

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра САПР**

**ОТЧЕТ**  
**по лабораторной работе №7**  
**по дисциплине «Объектно-ориентированное**  
**программирование»**  
**Тема: «Использование переменных ссылочного типа»**

Студенты гр. 1302

\_\_\_\_\_ Романова О.В.

\_\_\_\_\_ Новиков Г.В.

Преподаватель:

\_\_\_\_\_ Васильев А.А.

Санкт-Петербург

2023

## **1. Цель работы**

Изучение переменных ссылочного типа в языке C# с помощью программного продукта компании Microsoft VS 2022.

## **2. Анализ задачи**

Необходимо:

- 1) Написать программу, которая обеспечивает перевод денег с одного счета на другой;
- 2) Написать программу, которая читает строку символов и переносит ее в другую переменную в обратном порядке;
- 3) Написать программу, которая считывает текст из одного файла и переписывает в другой файл, переводя все буквы в верхний регистр.
- 4) Написать программу, которая добавляет статический метод `IsItFormattable`, который определяет поддерживается ли передающийся в этот метод объект `IFormattable`, в класс `Utils`;
- 5) Написать программу, которая добавляет метод `Display`, который использует оператор `as` для определения передачи объекта как параметра в интерфейс `IPrintable`.

## **3. Ход выполнения работы**

### **3.1 Упражнение 1**

В ходе выполнения данного упражнения написана программа, которая переводит деньги с одного аккаунта на другой.

#### **3.1.1 Пошаговое описание алгоритма**

На счет каждого аккаунта кладется 100.

На экран пользователя выводится состояние аккаунтов до операции перевода.

Вызывается метод `TransferFrom` для перевода денег с одного аккаунта на другой.

На экран пользователя выводится состояние аккаунтов после операции перевода.

### **3.1.2 Используемые классы и методы**

В программе, написанной в данном упражнении, используются следующие методы:

- `System.Console.WriteLine()` – служит для отображения на экране строк и значений переменных, переданных в метод в качестве параметров, с переходом на новую строку;

- `System.Console.ReadKey()` – ожидает следующего нажатия клавиши пользователем;

- `Main()` – служит для запуска программы.

- `BankAccount` – класс, который описывает банковский аккаунт с помощью методов: `TransferFrom` (перевод денег), `Number` (возвращает номер аккаунта), `Balance` (возвращает текущий баланс), `Type` (возвращает тип аккаунта), `Withdraw` (позволяет произвести снятие денег) и `Deposit` (позволяет пополнить счет), `Populate` (создание), `NextNumber` (следующий номер счета).

### **3.1.3 Контрольный пример**

На рис.3.1 представлены результаты выполнения программы.

```
Before:
Account number is 123
Account balance is 100
Account type is Checking
Account number is 124
Account balance is 100
Account type is Checking
After:
Account number is 123
Account balance is 110
Account type is Checking
Account number is 124
Account balance is 90
Account type is Checking
```

Рис.3.1 Контрольный пример для программы 1

Как видно из рисунка, на экран выводятся данные аккаунтов до и после перевода денег.

#### 4. Листинг программы

##### Test.cs:

```
using System;

/// <summary>
///     Test harness.
/// </summary>

public class Test
{
    public static void Main()
    {
        BankAccount b1 = new BankAccount(), b2 = new BankAccount();
        b1.Populate(100);
        b2.Populate(100);

        Console.WriteLine("Before: ");
        Console.WriteLine("Account number is {0}", b1.Number());
        Console.WriteLine("Account balance is {0}", b1.Balance());
        Console.WriteLine("Account type is {0}", b1.Type());

        Console.WriteLine("Account number is {0}", b2.Number());
        Console.WriteLine("Account balance is {0}", b2.Balance());
        Console.WriteLine("Account type is {0}", b2.Type());

        b1.TransferFrom(b2, 10);

        Console.WriteLine("After: ");
        Console.WriteLine("Account number is {0}", b1.Number());
        Console.WriteLine("Account balance is {0}", b1.Balance());
        Console.WriteLine("Account type is {0}", b1.Type());

        Console.WriteLine("Account number is {0}", b2.Number());
```

```

        Console.WriteLine("Account balance is {0}", b2.Balance());
        Console.WriteLine("Account type is {0}", b2.Type());
    }
}

```

## AccountType.cs

```

enum AccountType
{
    Checking,
    Deposit
}

```

## BankAccount.cs

```

class BankAccount
{
    private long accNo;
    private decimal accBal;
    private AccountType accType;

    private static long nextNumber = 123;

    public void Populate(decimal balance)
    {
        accNo = NextNumber();
        accBal = balance;
        accType = AccountType.Checking;
    }

    public bool Withdraw(decimal amount)
    {
        bool sufficientFunds = accBal >= amount;
        if (sufficientFunds) {
            accBal -= amount;
        }
        return sufficientFunds;
    }

    public decimal Deposit(decimal amount)
    {
        accBal += amount;
        return accBal;
    }

    public long Number()
    {
        return accNo;
    }

    public decimal Balance()
    {
        return accBal;
    }

    public string Type()
    {
        return accType.ToString();
    }

    private static long NextNumber()
    {
        return nextNumber++;
    }
}

```

```

public void TransferFrom(BankAccount accForm, decimal amount)
{
    if (accForm.Withdraw(amount))
        this.Deposit(amount);
}
}

```

## 3.2 Упражнение 2

В ходе выполнения данного упражнения, написана программа, которая переносит строку символов в другую переменную в обратном порядке.

### 3.2.1 Пошаговое описание алгоритма

Ввод пользователем сообщения.

Вызов функции Reverse для изменения порядка на обратный.

Вывод сообщения введенного пользователем в обратном порядке.

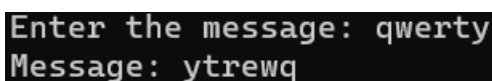
### 3.2.2 Используемые классы и методы

В программе, написанной в данном упражнении, используются следующие методы:

- `System.Console.WriteLine()` – служит для отображения на экране строк и значений переменных, переданных в метод в качестве параметров, с переходом на новую строку;
- `System.Console.ReadKey()` – ожидает следующего нажатия клавиши пользователем;
- `Main()` – служит для запуска программы.
- `Utils` – класс, который взят из прошлой лабораторной работы и добавлен метод `Reverse`.

### 3.2.3 Контрольный пример

На рис.3.2 представлены результаты выполнения программы.



```

Enter the message: qwerty
Message: ytrewq

```

Рис.3.2 Контрольный пример для программы 2

Как видно из рисунка, пользователь вводит сообщение, затем выводится перевернутое сообщение.

#### 4. Листинг программы

##### Test.cs:

```
namespace Utils
{
    using System;

    /// <summary>
    ///     This the test harness
    /// </summary>

    public class Test
    {
        public static void Main()
        {
            string message;
            Console.Write("Enter the message: ");
            message = Console.ReadLine();
            Utils.Reverse(ref message);
            Console.WriteLine("Message: {0}", message);
        }
    }
}
```

##### Utils.cs:

```
namespace Utils
{
    using System;

    class Utils
    {
        //
        // Return the larger of two integer values
        //

        public static int Greater(int a, int b)
        {
            if (a > b)
                return a;
            else
                return b;

            // Alternative version - more terse
            // return (a>b) ? (a) : (b);
        }

        //
        // Swap two integers, passed by reference
        //

        public static void Swap(ref int a, ref int b)
        {
            int temp;
            temp = a;
            a = b;
        }
    }
}
```

```

    b = temp;
}

//
// Calculate factorial
// and return the result as an out parameter
//

public static bool Factorial(int n, out int answer)
{
    int k;          // loop counter
    int f;          // working value
    bool ok = true; // true if ok, false if not

    // Check the input value

    if (n < 0)
        ok = false;

    // Calculate the factorial value as the
    // product of all the numbers from 2 to n

    try
    {
        checked
        {
            f = 1;
            for (k = 2; k <= n; ++k)
            {
                f = f * k;
            }

            // Here is a terse alternative
            // for (f=1,k=2;k<=n;++k)
            //     f*=k;

        }
    }
    catch (Exception)
    {
        // If something goes wrong in the calculation,
        // catch it here. All exceptions
        // are handled the same way: set the result to
        // to zero and return false.

        f = 0;
        ok = false;
    }

    // assign result value
    answer = f;

    // return to caller
    return ok;
}

//
// Another way to solve the factorial problem, this time
// as a recursive function
//

public static bool RecursiveFactorial(int n, out int f)
{
    bool ok = true;

```



```

        // Trap negative inputs
        if (n < 0)
        {
            f = 0;
            ok = false;
        }

        if (n <= 1)
            f = 1;
        else
        {
            try
            {
                int pf;
                checked
                {
                    ok = RecursiveFactorial(n - 1, out pf);
                    f = n * pf;
                }
            }
            catch (Exception)
            {
                // Something went wrong. Set error
                // flag and return zero.
                f = 0;
                ok = false;
            }
        }

        return ok;
    }

    public static void Reverse (ref string s)
    {
        string sRev = "";
        for (int i = s.Length - 1; i >= 0; i--)
        {
            sRev += s[i];
        }
        s = sRev;
    }
}

```

### 3.3 Упражнение 3

В ходе выполнения данного упражнения, написана программа, которая считывает текст из одного файла и переписывает в другой файл, переводя все буквы в верхний регистр.

#### 3.3.1 Пошаговое описание алгоритма

Пользователем вводится название входного и выходного файла.

Сообщение из одного файла передается в другой с верхним регистром.

В случае ошибки выводится сообщение об ошибке.

### 3.3.2 Используемые классы и методы

В программе, написанной в данном упражнении, используются следующие методы:

- `System.Console.WriteLine()` – служит для отображения на экране строк и значений переменных, переданных в метод в качестве параметров, с переходом на новую строку;
- `System.Console.ReadKey()` – ожидает следующего нажатия клавиши пользователем;
- `Main()` – служит для запуска программы.

### 3.3.3 Контрольный пример

На рис.3.3 представлены результаты выполнения программы.

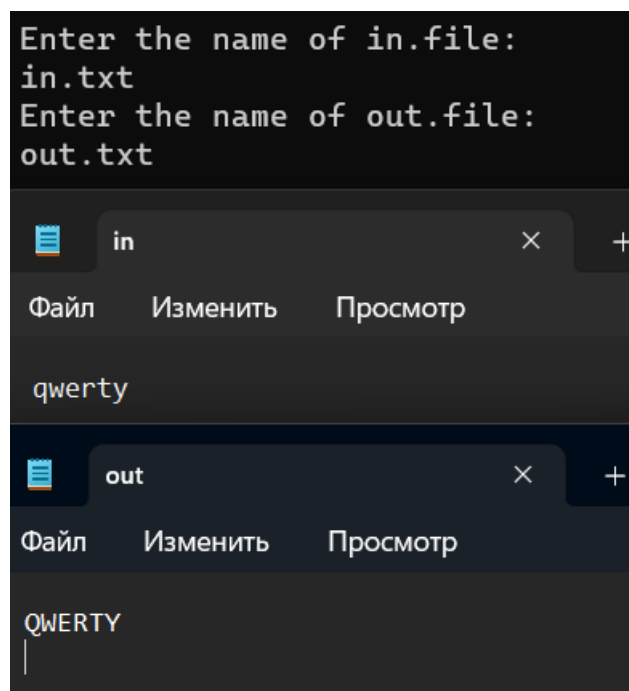


Рис.3.3 Контрольный пример для программы 3

Как видно из рисунка, пользователем вводятся названия файлов и в выходном файле появляется сообщение из входного файла в верхнем регистре.

## 4. Листинг программы

```
using System;
using System.IO;

/// <summary>
///     Class to create an upper case copy of a file
/// </summary>
public class CopyFileUpper
{
    public static void Main()
    {
        string sFrom, sTo;
        StreamReader srFrom;
        StreamWriter swTo;

        Console.WriteLine("Enter the name of in.file: ");
        sFrom = Console.ReadLine();

        Console.WriteLine("Enter the name of out.file: ");
        sTo = Console.ReadLine();

        try
        {
            srFrom = new StreamReader(sFrom);
            swTo = new StreamWriter(sTo);

            while (srFrom.Peek() != -1)
            {
                string sBuffer = srFrom.ReadLine();
                sBuffer = sBuffer.ToUpper();
                swTo.WriteLine(sBuffer);
            }
            swTo.Close();
            srFrom.Close();
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine("File not found.");
        }
        catch (Exception exc)
        {
            Console.WriteLine("Exception: {0}", exc.ToString());
        }
    }
}
```

### 3.4 Упражнение 4

В ходе выполнения данного упражнения, написана программа, которая добавляет статический метод.

#### 3.4.1 Пошаговое описание алгоритма

Инициализируются три переменные.

Переменные выводятся на экран после метода `IsItFormattable`.

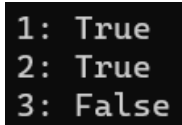
### 3.4.2 Используемые классы и методы

В программе, написанной в данном упражнении, используются следующие методы:

- `System.Console.WriteLine()` – служит для отображения на экране строк и значений переменных, переданных в метод в качестве параметров, с переходом на новую строку;
- `System.Console.ReadKey()` – ожидает следующего нажатия клавиши пользователем;
- `Main()` – служит для запуска программы.
- `Utils` – класс, который взят из прошлой лабораторной работы и добавлен метод `IsItFormattable`.

### 3.4.3 Контрольный пример

На рис.3.4 представлены результаты выполнения программы.



```
1: True
2: True
3: False
```

Рис.3.4 Контрольный пример для программы 4

Как видно из рисунка, выводятся результаты после метода `IsItFormattable`.

## 4. Листинг программы

```
Test.cs
namespace Utils
{
    using System;

    /// <summary>
    ///     This the test harness
    /// </summary>

    public class Test
    {
        public static void Main()
```

```

    {
        int i = 0;
        ulong ul = 0;
        string s = "Test";

        Console.WriteLine("1: {0}", Utils.IsItFormattable(i));
        Console.WriteLine("2: {0}", Utils.IsItFormattable(ul));
        Console.WriteLine("3: {0}", Utils.IsItFormattable(s));
    }
}

```

## Utils.cs

```

namespace Utils
{
    using System;

    class Utils
    {
        //
        // Return the larger of two integer values
        //

        public static int Greater(int a, int b)
        {
            if (a > b)
                return a;
            else
                return b;

            // Alternative version - more terse
            // return (a>b) > (a) : (b);
        }

        //
        // Swap two integers, passed by reference
        //

        public static void Swap(ref int a, ref int b)
        {
            int temp;
            temp = a;
            a = b;
            b = temp;
        }

        //
        // Calculate factorial
        // and return the result as an out parameter
        //

        public static bool Factorial(int n, out int answer)
        {
            int k;          // loop counter
            int f;          // working value
            bool ok = true; // true if ok, false if not

            // Check the input value

            if (n < 0)
                ok = false;

            // Calculate the factorial value as the
            // product of all the numbers from 2 to n

```

```

try
{
    checked
    {
        f = 1;
        for (k = 2; k <= n; ++k)
        {
            f = f * k;
        }

        // Here is a terse alternative
        // for (f=1,k=2;k<=n;++k)
        //     f*=k;
    }
}
catch (Exception)
{
    // If something goes wrong in the calculation,
    // catch it here. All exceptions
    // are handled the same way: set the result to
    // to zero and return false.

    f = 0;
    ok = false;
}

// assign result value
answer = f;

// return to caller
return ok;
}

//
// Another way to solve the factorial problem, this time
// as a recursive function
//

public static bool RecursiveFactorial(int n, out int f)
{
    bool ok = true;

    // Trap negative inputs
    if (n < 0)
    {
        f = 0;
        ok = false;
    }

    if (n <= 1)
        f = 1;
    else
    {
        try
        {
            int pf;
            checked
            {
                ok = RecursiveFactorial(n - 1, out pf);
                f = n * pf;
            }
        }
        catch (Exception)
    }
}

```

```

        {
            // Something went wrong. Set error
            // flag and return zero.
            f = 0;
            ok = false;
        }

    }

    return ok;
}

public static bool IsItFormattable(object x)
{
    if( x is IFormattable)
    {
        return true;
    }
    else
        return false;
}
}
}

```

### 3.5 Упражнение 5

В ходе выполнения данного упражнения, написана программа, которая добавляет метод Display, который использует оператор as для определения передачи объекта как параметра в интерфейс IPrintable.

#### 3.5.1 Пошаговое описание алгоритма

Переменные инициализируются.

Метод Display.

Вывод результата на экран.

#### 3.5.2 Используемые классы и методы

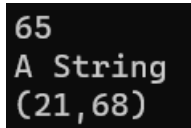
В программе, написанной в данном упражнении, используются следующие методы:

- System.Console.WriteLine() – служит для отображения на экране строк и значений переменных, переданных в метод в качестве параметров, с переходом на новую строку;
- System.Console.ReadKey() – ожидает следующего нажатия клавиши пользователем;
- Main() – служит для запуска программы.

- Utils – класс, который взят из прошлой лабораторной работы и добавлен метод Display.

### 3.5.3 Контрольный пример

На рис.3.5 представлены результаты выполнения программы.



```
65
A String
(21,68)
```

Рис.3.5 Контрольный пример для программы 4

Как видно из рисунка, выводятся результаты после метода Display.

## 4. Листинг программы

### Test.cs:

```
namespace Utils
{
    using System;

    /// <summary>
    ///   This the test harness
    /// </summary>

    public class Test
    {
        public static void Main()
        {
            int num = 65;
            string msg = "A String";
            Coordinate c = new Coordinate(21.0,68.0);
            Utils.Display(num);
            Utils.Display(msg);
            Utils.Display(c);
        }
    }
}
```

### Coordinate.cs:

```
namespace Utils
{
    using System;

    // Sample class that implements the IPrintable interface

    class Coordinate : IPrintable
    {
        private double x;
        private double y;
    }
}
```



```

        public Coordinate() {
            x=0.0;
            y=0.0;
        }

        public Coordinate(double px,double py) {
            x=px;
            y=py;
        }

        public void Print() {
            Console.WriteLine("{0},{1}", x, y);
        }
    }
}

```

### Utils.cs:

```

namespace Utils
{
    using System;

    class Utils
    {

        public static bool IsItFormattable(object x)
        {
            // Use is to test if the object has the
            // IFormattable interface

            if (x is IFormattable)
                return true;
            else
                return false;
        }

        //
        // Return the larger of two integer values
        //
        public static int Greater(int a, int b)
        {
            if (a > b)
                return a;
            else
                return b;

            // Alternative version - more terse
            // return (a>b) ? (a) : (b);
        }

        //
        // Swap two integers, passed by reference
        //
        public static void Swap(ref int a, ref int b)
        {
            int temp;
            temp = a;
            a = b;
            b = temp;
        }

        //
        // Calculate factorial
    }
}

```

```

// and return the result as an out parameter
//

public static bool Factorial(int n, out int answer)
{
    int k;          // loop counter
    int f;          // working value
    bool ok = true; // true if ok, false if not

    // Check the input value

    if (n < 0)
        ok = false;

    // Calculate the factorial value as the
    // product of all the numbers from 2 to n

    try
    {
        checked
        {
            f = 1;
            for (k = 2; k <= n; ++k)
            {
                f = f * k;
            }

            // Here is a terse alternative
            // for (f=1,k=2;k<=n;++k)
            //     f*=k;
        }
    }
    catch (Exception)
    {
        // If something goes wrong in the calculation,
        // catch it here. All exceptions
        // are handled the same way: set the result to
        // to zero and return false.

        f = 0;
        ok = false;
    }

    // assign result value
    answer = f;

    // return to caller
    return ok;
}

//
// Another way to solve the factorial problem, this time
// as a recursive function
//

public static bool RecursiveFactorial(int n, out int f)
{
    bool ok = true;

    // Trap negative inputs
    if (n < 0)
    {
        f = 0;
        ok = false;
    }

```

```

    }

    if (n <= 1)
        f = 1;
    else
    {
        try
        {
            int pf;
            checked
            {
                ok = RecursiveFactorial(n - 1, out pf);
                f = n * pf;
            }
        }
        catch (Exception)
        {
            // Something went wrong. Set error
            // flag and return zero.
            f = 0;
            ok = false;
        }
    }

    return ok;
}

public static void Display(object item)
{
    IPrintable ip;

    ip = item as IPrintable;

    if (ip!=null)
    {
        ip.Print();
    }
    else if (ip == null)
    {
        Console.WriteLine(item.ToString());
    }
}
}
}

```

### **IPrintable.cs:**

```

namespace Utils
{
    using System;

    interface IPrintable
    {
        void Print();
    }
}

```

## **5. Полученные результаты**

В ходе выполнения данной лабораторной работы нами были получены следующие результаты:

- В ходе работы программы 1 были созданы методы с параметрами и добавлены в класс.
- В ходе программы 2 были использованы методы со ссылочными параметрами.
- В ходе программы 3 были преобразованы символы файла в верхний регистр.
- В ходе программы 4 была проведена проверка реализации интерфейса.
- В ходе программы 5 была произведена работа с интерфейсами.

## **6. Выводы**

В ходе выполнения данной лабораторной работы:

- были изучены переменные ссылочного типа в языке C#;
- были изучены интерфейсы в языке C#.

## **7. Список использованной литературы**

1. MSDN — сеть разработчиков Microsoft. URL:  
<https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/reference-types> (дата обращения: 14.04.2023)