# 实验六 数组、索引器、集合编程练习

## 【实验目的】

1.了解数组的概念，掌握数组的声明、初始化方式；

2. 熟练掌握foreach语句的用法，理解params关键字的使用；

3.了解多维数组的定义和声明，理解规则数组和不规则数组的使用区别；

4. 掌握数组转换和数组排序方法的使用；

5. 理解索引器的概念，掌握在整数索引和在字符串上索引的应用；

## 【实验要求】

1. 理解应用数组实现功能的程序；

2. 写出使用数组实现的程序；

3. 调试程序，要给出测试数据和实验结果。

4. 整理上机步骤，总结经验和体会。

5. 完成实验日志和上交程序。

## 【实验内容】

### 一、数组和循环语句的结合应用

程序功能要求：创建帐户数组accountBanlances，结合for循环实现允许用户输入5个帐户结算，将利息加到每个帐户结算上，打印5个结算的最后结果。

参考代码如下：

using System;

class AccountBalanceTraversal

{

public static void Main()

{

const decimal interestRate = 0.1m;

decimal [] accountBalances;

accountBalances = new decimal [5];

Console.WriteLine("Please enter {0} account balances:", accountBalances.Length);

for (int i = 0; i < accountBalances.Length; i++)

{

Console.Write("Enter balance with index {0}: ", i);

accountBalances[i] = Convert.ToDecimal(Console.ReadLine());

}

Console.WriteLine("\nAccount balances after adding interest\n");

for (int i = 0; i < accountBalances.Length; i++)

{

accountBalances[i] = accountBalances[i]

+ (accountBalances[i] \* interestRate);

Console.WriteLine("Account balance with index {0}: {1:C}",

i, accountBalances[i]);

}

}

}

以上程序中为何用decimal数据类型？将以上程序改造成你自己的程序也实现以上功能，如将for语句用foreach语句实现或将for语句改造成其它循环语句。

**答：使用decimal数据类型可以使变量保持固定的小数位，使存储更加精确。**

改造程序：

**int index = 0;**

**foreach(decimal account in accountBalances)**

**{**

**decimal result = account \* (1 + interestRate);**

**Console.WriteLine("Account balance with index {0}: {1:C}", index++, result);**

**}**

### 二、一个强大而复杂的银行模拟程序

**注意：本程序不仅给出了程序的功能要求，更从软件的角度分析了实现过程，此后给予的示例代码，同学们可以从本程序中收获不少启示。**

程序功能要求：

模拟一个持有若干银行账号的银行，银行帐户可以通过控制台窗口提供的一个简单用户界面来访问和操作。用户通过发出简单命令必须能：

开始指定由银行管理的账户数；

在指定帐户上存款；

从指定帐户上提款；

设置指定帐户的利率；

将利息加到所有帐户上；

打印帐户结算；

打印支付给每个帐户的利息；

打印每个帐户的利率；

结束模拟。

软件分析：

确定两个明显的类：Account和Bank及将二者功能对应的包含Main方法的BankSimulation；

Account帐户类包含实例变量：结算总额，当前利率，总支付利息；另外帐户类应含有对帐户结算增减、利率计算等的方法；所有实例变量在构造函数中被初始化。

Bank类的实例变量：一个帐户数组，先要求输入帐户数组元素个数；通过构造方法初始化帐户数组。其它涉及信息都可以在帐户类生成的对象里获得；

因为Account对象内一般实例变量为private，无法被外部访问，所以Bank类要想访问，可以用属性或存取器、变异器方法。如currentInterestRate用setInterestRate和GetInterestRate来对当前利率赋值和读取，从而实现通过存取器对私有变量的外部访问。Balance和totalInterestPaid也分别通过响应的存取方法返回值。

注意：设第一个帐户account number为1，其对应的数组索引为0，所以，Bank类中有accounts[accountNumber - 1]的应用。

BankSimulation仅需要一个实例变量，一个Bank对象，其它对应到前两个类中；

示例代码如下，阅读后回答问题：

using System;

class Account

{

private decimal balance;

private decimal currentInterestRate;

private decimal totalInterestPaid;

public Account()

{

balance = 0;

currentInterestRate = 0;

totalInterestPaid = 0;

}

public void SetInterestRate(decimal newInterestRate)

{

currentInterestRate = newInterestRate;

}

public decimal GetInterestRate()

{

return currentInterestRate;

}

public void UpdateInterest()

{

totalInterestPaid += balance \* currentInterestRate;

balance += balance \* currentInterestRate;

}

public void Withdraw (decimal amount)

{

balance -= amount;

}

public void Deposit (decimal amount)

{

balance += amount;

}

public decimal GetBalance()

{

return balance;

}

public decimal GetTotalInterestPaid()

{

return totalInterestPaid;

}

}

class Bank

{

private Account [] accounts;

public Bank()

{

Console.WriteLine("Congratulations! You have created a new bank");

Console.Write("Please enter number of accounts in bank: ");

accounts = new Account[Convert.ToInt32(Console.ReadLine())];

for (int i = 0; i < accounts.Length; i++)

{

accounts[i] = new Account();

}

}

public void Deposit()

{

int accountNumber;

decimal amount;

Console.Write("Deposit. Please enter account number: ");

accountNumber = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter amount to deposit: ");

amount = Convert.ToDecimal(Console.ReadLine());

accounts[accountNumber - 1].Deposit(amount);

Console.WriteLine("New balance of account {0}: {1:C}",

accountNumber, accounts[accountNumber - 1].GetBalance());

}

public void Withdraw()

{

int accountNumber;

decimal amount;

Console.Write("Withdraw. Please enter account number: ");

accountNumber = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter amount to withdraw: ");

amount = Convert.ToDecimal(Console.ReadLine());

accounts[accountNumber - 1].Withdraw(amount);

Console.WriteLine("New balance of account {0}: {1:C}",

accountNumber, accounts[accountNumber - 1].GetBalance());

}

//注意，{1：C}这里代表本位置上输出的数字以货币形式显式。如$100.

public void SetInterestRate()

{

int accountNumber;

decimal newInterestRate;

Console.Write("Set interest rate. Please enter account number: ");

accountNumber = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter interest rate: ");

newInterestRate = Convert.ToDecimal(Console.ReadLine());

accounts[accountNumber - 1].SetInterestRate(newInterestRate);

}

public void PrintAllInterestRates()

{

Console.WriteLine("Interest rates for all accounts:");

for (int i = 0; i < accounts.Length; i++)

{

Console.WriteLine("Account {0,-3}: {1,-10}",

(i + 1), accounts[i].GetInterestRate());

}

}

public void PrintAllBalances()

{

Console.WriteLine("Account balances for all accounts:");

for (int i = 0; i < accounts.Length; i++)

{

Console.WriteLine("Account {0,-3}: {1,12:C}",

(i + 1), accounts[i].GetBalance());

}

}

public void PrintTotalInterestPaidAllAccounts()

{

Console.WriteLine("Total interest paid for each individual account");

for (int i = 0; i < accounts.Length; i++)

{

Console.WriteLine("Account {0,-3}: {1,12:C}",

(i + 1), accounts[i].GetTotalInterestPaid());

}

}

public void UpdateInterestAllAccounts()

{

for (int i = 0; i < accounts.Length; i++)

{

Console.WriteLine("Interest added to account number {0,-3}: {1,12:C}",

(i + 1), accounts[i].GetBalance() \* accounts[i].GetInterestRate());

accounts[i].UpdateInterest();

}

}

}

class BankSimulation

{

private static Bank bigBucksBank;

public static void Main()

{

string command;

bigBucksBank = new Bank();

do

{

PrintMenu();

command = Console.ReadLine().ToUpper();

switch (command)

{

case "D":

bigBucksBank.Deposit();

break;

case "W":

bigBucksBank.Withdraw();

break;

case "S":

bigBucksBank.SetInterestRate();

break;

case "U":

bigBucksBank.UpdateInterestAllAccounts();

break;

case "P":

bigBucksBank.PrintAllBalances();

break;

case "T":

bigBucksBank.PrintTotalInterestPaidAllAccounts();

break;

case "I":

bigBucksBank.PrintAllInterestRates();

break;

case "E":

Console.WriteLine("Bye Bye!");

break;

default:

Console.WriteLine("Invalid choice");

break;

}

} while (command != "E");

}

private static void PrintMenu()

{

Console.WriteLine("\nWhat would you like to do?\n" +

"D)eposit\n" +

"W)ithdraw\n" +

"S)et interest rate\n" +

"U)pdate all accounts for interest\n" +

"P)rint all balances\n" +

"T)otal interest paid printed for all accounts\n" +

"I)nterest rates printed for all accounts\n" +

"E)nd session\n" +

"Note: First account has account number one");

}

}

**问题：**

* 理解数组的声明语句

accounts = new Account[Convert.ToInt32(Console.ReadLine())];并给出数组声明的其它方式；

**答：该声明方式可以由IO输入动态指定数组大小，实现数组空间的合理利用。**

**还可以通过以下等方式声明：**

1. **int[] intArray = new int[10];**
2. **int[] intArray = new int[3] {1,2,3};**
3. **int[] intArray = new int[] {1,2,3};**
4. **int[] intArray = {1,2,3};**
5. **二维数组：int[,] intArray = new int[2,3] {{1,2,3},{4,5,6}};**

* 语句Console.WriteLine("Account {0,-3}: {1,-10}"和Console.WriteLine("Account {0,-3}: {1,12:C}"中的输出参数表示什么意思？

**答：1:C表示输出数字前放置一个货币符号。**

* 尝试将程序简化，如可将存取器改为属性，将打印方法和响应方法合并，for语句用foreach语句代替等；将改造后并调试通过的关键代码部分填写到实验报告中；

改造后的代码如下：

...

public decimal InterestRate

{

set { currentInterestRate = value; }

get { return currentInterestRate; }

}

public decimal Balance

{

get { return balance; }

}

public decimal TotalInterestPaid

{

get { return totalInterestPaid; }

}

...

int index=0;

foreach (Account account in accounts)

{

Console.WriteLine("Account {0,-3}: {1,12:C}",

index, account.Balance);

}

...

int index=0;

foreach (Account account in accounts)

{

Console.WriteLine("Account {0,-3}: {1,12:C}",

index, account.TotalInterestPaid);

}

### 三、二维数组的使用

程序功能：一个电梯需求跟踪的类能根据一周内每天每时的随即电梯需求进行统计跟踪。具体实现：

随机产生一周7天内每小时的随机数，当8时至18时之间时电梯随机请求数在1-10楼之间，除此以外的时间在20-99之间产生；

以列表形式打印出每天每个小时产生的需求数；

打印出每天请求的总数；

打印出一周内每小时的平均请求数；

参考代码如下：

using System;

class ElevatorRequestTracker

{

public static void Main()

{

ushort[,] requests = new ushort [7,24];

Random randomizer = new Random();

int sum;

// Randomly generate number of requests received for every

// hour of the day and every day of the week.

for (int i = 0; i < 7; i++)

{

for (int j = 0; j < 24 ; j++)

{

if ((j < 8) || (j > 18))

requests[i,j] = (ushort)randomizer.Next(1,10);

else

requests[i,j] = (ushort)randomizer.Next(20,99);

}

}

//Print out table showing requests of all hours of every day

Console.WriteLine(" Hour\n");

Console.Write(" ");

for (int i = 0; i < 24; i++)

{

Console.Write("{0,2} ",i);

}

Console.Write("\nDay");

for (int i = 0; i < 7; i++)

{

Console.Write("\n{0} ", (i + 1));

for (int j = 0; j < 24; j++)

{

Console.Write("{0,2} ", requests[i,j]);

}

}

// Calculate and print total number of requests on a daily basis

Console.WriteLine("\n\nTotal number of request per day:\n");

for (int i = 0; i < 7; i++)

{

sum = 0;

for (int j = 0; j < 24; j++)

sum += requests[i,j];

Console.WriteLine("Day {0}: {1}", (i + 1), sum);

}

// Calculate and print average requests on an hourly basis

Console.Write("\nAverage requests per hour:\n\nHour:");

for (int i = 0; i < 24; i++)

{

Console.Write("{0,2} ",i);

}

Console.Write("\nAver:");

for (int j = 0; j < 24; j++)

{

sum = 0;

for (int i = 0; i < 7; i++)

sum += requests[i,j];

Console.Write("{0,2} ", (sum / 7));

}

}

}

注意：random类的使用；语句Console.Write("\nDay");或Console.Write("\nAver:")中转义符的作用。

### 练习题：

1. 输入、打印5个人的年龄并计算和打印平均年龄，要求输出如下格式：

“Enter age for number ？：\*\*\* ”“Average age for 5 people is \*\*\* ”，要求“？”处由方法或参数来得出，“\*\*\*”处为输入或计算结果，打印用foreach语句实现。

**代码如下：**

**#region using derectives**

**using System;**

**using System.Collections.Generic;**

**using System.Text;**

**#endregion**

**class AgeCalculation{**

**public static void Main()**

**{**

**int[] persons = new int[5];**

**double sum = 0.0d;**

**for(int i = 0; i < persons.Length; i++)**

**{**

**Console.Write("Enter age for number {0}: ", i + 1);**

**persons[i] = Convert.ToInt32(Console.ReadLine());**

**sum += persons[i];**

**}**

**Console.WriteLine();**

**int index = 1;**

**foreach(int person in persons)**

**{**

**Console.WriteLine("Person {0} aged {1}", index++, person);**

**}**

**Console.WriteLine("The average age of the five persons is {0}", sum/5);**

**}**

**}**