**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. 9301** |  | **Власов Е.А.** |
|  |  | **Токарев С.В.** |
|  |  | **Служевская А.С.** |
| **Преподаватель** |  | **Новакова Н.Е.** |

**Санкт-Петербург**

**2021**

# Цель работы

# Научиться передавать данные по ссылке и работать с файлами, ознакомиться с интерфесами.

# Ход работы Упражнение 1

* 1. Отредактировать файл BankAccount.cs; внести метод TransferFrom(), который переводит деньги с одного аккаунта на другой.
  2. Протестировать новый метод с выводом на экран информации о банковских счетах до и после перевода денег.
  3. Метод TranferFrom() содержит два параметра. Первый имеет тип BankAccount и передается по ссылке, второй имеет тип decimal. У первого параметра будет вызван метод Withdraw(), снимающий указанную сумму со счета, если таковая имеется; у второго будет вызван метод Deposite(), зачисляющий указанную сумму на счет.

**Упражнение 2**

* 1. Добавить метод Reverse(), который переворачивает строку.
  2. Протестировать метод.
  3. Метод Reverse() имеет параметр типа string. Внутри находится цикл, который проходит от конца до начала строки из параметра и побуквенно записывает ее во временную строку. После конца цикла строка-параметр получает значение из временной строки.

# Упражнение 3

* 1. Программа получает на вход имена файлов для чтения данных и записи результата.
  2. Создаются StreamReader — файловый поток на чтение с первым именем и StreamWriter — файловый поток на запись со вторым именем.
  3. В цикле while выполняется построчное считывание файла, перевод строки в верхний регистр и запись полученной строки в другой файл, пока метод Peek, который читает файл посимвольно, не выдаст -1, что означает конец файла.
  4. При неправильно указанном пути к любому из файлов или возникновении другой ошибки программа сообщает об этом.

# Упражнение 4

# Отредактировать класс Utils, добавив в него метод IsItFormattable.

# Метод IsItFormattable содержит в себе один параметр типа object.

# Метод проверяет, поддерживает ли объект, который в него передается, System.IFormattable и возвращает соответствующее значение типа bool

# Упражнение 5

# Отредактировать класс Utils, добавив в него метод Display.

# Метод Display получает параметр типа object.

# Этот метод проверяет реализует ли параметр IPrintable. Если да, то вызывает метод Print, если нет, то присваивает значение null.

# Диаграмма классов

# 

# Текст программы

# Упражнение 1

using System;

namespace main

{

enum AccountType

{

Checking,

Deposit

}

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()

{

string accountType = accType.ToString();

return accountType;

}

private static long NextNumber()

{

return nextNumber++;

}

public void TransferFrom(ref BankAccount accForm, decimal ammount)

{

if (accForm.Withdraw(ammount))

{

Deposit(ammount);

}

}

}

public class Test

{

public static void Main()

{

BankAccount b1 = new BankAccount();

BankAccount b2 = new BankAccount();

b1.Populate(100);

b2.Populate(100);

AccountInfo(ref b1);

AccountInfo(ref b2);

b1.TransferFrom(ref b2, 10);

AccountInfo(ref b1);

AccountInfo(ref b2);

}

static void AccountInfo(ref BankAccount getInfo)

{

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

Console.WriteLine("Account balance is {0}", getInfo.Balance());

Console.WriteLine("Account type is {0}", getInfo.Type());

Console.WriteLine(" ");

}

}

}

# Упражнение 2

using System;

namespace main

{

enum AccountType

{

Checking,

Deposit

}

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()

{

string accountType = accType.ToString();

return accountType;

}

private static long NextNumber()

{

return nextNumber++;

}

public void TransferFrom(ref BankAccount accForm, decimal ammount)

{

if (accForm.Withdraw(ammount))

{

Deposit(ammount);

}

}

public static void Reverse(ref string s)

{

string sRev = "";

for (int i=s.Length - 1; i >= 0; i--)

{

sRev += s[i];

}

s = sRev;

}

}

public class Test

{

public static void Main()

{

string message = Console.ReadLine();

BankAccount.Reverse(ref message);

Console.WriteLine(message);

}

static void AccountInfo(ref BankAccount getInfo)

{

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

Console.WriteLine("Account balance is {0}", getInfo.Balance());

Console.WriteLine("Account type is {0}", getInfo.Type());

Console.WriteLine(" ");

}

}

}

# Упражнение 3

using System;

using System.IO;

namespace OOP\_7\_3

{

public class CopyFileUpper

{

public static void Main()

{

string sFrom, sTo;

StreamReader srFrom;

StreamWriter swTo;

Console.WriteLine("Enter file name for input: ");

sFrom = Console.ReadLine();

Console.WriteLine("Enter file name for output: ");

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("Couldn't find a file");

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

}

# Упражнение 4

using System;

namespace упражнение4

{

using System;

class main

{

//

// 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;

}

}

public static void Main()

{

int i = 0;

ulong ul = 0;

string s = "Test";

Console.WriteLine(IsItFormattable(i));

Console.WriteLine(IsItFormattable(ul));

Console.WriteLine(IsItFormattable(s));

}

}

}

# Упражнение 5

using System;

namespace main

{

class Utils

{

interface IPrintable

{

void Print();

}

// 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);

}

}

public static bool IsItFormattable(object x)

{

// Use is to test if the object has the

// IFormattable iterface

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 = item as IPrintable;

if (ip != null)

{

ip.Print();

}

else

{

Console.WriteLine(Convert.ToString(item));

}

}

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);

}

}

}

}

# Примеры работы программы

На Рис. 1 представлена работа программы упражнения 1.

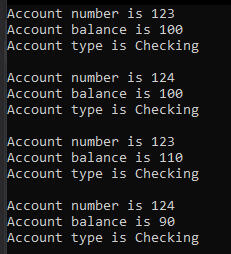


Рис. 1

На Рис. 2 представлена работа программы упражнения 2.

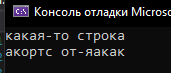


Рис. 2

На Рис. 3 представлена работа программы упражнения 3.

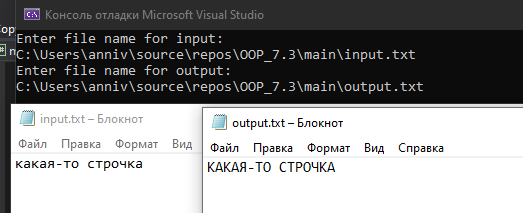


Рис. 3

На Рис. 4 представлена работа программы упражнения 4.

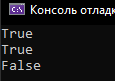


Рис. 4

На Рис. 5 представлена работа программы упражнения 5.



Рис. 5

# Вывод

В ходе работы была проведена работы с вводом и выводом из файла, были рассмотрен интерфейсы. Так же было выяснено, что типы данных int и ulong поддерживают интерфейс IFormatable, а string — нет.