

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

По дисциплине Объектно-ориентированное программирование

Тема работы Создание и использование размерных типов данных

Обучающийся Буров Глеб Максимович

Факультет факультет инфокоммуникационных технологий

Группа K3223

Направление подготовки 11.03.02 Инфокоммуникационные технологии и системы связи

Образовательная программа Программирование в инфокоммуникационных системах

Обучающийся

(дата)

(подпись)

Буров Г.М.
(Ф.И.О.)

Руководитель

(дата)

(подпись)

Иванов С.Е.
(Ф.И.О.)

СОДЕРЖАНИЕ

Стр.

ВВЕДЕНИЕ	3
1 Ход работы	4
1.1 Упражнение 1	4
1.2 Упражнение 2	5
1.3 Упражнение 3	6
ЗАКЛЮЧЕНИЕ	9

ВВЕДЕНИЕ

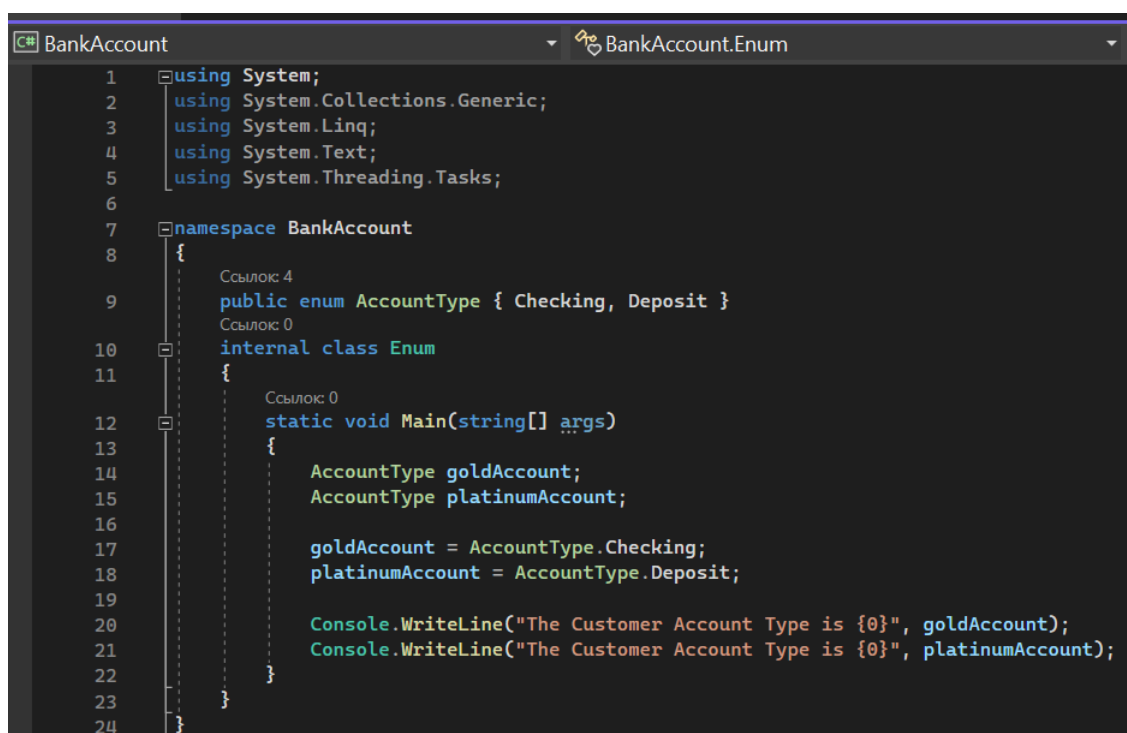
Целью данной лабораторной работы является изучение размерных типов, данных и приобретение навыков работы со структурными типами.

1 Ход работы

1.1 Упражнение 1

В первом упражнении необходимо создать перечисление для представления различных типов банковских счетов. Для этого был создан новый проект под названием BankAccount, в коде которого был описан enum AccountType, содержащий значения Checking и Deposit — типы банковского аккаунта.

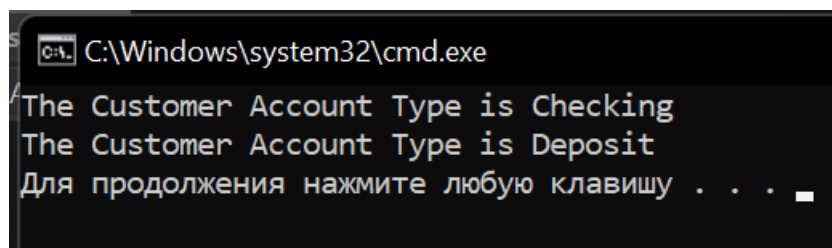
После этого в методе Main были определены две переменные типа AccountType, одной из которых было присвоено значение Checking, а другой — Deposit (рис. 1.1).



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace BankAccount
8 {
9     public enum AccountType { Checking, Deposit }
10    internal class Enum
11    {
12        static void Main(string[] args)
13        {
14            AccountType goldAccount;
15            AccountType platinumAccount;
16
17            goldAccount = AccountType.Checking;
18            platinumAccount = AccountType.Deposit;
19
20            Console.WriteLine("The Customer Account Type is {0}", goldAccount);
21            Console.WriteLine("The Customer Account Type is {0}", platinumAccount);
22        }
23    }
24 }
```

Рисунок 1.1 — Код программы из упражнения 1

Результат вывода программы показан на рисунке 1.2.



```
C:\Windows\system32\cmd.exe
The Customer Account Type is Checking
The Customer Account Type is Deposit
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1.2 — Результат работы программы из упражнения 1

1.2 Упражнение 2

Во втором упражнении необходимо было создать структуру, которую можно использовать для представления банковских счетов. При этом, для хранения номеров счетов (тип данных `long`), балансов счетов (тип данных `decimal`) и типов счетов (перечисление из прошлого упражнения) используются переменные.

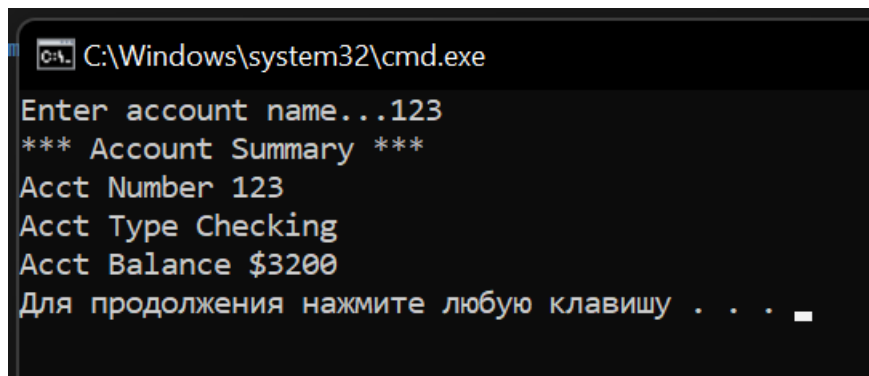
Упражнение было выполнено в новом проекте `StructType`, в котором была объявлена и описана структура `BankAccount` с публичным модификатором доступа. Все поля также имели модификатор доступа `public`.

Далее в методе `Main` был объявлен экземпляр `goldAccount` структуры `BankAccount`, полям которого были установлены значения (см. рис 1.3). Также был реализован диалог с пользователем — программа принимает на вход номер аккаунта.

```
8 namespace StructType
9 {
10     Ссылка: 0
11     internal class Struct
12     {
13         Ссылка: 2
14         public enum AccountType { Checking, Deposit }
15
16         Ссылка: 1
17         public struct BankAccount
18         {
19             public long accNo;
20             public decimal accBal;
21             public AccountType accType;
22         }
23
24         Ссылка: 0
25         static void Main(string[] args)
26         {
27             BankAccount goldAccount;
28             goldAccount.accType = AccountType.Checking;
29             goldAccount.accBal = (decimal)3200.00;
30             Console.WriteLine("Enter account name...");
31             goldAccount.accNo = long.Parse(Console.ReadLine());
32
33             Console.WriteLine("*** Account Summary ***");
34             Console.WriteLine("Acct Number {0}", goldAccount.accNo);
35             Console.WriteLine("Acct Type {0}", goldAccount.accType);
36             Console.WriteLine("Acct Balance ${0} ",goldAccount.accBal);
37         }
38     }
39 }
```

Рисунок 1.3 — Код программы из упражнения 2

Пример работы программы программы показан на рисунке 1.4.



```
C:\Windows\system32\cmd.exe
Enter account name...123
*** Account Summary ***
Acct Number 123
Acct Type Checking
Acct Balance $3200
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1.4 — Результат работы программы из упражнения 2

1.3 Упражнение 3

В третьем задании необходимо было самостоятельно реализовать структуру `Distance`, которая определяет длину в английской системе мер. Полями в данной структуре послужили `inch` (дюймы) и `ft` (футы), оба типа `int`.

После этого в методе `Main` создаются три экземпляра данной структуры, два из которых инициализируются пользователем с клавиатуры. Стоит отметить, что можно было обернуть преобразование вводимых данных в тип `int` в конструкцию `try/catch`, чтобы предусмотреть ситуацию некорректного ввода — я оставил это на совести пользователя.

После ввода данных, они приводятся к корректному виду: футов должно быть не более 12.

Третий экземпляр получается сложением значений первого и второго экземпляров (см. рис. 1.5). Данные выводятся на экран.

```

Ссылка: 3
public struct Distance
{
    public int inch;
    public int ft;
}

Ссылка: 0
static void Main(string[] args)
{
    Distance firstDistance;
    Distance secondDistance;
    Distance thirdDistance;

    Console.WriteLine("*** First variable initialization *** ");

    Console.WriteLine("Enter inch count: ");
    firstDistance.inch = int.Parse(Console.ReadLine());
    Console.WriteLine("Enter the feet count: ");
    firstDistance.ft = int.Parse(Console.ReadLine());

    firstDistance.inch += (int)(firstDistance.ft / 12);
    firstDistance.ft = (int)(firstDistance.ft % 12);

    Console.WriteLine("First variable length: {0}\' - {1}\'",
        firstDistance.inch, firstDistance.ft);
    Console.WriteLine();

    Console.WriteLine("*** Second variable initialization *** ");

    Console.WriteLine("Enter inch count: ");
    secondDistance.inch = int.Parse(Console.ReadLine());
    Console.WriteLine("Enter the feet count: ");
    secondDistance.ft = int.Parse(Console.ReadLine());

    secondDistance.inch += (int)(secondDistance.ft / 12);
    secondDistance.ft = (int)(secondDistance.ft % 12);

    Console.WriteLine("Second variable length: {0}\' - {1}\'",
        secondDistance.inch, secondDistance.ft);
    Console.WriteLine();

    thirdDistance.inch = firstDistance.inch + secondDistance.inch;
    thirdDistance.ft = firstDistance.ft + secondDistance.ft;

    thirdDistance.inch += (int)(thirdDistance.ft / 12);
    thirdDistance.ft = (int)(thirdDistance.ft % 12);

    Console.WriteLine("Third variable length: {0}\' - {1}\'",
        thirdDistance.inch, thirdDistance.ft);
}
}

```

Рисунок 1.5 — Код программы из упражнения 3

Пример работы программы программы показан на рисунке 1.6.

```
*** First variable initialization ***  
Enter inch count:  
6  
Enter the feet count:  
14  
First variable length: 7' - 2"  
  
*** Second variable initialization ***  
Enter inch count:  
2  
Enter the feet count:  
11  
Second variable length: 2' - 11"  
  
Third variable length: 10' - 1"  
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1.6 — Результат работы программы из упражнения 3

ЗАКЛЮЧЕНИЕ

В ходе этой лабораторной работы я поработал с размерными типами, ознакомился со структурами, и перечислениями, создав свои собственные. Все упражнения были выполнены.