# Решения на задачите от книгата "Въведение в програмирането със C#"

Предлагаме ви решения на задачите от [книгата "Въведение в програ­мирането със C#"](http://www.introprogramming.info), заедно с анализ на задачата, описание на използва­ните идеи, алгоритми, подходи за решаване и тестове.

## Авторски колектив

**Име Презиме Фамилия**

Контакти: (по желание може да се остави уеб сайт, мейл, Facebook, LinkedIn, Twitter профил, …)

**Име Презиме Фамилия**

Контакти: (по желание може да се остави уеб сайт, мейл, Facebook, LinkedIn, Twitter профил, …)

**Име Презиме Фамилия**

Контакти: (по желание може да се остави уеб сайт, мейл, Facebook, LinkedIn, Twitter профил, …)

# Шаблон за описание на задачите

|  |  |
| --- | --- |
| Задача XX. (използвайте номерацията от съответната глава на книгата и измислете подходящо име на задачата на български или на английски език) | |
| **Условие**  (копирайте условието на задачата както е дадено в книгата, на български език; ако има нещо неясно, може да го уточните, да дадете примери и да допълните описанието с ваши думи) | |
| **Описание на входа**  (опишете формата на входните данни; форматът на данните трябва да е фиксиран и еднозначно описан, така че да може да се ползва в състезателната система [BG Coder](http://www.bgcoder.com)) | |
| **Описание на изхода**  (опишете формата на изходните данни по еднозначен начин; помислете ако има няколко решения, кое от тях трябва да се отпечата, така че да се позволи автоматична проверка чрез сравнение на изхода от програмата с очаквания резултат байт по байт, както обикновено се прави в [BG Coder](http://www.bgcoder.com)) | |
| **Анализ на задачата**  (опишете идеята си за решаване на задачата в свободен текст; описанието трябва да е толкова подробно, че да бъде разбираемо; трябва да се акцентира на идеята, не толкова на детайлите) | |
| **Решение (сорс код)** | |
| (копирайте форматиран и оцветен сорс код от Visual Studio, шрифт Consolas, размер 10pt; ако имате няколко класа, ги сложете един след друг с 2 празни реда между тях) | |
| **Тестове**  (опишете как сте тествали решението си; започнете с номинален случай и дайте списък с интересните гранични случаи, тест за бързодействие и всички други интересни случаи, които сте разгледали и вашето решение покрива; тестовете трябва да са изчерпателни и да покриват всички частни и специални случаи, т.е. ако някое решение на задачата работи вярно на всички тестове, то трябва да се счита за вярно; при някои задачи е удобно да си напишете тестов генератор; ако даден тест е прекалено обемен, можете да го изкарате във външен текстов файл и да качите линк към файла в настоящия документ; минимален брой тестове: 5) | |
| **Вход** | **Изход** |
| (дайте примерен вход според тестовете, които сте описали в предната точка) | (дайте коректния отговор на примерния вход; ако е много дълъг, изпишете го съкратено използвайки …) |
| **Вход** | **Изход** |
| (дайте примерен вход според тестовете, които сте описали в предната точка) | (дайте коректния отговор на примерния вход; ако е много дълъг, изпишете го съкратено използвайки …) |

# Примерно решение и описание на задача

|  |  |
| --- | --- |
| Задача 5. Нарастваща редица последователни елементи | |
| **Условие**  Напишете програма, която намира максималната редица от последова­телни нараст­ващи елементи в масив. Пример: {3, **2, 3, 4**, 2, 2, 4} 🡪 {2, 3, 4}. | |
| **Описание на входа**  Входните данни се въвеждат като поредица от числа, на един единствен ред, разделени едно от друго с единичен интервал. | |
| **Описание на изхода**  Изходът се извежда като поредица от числа на един ред с интервали между тях. Ако има повече от една най-дълга редица, трябва да се изведе тази най-дълга редица, която започва най-наляво. | |
| **Анализ на задачата**  Сканираме с един цикъл (**for i**) елементите на масива от първия към последния и пазим във всеки един момент в променливите **start** и **len** съответно началото и дължината на нарастващата редица, която е започнала на някоя предходна стъпка и завършва в текущия елемент на масива.  На всяка стъпка от алгоритъма, когато стигнем до дадено число имаме две възможности:   * Текущото число е по-голямо от предходното (или няма предходно) 🡪 текущото число е продължение на текущо-започнатата редица и трябва да увеличим **len**. * Текущото число е по-малко или равно на предходното 🡪 текущата редица не може да продължи и трябва да започнем нова редица (**start=i**; **len=1**).   В края на всяка стъпка проверяваме дали текущата дължина **len** е по-добра от най-добрата за момента **bestLen**. Ако е така запомняме текущата редица в **bestLen** и **bestStart**.  Първоначално установяваме **bestLen=0**, **bestStart=0**, **start=0**, **len=0**.  Накрая отпечатваме редицата, започваща от **bestStart** с дължина **bestLen**. | |
| **Решение (сорс код)** | |
| using System;  class LongestIncreasingSequence  {  static void Main()  {  // Sample sequence (for quick testing)  //int[] sequence = { 3, 2, 3, 4, 2, 2, 4 };  // Read the input sequence  string sequenceStr = Console.ReadLine();  string[] sequenceTokens = sequenceStr.Split(  new char[] {' '}, StringSplitOptions.RemoveEmptyEntries);  int[] sequence = new int[sequenceTokens.Length];  for (int i = 0; i < sequenceTokens.Length; i++)  {  sequence[i] = int.Parse(sequenceTokens[i]);  }  // Find the longest increasing sequence of numbers  int start = 0;  int len = 0;  int bestStart = 0;  int bestLen = 0;  for (int i = 0; i < sequence.Length; i++)  {  if (i == 0 || sequence[i] > sequence[i - 1])  {  len++;  }  else  {  start = i;  len = 1;  }  if (len > bestLen)  {  bestLen = len;  bestStart = start;  }  }  // Print the maximal increasing sequence  for (int i = 0; i < bestLen; i++)  {  Console.Write(sequence[i + bestStart] + " ");  }  Console.WriteLine();  }  } | |
| **Тестове**  Интересните случаи за тестване са следните:   * Примерният вход от условието на задачата * Масив от намаляващи елементи – тогава нарастващата редица е първият елемент * Масив от нарастващи елементи – тогава целият масив е търсеният резултат * Масив с две еднакво-дълги нарастващи редици – трябва да се изведе първата от тях * Масив от един елемент – тогава този елемент трябва да е търсеният резултат * Празен масив (нула елемента) – резултатът е празна редица * Масив от 100 000 елемента (тест за скорост) | |
| **Вход** | **Изход** |
| 3 2 3 4 2 2 4 | 2 3 4 |
| **Вход** | **Изход** |
| 5 4 3 2 1 | 5 |
| **Вход** | **Изход** |
| 1 2 3 4 | 1 2 3 4 |
| **Вход** | **Изход** |
| 1 2 3 0 1 2 | 1 2 3 |
| **Вход** | **Изход** |
| 1 | 1 |
| **Вход** | **Изход** |
| (празен масив) | (празен масив) |
| **Вход** | **Изход** |
| 1 2 3 … 100000 | 1 2 3 … 100000 |

# Глава 1. Въведение в програмирането

1. Запознайте се с Microsoft Visual Studio, Microsoft Developer Network (MSDN) Library Documentation. Инсталирайте си Visual Studio в къщи.
2. Да се намери описанието на класа System.Console в стандартната .NET API документация (MSDN Library).
3. Да се намери описанието на метода System.Console.WriteLine() с различ­ните негови възможни параметри в MSDN Library.
4. Да се компилира и изпълни примерната програма от примерите в тази глава през командния ред (конзолата) и с помощта на Visual Studio.
5. Да се модифицира примерната програма, така че да изписва различно поздравление, например "Добър ден!".
6. Напишете програма, която изписва вашето име и фамилия на конзолата.
7. Напишете програма, която извежда на конзолата числата 1, 101, 1001 на нов ред.
8. Напишете програма, която извежда на конзолата текущата дата и час.
9. Напишете програма, която извежда корен квадратен от числото 12345.
10. Напишете програма, която извежда първите 100 члена на редицата 2, -3, 4, -5, 6, -7, 8.
11. Направете програма, която прочита от конзолата вашата възраст и изписва (също на конзолата) каква ще бъде вашата възраст след 10 години.
12. Опишете разликите между C# и .NET Framework.
13. Направете списък с най популярните програмни езици. С какво те се различават от C#?
14. Да се декомпилира примерната програма от задача 5.

# Глава 2. Примитивни типове и променливи

1. Декларирайте няколко променливи, като изберете за всяка една най-подходящия от типовете sbyte, byte, short, ushort, int, uint, long и ulong, за да им присвоите следните стойности: 52130, -115, 4825932, 97, -10000, 20000; 224; 970700000; 112; -44; -1000000; 1990; 123456789123456789.
2. Кои от следните стойности може да се присвоят на променливи от тип float, double и decimal: 34.567839023; 12.345; 8923.1234857; 3456.091124875956542151256683467?
3. Напишете програма, която изчислява вярно променливи с плаваща запетая с точност до 0.000001.
4. Инициализирайте променлива от тип int със стойност 256 в шестна­десетичен формат (256 е 100 в бройна система с основа 16).
5. Декларирайте променлива от тип char и присвоете като стойност символа който има Unicode код 72 (използвайте калкулатора на Windows за да намерите шестнайсетичното представяне на 72).
6. Декларирайте променлива isMale от тип boolи присвоете стойност на последната в зависимост от вашия пол.
7. Декларирайте две променливи от тип string със стойности "Hello" и "World". Декларирайте променлива от тип object. Присвоете на тази променлива стойността, която се получава от конкатенацията на двете стрингови променливи (добавете интервал, ако е необходимо). Отпе­чатайте променливата от тип object.
8. Декларирайте две променливи от тип string и им присвоете стойности "Hello" и "World". Декларирайте променлива от тип object и и присвоете стойността на конкатенацията на двете променливи от тип string (не изпускайте интервала по средата). Декларирайте трета променлива от тип string и я инициализирайте със стойността на променливата от тип object ( трябва да използвате type casting).
9. Декларирайте две променливи от тип string и им присвоете стойност "The "use" of quotations causes difficulties." (без първите и последни кавички). В едната променлива използвайте quoted string, а в другата не го използвайте.
10. Напишете програма, която принтира фигура във формата на сърце със знака "o".
11. Напишете програма, която принтира на конзолата равнобедрен триъгълник, като страните му са очертани от символа звездичка "©".
12. Фирма, занимаваща се с маркетинг, иска да пази запис с данни на нейните служители. Всеки запис трябва да има следната характе­ристика – първо име, фамилия, възраст, пол (‘м’ или ‘ж’) и уникален номер на служителя (27560000 до 27569999). Декларирайте необходи­мите променливи, нужни за да се запази информацията за един служи­тел, като използвате подходящи типове данни и описателни имена.
13. Декларирайте две променливи от тип int. Задайте им стойности съответно 5 и 10. Разменете стойностите им и ги отпечатайте.

# Глава 3. Оператори и изрази

1. Напишете израз, който да проверява дали дадено цяло число е четно или нечетно.
2. Напишете булев израз, който да проверява дали дадено цяло число се дели на 5 и на 7 без остатък.
3. Напишете израз, който да проверява дали третата цифра (отдясно на ляво) на дадено цяло число е 7.
4. Напишете израз, който да проверява дали третия бит на дадено число е 1 или 0.
5. Напишете израз, който изчислява площта на трапец по дадени a, b и h.
6. Напишете програма, която за подадени от потребителя дължина и височина на право­ъгълник, пресмята и отпечатва на конзолата неговия периметър и лице.
7. Силата на гравитационното поле на Луната е приблизително 17% от това на Земята. Напишете програма, която да изчислява тежестта на човек на Луната, по дадената тежест на Земята.
8. Напишете програма, която проверява дали дадена точка О (x, y) е вътре в окръжността К ((0,0), 5). Пояснение: точката (0,0) е център на окръжността, а радиусът й е 5.
9. Напишете програма, която проверява дали дадена точка О (x, y) е вътре в окръжността К ((0,0), 5) и едновременно с това извън право­ъгълника ((-1, 1), (5, 5). Пояснение: правоъгълникът е зададен чрез координатите на горния си ляв и долния си десен ъгъл.
10. Напишете програма, която приема за вход четирицифрено число във формат *abcd* (например числото 2011) и след това извършва следните действия върху него:
    * Пресмята сбора от цифрите на числото (за нашия пример 2+0+1+1 = 4).
    * Разпечатва на конзолата цифрите в обратен ред: *dcba* (за нашия пример резултатът е 1102).
    * Поставя последната цифра, на първо място: *dabc* (за нашия пример резултатът е 1201).
    * Разменя мястото на втората и третата цифра: *acbd* (за нашия пример резултатът е 2101).
11. Дадено е число n и позиция p. Напишете поредица от операции, които да отпечатат стойността на бита на позиция p от числото n (0 или 1). Пример: n=35, p=5 -> 1. Още един пример: n=35, p=6 -> 0.
12. Напишете булев израз, който проверява дали битът на позиция p на цялото число v има стойност 1. Пример v=5, p=1 -> false.
13. Дадено е число n, стойност v (v = 0 или 1) и позиция p. Напишете поредица от операции, които да променят стойността на n, така че битът на позиция p да има стойност v. Пример n=35, p=5, v=0 -> n=3. Още един пример: n=35, p=2, v=1 -> n=39.
14. Напишете програма, която проверява дали дадено число n (1 < n < 100) е просто (т.е. се дели без остатък само на себе си и на единица).
15. \* Напишете програма, която разменя стойностите на битовете на позиции 3, 4 и 5 с битовете на позиции 24, 25 и 26 на дадено цяло положително число.
16. \* Напишете програма, която разменя битовете на позиции {p, p+1, …, p+k-1) с битовете на позиции {q, q+1, …, q+k-1} на дадено цяло положително число.

# Глава 4. Вход и изход от конзолата

1. Напишете програма, която чете от конзолата три числа от тип int и отпечатва тяхната сума.
2. Напишете програма, която чете от конзолата радиуса "r" на кръг и отпечатва неговия периметър и обиколка.
3. Дадена фирма има име, адрес, телефонен номер, факс номер, уеб сайт и мениджър. Мениджърът има име, фамилия и телефонен номер. Напишете програма, която чете информацията за фирмата и нейния мениджър и я отпечатва след това на конзолата.
4. Напишете програма, която отпечатва три числа в три виртуални колони на конзолата. Всяка колона трябва да е с широчина 10 символа, а числата трябва да са ляво подравнени. Първото число трябва да е цяло число в шестнадесетична бройна система, второто да е дробно положи­телно, а третото – да е дробно отрицателно. Последните две числа да се закръглят до втория знак след десетичната запетая.
5. Напишете програма, която чете от конзолата две цели числа (int) и отпечатва, колко числа има между тях, такива, че остатъкът им от деленето на 5 да е 0. Пример: в интервала (17, 25) има 2 такива числа.
6. Напишете програма, която чете две числа от конзолата и отпечатва по-голямото от тях. Решете задачата без да използвате условни конструкции.
7. Напишете програма, която чете пет числа и отпечатва тяхната сума. При невалидно въведено число да се подкани потребителя да въведе друго число.
8. Напишете програма, която чете пет числа от конзолата и отпечатва най-голямото от тях.
9. Напишете програма, която чете коефициентите a, b и c от конзолата и решава уравнението: ax2+bx+c=0. Програмата трябва да принтира реалните решения на уравнението на конзолата.
10. Напишете програма, която прочита едно цяло число n от конзолата. След това прочита още n на брой числа от конзолата и отпечатва тяхната сума.
11. Напишете програма, която прочита цяло число n от конзолата и отпечатва на конзолата всички числа в интервала [1…n], всяко на отделен ред.
12. Напишете програма, която отпечатва на конзолата първите 100 числа от редицата на Фибоначи: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, ...
13. Напишете програма, която пресмята сумата (с точност до 0.001): 1+ 1/2 + 1/3 + 1/4 + 1/5 + ...

# Глава 5. Условни конструкции

1. Да се напише if-конструкция, която проверява стойността на две целочислени променливи и разменя техните стойности, ако стойността на първата променлива е по-голяма от втората.
2. Напишете програма, която показва знака (+ или -) от произведението на три реални числа, без да го пресмята. Използвайте последователност от if оператори.
3. Напишете програма, която намира най-голямото по стойност число, измежду три дадени числа.
4. Сортирайте 3 реални числа в намаляващ ред. Използвайте вложени if оператори.
5. Напишете програма, която за дадена цифра (0-9), зададена като вход, извежда името на цифрата на български език.
6. Напишете програма, която при въвеждане на коефициентите (a, b и c) на квадратно уравнение: ax2+bx+c, изчислява и извежда неговите реални корени (ако има такива). Квадратните уравнения могат да имат 0, 1 или 2 реални корена.
7. Напишете програма, която намира най-голямото по стойност число измежду дадени 5 числа.
8. Напишете програма, която по избор на потребителя прочита от конзолата променлива от тип int, double или string. Ако променливата е int или double, трябва да се увеличи с 1. Ако променливата е string, трябва да се прибави накрая символа "\*". Отпечатайте получения резултат на конзолата. Използвайте switch конструкция.
9. Дадени са пет цели числа. Напишете програма, която намира онези подмножества от тях, които имат сума 0. Примери:

* Ако са дадени числата {3, -2, 1, 1, 8}, сумата на -2, 1 и 1 е 0.
* Ако са дадени числата {3, 1, -7, 35, 22}, няма подмножества със сума 0.

1. Напишете програма, която прилага бонус точки към дадени точки в интервала [1..9] чрез прилагане на следните правила:

* Ако точките са между 1 и 3, програмата ги умножава по 10.
* Ако точките са между 4 и 6, ги умножава по 100.
* Ако точките са между 7 и 9, ги умножава по 1000.
* Ако точките са 0 или повече от 9, се отпечатва съобщение за грешка.

1. \* Напишете програма, която преобразува дадено число в интервала [0..999] в текст, съответстващ на българското произношение на числото. Примери:

* 0 → "Нула"
* 12 → "Дванадесет"
* 98 → "Деветдесет и осем"
* 273 → "Двеста седемдесет и три"
* 400 → "Четиристотин"
* 501 → "Петстотин и едно"
* 711 → "Седемстотин и единадесет"

# Глава 6. Цикли

1. Напишете програма, която отпечатва на конзолата числата от 1 до N. Числото N трябва да се чете от стандартния вход.
2. Напишете програма, която отпечатва на конзолата числата от 1 до N, които не се делят едновременно на 3 и 7. Числото N да се чете от стандартния вход.
3. Напишете програма, която чете от конзолата поредица от цели числа и отпечатва най-малкото и най-голямото от тях.
4. Напишете програма, която отпечатва всички възможни карти от стан­дартно тесте карти без джокери (имаме 52 карти: 4 бои по 13 карти).
5. Напишете програма, която чете от конзолата числото N и отпечатва сумата на първите N члена от редицата на Фибоначи: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, ...
6. Напишете програма, която пресмята N!/K! за дадени N и K (1<K<N).
7. Напишете програма, която пресмята N!\*K!/(N-K)! за дадени N и K (1<K<N).
8. В комбинаториката числата на Каталан (Catalan’s numbers) се изчис­ляват по следната формула: , за n ≥ 0. Напи­шете програма, която изчислява n-тото число на Каталан за дадено n.
9. Напишете програма, която за дадено цяло число n, пресмята сумата: 
10. Напишете програма, която чете от конзолата положително цяло число N (N < 20) и отпечатва матрица с числа като на фигурата по-долу:

**N = 3**  **N = 4**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 1 | 2 | 3 | | 2 | 3 | 4 | | 3 | 4 | 5 | | |  |  |  |  | | --- | --- | --- | --- | | 1 | 2 | 3 | 4 | | 2 | 3 | 4 | 5 | | 3 | 4 | 5 | 6 | | 4 | 5 | 6 | 7 | |

1. Напишете програма, която пресмята с колко нули завършва факториелът на дадено число. Примери:

N = 10 -> N! = 36288**00** -> 2

N = 20 -> N! = 243290200817664**0000** -> 4

1. Напишете програма, която преобразува дадено число от десетична в двоична бройна система.
2. Напишете програма, която преобразува дадено число от двоична в десетична бройна система.
3. Напишете програма, която преобразува дадено число от десетична в шестнайсетична бройна система.
4. Напишете програма, която преобразува дадено число от шестнайсе­тична в десетична бройна система.
5. Напишете програма, която по дадено число N отпечатва числата от 1 до N, разбъркани в случаен ред.
6. Напишете програма, която за дадени две числа, намира най-големия им общ делител.
7. Напишете програма, която по дадено число n, извежда матрица във формата на спирала:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 12 | 13 | 14 | 5 |
| 11 | 16 | 15 | 6 |
| 10 | 9 | 8 | 7 |

Пример при n=4

# Глава 7. Масиви

1. Да се напише програма, която създава масив с 20 елемента от целочислен тип и инициализира всеки от елементите със стойност равна на индекса на елемента умножен по 5. Елементите на масива да се изведат на конзолата.
2. Да се напише програма, която чете два масива от конзолата и прове­рява дали са еднакви.
3. Да се напише програма, която сравнява два масива от тип char лексикографски (буква по буква) и проверява кой от двата е по-рано в лексикографската подредба.
4. Напишете програма, която намира максимална редица от последова­телни еднакви елементи в масив. Пример: {2, 1, 1, 2, 3, 3, **2, 2, 2**, 1} 🡪 {2, 2, 2}.
5. Напишете програма, която намира максималната редица от последова­телни нараст­ващи елементи в масив. Пример: {3, **2, 3, 4**, 2, 2, 4} 🡪 {2, 3, 4}.
6. Напишете програма, която намира максималната подредица от нараст­ващи елементи в масив arr[n]. Елементите може и да не са последо­вателни. Пример: {9, 6, **2**, 7, **4**, 7, **6**, 5, **8**, 4} 🡪 {2, 4, 6, 8}.
7. Да се напише програма, която чете от конзолата две цели числа N и K (K<N), и масив от N елемента. Да се намерят тези K поредни елемента, които имат максимална сума.
8. Сортиране на масив означава да подредим елементите му в нарастващ (намаляващ) ред. Напишете програма, която сортира масив. Да се използва алгоритъма "Selection sort".
9. Напишете програма, която намира последователност от числа, чиито сума е максимална. Пример: {2, 3, -6, -1, **2, -1, 6, 4**, -8, 8} 🡪 **11**
10. Напишете програма, която намира най-често срещания елемент в масив. Пример: {**4**, 1, 1, **4**, 2, 3, **4**, **4**, 1, 2, **4**, 9, 3} 🡪 4 (среща се 5 пъти).
11. Да се напише програма, която намира последователност от числа в масив, които имат сума равна на число, въведено от конзолата (ако има такава). Пример: {4, 3, 1, **4, 2, 5**, 8}, S=11 🡪 {4, 2, 5}.
12. Напишете програма, която създава следните квадратни матрици и ги извежда на конзолата във форматиран вид. Размерът на матриците се въвежда от конзолата. Пример за (4,4):

Matrices

1. Да се напише програма, която създава правоъгълна матрица с размер n на m. Размерността и елементите на матрицата да се четат от конзолата. Да се намери подматрицата с размер (3,3), която има максимална сума.
2. Да се напише програма, която намира най-дългата последователност от еднакви string елементи в матрица. Последователност в матрица дефинираме като елементите са на съседни и са на същия ред,колона или диагонал.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ha** | fifi | ho | hi |  |  |
| fo | **ha** | hi | xx |  | ha, ha, ha |
| xxx | ho | **ha** | xx |  |  |

1. Да се напише програма, която създава масив с всички букви от латинската азбука. Да се даде възможност на потребител да въвежда дума от конзолата и в резултат да се извеждат индексите на буквите от думата.
2. Да се реализира двоично търсене (binary search) в **сортиран** целочислен масив.
3. Напишете програма, която сортира целочислен масив по алгоритъма "merge sort".
4. Напишете програма, която сортира целочислен масив по алгоритъма "quick sort".
5. Напишете програма, която намира всички прости числа в диапазона [1…10 000 000].
6. Напишете програма, която по дадени N числа и число S, проверявадали може да се получи сума равна на S с използване на подмасив от N-те числа (не непременно последователни).

Пример: {2, **1, 2**, 4, 3, **5**, 2, **6**}, S = 14 🡪 yes (1 + 2 + 5 + 6 = 14)

1. Напишете програма, която по дадени N, K и S, намира К на брой елементи измежду N-те числа, чиито сума е точно S или показва, че това е невъзможно.
2. Напишете програма, която прочита от конзолата масив от цели числа и премахва минимален на брой числа, така че останали числа да са сортирани в нарастващ ред. Отпечатайте резултата.

Пример: {6, **1,** 4, **3**, 0, **3**, 6, **4**, **5**} 🡪 {1, 3, 3, 4, 5}

1. Напишете програма, която прочита цяло число N от конзолата и отпечатва всички пермутации на числата [1…N].

Пример: N = 3 🡪 {1, 2, 3}, {1, 3, 2}, {2, 1, 3}, {2, 3, 1}, {3, 1, 2}, {3, 2, 1}

1. Напишете програма, която прочита цели числа N и K от конзолата и отпечатва всички вариации от К елемента на числата [1…N].

Пример: N = 3, K = 2 🡪 {1, 1}, {1, 2}, {1, 3}, {2, 1}, {2, 2}, {2, 3}, {3, 1}, {3, 2}, {3, 3}

1. Напишете програма, която прочита цяло число N от конзолата и отпечатва всички комбинации от К елемента на числата [1…N].

Пример: N = 5, K = 2 🡪 {1, 2}, {1, 3}, {1, 4}, {1, 5}, {2, 1}, {2, 3}, {2, 4}, {2, 5}, {3, 1}, {3, 4}, {3, 5}, {4, 5}

1. Напишете програма, която обхожда матрица (NxN) по следния начин:

Пример за N=4:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 16 | 15 | 13 | 10 |  | 7 | 11 | 14 | 16 |
| 14 | 12 | 9 | 6 | 4 | 8 | 12 | 15 |
| 11 | 8 | 5 | 3 | 2 | 5 | 9 | 13 |
| 7 | 4 | 2 | 1 | 1 | 3 | 6 | 10 |

1. \* Напишете програма, която по подадена матрица намира най-голя­мата област от еднакви числа. Под област разбираме съвкупност от съседни (по ред и колона) елементи. Ето един пример, в който имаме област, съставена от 13 на брой еднакви елементи със стойност 3:

same-number-area

# Глава 8. Бройни системи

1. Превърнете числата 151, 35, 43, 251 и -0,41 в двоична бройна система.
2. Превърнете числото 1111010110011110**(2)**в шестнадесетична и в десе­тична бройна система.
3. Превърнете шестнайсетичните числа 2A3E, FA, FFFF, 5A0E9 в двоична и десетична бройна система.
4. Да се напише програма, която преобразува десетично число в дво­ично.
5. Да се напише програма, която преобразува двоично число в десе­тично.
6. Да се напише програма, която преобразува десетично число в шестнадесетично.
7. Да се напише програма, която преобразува шестнадесетично число в десетично.
8. Да се напише програма, която преобразува шестнадесетично число в двоично.
9. Да се напише програма, която преобразува двоично число в шестна­десетично.
10. Да се напише програма, която преобразува двоично число в десетично по схемата на Хорнер.
11. Да се напише програма, която преобразува римските числа в арабски.
12. Да се напише програма, която преобразува арабските числа в римски.
13. Да се напише програма, която по зададени N, S, D (2 ≤ S, D ≥ 16) преобразува числото N от бройна система с основа S към бройна система с основа D.
14. Да се напише програма, която по дадено цяло число извежда на конзолата двоичното представяне на числото.
15. Опитайте да сумирате 50 000 000 пъти числото 0.000001. Използвайте цикъл и събиране (не директно умножение). Опитайте с типовете float и double и след това с decimal. Забелязвате ли разликата в резултатите и в скоростта?
16. \* Да се напише програма, която отпечатва стойността на мантисата, знака на мантисата и стойността на експонентата за числа тип float (32-битови числа с плаваща запетая съгласно стандарта IEEE 754). Пример: за числото -27,25 да се отпечата: знак = 1, експонента = 10000011, мантиса = 10110100000000000000000.

# Глава 9. Методи

1. Напишете метод, който при подадено име отпечатва в конзолата "Hello, <name>!" (например "Hello, Peter!"). Напишете програма, която тества този метод дали работи правилно.
2. Създайте метод GetMax() с два целочислени (int) параметъра, който връща по-голямото от двете числа. Напишете програма, която прочита три цели числа от конзолата и отпечатва най-голямото от тях, изпол­звайки метода GetMax().
3. Напишете метод, който връща английското наименование на послед­ната цифра от дадено число. Примери: за числото 512 отпечатва "two"; за числото 1024 – "four".
4. Напишете метод, който намира колко пъти дадено число се среща в даден масив. Напишете програма, която проверява дали този метод работи правилно.
5. Напишете метод, който проверява дали елемент, намиращ се на дадена позиция от масив, е по-голям, или съответно по-малък от двата му съседа. Тествайте метода дали работи коректно.
6. Напишете метод, който връща позицията на първия елемент на масив, който е по-голям от двата свои съседи едновременно, или -1, ако няма такъв елемент.
7. Напишете метод, който отпечатва цифрите на дадено десетично число в обратен ред. Например 256, трябва да бъде отпечатано като 652.
8. Напишете метод, който пресмята сумата на две цели положителни цели числа. Числата са представени като масив от цифрите си, като последната цифра (единиците) са записани в масива под индекс 0. Направете така, че метода да работи за числа с дължина до 10 000 цифри.
9. Напишете метод, който намира най-големия елемент в част от масив. Използвайте метода за да сортирате възходящо/низходящо даден масив.
10. Напишете програма, която пресмята и отпечатва n! за всяко n в интервала [1…100].
11. Напишете програма, която решава следните задачи:

* Обръща последователността на цифрите на едно число.
* Пресмята средното аритметично на дадена поредица от числа.
* Решава линейното уравнение a \* x + b = 0.

Създайте подходящи методи за всяка една от задачите.

Напишете програмата така, че на потребителя да му бъде изведено текстово меню, от което да избира коя от задачите да решава.

Направете проверка на входните данни:

* Десетичното число трябва да е неотрицателно.
* Редицата не трябва да е празна.
* Коефициентът a не трябва да е 0.

1. Напишете метод, който събира два полинома с цели коефициенти, например (3x2 + x - 3) + (x - 1) = (3x2 + 2x - 4).
2. Напишете метод, който умножава два полинома с цели коефициенти, например (3x2 + x - 3) \* (x - 1) = (3x3 - 2x2 - 4x + 3).

# Глава 10. Рекурсия

1. Напишете програма, която симулира изпълнението на n вложени цикъла от 1 до n. Пример:

1 1 1

1 1 2

1 1 3

1 1 1 2 1

n=2 -> 1 2 n=3 -> ….

2 1 3 2 3

2 2 3 3 1

3 3 2

3 3 3

1. Напишете рекурсивна програма, която генерира и отпечатва всички комбинации с повторение на k елемента над n-елементно множество.

Примерен вход:

|  |
| --- |
| n = 3  k = 2 |

Примерен изход:

|  |
| --- |
| (1 1), (1 2), (1 3), (2 2), (2 3), (3 3) |

Измислете и реализирайте итеративен алгоритъм за същата задача.

1. Напишете рекурсивна програма, която генерира всички вариации с повторение на n елемента от k-ти клас.

Примерен вход:

|  |
| --- |
| n = 3  к = 2 |

Примерен изход:

|  |
| --- |
| (1 1), (1 2), (1 3), (2 1), (2 2), (2 3), (3 1), (3 2), (3 3) |

Измислете и реализирайте итеративен алгоритъм за същата задача.

1. Нека е дадено множество от символни низове. Да се напише рекур­сивна програма, която генерира всички подмножества съставени от точно k на брой символни низа, избрани измежду елементите на това множество.

Примерен вход:

|  |
| --- |
| strings = {'test', 'rock', 'fun'}  k = 2 |

Примерен изход:

|  |
| --- |
| (test rock), (test fun), (rock fun) |

Измислете и реализирайте итеративен алгоритъм за същата задача.

1. Напишете рекурсивна програма, която отпечатва всички подмножества на дадено множество от думи.

Примерен вход:

|  |
| --- |
| words = {'test', 'rock', 'fun'} |

Примерен изход:

|  |
| --- |
| (), (test), (rock), (fun), (test rock), (test fun), (rock fun), (test rock fun) |

Измислете и реализирайте итеративен алгоритъм за същата задача.

1. Реализирайте алгоритъма "сортиране чрез сливане" (merge-sort). При него началният масив се разделя на две равни по големина части, които се сортират (рекурсивно чрез merge-sort) и след това двете сортирани части се сливат, за да се получи целият масив в сортиран вид.
2. Напишете рекурсивна програма, която генерира и отпечатва пермута­циите на числата 1, 2, …, n, за дадено цяло число n.

Примерен вход:

|  |
| --- |
| n = 3 |

Примерен изход:

|  |
| --- |
| (1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1) |

1. Даден е масив с цели числа и число N. Напишете рекурсивна прог­рама, която намира всички подмножества от числа от масива, които имат сума N. Например ако имаме масива {2, 3, 1, -1} и N=4, можем да получим N=4 като сума по следните два начина: 4=2+3-1; 4=3+1.
2. Даден е масив с цели **положителни** числа. Напишете програма, която проверява дали в масива съществуват едно или повече числа, чиято сума е N. Можете ли да решите задачата без рекурсия?
3. Дадена е матрица с проходими и непроходими клетки. Напишете рекурсивна програма, която намира всички пътища между две клетки в матрицата.
4. Модифицирайте горната програма, за да проверява дали съществува път между две клетки без да се намират всички възможни пътища. Тествайте за матрица 100х100 пълна само с проходими клетки.
5. Напишете програма, която намира най-дългата поредица от съседни проходими клетки в матрица.
6. Даден е двумерен масив с проходими и непроходими клетки. Напишете програма, която намира всички площи съставени само от проходими клетки.
7. Реализирайте алгоритъма BFS (breath-first search) за търсене на най-кратък път в лаби­ринт. Ако се затруднявате, потърсете информация в Интернет.
8. Напишете рекурсивна програма, която обхожда целия твърд диск C:\ рекурсивно и отпечатва всички папки и файловете в тях.

# Глава 11. Създаване и използване на обекти

1. Напишете програма, която прочита от конзолата година и проверява дали е високосна.
2. Напишете програма, която генерира и принтира на конзолата 10 случайни числа в интервала [100, 200].
3. Напишете програма, която извежда на конзолата кой ден от седмицата е днес.
4. Напишете програма, която извежда на стандартния изход броя на дните, часовете и минутите, които са изтекли от включването на ком­пютъра до момента на изпълнението на програмата. За реализацията изпол­звайте класа Environment.
5. Напишете програма, която по дадени два катета намира хипотенузата на правоъгълен триъгълник. Реализирайте въвеждане на дължините на катетите от стандартния вход, а за пресмятането на хипотенузата използвайте методи на класа Math.
6. Напишете програма, която пресмята лице на триъгълник по:
   1. дължините на трите му страни;
   2. дължината на една от страните и височината към нея;
   3. дължините на две от страните му и ъгъла между тях в градуси.
7. Дефинирайте свое собствено пространство Chapter11 и поставете в него двата класа Cat и Sequence, които използвахме в примерите на текущата тема. Направете още едно собствено пространство с име Chapter11.Examples и в него направете клас, който извиква класовете Cat и Sequence.
8. Напишете програма, която създава 10 обекта от тип Cat, дава им имена от вида CatN, където N e уникален пореден номер на обекта, и накрая извиква метода SayMiau() на всеки от тях. За реализацията използвайте вече дефинираното пространство Chapter11.
9. Напишете програма, която пресмята броя работни дни между днешната дата и дадена друга дата след днешната (включително). Работните дни са всички дни без събота и неделя, които не са официални празници, като по изключение събота може да е работен ден, когато се отра­ботват почивни дни около празниците. Програмата трябва да пази списък от предварително зададени официални празници, както и списък от предварително зададени работни съботи.
10. Дадена е последователност от цели положителни числа, записани едно след друго като символен низ, разделени с интервал. Да се напише програма, която пресмята сумата им. Пример: "43 68 9 23 318" 🡪 461.
11. Напишете програма, която генерира случайно рекламно съобщение за някакъв продукт. Съобщенията трябва да се състоят от хвалебст­вена фраза, следвани от хвалебствена случка, следвани от автор (първо и второ име) и град, които се избират от предварително подготвени списъци. Например, нека имаме следните списъци:

* Хвалебствени фрази: {"Продуктът е отличен.", "Това е страхотен продукт.", "Постоянно ползвам този продукт.", "Това е най-добрият продукт от тази категория."}.
* Хвалебствени случки: {"Вече се чувствам добре.", "Успях да се променя.", "Той направи чудо.", "Не мога да повярвам, но вече се чувствам страхотно.", "Опитайте и вие. Аз съм много доволна."}.
* Първо име на автор: {"Диана", "Петя", "Стела", "Елена", "Катя"}.
* Второ име на автор: {"Иванова", "Петрова", "Кирова"}.
* Градове: {"София", "Пловдив", "Варна", "Русе", "Бургас"}.

Тогава програма би могла да изведе следното случайно-генерирано рекламно съобщение:

|  |
| --- |
| Постоянно ползвам този продукт. Опитайте и вие. Аз съм доволна. -– Елена Петрова, Варна |

1. \* Напишете програма, която изчислява стойността на даден числен израз, зададен като стринг. Численият израз се състои от:

* реални числа, например 5, 18.33, 3.14159, 12.6;
* аритметични оператори: +, -, \*, / (със стандартните им приоритети);
* математически функции: ln(x), sqrt(x), pow(x,y);
* скоби за промяна на приоритета на операциите: ( и ).

Обърнете внимание, че числовите изрази имат приоритет, например изразът -1 + 2 + 3 \* 4 - 0.5 = (-1) + 2 + (3 \* 4) - 0.5 = 12.5.

# Глава 12. Обработка на изключения

1. Да се намерят всички стандартни изключения от йерархията на System.IO.IOException.
2. Да се намерят всички стандартни изключения от йерархията на System.IO.FileNotFoundException.
3. Да се намерят всички стандартни изключения от йерархията на System.ApplicationException.
4. Обяснете какво представляват изключенията, кога се използват и как се прихващат.
5. Обяснете ситуациите, при които се използва try-finally конструк­цията. Обяснете връзката между try-finally и using конструкциите.
6. Обяснете предимствата на използването на изключения.
7. Напишете програма, която прочита от конзолата цяло положително число и отпечатва на конзолата корен квадратен от това число. Ако числото е отрицателно или невалидно, да се изпише "Invalid Number" на конзолата. Във всички случаи да се принтира на конзолата "Good Bye".
8. Напишете метод ReadNumber(int start, int end), който въвежда от конзолата число в диапазона [start…end]. В случай на въведено невалидно число или число, което не е в подадения диапазон хвърлете подходящо изключение. Използвайки този метод напишете програма, която въвежда 10 числа a1, a2, …, a10, такива, че 1 < a1 < … < a10 < 100.
9. Напишете метод, който приема като параметър име на текстов файл, прочита съдържанието му и го връща като string. Какво е правилно да направи методът с евентуално възникващите изключения?
10. Напишете метод, който приема като параметър име на бинарен файл и прочита съдържанието на файла и го връща като масив от байтове. Напишете метод, който записва прочетеното съдържание в друг файл. Сравнете двата файла.
11. Потърсете информация в Интернет и дефинирайте собствен клас за изключение FileParseException. Вашето изключение трябва да съ­държа в себе си името на файл, който се обработва и номер на ред, в който е възникнал проблем. Добавете подходящи конструктори за вашето изключение. Напишете програма, която чете от текстов файл числа. Ако при четенето се стигне до ред, който не съдържа число, хвърлете FileParseException и го обработете в извикващия метод.
12. Напишете програма, която прочита от потребителя пълен път до даден файл (например C:\Windows\win.ini), прочита съдържанието на файла и го извежда на конзолата. Намерете в MSDN как да използвате метода System.IO.File.ReadAllText(…). Уверете се, че прихващате всички възможни изключения, които могат да възникнат по време на работа на метода и извеждайте на конзолата съобщения за грешка, разбираеми за обикновения потребител.
13. Напишете програма, която изтегля файл от Интернет по даден URL адрес, примерно (<http://www.devbg.org/img/Logo-BASD.jpg>).

# Глава 13. Символни низове

1. Разкажете за низовете в C#. Какво е типично за типа string? Обяснете кои са най-важните методи на класа string.
2. Напишете програма, която прочита символен низ, обръща го отзад напред и го принтира на конзолата. Например: "introduction" 🡪 "noitcudortni".
3. Напишете програма, която проверява дали в даден аритметичен израз скобите са поставени коректно. Пример за израз с коректно поставени скоби: ((a+b)/5-d). Пример за некоректен израз: )(a+b)).
4. Колко обратни наклонени черти трябва да посочите като аргумент на метода Split(…), за да разделите текста по обратна наклонена черта?

Пример: one\two\three

Забележка: В C# обратната наклонена черта е екраниращ символ.

1. Напишете програма, която открива колко пъти даден подниз се съдържа в текст. Например нека търсим подниза "in" в текста:

|  |
| --- |
| We are liv**in**g **in** a yellow submar**in**e. We don't have anyth**in**g else. **In**side the submar**in**e is very tight. So we are dr**in**k**in**g all the day. We will move out of it **in** 5 days. |

Резултатът е 9 срещания.

1. Даден е текст. Напишете програма, която променя регистъра на буквите до главни на всички места в текста, заградени с таговете <upcase> и </upcase>. Таговете не могат да бъдат вложени.

Пример:

|  |
| --- |
| We are living in a <upcase>yellow submarine</upcase>. We don't have <upcase>anything</upcase> else. |

Резултат:

|  |
| --- |
| We are living in a YELLOW SUBMARINE. We don't have ANYTHING else. |

1. Напишете програма, която чете от конзолата стринг от максимум 20 символа и ако е по-кратък го допълва отдясно със "\*" до 20 символа.
2. Напишете програма, която преобразува даден стринг във вид на поредица от Unicode екраниращи последователности. Примерен входен стринг: "Наков". Резултат: "\u041d\u0430\u043a\u043e\u0432".
3. Напишете програма, която кодира текст по даден шифър като прилага шифъра побуквено с операция XOR (изключващо или) върху текста. Кодирането трябва да се извършва като се прилага XOR между пър­вата буква от текста и първата буква на шифъра, втората буква от текста и втората буква от шифъра и т.н. до последната буква от шифъра, след което се продължава отново с първата буква от шифъра и поредната буква от текста. Отпечатайте резултата като поредица от Unicode кодирани екраниращи символи.

Примерен текст: "Nakov". Примерен шифър: "ab". Примерен резултат: "\u002f\u0003\u000a\u000d\u0017".

1. Напишете програма, която извлича от даден текст всички изречения, които съдържат определена дума. Считаме, че изреченията са разде­лени едно от друго със символа ".", а думите са разделени една от друга със символ, който не е буква. Примерен текст:

|  |
| --- |
| We are living **in** a yellow submarine. We don't have anything else. Inside the submarine is very tight. So we are drinking all the day. We will move out of it **in** 5 days. |

Примерен резултат:

|  |
| --- |
| We are living in a yellow submarine.  We will move out of it in 5 days. |

1. Даден е символен низ, съставен от няколко "забранени" думи, разде­лени със запетая. Даден е и текст, съдържащ тези думи. Да се напише програма, която замества забранените думи в текста със звездички. Примерен текст:

|  |
| --- |
| Microsoft announced its next generation C# compiler today. It uses advanced parser and special optimizer for the Microsoft CLR. |

Примерен низ от забранените думи: "C#,CLR,Microsoft".

Примерен съответен резултат:

|  |
| --- |
| \*\*\*\*\*\*\*\*\* announced its next generation \*\* compiler today. It uses advanced parser and special optimizer for the \*\*\*\*\*\*\*\*\* \*\*\*. |

1. Напишете програма, която чете число от конзолата и го отпечатва в 15-символно поле, подравнено вдясно по няколко начина: като десе­тично число, като шестнайсетично число, като процент, като валутна сума и във вид на експоненциален запис (scientific notation).
2. Напишете програма, която приема URL адрес във формат:

|  |
| --- |
| [protocol]://[server]/[resource] |

и извлича от него протокол, сървър и ресурс. Например при подаден адрес: <http://www.devbg.org/forum/index.php> резултатът е:

[protocol]="http"

[server]="www.devbg.org"

[resource]="/forum/index.php"

1. Напишете програма, която обръща думите в дадено изречение без да променя пунктуацията и интервалите. Например: "C# is not C++ and PHP is not Delphi" -> "Delphi not is PHP and C++ not is C#".
2. Даден е тълковен речник, който се състои от няколко реда текст. На всеки ред има дума и нейното обяснение, разделени с тире:

|  |
| --- |
| .NET – platform for applications from Microsoft CLR – managed execution environment for .NET namespace – hierarchical organization of classes |

Напишете програма, която парсва речника и след това в цикъл чете дума от конзолата и дава обяснение за нея или съобщение, че думата липсва в речника.

1. Напишете програма, която заменя в HTML документ всички препратки (hyperlinks) от вида <a href="…">…</a> с препратки стил "форум", които имат вида [URL=…]…/URL].

Примерен текст:

|  |
| --- |
| <p>Please visit <a href="http://academy.telerik.com">our site</a> to choose a training course. Also visit <a href="www.devbg.org">our forum</a> to discuss the courses.</p> |

Примерен съответен резултат:

|  |
| --- |
| <p>Please visit [URL=http://academy.telerik.com]our site[/URL] to choose a training course. Also visit [URL=www.devbg.org]our forum[/URL] to discuss the courses.</p> |

1. Напишете програма, която чете две дати, въведени във формат "ден.месец.година" и изчислява броя дни между тях.

|  |
| --- |
| Enter the first date: 27.02.2006  Enter the second date: 3.03.2004  Distance: 4 days |

1. Напишете програма, която чете дата и час, въведени във формат "ден.месец.година час:минути:секунди" и отпечатва датата и часа след 6 часа и 30 минути, в същия формат.
2. Напишете програма, която извлича от даден текст всички e-mail адреси. Това са всички поднизове, които са ограничени от двете страни с край на текст или разделител между думи и съответстват на формата <sender>@<host>…<domain>. Примерен текст:

|  |
| --- |
| Please contact us by phone (+359 222 222 222) or by email at [example@abv.bg](mailto:example@abv.bg) or at [baj.ivan@yahoo.co.uk](mailto:baj.ivan@yahoo.co.uk). This is not email: test@test. This also: @telerik.com. Neither this: a@a.b. |

Извлечени e-mail адреси от примерния текст:

|  |
| --- |
| example@abv.bg  baj.ivan@yahoo.co.uk |

1. Напишете програма, която извлича от даден текст всички дати, които се срещат изписани във формат DD.MM.YYYY и ги отпечатва на конзолата в стандартния формат за Канада. Примерен текст:

|  |
| --- |
| I was born at 14.06.1980. My sister was born at 3.7.1984. In 5/1999 I graduated my high school. The law says (see section 7.3.12) that we are allowed to do this (section 7.4.2.9). |

Извлечени дати от примерния текст:

|  |
| --- |
| 14.06.1980  3.7.1984 |

1. Напишете програма, която извлича от даден текст всички думи, които са палиндроми, например "ABBA", "lamal", "exe".
2. Напишете програма, която чете от конзолата символен низ и отпечатва в азбучен ред всички букви от въведения низ и съответно колко пъти се среща всяка от тях.
3. Напишете програма, която чете от конзолата символен низ и отпечатва в азбучен ред всички думи от въведения низ и съответно колко пъти се среща всяка от тях.
4. Напишете програма, която чете от конзолата символен низ и заменя в него всяка последователност от еднакви букви с единична съответна буква. Пример: "aaaaabbbbbcdddeeeedssaa" 🡪 "abcdedsa".
5. Напишете програма, която чете от конзолата списък от думи, разде­лени със запетайки и ги отпечатва по азбучен ред.
6. Напишете програма, която изважда от даден HTML документ всичкия текст без таговете и техните атрибути.

Примерен текст:

|  |
| --- |
| <html>  <head><title>News</title></head>  <body><p><a href="http://academy.telerik.com">Telerik  Academy</a>aims to provide free real-world practical  training for young people who want to turn into  skillful .NET software engineers.</p></body>  </html> |

Примерен съответен резултат:

|  |
| --- |
| Title: News  Body:  Telerik Academy aims to provide free real-world practical training for young people who want to turn into skillful .NET software engineers. |

# Глава 14. Дефиниране на класове

1. Дефинирайте клас Student, който съдържа следната информация за студентите: трите имена, курс, специалност, университет, електронна поща и телефонен номер.
2. Декларирайте няколко конструктора за класа Student, които имат различни списъци с параметри (за цялостната информация за даден студент или част от нея). Данните, за които няма входна информация да се инициализират съответно с null или 0.
3. Добавете статично поле в класа Student, в което се съхранява броя на създадените обекти от този клас.
4. Добавете метод в класа Student, който извежда пълна информация за студента.
5. Модифицирайте текущия код на класа Student така, че да капсулирате данните в класа чрез свойства.
6. Напишете клас StudentTest, който да тества функционалността на класа Student.
7. Добавете статичен метод в класа StudentTest, който създава няколко обекта от тип Student и ги съхранява в статични полета. Създайте статично свойство на класа, което да ги достъпва. Напишете тестова програма, която да извежда информацията за тях в конзолата.
8. Дефинирайте клас, който съдържа информация за мобилен телефон: модел, производител, цена, собственик, характеристики на батерията (модел, idle time и часове разговор /hours talk/) и характеристики на екрана (големина и цветове).
9. Декларирайте няколко конструктора за всеки от създадените класове от предходната задача, които имат различни списъци с параметри (за цялостната информация за даден студент или част от нея). Данните за полетата, които не са известни трябва да се инициализират съответно със стойности с null или 0.
10. Към класа за мобилен телефон от предходните две задачи, добавете статично поле nokiaN95, което да съхранява информация за мобилен телефон модел Nokia 95. Добавете метод, в същия клас, който извежда информация за това статично поле.
11. Добавете изброим тип BatteryType, който съдържа стойности за тип на батерията (Li-Ion, NiMH, NiCd, …) и го използвайте като ново поле за класа Battery.
12. Добавете метод в класа GSM, който да връща информация за обекта под формата на string.
13. Дефинирайте свойства, за да капсулирате данните в класовете GSM, Battery и Display.
14. Напишете клас GSMTest, който тества функционалностите на класа GSM. Създайте няколко обекта от дадения клас и ги запазете в масив. Изведете информация за създадените обекти. Изведете информация за статичното поле nokiaN95.
15. Създайте клас Call, който съдържа информация за разговор, осъщес­твен през мобилен телефон. Той трябва да съдържа информация за датата, времето на започване и продължителността на разговора.
16. Добавете свойство архив с обажданията – callHistory, което да пази списък от осъщест­вените разговори.
17. В класа GSM добавете методи за добавяне и изтриване на обаждания (Call) в архива с обаждания на мобилния телефон. Добавете метод, който изтрива всички обаждания от архива.
18. В класа GSM добавете метод, който пресмята общата сума на обажда­нията (Call) от архива с обаждания на телефона (callHistory) като нека цената за едно обаждане се подава като параметър на метода.
19. Създайте клас GSMCallHistoryTest, с който да се тества функционал­ността на класа GSM, от задача 12, като обект от тип GSM. След това, към него добавете няколко обаждания (Call). Изведете информация за всяко едно от обажданията. Ако допуснем, че цената за минута раз­говор е 0.37, пресметнете и отпечатайте общата цена на разговорите. Премахнете най-дългият разговор от архива с обаждания и пресмет­нете общата цена за всички разговори отново. Най-накрая изтрийте архива с обаждания.
20. Нека е дадена библиотека с книги. Дефинирайте класове съответно за библиотека и книга. Библиотеката трябва да съдържа име и списък от книги. Книгите трябва да съдържат информация за заглавие, автор, издателство, година на издаване и ISBN-номер. В класа, който описва библиотека, добавете методи за добавяне на книга към библиотеката, търсене на книга по предварително зададен автор, извеждане на информация за дадена книга и изтриване на книга от библиотеката.
21. Напишете тестов клас, който създава обект от тип библиотека, добавя няколко книги към него и извежда информация за всяка една от тях. Имплементирайте тестова функционалност, която намира всички книги, чийто автор е Стивън Кинг и ги изтрива. Накрая, отново изведете информация за всяка една от оставащите книги.
22. Дадено ни е училище. В училището имаме класове и ученици. Всеки клас има множество от преподаватели. Всеки преподавател има мно­жест­во от дисциплини, по които преподава. Учениците имат име и уникален номер в класа. Класовете имат уникален текстов иден­тификатор. Дисциплините имат име, брой уроци и брой упражне­ния.   
    Задачата е да се моделира училище с C# класове. Трябва да декларирате класове заедно с техните полета, свойства, методи и конструктори. Дефинирайте и тестов клас, който демонстрира, че останалите класове работят коректно.
23. Напишете типизиран клас GenericList<T>, който пази списък от елементи от тип T. Пазете елементите от списъка в масив с фиксиран капацитет, който е зададен като параметър на конструктора на класа. Добавете методи за добавяне на елемент, достъпване на елемент по индекс, премахване на елемент по индекс, вмъкване на елемент на зададена позиция, изчистване на списъка, търсене на елемент по стойност и предефинирайте метода ToString().
24. Имплементирайте автоматично преоразмеряване на масива от предната задача, когато при добавяне на елемент се достигне капацитета на масива.
25. Дефинирайте клас Fraction, който съдържа информация за рационална дроб (например ¼, ½). Дефинирайте статичен метод Parse(), който да опитва да създаде дроб от символен низ (например -3/4). Дефинирайте подходящи свойства и конструктори на класа. Напишете и свойство от тип Decimal, което връща десетичната стойност на дробта (например 0.25).
26. Напишете клас FractionTest, който тества функционалността на класа от предната задача Fraction. Отделете специално внимание на тестването на функцията Parse с различни входни данни.
27. Напишете функция, която съкращава дробта (Например ако числителя и знаменателя са съответно 10 и 15, дробта да се съкращава до 2/3).

# Глава 15. Текстови файлове

1. Напишете програма, която чете от текстов файл и отпечатва нечетните му редове на конзолата.
2. Напишете програма, която съединява два текстови файла и записва резултата в трети файл.
3. Напишете програма, която прочита съдържанието на текстов файл и вмъква номерата на редовете в началото на всеки ред и след това записва обратно съдържанието на файла.
4. Напишете програма, която сравнява ред по ред два текстови файла с еднакъв брой редове и отпечатва броя съвпадащи и броя различни редове.
5. Напишете програма, която чете от файл квадратна матрица от цели числа и намира подматрицата с размери 2 х 2 с най-голяма сума и записва тази сума в отделен текстов файл. Първият ред на входния файл съдържа големината на записаната матрица (N). Следващите N реда съдържат по N числа, разделени с интервал.

Примерен входен файл:

|  |
| --- |
| 4  2 3 3 4  0 2 3 4  **3 7** 1 2  **4 3** 3 2 |

Примерен изход: 17.

1. Напишете програма, която чете списък от имена от текстов файл, подрежда ги по азбучен ред и ги запазва в друг файл. Имената са записани по едно на ред.
2. Напишете програма, която заменя всяко срещане на подниза "start" с "finish" в текстов файл. Можете ли да пренапишете програмата така, че да заменя само цели думи? Работи ли програмата за големи фай­лове (примерно 800 MB)?
3. Напишете предната програма така, че да заменя само целите думи (не части от думи).
4. Напишете програма, която изтрива от текстов файл всички нечетни редове.
5. Напишете програма, която извлича от XML файл всичкия текст без таговете. Примерен входен файл:

|  |
| --- |
| <?xml version="1.0"><student><name>**Pesho**</name> <age>**21**</age><interests count="3"><interest> **Games**</instrest><interest>**C#**</instrest><interest> **Java**</instrest></interests></student> |

Примерен резултат:

|  |
| --- |
| Pesho  21  Games  C#  Java |

1. Напишете програма, която изтрива от текстов файл всички думи, които започват с "test". Думите съдържат само символите 0...9, a…z, A…Z,\_.
2. Даден е текстов файл words.txt, съдържащ списък от думи, по една на ред. Напишете програма, която изтрива от файла text.txt всички думи, които се срещат в другия файл. Прихванете всички възможни изключения (Exceptions).
3. Напишете програма, която прочита списък от думи от файл, наречен words.txt, преброява колко пъти всяка от тези думи се среща в друг файл text.txt и записва резултата в трети файл – result.txt, като преди това ги сортира по броя срещания в намаляващ ред. Прихванете всички възможни изключения (Exceptions).

# Глава 16. Линейни структури от данни

1. Напишете програма, която прочита от конзолата поредица от цели положителни числа. Поредицата спира когато се въведе празен ред. Програмата трябва да изчислява сумата и средното аритметично на поредицата. Използвайте List<int>.
2. Напишете програма, която прочита N цели числа от конзолата и ги отпечатва в обратен ред. Използвайте класа Stack<int>.
3. Напишете програма, която прочита от конзолата поредица от цели положителни числа, поредицата спира когато се въведе празен ред, и ги сортира възходящо.
4. Напишете метод, който намира най-дългата подредица от равни числа в даден List<int> и връща като резултат нов List<int> със тази подредица. Напишете програма, която проверява дали този метод работи коректно.
5. Напишете програма, която премахва всички отрицателни числа от дадена редица.

Пример: array = {19, -10, 12, -6, -3, 34, -2, 5} 🡪 {19, 12, 34, 5}

1. Напишете програма, която при дадена редица изтрива всички числа, които се срещат нечетен брой пъти.

Пример: array = {4, 2, 2, 5, 2, 3, 2, 3, 1, 5, 2} 🡪 {5, 3, 3, 5}

1. Напишете програма, която по даден масив от цели числа в интервала [0..1000], намира по колко пъти се среща всяко число.

Пример: array = {3, 4, 4, 2, 3, 3, 4, 3, 2}

2 🡪 2 пъти

3 🡪 4 пъти

4 🡪 3 пъти

1. Мажорант на масив от N елемента е стойност, която се среща поне N/2+1 пъти. Напишете програма, която по даден масив от числа намира мажоранта на масива и го отпечатва. Ако мажоранта не съществува – отпечатва "The majorant does not exists!”.

Пример: {2, 2, 3, 3, 2, 3, 4, 3, 3} 🡪 3

1. Дадена е следната поредица:

S1 = N;

S2 = S1 + 1;

S3 = 2\*S1 + 1;

S4 = S1 + 2;

S5 = S2 + 1;

S6 = 2\*S2 + 1;

S7 = S2 + 2;

...

Използвайки класа Queue<T> напишете програма, която по дадено N отпечатва на конзолата първите 50 числа от тази поредица.

Пример: N=2 🡪 2, 3, 5, 4, 4, 7, 5, 6, 11, 7, 5, 9, 6, ...

1. Дадени са числа N и M и следните операции:

N = N+1

N = N+2

N = N\*2

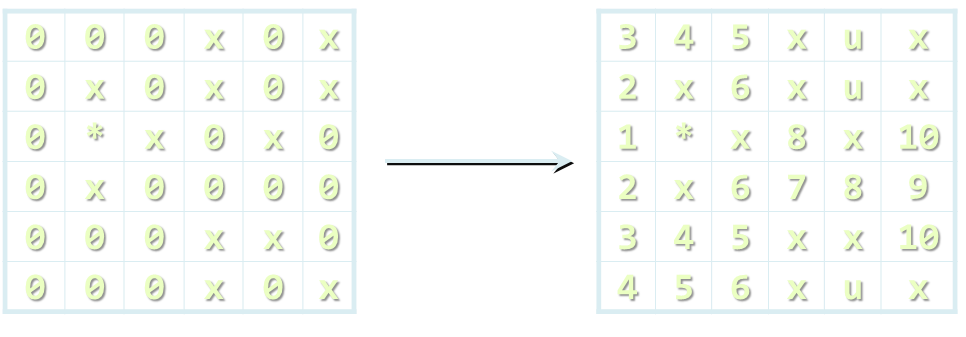
Напишете програма, която намира най-кратката поредица от посочените операции, която започва от N и завършва в M. Използвайте опашка.

Пример: N = 5, M = 16

Поредицата е: 5 🡪 7 🡪 8 🡪 16

1. Реализирайте структурата двойно свързан динамичен списък – списък, чиито елементи имат указател, както към следващия така и към пред­хождащия го елемент. Реализирайте операциите добавяне, премахване и търсене на елемент, добавяне на елемент на определено място (индекс), извличане на елемент по индекс и метод, който връща масив с елементите на списъка.
2. Създайте клас DynamicStack представляващ динамична реализация на стек. Добавете методи за необходимите операции.
3. Реализирайте структурата от данни "дек". Това е специфична списъчна структура, подобна на стек и опашка, позволя­ваща елементи да бъдат добавяни и премахвани от двата й края. Нека освен това, елемент поставен от едната страна да може да бъде премахнат само от същата. Реализирайте операции за премах­ване добавяне и изчистване на дека. При невалидна операция подавайте подходящо изключение.
4. Реализирайте структурата "зациклена опашка" с масив, който при нужда удвоява размера си. Имплементирайте необходимите методи за добавяне към опашката, извличане на елемента, който е наред и поглеждане на елемента, който е наред, без да го премахвате от опашката. При невалидна операция подавайте подходящо изключение.
5. Реализирайте сортиране на числа в динамичен свързан списък, без да използвате допълнителен масив.
6. Използвайки опашка реализирайте пълно обхождане на всички дирек­тории на твърдия ви диск и ги отпечатвайте на конзолата. Реализи­райте алгоритъма "обхождане в ширина" – Breadth-First-Search (BFS) – може да намерите стотици статии за него в Интернет.
7. Използвайки опашка реализирайте пълно обхождане на всички дирек­тории на твърдия ви диск и ги отпечатвайте на конзолата. Реализи­райте алгоритъма "обхождане в дълбочина" – Depth-First-Search (DFS) – може да намерите стотици статии за него в Интернет.
8. Даден е лабиринт с размери N x N. някои от клетките на лабиринта са празни (0) а други са запълнени (х). Можем да се движим от празна клетка до друга празна клетка, ако двете имат обща стена. При дадена начална позиция (\*) изчислете и попълнете лабиринта с минималната дължина от началната позиция до всяка друга. Ако някоя клетка не може да бъде достигната я попълнете с "u”.

Пример:



# Глава 17. Дървета и графи

1. Да се напише програма, която намира броя на срещанията на дадено число в дадено дърво от числа.
2. Да се напише програма, която извежда корените на онези поддървета на дадено дърво, които имат точно **k** на брой върха, където **k** e дадено естествено число.
3. Да се напише програма, която намира броя на листата и броя на вътрешните върхове на дадено дърво.
4. Напишете програма, която по дадено двоично дърво от числа намира сумата на върховете от всяко едно ниво на дървото.
5. Да се напише програма, която намира и отпечатва всички върхове на двоично дърво, които имат за наследници само листа.
6. Да се напише програма, която проверява дали дадено двоично дърво е идеално балансирано.
7. Нека е даден граф **G(V, E)** и два негови върха **x** и **y**. Напишете програма, която намира най-краткия път между два върха по брой на върховете.
8. Нека е даден граф **G(V, E)**. Напишете програма, която проверява дали графът е цикличен.
9. Напишете рекурсивно обхождане в дълбочина и програма, която да го тества.
10. Напишете обхождане в ширина (BFS), базирано на опашка.
11. Напишете програма, която обхожда директорията C:\Windows\ и всичките и поддиректории рекурсивно и отпечатва всички файлове, който имат разширение \*.exe.
12. Дефинирайте класове File { string name, int size } и Folder { string name, File[] files, Folder[] childFolders }. Използвайки тези класове, постройте дърво, което съдържа всички файлове и директории на твърдия диск, като започнете от C:\Windows\. Напишете метод, който изчислява сумата от големините на файловете в дадено поддърво и програма, която тества този метод. За обхождането на директориите използвайте рекурсивно обхождане в дълбочина (DFS).
13. Напишете програма, която намира всички цикли в даден граф.
14. Нека е даден граф **G(V, E)**. Напишете програма, която намира всички компоненти на свързаност на графа, т.е. намира всички негови максимални свързани подграфи. Максимален свързан подграф на **G** е свързан граф такъв, че няма друг подграф на **G**, който да е свързан и да го съдържа.
15. Нека е даден претеглен ориентиран граф **G(V, E)**, в който теглата по ребрата са неотрицателни числа. Напишете програма, която по зададен връх **x** от графа намира минималните пътища от него до всички останали.
16. Имаме N задачи, които трябва да бъдат изпълнени последователно. Даден е списък с двойки задачи, за които втората зависи от резултата от първата и трябва да бъде изпълнена след нея. Напишете програма, която подрежда задачите по такъв начин, че всяка задача да се изпълни след всички задачи, от които зависи. Ако не съществува такава наредба, да се отпечата подходящо съобщение.

Пример: {1, 2}, {2, 5}, {2, 4}, {3, 1} 🡪 3, 1, 2, 5, 4

1. Ойлеров цикъл в граф се нарича цикъл, който започва от даден връх, минава точно по веднъж през всички негови ребра и се връща в началния връх. При това обхождане всеки връх може да бъде посетен многократно. Напишете програма, която по даден граф намира в него Ойлеров цикъл или установява, че такъв няма.
2. Хамилтонов цикъл в граф се нарича цикъл, съдържащ всеки връх в графа точно по веднъж. Да се напише програма, която при даден претеглен ориентиран граф **G(V, E)**, намира Хамилтонов цикъл с минимална дължина, ако такъв съществува.

# Глава 18. Речници, хеш-таблици и множества

1. Напишете програма, която брои колко пъти се среща всяко число в дадена редица от числа.

Пример: array = {3, 4, 4, 2, 3, 3, 4, 3, 2}

2 🡪 2 пъти

3 🡪 4 пъти

4 🡪 3 пъти

1. Напишете програма, която премахва всички числа, които се срещат нечетен брой пъти в дадена редица. Например, ако имаме началната редица {4, 2, 2, 5, 2, 3, 2, 3, 1, 5, 2, 6, 6, 6}**,** трябва да я редуцираме до редицата {5, 3, 3, 5}.
2. Напишете програма, която по даден текст във текстов файл, преброява колко пъти се среща всяка дума. Отпечатайте на конзолата всички думи и по колко пъти се срещат, подредени по брой срещания.

Пример: "This is the TEXT. Text, text, text – THIS TEXT! Is this the text?"

Резултат:

is 🡪 2, the 🡪 2, this 🡪 3, text 🡪 6

1. Реализирайте клас DictHashSet<Т>, базиран на класа HashDictionary <K, V>, който разгледахме по-горе.
2. Реализирайте хеш-таблица, която съхранява тройки стойности (ключ1, ключ2, стойност) и позволява бързо търсене по двойка ключове и добавяне на тройки стойности.
3. Реализирайте хеш-таблица, която позволява по даден ключ да съхраняваме повече от една стойност.
4. Реализирайте хеш-таблица, която използва "кукувиче хеширане" с 3 хеш-функции за разрешаване на колизиите.
5. Реализирайте структурата данни хеш-таблица в клас HashTable<K, T>. Пазете данните в масив от списъци от двойки ключ-стойност (LinkedList<KeyValuePair<K,T>>[]) с начален капацитет от 16 елемента. Когато хеш-таблиц­­ата ­­достигне 75% от своя капацитет да се удвоява капацитета. Реализирайте следните операции: Add(key, value), Find(key)🡪value, Remove(key), Count, Clear(), this[], Keys. Реализирайте и итериране по елементите на хеш-таблицата с foreach.
6. Реализирайте структурата от данни "Set" в клас HashedSet<T>. Използвайте класа от предната задача HashTable<K, T>, за да пазите елементите. Имплементирайте всички стандартни операции за типа данни Set: Add(T), Find(T), Remove(T), Count, Clear(),обединение и сечение.
7. Дадени са три редици от числа, дефинирани чрез формулите:

* f1(0) = 1; f1(k) = 2\*f1(k-1) + 3; f1 = {1, 5, 13, 29, …}
* f2(0) = 2; f2(k) = 3\*f2(k-1) + 1; f2 = {2, 7, 22, 67, …}
* f3(0) = 2; f3(k) = 2\*f3(k-1) - 1; f3 = {2, 3, 5, 9, …}

Напишете програма, която намира сечението и обединението на множе­ствата от членовете на редиците в интервала [0; 100000]: f1 \* f2; f1 \* f3; f2 \* f3; f1 \* f2 \* f3; f1 + f2; f1 + f3; f2 + f3; f1 + f2 + f3. Със символите + и \* означаваме съответно обединение и сечение на множества.

1. \* Дефинирайте клас TreeMultiSet<T>, който позволява да пазим съвкупност от елементи, подредени по големина и позволява повто­рения на някои от елементите. Реализирайте операциите добавяне на елемент, търсене на броя срещания на даден елемент, изтриване на елемент, итератор, намиране на най-малък / най-голям елемент, изтриване на най-малък / най-голям елемент. Реализирайте възможност за подаване на външен Comparer<T> за сравнение на елементите.
2. \* Даден е списък с времената на пристигане и заминаване на всички автобуси от дадена автогара. Да се напише програма, която използ­вайки HashSet класa по даден интервал (начало, край) намира броя автобуси, които успяват да пристигнат и да напуснат авто­гарата. Пример:

Имаме данните за следните автобуси: [08:24-08:33], [08:20-09:00], [08:32-08:37], [09:00-09:15]. Даден е интервалът [08:22-09:05]. Броят автобуси, които идват и си тръгват в рамките на този интервал е 2.

1. \* Дадена е редица P с цели числа (1 < P < 50 000) и число N. Щастлива под-редица в редицата P наричаме всяка съвкупност, състояща се от последо­вателни числа от P, чиято сума е N. Да си представим, че имаме редицата S, състояща се от всички щастливи под-редици в P, подредени в намаляващ ред спрямо дължината им. Напишете програма, която извежда първите 10 елемента на S.

Пример: Имаме N=5 и редицата P={1, 1, 2, 1, -1, 2, 3, -1, 1, 2, 3, 5, 1, -1, 2, 3}. Редицата S се състои от следните 13 под-редици на P:

* **[1, -1, 2, 3, -1, 1]**
* **[1, 2, 1, -1, 2]**
* **[1, -1, 2, 3]**
* **[2, 3, -1, 1]**
* **[3, -1, 1, 2]**
* **[-1, 1, 2, 3]**
* **[1, -1, 2, 3]**
* **[1, 1, 2, 1]**
* **[5, 1, -1]**
* **[2, 3]**
* [2, 3]
* [2, 3]
* [5]

Първите 10 елемента на P са дадени с удебелен шрифт.

# Глава 19. Структури от данни – съпоставка и препоръки

1. Хеш-таблиците не позволяват в един ключ да съхраняваме повече от една стойност. Как може да се заобиколи това ограничение?
2. Реализирайте структура от данни, която изпълнява бързо следните две операции: добавяне на елемент и извличане на най-малкия елемент. Структурата трябва да позволява включването на повтарящи се еле­менти.
3. Текстов файл students.txt съдържа информация за студенти и техните специалност в следния формат:

|  |
| --- |
| Spas Delev | Computer Sciences  Ivan Ivanov | Software Engeneering  Gergana Mineva | Public Relations  Nikolay Kostov | Computer Sciences  Stanimira Georgieva | Public Relations  Vasil Ivanov | Software Engeneering |

Като използвате SortedDictionary<K,T> изведете на конзолата в азбучен ред специалностите и за всеки от тях изведете имената на студентите, сортирани първо по фамилия, после по първо име, както е показано:

|  |
| --- |
| Computer Sciences: Spas Delev, Nikolay Kostov  Public Relations: Stanimira Georgieva, Gergana Mineva  Software Engeneering: Ivan Ivanov, Vasil Ivanov |

1. Имплементирайте клас BiDictionary<K1,K2,T>, който позволява добавяне на тройки {key1, key2, value} и бързо търсене по ключовете key1, key2 и търсене по двата ключа. Заб.: Разрешено е добавянето на много елементи с един и същ ключ.
2. В една голяма верига супермаркети се продават милиони стоки. Всяка от тях има уникален номер (баркод), производител, наименование и цена. Каква структура от данни можем да използваме, за да можем бързо да намерим всички стоки, които струват между 5 и 10 лева?
3. Голяма компания за продажби притежава милиони статии, всяка от които има баркод, производител, заглавие и цена. Имплементирайте структура от данни, която позволява бързи заявки за статии по цена на статията в определен интервал [x…y].
4. Разписанието на дадена конгресна зала представлява списък от събития във формат [начална дата и час; крайна дата и час; наименование на събитието]. Какви структури от данни можем да ползваме, за да можем бързо да проверим дали залата е свободна в даден интервал [начална дата и час; крайна дата и час]?
5. Имплементирайте структурата от данни PriorityQueue<T>, която предоставя бърз достъп за изпълнение на следните операции: добавяне на елемент, изкарване на най-малкия елемент.
6. Представете си, че разработвате търсачка в обявите за продажба на коли на старо, която обикаля десетина сайта за обяви и събира от тях всички обяви за последните няколко години. След това търсачката позволява бързо търсене по един или няколко критерии: марка, модел, цвят, година на производство и цена. Нямате право да ползвате система за управление на бази от данни и трябва да реализирате собствено индексиране на обявите в паметта, без да пишете на твър­дия диск и без да използвате LINQ. При търсене по цена се подава минимална и максимална цена. При търсене по година на производство се задава начална и крайна година. Какви структури от данни ще ползвате, за да осигурите бързо търсене по един или няколко критерия?

# Глава 20. Принципи на обектно-ориентираното програмиране

1. Нека е дадено едно училище. В училището име класове от ученици. Всеки клас има множество от учители. Всеки учител преподава множество от предмети. Учениците имат име и уникален номер в класа. Класовете имат уникален текстов идентификатор. Учителите имат име. Предметите имат име, брой на часове и брой упражнения. Както учителите, така и студентите са хора. Вашата задача е да моделирате класовете (в контекста на ООП) заедно с техните атрибути и операции, дефинирате класовата йерархия и създайте диаграма с Visual Studio.
2. Дефинирайте клас Human със свойства "собствено име" и "фамилно име". Дефинирайте клас Student, наследяващ Human, който има свойство "оценка". Дефинирайте клас Worker, наследяващ Human, със свойства "надница" и "изработени часове". Имплементирайте и метод "изчисли надница за 1 час", който смята колко получава работникът за 1 час работа, на базата на надницата и изработените часове. Напишете съответните конструктори и методи за достъп до полетата (свойства).
3. Инициализирайте масив от 10 студента и ги сортирайте по оценка в нарастващ ред. Използвайте интерфейса System.IComparable.
4. Инициализирайте масив от 10 работника и ги сортирайте по заплата в намаляващ ред.
5. Дефинирайте клас Shape със само един метод calculateSurface() и полета width и height. Дефинирайте два нови класа за триъгълник и правоъгълник, които имплементират споменатия виртуален метод. Този метод трябва да връща площта на правоъгълника (height\*width) и триъгълника (height\*width/2). Дефинирайте клас за кръг с подхо­дящ конструктор, при когото при инициализация и двете полета (height и width) са с еднаква стойност (радиуса), и имплементирайте виртуалния метод за изчисляване на площта. Направете масив от различни фигури и сметнете площта на всичките в друг масив.
6. Имплементирайте следните обекти: куче (Dog), жаба (Frog), котка (Cat), котенце (Kitten), котарак (Tomcat). Всички те са животни (Animal). Животните се характеризират с възраст (age), име (name) и пол (gender). Всяко животно издава звук (виртуален метод на Animal).  
   Направете масив от различни животни и за всяко изписвайте на конзолата името, възрастта и звука, който издава.
7. Изтеглете си някакъв инструмент за работа с UML и негова помощ генерирайте клас диаграма на класовете от предходната задача.
8. Дадена банка предлага различни типове сметки за нейните клиенти: депозитни сметки, сметки за кредит и ипотечни сметки. Клиентите могат да бъдат физически лица или фирми. Всички сметки имат клиент, баланс и месечен лихвен процент. Депозитните сметки дават възможност да се внасят и теглят пари. Сметките за кредит и ипотечните сметки позволяват само да се внасят пари. Всички сметки могат да изчисляват стойността на лихвата си за даден период (в месеци). В общия случай това става като се умножи броят\_на\_месеците \* месечния\_лихвен\_процент. Кредитните сметки нямат лихва за първите три месеца ако са на физически лица. Ако са на фирми – нямат лихва за първите два месеца. Депозитните сметки нямат лихва ако техният баланс е положителен и по-малък от 1000. Ипотечните сметки имат ½ лихва за първите 12 месеца за фирми и нямат лихва за първите 6 месеца за физически лица. Вашата задача е да напишете обектно- ориентиран модел на банковата система чрез класове и интерфейси. Трябва да моделирате класовете, интерфейсите, базовите класове и абстрактните операции и да имплементирате съответните изчисления за лихвите.
9. Прочетете за шаблона "Abstract Factory" и го имплементирайте.

# Глава 21. Качествен програмен код

1. Вземете кода от първия пример в тази глава и го направете качествен.
2. Прегледайте собствения си код досега и вижте какви грешки допускате. Обърнете особено внимание на тях и помислете защо ги допускате. Постарайте се в бъдеще да не правите същите грешки.
3. Отворете чужд код и се опитайте само на базата на кода и доку­ментацията да разберете какво прави той. Има ли неща, които не ви стават ясни от първия път? А от втория? Какво бихте променили в този код? Как бихте го написали вие?
4. Разгледайте класове от CTS. Намирате ли примери за некачествен код?
5. Използвали ли сте (виждали ли сте) някакви код конвенции. През приз­мата на тази глава смятате ли, че са добри или лоши?
6. Дадена е квадратна матрица с големина n x n клетки. Въртящо обхождане на матрица наричаме такова обхождане, което започва от най-горната най-лява клетка на матрицата и тръгва към най-долната дясна. Когато обхождането не може да продължи в текущата посока (това може да се случи, ако е стигнат краят на матрицата или е достигната вече обходена клетка) посоката се сменя на следващата възможна по часовниковата стрелка. Осемте възможни посоки са:Description: 8-directions

Когато няма свободна празна клетка във всички възможни посоки, обхождането продължава от първата свободна клетка с възможно най-малък ред и възможно най-близко до началото на този ред. Обхождането приключва, когато няма свободна празна клетка в цялата матрица. Задачата е да се напише програма, която чете от конзолата цяло число n (1 ≤ n ≤ 100) и изписва запълнената матрица също на конзолата.

|  |  |  |  |
| --- | --- | --- | --- |
| Примерен вход:   |  | | --- | | n = 6 | | Примерен изход:   |  | | --- | | 1 16 17 18 19 20  15 2 27 28 29 21  14 31 3 26 30 22  13 36 32 4 25 23  12 35 34 33 5 24  11 10 9 8 7 6 | |

Вашата задача е да свалите от този адрес решение на горната задача:  
<http://introcsharpbook.googlecode.com/svn/trunk/book/resources/High-Quality-Code.rar> и да го преработите според концепциите за качествен код. Може да ви се наложи да оправяте и бъгове в решението.

# Глава 22. Ламбда изрази и LINQ заявки

1. Имплементирайте разширяващ метод Substring(int index, int length) за класа StringBuilder, който връща нов StringBuilder и има същата функционалност като метода Substring(…) на класа String.
2. Имплементирайте следните разширяващи методи за класовете, имплементиращи интерфейса IEnumerable<T>: Sum, Min, Max, Average.
3. Напишете клас Student със следните свойства: първо име, фамилия и възраст. Напишете метод, който по даден масив от студенти намира всички студенти, на които името им е по-малко лексикографски от фамилията. Използвайте LINQ заявка.
4. Напишете LINQ заявка, която намира първото име и фамилията на всички студенти, които са на възраст между 18 и 24 години включи­телно. Използвайте класа Student от предната задача.
5. Като използвате разширяващите методи OrderBy(…) и ThenBy(…) с ламбда израз, сортирайте списък от студенти по първо име и по фамилия в намаляващ лексикографски ред. Напишете същата функционалност, използвайки LINQ заявка.
6. Напишете програма, която отпечатва на конзолата всички числа в даден масив (или списък), които се делят едновременно на 7 и на 3. Използвайте вградените разширяващи методи с ламбда изрази и после напишете същото, само че с LINQ заявка.
7. Напишете разширяващ метод на класа String, който прави главна, всяка буква, която е начало на дума в изречение на английски език. Например текстът "this iS a Sample sentence." трябва да стане на "This Is A Sample Sentence.".

# Глава 23. Как да решаваме задачи по програмиране?

1. Използвайки описаната в тази глава методология за решаване на задачи по програмиране решете следната задача: дадени са N точки (N < 100 000) в равнината. Точките са представени с целочислени координати (xi, yi). Напишете програма, която намира всички възможни хоризонтални или вертикални прави (отново с целочислени координати), които разделят равнината на две части, така че двете части да съдържат по равен брой точки (точките попадащи върху линията не се броят).
2. Използвайки описаната в тази глава методология за решаване на задачи по програмиране решете следната задача: дадено е множество S от n цели числа и положително цяло число k (k <= n <= 10). Алтернативна редица от числа е редица, която алтернативно сменя поведението си от растяща към намаляваща и обратно след всеки неин елемент. Напишете програма, която генерира всички алтернативни редици s1, s2, …, sk състояща се от k различни елемента от S. Пример: S={ 2, 5, 3, 4 }, K=3: {2, 4, 3}, {2, 5, 3}, {2, 5, 4}, {3, 2, 4}, {3, 2, 5}, {3, 4, 2}, {3, 5, 2}, {3, 5, 4}, {4, 2, 3}, {4, 2, 5}, {4, 3, 5}, {5, 2, 3}, {5, 2, 4}, {5, 3, 4}
3. Използвайки описаната в тази глава методология за решаване на задачи по програмиране решете следната задача: разполагаме с карта на един град. Картата се състои от улици и кръстовища. За всяка улица на картата е отбелязана нейната дължината. Едно кръстовище свързва няколко улици. Задачата е да се намери и отпечата най-късият път между двойка кръстовища (измерен като суми от дължи­ните на улиците, през които се преминава).

Ето как изглежда схематично картата на един примерен град:

Towns-map-graph

На тази карта най-късият път между кръстовища A и D е с дължина 70 и е показан на фигурата с удебелени линии. Както виждате, между A и D има много пътища с най-различна дължина. Не винаги най-късото начало води към най-късия път и не винаги най-малкият брой улици води до най-къс път. Между някои двойки кръстовища дори въобще не съществува път. Това прави задачата доста интересна.

Входните данни се задават в текстов файл map.txt. Файлът започва със списък от улици и техните дължини, след което следва празен ред и след него следват двойки кръстовища, между които се търси най-кратък път. Файлът завършва с празен ред. Ето пример:

|  |
| --- |
| A B 20  A H 30  B H 5  ...  L M 5  (празен ред)  A D  H K  A E  (празен ред) |

Резултатът от изпълнението на програмата за всяка двойка кръсто­вища от списъка във входния файл трябва да е дължината на най-късия път, следвана от самия път. За картата от нашия пример изходът трябва да изглежда така:

|  |
| --- |
| 70 ABJCFD  No path!  35 ABHE |

1. \* В равнината са дадени са N точки с координати цели, положителни числа. Тези точки представляват дръвчета в една нива. Стопа­нинът на нивата иска да огради дръвчетата, като използва минимално коли­чество ограда. Напишете програма, която намира през кои точки трябва да минава оградата. Използвайте методологията за решаване на задачи по програмиране!

Ето как би могла да изглежда градината:

Points-in-the-plane

Входните данни се четат от файл garden.txt. На първия ред на файла е зададен броят на точките. Следват координатите на точките. За нашия пример входният файл би могъл да има следното съдържание:

|  |
| --- |
| 13  60 50  100 30  40 40  20 70  50 20  30 70  10 10  110 70  90 60  80 20  70 80  20 20  30 60 |

Изходните данни трябва да се отпечатат на конзолата в като последователност от точки, през които оградата трябва да мине. Ето примерен изход:

|  |
| --- |
| (10, 10) - (20, 70) - (70, 80) - (110, 70) - (100, 30) - (80, 20) - (10, 10) |

# Глава 24. Практически задачи за изпит по програмиране – тема 1

1. Даден входен файл mails.txt, който съдържа имена на потребители и техните email адреси. Всеки ред от файла изглежда така:

|  |
| --- |
| **<first name> <last name> <username>@<host>.<domain>** |

Има изискване за имейл адресите – <username> може да е последова­телност от латински букви (a-z, A-Z) и долна черна (\_), <host> е последователност от малки латински букви (a-z), а <domain> има огра­ничение от 2 до 4 малки латински букви (a-z). Да се напише програма, която намира валидните email адреси и ги записва заедно с имената на потребителите в изходен файл validMails.txt.

1. Даден е лабиринт, който се състои от N x N квадратчета, всяко от които може да е проходимо (0) или не (x).

В едно от квадратчетата се намира отново нашият герой Минчо (\*). Две квадратчета са съседни, ако имат обща стена. Минчо може на една стъпка да преминава от едно проходимо квадратче в съседно на него проходимо квадратче. Напишете програма, която по даден лабиринт отпечатва броя на възможните изходи от лабиринта.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **x** | **x** | **x** | **0** | **x** | **x** |
| **0** | **x** | **0** | **0** | **0** |  |
| **0** | **\*** | **0** | **x** | **0** | **0** |
| **x** | **x** | **x** | **x** | **0** | **x** |
| **0** | **0** | **0** | **0** | **0** | **x** |
| **0** | **x** | **0** | **x** | **x** | **0** |

Входните данни се четат от текстов файл с име Problem.in. На първия ред във файла стои числото N (2 < N < 1000). На следващите N реда стоят по N символа, всеки от които е или "**0**" или "**x**" или "**\***". Изходът представлява едно число и трябва да се изведе във файла Problem.out.

1. Даден е лабиринт, който се състои от N x N квадратчета, всяко от които може да е проходимо или не. Проходимите клетки съдържат малка латинска буква между "а" и "z", а непроходимите – '#'. В едно от квадратчетата се намира Минчо. То е означено с "\*".

Две квадратчета са съседни, ако имат обща стена. Минчо може на една стъпка да преминава от едно проходимо квадратче в съседно на него проходимо квадратче. Когато Минчо минава през проходимите квадрат­чета, той си записва буквите от всяко квадратче. На всеки изход получава дума. Напишете програма, която по даден лабиринт отпе­чатва думите, които се образуват при всички възможни изходи от лабиринта.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **a** | **#** | **#** | **k** | **m** | **#** |
| **z** | **#** | **a** | **d** | **a** | **#** |
| **a** | **\*** | **m** | **#** | **#** | **#** |
| **#** | **d** | **#** | **#** | **#** | **#** |
| **r** | **i** | **f** | **i** | **d** | **#** |
| **#** | **d** | **#** | **d** | **#** | **t** |

Входните данни се четат от текстов файл с име Problem.in. На първия ред във файла стои числото N (2 < N < 10). На следващите N реда стоят по N символа, всеки от които е или латинска буква между "а" и "z" или "**#**" или "**\***". Изходът трябва да се изведе във файла Problem.out.

1. Фирма планира създаване на система за управление на звукозаписна компания. Звукозаписната компания има име, адрес, собственик и из­пъл­нители. Всеки изпълнител има име, псевдоним и създадени албуми. Албумите се описват с име, жанр, година на издаване, брой на прода­дените копия и списък от песни. Песните, от своя страна се описват с име и времетраене. Да се проектира съвкупност от класове с връзки между тях, които моделират данните за звукозаписната компания. Да се реализира тестов клас, който демонстрира работата на всички останали класове.
2. Фирма планира създаване на система за управление на компания за недвижими имоти. Компанията има име, собственик, Булстат, служи­тели и разполага със списък от имоти за продажба. Служители се опис­ват с име, длъжност и стаж. Компанията продава няколко вида имоти – апартаменти, къщи, незастроени площи и магазини. Всички те се характеризират с площ, цена на квадратен метър и местоположение. За някои от тях има допълнителна информация. За апартамента има данни за номер на етажа, дали в блока има асансьор и дали е обзаведен. За къщите се зная квадратните метри на застроена част и на незастроената (двора), на колко етажа е и дали е обзаведена. Да се проектира съвкупност от класове с връзки между тях, които моделират данните за компа­нията. Да се реализира тестов клас, който демонстрира работата на всички останали класове.

# Глава 25. Практически задачи за изпит по програмиране – тема 2

1. Решете задачата "броене на думи в текст", използвайки само един буфер за четене (StringBuilder). Промени ли се сложността на алгоритъмът ви?
2. Реализирайте по-ефективно решение на задачата "матрица с прости числа" като търсите простите числа с "решето на Ератостен": <http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes>.
3. Добавете поддръжка на операциите умножение и целочислено деление в задачата "аритметичен израз". Имайте предвид, че те са с по-висок приоритет от събирането и изваждането!
4. Добавете поддръжка на реални числа, не само цели.
5. Добавете поддръжка на скоби в задачата "аритметичен израз".
6. Напишете програма, която валидира аритметичен израз. Например "2\*(2.25+5.25)-17/3" е валиден израз, докато "\*232\*-25+(33+а" е невалиден.

# Глава 26. Практически задачи за изпит по програмиране – тема 3

1. Напишете програма, която отпечатва спирална квадратна матрица, започвайки от числото 1 в горния десен ъгъл и движейки се по часовниковата стрелка. Примери при N=3 и N=4:

Spiral-Squares

1. Напишете програма, която брои думите в текстов файл, но за дума счита всяка последователност от символи (подниз), а не само отделените с разделители. Например в текста "Аз съм студент в София" поднизовете "с", "сту", "а" и "аз съм" се срещат съответно 3, 1, 2 и 1 пъти.
2. Моделирайте със средствата на ООП файловата система в един компютър. В нея имаме устройства, директории и файлове. Устрой­ствата са примерно твърд диск, флопи диск, CD-ROM устройство и др. Те имат име и дърво на директориите и файловете. Една директория има име, дата на последна промяна и списък от файлове и директории, които се съдържат в нея. Един файл има име, дата на създаване, дата на последна промяна и съдържание. Файлът се намира в някоя от директориите. Файлът може да е текстов или бинарен. Текстовите файлове имат за съдържание текст (string), а бинарните – поредица от байтове (byte[]). Направете клас, който тества другите класове и показва, че с тях можем да построим модел на устройствата, директо­риите и файловете в компютъра.
3. Използвайки класовете от предходната задача с търсене в Интернет напи­шете програма, която взима истинските файлове от компютъра и ги записва във вашите класове (без съдържанието на файловете, защото няма да стигне паметта).