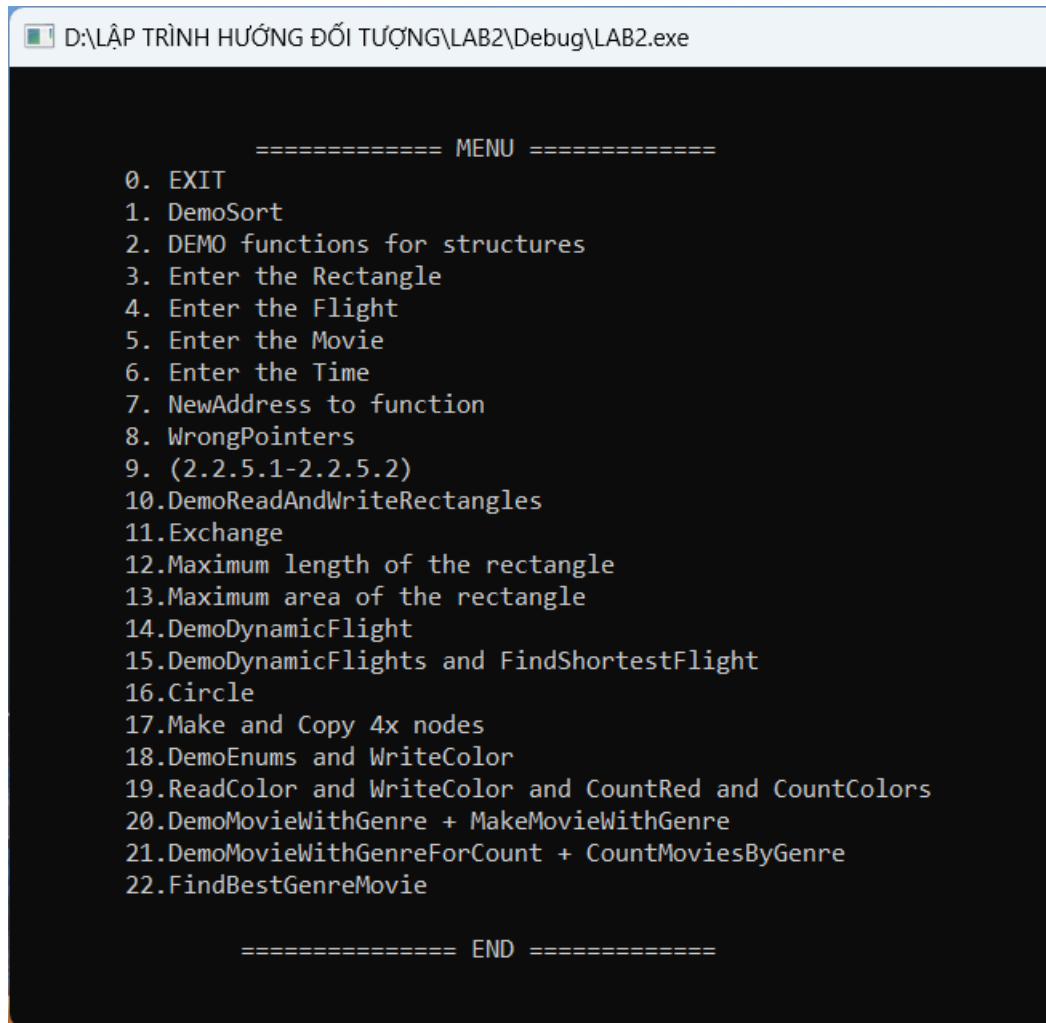
```
ярлык имя_перечисления;
```

Также как и у структур, можно объявить перечисление без создания ярлыка (зачем нам лишняя строчка?):

```
enum {имя1, имя2, имя3, имя4, имя5} имя_перечисления;
```

3. Ход работы

Ниже представлен интерфейс лабораторной работы:



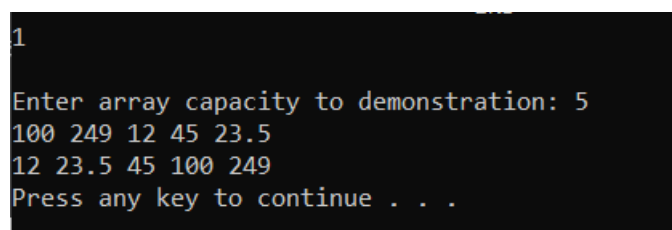
```
D:\LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG\LAB2\Debug\LAB2.exe

===== MENU =====
0. EXIT
1. DemoSort
2. DEMO functions for structures
3. Enter the Rectangle
4. Enter the Flight
5. Enter the Movie
6. Enter the Time
7. NewAddress to function
8. WrongPointers
9. (2.2.5.1-2.2.5.2)
10.DemoReadAndWriteRectangles
11.Exchange
12.Maximum length of the rectangle
13.Maximum area of the rectangle
14.DemoDynamicFlight
15.DemoDynamicFlights and FindShortestFlight
16.Circle
17.Make and Copy 4x nodes
18.DemoEnums and WriteColor
19.ReadColor and WriteColor and CountRed and CountColors
20.DemoMovieWithGenre + MakeMovieWithGenre
21.DemoMovieWithGenreForCount + CountMoviesByGenre
22.FindBestGenreMovie

===== END =====
```

Рисунок 1 – Интерфейс работы

В функции DemoSort() дважды вызовите функцию Sort(), сначала для count равного 5, а затем равным -1. Убедитесь, что при положительных значениях count сортировка выполняется корректно, а при отрицательных генерируется исключение (Рисунок 2)



```
1
Enter array capacity to demonstration: 5
100 249 12 45 23.5
12 23.5 45 100 249
Press any key to continue . . .
```

Рисунок 2 – Вывод функции демонстрационной сортировки

В задаче 2 представлена демонстрация функции в структуре (Рисунок 3)

```
DemoRectangle:

Color: Red
Rectangle size: 1x7

DemoFlight:

Flight: Moscow -- Tomsk

DemoMovie:

Film: Click
Duration(minutes): 100
Year of issue: 2001
Genre: Comedy
Rate: 8.3

DemoTime:

Time: 20 hour 10 minute 59 second.
```

Рисунок 3 – Демонстрационная функция в структуре

В задаче 3 есть демонстрационная функция для ввода прямоугольника (Рисунок 4)

```
3
Enter the number of times: 2

Enter rectangle 1th :
Enter color: Red

Enter length: 10

Enter width: 8
Enter rectangle 2th :
Enter color: Blue

Enter length: 8

Enter width: 6

Rectangle 1th :
Color: Red
Rectangle size: 10x8

Rectangle 2th :
Color: Blue
Rectangle size: 8x6
```

Рисунок 4 – Демонстрация функции прямоугольного ввода

В задаче 4 представлена демонстрация функции ввода полета (Рисунок 5):

```

4
Enter the number of times: 2

Enter flight 1:
Enter the starting point: Tomsk

Enter destination: Moscow

Enter time: 10

Enter flight 2:
Enter the starting point: Moscow

Enter destination: Tomsk

Enter time: 18


Flight 1th :

Flight: Tomsk -- Moscow
Flight 2th :

Flight: Moscow -- Tomsk

```

Рисунок 5 – Демонстрация функции ввода полета.

В задаче 5 представлена демонстрация функции ввода информации о фильме.
(Рисунок 6):

```

Film 1th :

Film: haha
Duration(minutes): 120
Year of issue: 1999
Genre: comandy
Rate: 10
Press any key to continue . . .

```

Рисунок 6 – Демонстрация функции ввода информации о фильме.

В задаче 7 представлена демонстрационная функция, которая вводит информацию о фильме и выводит адрес (Рисунок 7)

```
Enter the title of the movieHaha
Enter rating of the movie: 9
Enter genre of the movie: Comedy
Duration(minutes): 120
Enter year of issue: 1888

Address Flight: 00CFF96C
Address flight: 00FD0E90
Address Movie: 00CFF91C
Address movie: 00FD0EF8
If the program started, then you did not remove /**/ in the function.
Press any key to continue . . .
```

Рисунок 7 – Демонстрация функции

В задаче 8 представлена демонстрационная функция, которая считывает и записывает прямоугольную информацию. (Рисунок 8)

```
10
Enter number of rectangles: 2

Enter length: 10
Enter width: 8
Enter length: 10
Enter width: 7
Rectangle size: 10x8
Rectangle size: 10x7
Press any key to continue . . .
```

Рисунок 8 – Демонстрация функции считывает и записывает прямоугольную информацию

В задаче 9 представлена демонстрационная функция, которая считывает и записывает информацию о двух прямоугольниках и одновременно изменяет информацию (Рисунок 9)


```
11
Enter length: 10
Enter width: 8
Enter length: 10
Enter width: 7

Information rectangles:
Rectangle1:
Rectangle size: 10x8

Rectangle2:
Rectangle size: 10x7

After Exchange:
Rectangle1:
Rectangle size: 10x7

Rectangle2:
Rectangle size: 10x8
Press any key to continue . . .
```

Рисунок 8 – Демонстрационная функция для изменения прямоугольной информации

В задаче 10 есть демонстрационная функция для ввода прямоугольной информации и поиска самого большого прямоугольника (Рисунок 10)

```
Enter the number of rectangle: 3
Rectangle 1
Enter length: 5
Enter width: 4
Rectangle 2
Enter length: 6
Enter width: 5
Rectangle 3
Enter length: 7
Enter width: 6
Rectangle with the biggest length: 7x6
```

Рисунок 10– Демонстрационная функция для нахождения наибольшей длины

В задаче 11 есть демонстрационная функция для ввода прямоугольной информации и поиска прямоугольника с наибольшей площадью (рис. 11).

```

Enter length: 5

Enter width: 4
||fTÇTÄ||J||Tâ||J||Tî||J|| 2

Enter length: 7

Enter width: 3

Rectangle with the biggest area: 7x3
Press any key to continue . . .

```

Рисунок 11– Демонстрационная функция для нахождения наибольшей площади

В задаче 12 есть демонстрационная функция для вывода циклической информации (рис. 12).

```

Circle:
X = 3; Y = 5; Radius = 6.5; Color = Red;
X = 4; Y = -5; Radius = 7; Color = Pink;
X = 0; Y = 0; Radius = 5; Color = Blue;
X = 3; Y = 5; Radius = 6.5; Color = Red;
X = 4; Y = -5; Radius = 7; Color = Pink;
X = 0; Y = 0; Radius = 5; Color = Blue;
Press any key to continue . . .

```

Рис. 12 – Демонстрационная функция для экспорта циклической информации

В задаче 13 есть демонстрация, которая ищет фильмы в списке (рис. 13).

```

Duration(minute): 120

Year of issue: 1999

Enter film title: Fast and Furius 2

Enter film rating:
Enter genre:
0 - Comedy; 1 - Drama; 2 - Thriller; 3 - Action; 4 - Horrors; 5 - Crimainal
Enter genre: Please enter a valid value!
3

Duration(minute): 120

Year of issue: 1999

0 - Comedy; 1 - Drama; 2 - Thriller; 3 - Action; 4 - Horrors; 5 - Crimainal
Enter the number of genre looking for: 2

Highest rated movie in selected genre:

Not in the movie list!Press any key to continue . . .

```

Рис. 13 – Демо поиск фильмов в списке

В задаче 14 есть демонстрация функция инициализирует информацию о рейсах и фильмах, полет. время и создает копии (рис. 14).

```
Flight:
Flight: Moscow -- Tokyo
FlightCopy:
Flight: Moscow -- Tokyo
Press any key to continue . . .

Movie:
Film: Tom and Jerry
Duration(minutes): 400
Year of issue: 1999
Genre: Comedy
Rate: 9

MovieCopy:
Film: Tom and Jerry
Duration(minutes): 400
Year of issue: 1999
Genre: Comedy
Rate: 9
Press any key to continue . . .
```

Рис. 14 – Демонстрационная функция инициализирует информацию о создании копии

4. Вывод

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