Курс по програмиране на C#

ДАВИД холдинг АД | гр.“Казанлък“, ул.“Стара река“ 2, ДК „Арсенал“, ет.4

Упражнения

Валери Дачев и Александър Далемски

2013

# Съдържание

[Съдържание 1](#_Toc364377820)

[Намиране на факториел 3](#_Toc364377821)

[Задача 3](#_Toc364377822)

[Изисквания 3](#_Toc364377823)

[Допълнителни задачи 3](#_Toc364377824)

[Обръщане на масив с рекурсия 3](#_Toc364377825)

[Задача 3](#_Toc364377826)

[Изисквания 3](#_Toc364377827)

[Допълнителни задачи 3](#_Toc364377828)

[Двоично търсене в сортиран масив 4](#_Toc364377829)

[Задача 4](#_Toc364377830)

[Изисквания 4](#_Toc364377831)

[Допълнителни задачи 4](#_Toc364377832)

[Търсене на път в лабиринт 4](#_Toc364377833)

[Задача 4](#_Toc364377834)

[Изисквания 4](#_Toc364377835)

[Допълнителни задачи 4](#_Toc364377836)

[Сняг 4](#_Toc364377837)

[Задача 4](#_Toc364377838)

[Описание 5](#_Toc364377839)

[Изисквания 5](#_Toc364377840)

[Допълнителни задачи 5](#_Toc364377841)

[Адресна книга 5](#_Toc364377842)

[Задача 5](#_Toc364377843)

[Изисквания 5](#_Toc364377844)

[Допълнителни задачи 5](#_Toc364377845)

[Рационални числа 6](#_Toc364377846)

[Задача 6](#_Toc364377847)

[Изисквания 6](#_Toc364377848)

[Допълнителни задачи 6](#_Toc364377849)

[Печатни произведения 6](#_Toc364377850)

[Задача 6](#_Toc364377851)

[Изисквания 6](#_Toc364377852)

[Допълнителни задачи 7](#_Toc364377853)

[Аритметични изрази 7](#_Toc364377854)

[Задача 7](#_Toc364377855)

[Изисквания 7](#_Toc364377856)

[Допълнителни задачи 7](#_Toc364377857)

[Оказион 8](#_Toc364377858)

[Задача 8](#_Toc364377859)

[Изисквания 8](#_Toc364377860)

[Допълнителни задачи 8](#_Toc364377861)

[Геометрични фигури 9](#_Toc364377862)

[Задача 9](#_Toc364377863)

[Изисквания 9](#_Toc364377864)

[Допълнителни задачи 10](#_Toc364377865)

# Намиране на факториел

## Задача

Напишете програма, която изчислява факториел на дадено число ***n***.

## Изисквания

* Намирането на факториел от число оформете като функция ***GetFactorialIteratively***, която използва итерация (т.е. използва цикли).

## Допълнителни задачи

* Променете програмата така, че тя да може да работи с големи числа като резултат от изчислението;
* Модифицирайте програмата така, че да дава възможност на потребителя да въведе цялото число ***n***, като проверявате за коректен вход;
* Модифицирайте програмата така, че да попита потребителя дали иска да продължи. Ако избере да продължи, изпълнението да започне отначало. За целта използвайте подходящ цикъл;
* Добавете функция ***GetFactorialRecursively***, която намира факториел рекурсивно на подадено число използвайки свойството:

n! = n . (n-1)!

* Дайте възможност на потребителя да избере начина на изчисляване – итеративно или рекурсивно;
* Реализирайте приложението като Windows Forms Application. Използвайте ***NumericUpDown*** за въвеждане на числото ***n***, ***Label*** за извеждане на резултата, ***Button*** за рекурсивно изчисление и ***Button*** за итеративно изчисление.

# Обръщане на масив с рекурсия

## Задача

Напишете програма, която обръща масив от числа отзад-напред.

## Изисквания

* Обръщането на масива оформете като процедура ***ReverseArrayIteratively***, която използва итерация (т.е. използва цикли);
* Покажете резултата с функция ***ShowArray***.

## Допълнителни задачи

* Модифицирайте програмата така, че да чете масива от клавиатурата, като:
	+ Потребителят въвежда броя на числата;
	+ Всяко число потребителят въвежда на отделен ред. Проверявайте за коректен вход. Ако дадено число не е въведено коректно, подканете потребителя да го въведе отново.
* Модифицирайте програмата така, че потребителят да не въвежда броя на числата, а всяко ново число да се добавя към масива и въвеждането да приключва с въвеждането на празен ред;
* Добавете функция ***ReverseArrayRecursively***, която да обръща масива рекурсивно последователно разменя крайните две числа, след което предпоследните две числа и т.н.;
* Дайте възможност на потребителя да избере начина на изчисляване – итеративно или рекурсивно.

# Двоично търсене в сортиран масив

## Задача

Напишете програма, която използва [двоично търсене](http://en.wikipedia.org/wiki/Binary_search_algorithm) за намиране на число в сортиран масив от числа.

## Изисквания

* Двоичното търсене в сортиран масив оформете във функция ***BinSearchArrayRecursively***, която използва рекурсия намиране на число;
* Функцията ***BinSearchArrayRecursively*** трябва връща ***boolean***, указващ дали числото е намерено, и ***int*** – индекса на неговия пореден номер.

## Допълнителни задачи

* Добавете функция ***CheckIfSorted***, която проверява дали подаденият масив е сортиран, и я използвайте. Ако масивът не е сортиран, изведете съобщение за грешка и не извършвайте търсене;
* Добавете аналогична функция ***BinSearchArrayIteratively***, която осъществява двоично търсене с използване на цикли.

# Търсене на път в лабиринт

## Задача

Напишете програма, която [намира път в лабиринт рекурсивно](http://www.cs.bu.edu/teaching/alg/maze/).

## Изисквания

* Лабиринтът се представя като двумерен масив от символи с размери ***m*** x ***n***, в който масив символът интервал (***‘ ‘***) означава път, символът звезда (***‘\*’***) – стена, а символът диез (‘***#***’) – стъпка. Първоначалните координати са (***x0***,***y0***), а крайните координати – (***xt***,***yt***) (от думата target). Движението може да става само с една позиция в някоя от четирите посоки (нагоре, надолу, наляво и надясно), като не може да се стъпва на стена или място, на което вече има стъпки;
* Търсенето на път оформете във функция ***FindMazePath***, която използва рекурсия за намиране на пътя в подаден лабиринт.

## Допълнителни задачи

* Добавете функция ***ShowMaze***, която показва текущото състояние на лабиринта (с пътя, стената и стъпките);
* Извиквайте функцията ***ShowMaze*** при всеки ход;
* Добавете функция ***GenerateMaze***, която да създава лабиринт по зададени размери ***m*** и ***n***, като използвате [търсене в дълбочина](http://en.wikipedia.org/wiki/Maze_generation_algorithm#Depth-first_search). Използвайте нейния резултат, за да намерите път с функцията ***FindMazePath***.

# Сняг

## Задача

Напишете програма, която показва “сняг” в конзолата, който се натрупва. Снегът представлява множество от “снежинки” (представени със символа “\*”).

## Описание

Работа на програмата:

1. Изчиства се конзолата при стартиране;
2. От горната част на екрана по широчината на целия екран започват да “вали сняг” – произволен брой снежинки;
3. С времето всички снежинки, под които няма нищо, се “падат” с един ред надолу;
4. Снежинките, които са стигнали дъното на екрана или под които има друга снежинка, остават на мястото си;
5. Движението на снежинките трябва да се забави с определен период от време, за да изглежда падането плавно;
6. Изпълнението на програмата продължава автоматично, докато не бъде натиснат клавиш.

## Изисквания

* Изрисуването на снежинките трябва да става без да се изчиства екрана, за да се избегне премигването му;
* Екранът трябва да бъде представен като двумерен масив от символи, размерът на който зависи от размерите на конзолата;
* Разбийте програмата на отделни функции – за инициализация на масива, за обработка на масива, за изрисуването на екрана и т.н..

## Допълнителни задачи

* Преди започване на “снега”, създайте произволен земен “релеф” (представен със символа символа “@”).
* Ако по време на изпълнение на програмата е натиснат клавиш “стрелка нагоре” или “стрелка надолу”, съответно да се увеличава или намалява скоростта на снега;
* Ако по време на изпълнение на програмата е натиснат клавиш “стрелка наляво” или “стрелка надясно”, да се симулира вятър, като летящите снежинки (тези, под които няма нищо), се преместват съответно наляво или надясно.

# Адресна книга

## Задача

Да се реализира адресна книга с информация за имената, адресите и телефоните на хора. Да се предостави възможност за добавяне на редове в края на книгата през конзолата, както и за извеждане на всички редове в конзолата.

## Изисквания

* Да се декларира клас AddressEntry, който ще представлява един ред от адресната книга.
* Да се реализират необходимите методи за въвеждане и извеждане на данните за реда от/в конзолата.
* Редовете в адресната книга да се съхраняват в масив.

## Допълнителни задачи

* Размерът на масива да се въвежда от потребителя в началото на програмата.
* Да се реализират опции за изтриване на ред от адресната книга и за вмъкване на ред на конкретно място в адресната книга.
* Да се реализира сортирането на редовете в адресната книга по азбучен ред на имената.
* **Ново:** да се замени масивът с шаблонен списък от обекти и да се преправят реализациите на операциите за добавяне и премахване на ред от книгата.

# Рационални числа

## Задача

Напишете програма, която декларира структура за рационално число със съответните аритметични операции и създава няколко пробни израза с рационални числа.

## Изисквания

* Структурата за рационално число да бъде с наименование Rational.
* Да се реализира конструктор с параметри числител и знаменател.
* Да се презапишат аритметичните операции събиране, изваждане, умножение и деление за параметри рационални числа.
* Да се презапишат преобразувания на типове както следва:
	+ неявно преобразуване от int към рационално число;
	+ неявно преобразуване от рационално число към float, double и decimal;
	+ явно преобразуване от рационално число към int (като се раздели целочислено числителя на знаменателя).
* Да се реализира метод за получаване на реципрочното на рационално число.
* Да се реализира метод за преобразуването на рационално число в подходящ символен низ.

## Допълнителни задачи

* След всяка аритметична операция числителят и знаменателят на полученото рационално число да нямат общи делители.
* Да се реализира в главната функция на програмата цикъл, във всяка итерация на който се избира от потребителя операция (събиране, изваждане, умножение, деление, намиране на реципрочно), въвеждат се необходимия брой цели или рационални числа и се извежда резултатът. Рационалните числа се въвеждат под формата на две цели числа, разделени с наклонена черта. Изпълнението на цикъла да приключи при въвеждане на празен низ при избора на операция.

# Печатни произведения

## Задача

Напишете програма, в която е декларирана йерархия от класове за различни видове печатни произведения с подходящи характеристики и поведение. Програмата трябва да поддържа масив с регистрирани произведения, като се предоставя възможност за регистриране на произведение през конзолата и извеждане на всички регистрирани до момента произведения.

## Изисквания

* Базовият клас на йерархията да бъде PrintedWork с характеристики *заглавие* и *език*.
* Печатните произведения се делят на **книги** (Book) с характеристики *година на издаване* и *автор* и **периодични издания** (Periodical) с характеристики *период на издаване* (от коя до коя година е било издавано периодичното произведение) и *честота на издаване*.
* Книгите се делят на **художествени** (FictionBook) с характеристики *жанр* (свободен текст) и *брой глави* и **технически** (TechnicalBook) с характеристики *техническа сфера* (свободен текст) и *ниво* (изброен тип със стойности начинаещ-напреднал-експерт).
* Периодичните издания се делят на **вестници** (Newspaper) с характеристики *вид на съдържанието* (изброен тип със стойности новинарски-спортен-жълт) и *цветен печат* (да/не) и **списания** (Magazine) с характеристика *целева аудитория* (свободен текст).
* Да се реализират необходимите методи за въвеждане и извеждане на книги от/в конзолата.
* Да се използва превръщане на типове надолу по йерархията (downcasting), където е необходимо.

## Допълнителни задачи

* Да се добавят други подходящи характеристики на гореописаните класове, както и други наследени класове за по-фино категоризиране на печатните произведения.
* Да се декларират класове за групи от печатни произведения, например *трилогии от художествени книги* или групи от *периодични издания на един и същи издател*.
* **Ново:** Да се преправят методите за въвеждане и извеждане на книги от/в конзолата така, че да бъдат виртуални, и да се премахне употребата на downcasting навсякъде в програмата с извиквания на виртуални методи.

# Аритметични изрази

## Задача

Напишете програма, която декларира класове за аритметични изрази, които съхраняват в себе си информацията за аритметичен израз и предоставят методи за извеждането му на екрана и пресмятане на стойността му.

## Изисквания

* Да се реализират базов клас за аритметичен израз и три наследени класа за константа, унарна операция и бинарна операция.
* Аритметичните операции, които се разглеждат, са събиране, изваждане, умножение, деление и намиране на противоположното число (унарен минус).
* Константите и стойностите на изразите да бъдат от тип int.
* Да се презапишат съответните оператори за създадените класове.
* Да се направи виртуален метод за пресмятане на стойността на израз.
* Да се предефинира методът ToString() така, че да извежда представяне на аритметичния израз в символен низ.
* Да не се използва преобразуване на типове надолу по йерархията (downcasting).
* Да се създадат няколко аритметични израза и да се изпечатат на екрана заедно със стойностите си.

## Допълнителни задачи

* Да се създаде интерфейс за аритметичен израз и с него да се замени базовия клас на йерархията.
* Да се променят класовете на шаблонни класове с един типов параметър, който определя типа на стойностите на аритметичните изрази.
* Да се реализира възможност за въвеждане на израз от конзолата, пресмятането на стойността му и извеждането й на екрана (прост калкулатор).

# Оказион

## Задача

Да се напише програма, която реализира оказион с обяви за продажба на артикули чрез търг или директно закупуване. Да се поддържа списък от потребители и списък от обявени за продан артикули. Да се предостави възможност за вписване и отписване от системата с потребител; регистриране на обява за продан на артикул; разглеждане на активните, регистрираните и спечелените обяви; наддаване по търг; директно закупуване на артикул.

## Изисквания

* Да се използва обектно-ориентирано програмиране с подходящи класове, наследяване и интерфейси.
* Да се спази принципът за капсулиране.
* Изключенията в програмата да бъдат обработвани по подходящ начин, а при въвеждането на данни от клавиатурата да се осигурява валидността им.
* Данните да се съхраняват в оперативната памет.
* Всеки потребител се характеризира с уникално потребителско име.
* Всяка обява се характеризира със състояние (активна/прикючена), дата на регистриране, описание на артикула и потребител, който я е регистрирал.
* Приключените обяви се характеризират и с дата на приключване и потребител, който я е спечелил.
* Обявите за директна продажба се характеризират с фиксирана цена за закупуване.
* Обявите за търг се характеризират с начална цена на търга и краен срок.
* В програмата да се реализира главно меню със следните опции:
	+ Вписване в системата;
	+ Разглеждане на списъка с активни обяви;
	+ Разглеждане на списъка с обяви, регистрирани от вписания потребител (изисква вписан потребител);
	+ Разглеждане на списъка от приключени обяви, които вписаният потребител е спечелил (изисква вписан потребител);
	+ Наддаване по търг (изисква вписан потребител);
	+ Директно закупуване на артикул (изисква вписан потребител);
	+ Отписване на текущия потребител от системата (изисква вписан потребител);
	+ Край на работата с програмата.

## Допълнителни задачи

* Да се реализира в главното меню опция за регистриране на потребител с парола, като вписването в системата изисква съответната парола.
* Да се реализира зареждане на списъците с потребители и обяви от текстови файлове с фиксирани имена и подходящ формат за съхранение на данните, както и записването на данните от оперативната памет обратно в тези файлове при приключване на работата с програмата.
* За обявите за търг да се реализира максимална цена на търга, при която предложилият я потребител автоматично печели търга преди изтичането на крайния му срок.
* Да се използват подходящи събития в реализацията на програмата.

# Геометрични фигури

## Задача

Да се напише програма, която реализира класове за геометрични обекти, като предоставя възможност за въвеждането и извеждането на характеристиките им, както и за изчисление на съответните периметри, лица, повърхнини и обеми.

## Изисквания

* Да се декларират следните класове и интерфейси:
	+ Базов клас GeometryObject, който се предоставя характеристика *наименование на обекта* и методи за прочитане на характеристиките на обекта от клавиатурата и извеждането им на екрана. Всички останали класове да наследяват GeometryObject;
	+ Интерфейс IObject2D, който предоставя метод за изчисление на *периметър* и *лице*;
	+ Интерфейс IObject3D, който предоставя метод за изчисление на *пълна повърхнина* и *обем*;
	+ Клас Point с характеристики *абсциса* и *ордината*;
	+ Клас Line с характеристики *начална точка* и *крайна точка*;
	+ Клас Triangle с характеристики *дължина на страна A*, *дължина на страна B* и *дължина на страна C*, който имплементира IObject2D;
	+ Клас Square с характеристика *дължина на страната*, който имплементира IObject2D;
	+ Клас Rectangle с характеристики *дължина на първата страна* и *дължина на втората страна*, който имплементира IObject2D;
	+ Клас Circle с характеристика *радиус*, който имплементира IObject2D;
	+ Клас Cube с характеристика *дължина на страната*, който имплементира IObject3D;
	+ Клас Pyramid (правилна пирамида) с характеристики *брой ъгли на основата* (3 или 4), *дължина на страната на основата* и *дължина на височината*, който имплементира IObject3D. При изчислението на пълната повърхнина трябва предварително да се изчисли височината на околната стена с помощта на питагоровата теорема;
	+ Клас Prism (правилна призма) с характеристики *брой ъгли на основата* (3 или 4), *дължина на страната на основата* и *дължина на височината*, който имплементира IObject3D;
	+ Клас Cone (прав кръгов конус) с характеристики *радиус на основата* и *дължина на височината*, който имплементира IObject3D. При изчислението на пълната повърхнина трябва предварително да се изчисли дължината на образуващата с помощта на питагоровата теорема;
	+ Клас Cylinder (прав кръгов цилиндър) с характеристики *радиус на основата* и *дължина на височината*, който имплементира IObject3D;
	+ Клас Sphere с характеристика *радиус*, който имплементира IObject3D.
* Всички величини да се съхраняват в полета/променливи с тип double;
* Да се създаде колекция от геометрични обекти и да бъде попълнена с няколко случайно генерирани примерни обекта;
* Да се реализира опция за въвеждане на нов геометричен обект през конзолата (чрез избор на тип на обекта и въвеждане на характеристиките му) и добавянето му в колекцията от геометрични обекти;
* Да се реализира опция за извеждане характеристиките на всички геометрични обекти в колекцията;
* При извеждането на характеристиките на геометричен обект да се проверява дали съответният обект имплементира един от интерфейсите IObject2D и IObject3D; ако това е изпълнено, да се извеждат и съответните пресметнати характеристики от интерфейса;
* Да бъде спазен принципът за капсулиране;
* Да се използва предефиниране на методи.

## Допълнителни задачи

* Да се реализират и класове за успоредник, трапец, произволен многоъгълник, паралелепипед, пресечена пирамида и пресечен конус.
* Да се създадат интерфейсите IPolygon и IRoundShape, които наследяват IObject2D. Съществуващите класове за равнинни фигури да имплементират съответните наследени интерфейси вместо интерфейса IObject2D. Да се декларира клас Ellipse с характеристики *голям радиус* и *малък радиус*, който имплементира IRoundShape. Характеристиката *основа* в класовете Pyramid и Prism да бъде от тип IPolygon, а характеристиката *радиус на основата* в класовете Cone и Cylinder да бъде заменена с характеристика *основа* от тип IRoundShape.