**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение**

**высшего профессионального образования**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТ «МИФИ»**

**ОБНИНСКИЙ ИНСТИУТ АТОМНОЙ ЭНЕРГЕТИКИ**

**Е.А. Виноградова, А.Г. Царина**

**Сборник задач**

**по программированию на языках С и С++**

Учебное пособие по курсам

«Программирование»,

«Программирование на языках высокого уровня»,

«Языки программирования и методы трансляции»

*Рекомендовано к изданию*

*Редакционно-издательским советом института*

Обнинск 2013

УДК 681.3.06(075)

Виноградова Е.А. Царина А.Г. Сборник задач по программированию на языках С и С++. Учебное пособие по курсам «Программирование», «Программирование на языках высокого уровня» и «Языки программирования и методы трансляции» - Обнинск: ИАТЭ НИЯУ МИФИ, 2013. – 68 с.

Сборник включает в себя задачи по всем содержательным линиям предмета и составлен с учетом материала, разработанного авторами за время преподавания в течение нескольких лет программирования на факультетах кибернетики (направления подготовки 230100 «Информатика и вычислительная техника» и 230400 «Информационные системы и технологии») и естественных наук (по направлению подготовки 010400 «Прикладная математика и информатика») ИАТЭ НИЯУ МИФИ.

Сборник содержит рекомендации по оформлению программ и дополняет учебное пособие по данному курсу.

Предназначен для студентов первых и вторых курсов, изучающих язык программирования С.

Рецензенты: к.т.н. О.А. Мирзеабасов,

к.ф.-м.н. Г.М. Жердев

Темплан 2013, поз. 18

© ИАТЭ НИЯУ МИФИ, 2013 г.

© Е.А. Виноградова, А.Г. Царина, 2013 г.

## 1. Линейные алгоритмы

**Требования к написанию и оформлению кода программы**

Линейный алгоритм вашей задачи разделить на две части, выделив в одну часть вычислительные операции этого алгоритма, а в другую – все операции ввода-вывода. Вычислительную часть алгоритма оформить как функцию с параметрами, передаваемыми по значению, и возвращаемым значением. Другую часть алгоритма оформить как функцию main()**,** вызывающую первую функцию нужное количество раз.

**Задачи**

1. Составить программу для нахождения среднего арифметического модулей трех действительных чисел.
2. Составить программу для нахождения сопротивления параллельного соединения трех сопротивлений *R*1, *R*2, *R*3.
3. Составить программу для нахождения сторон треугольника по трем углам и радиусу описанной окружности.
4. Составить программу для нахождения площади кольца по двум радиусам: внешнему *R* и внутреннему *r*.
5. Составить программу для нахождения радиуса окружности, описанной вокруг треугольника, заданного тремя своими сторонами.
6. Составить программу для нахождения периода колебания математического маятника по его длине *L*.
7. Составить программу для нахождения времени, через которое встретятся два тела, движущиеся равноускоренно навстречу друг другу, если известны их начальные скорости, ускорения и начальное расстояние между ними.
8. Составить программу для нахождения площади равнобочной трапеции, если известны две ее стороны и угол при большем основании.
9. Составить программу для нахождения катета прямоугольного треугольника по гипотенузе и другому катету.
10. Составить программу для нахождения радиуса окружности, вписанной в треугольник, заданный тремя своими сторонами.
11. Составить программу для нахождения сторон треугольника по трем углам и радиусу вписанной окружности.
12. Составить программу для нахождения силы притяжения между двумя телами массой *m*1 и *m*2, находящимися на расстоянии *r* друг от друга.
13. Составить программу для нахождения времени падения камня на поверхность земли с высоты *H*.
14. Составить программу для нахождения площади сектора радиуса *R*, содержащего *N* радиан.
15. Составить программу для нахождения среднего геометрического модулей трех действительных чисел.
16. Составить программу для нахождения расстояния между двумя точками на плоскости с координатами *X*1, *Y*1 и *X*2, *Y*2.
17. Составить программу для нахождения периметра треугольника, заданного координатами своих вершин.
18. Составить программу для нахождения объема шара радиусом *R*.
19. Составить программу для нахождения длин медиан треугольника по трем его сторонам.
20. Составить программу для нахождения вписанной окружности прямоугольного треугольника по гипотенузе и другому катету.
21. Составить программу для нахождения температуры смеси *V*1 литров воды температуры *t*1 c *V*2 литрами воды температуры *t*2.
22. Составить программу нахождения периметра правильного *n*-угольника, описанного около окружности радиуса *R*.
23. Составить программу нахождения кинетической энергии тела, массой *m*, движущегося со скоростью *V*.
24. Составить программу нахождения величины работы *A*, выполняемой силой *F*, действующей на тело под углом α на пути, длиной *l*.
25. Составить программу нахождения площади треугольника, заданного координатами своих вершин.
26. Составить программу нахождения объема смеси *V*1 литров воды температуры *t*1 c *V*2 литрами воды температуры *t*2.
27. Составить программу нахождения силы взаимодействия в вакууме на расстоянии *R* двух параллельных прямолинейных проводников длиной *l* c током  и .
28. Составить программу нахождения конечного объема тела *V* при нагревании его от температуры  до , если начальный его объем , а относительный коэффициент объемного расширения γ.
29. Составить программу нахождения активной мощности цепи переменного тока сопротивлением *R*, через которую проходит ток силой *I*.
30. Составить программу нахождения полезной мощности, затрачиваемой на подъем транспортером материала массой *m* на высоту *h*.
31. Составить программу нахождения угла *В*, смежного углу *А*. Угол *А* программа получает заданный в градусах, а смежный ему *В* вычисляет в радианах.
32. Составить программу, которая вычисляет причитающуюся к выплате сумму через *N* месяцев, если деньги в сумме, равной *S*, положены на депозит из расчета *P*% годовых.
33. Составить программу, которая подсчитывает количество набранных спортивной командой очков на чемпионате, если задано общее количество сыгранных встреч, а также количество выигранных и проигранных ею встреч. Полагается, что за выигрыш команда получает два очка, за ничью – одно очко, за проигрыш – ноль очков.
34. Составить программу для вычисления времени *t* встречи автомобилей, движущихся равноускоренно навстречу друг другу, если известны скорости *V*1 и *V*2, ускорения *a*1 и *a*2 и начальное расстояние *S* между ними. Первый автомобиль стартовал в момент времени *t*1, а второй в момент времени *t*2.

### 2. Условный оператор

**Требования к написанию и оформлению кода программы**

Необходимо написать функцию, в которую в качестве параметров по значению передаются координаты точки, и которая возвращает значение true, если точка находится в области, и значение false – в противном случае. Саму программу оформить как функцию main()**,** вызывающую первую функцию и содержащую необходимые функции или операторы ввода/вывода.

**Задачи**

Напишите программу, которая вводит координаты точки (*x*, *y*) и определяет, попадает ли точка в заштрихованную область на рисунке. Попадание на границу области, выделенную черной жирной линией, считать попаданием в область.

|  |  |  |
| --- | --- | --- |
| **2.1** | **2.2** | **2.3** |
| 1  1  -1  -1 | 1  -1  -1  1 | 1  1  -1  -1 |
| **2.4** | **2.5** | **2.6** |
| 1  1  -1  -1 | 1  1  -1  -1 | 1  1  -1  -1 |
| **2.7** | **2.8** | **2.9** |
| 1  -1  -1  1 | 1  -1  -1  1 | 1  -1  -1  1 |
|  |  |  |
| **2.10** | **2.11** | **2.12** |
| 1  1  -1  -1 | 1  1  -1  -1 | 1  1  -1  -1 |
| **2.13** | **2.14** | **2.15** |
| 1  1  -1  -1 | 1  1  -1  -1 | 1  -1  -1  1 |
| **2.16** | **2.17** | **2.18** |
| 1  1  -1  -1 | 1  1  -1  -1 | 1  -1  -1  1 |

### 3. Ряды

**Требования к написанию и оформлению кода программы**

Необходимо написать функцию, в которую в качестве параметров по значению передаются *x* (если требуется) и ε. Функция возвращает значение суммы ряда. Все остальные вычисляемые величины должны быть указаны как ссылочные параметры функции. Саму программу оформить как функцию main()**,** вызывающую первую функцию и содержащую необходимые функции или операторы ввода/вывода. Предусмотреть неоднократный ввод значений пользователем.

*Примечание*: вычисление суммы ряда с точностью ε означает, что в сумму необходимо включить только те слагаемые, которые по модулю больше заданной точности ε.

**Задачи**

1. Вычислить с точностью ε значение суммы

.

Определить количество слагаемых, включенных в сумму.

1. Вычислить с точностью ε значение суммы и определить количество слагаемых, включенных в сумму:

.

1. Вычислить с точностью ε значение суммы и определить количество слагаемых, включенных в сумму:

.

1. Вычислить с точностью ε значение суммы и определить количество слагаемых, включенных в сумму:

.

1. Вычислить с точностью ε значение суммы и определить количество слагаемых, включенных в сумму:

.

1. Вычислить с точностью ε значение суммы и определить количество слагаемых, включенных в сумму:

.

1. Вычислить с точностью ε значение суммы и определить количество слагаемых, включенных в сумму:

.

1. Вычислить с точностью ε значение суммы и определить количество слагаемых, включенных в сумму:

.

1. Вычислить с точностью ε значение суммы и определить количество слагаемых, включенных в сумму: .
2. Вычислить с точностью ε значение суммы и определить количество слагаемых, включенных в сумму: .
3. Вычислить с точностью ε значение суммы и определить количество слагаемых, включенных в сумму: .
4. Вычислить с точностью ε значение суммы и определить количество слагаемых, включенных в сумму: .
5. Для действительных чисел составить программу вычисления суммы ряда  с точностью ε, последнего слагаемого, включенного в сумму, и его номера.
6. Для действительных чисел  составить программу вычисления суммы ряда  с точностью ε, последнего слагаемого, включенного в сумму, и его номера.
7. Для действительных чисел  составить программу вычисления суммы ряда  с точностью ε, последнего слагаемого, включенного в сумму, и его номера.
8. Для действительных чисел  составить программу вычисления суммы ряда  с точностью ε, последнего слагаемого, включенного в сумму, и его номера.
9. Для действительных чисел  составить программу вычисления суммы ряда  с точностью ε, последнего слагаемого, включенного в сумму, и его номера.
10. Для действительных чисел  составить программу вычисления суммы ряда  с точностью ε, последнего слагаемого, включенного в сумму, и его номера.
11. Для действительных чисел  составить программу вычисления суммы ряда  с точностью ε, последнего слагаемого, включенного в сумму, и его номера.
12. Для действительных чисел  составить программу вычисления суммы ряда  с точностью ε, последнего слагаемого, включенного в сумму, и его номера.
13. Для действительных чисел  составить программу вычисления суммы ряда  с точностью ε, последнего слагаемого, включенного в сумму, и его номера.
14. Для действительных чисел  составить программу вычисления с точностью ε суммы ряда

,

последнего слагаемого, включенного в сумму, и его номера.

1. Для числа *х* (|*х*|<1) cоставить программу вычисления суммы



с точностью . Найти значение последнего слагаемого и его номер.

1. Составить программу вычисления суммы бесконечного ряда для заданного *х*, учитывая, что |*х*|<1:



Найти значение последнего слагаемого и его номер.

1. Составить программу вычисления суммы бесконечного ряда для заданного *х*, учитывая, что |*х*|<1:



Найти значение последнего слагаемого и его номер.

1. Составить программу вычисления суммы бесконечного ряда для заданного х, учитывая, что |*х*| <1:



Найти значение последнего слагаемого и его номер.

1. Даны действительные числа . Составить программу вычисления произведения  с точностью  и его последнего учтенного сомножителя.
2. Даны действительные числа . Составить программу вычисления суммы  с точностью ε и определить величину последнего учтенного слагаемого.
3. Даны действительные числа . Составить программу вычисления суммы  с точностью  и определить величину последнего учтенного слагаемого и его номер.
4. Для заданной точности ε и трех вещественных чисел *x*нач*, x*кон(*x*нач*<x*кон)и *h* найти сумму ряда  при значениях *x*= *x*нач, *x*нач*+h,* *x*нач*+2h…x*кон. Составить программу таким образом, чтобы результат выводился в виде таблицы (обязательно использовать функции форматного вывода). Для указанных значений аргумента сравнить полученные результаты со значениями функции , вычисленными с помощью математической библиотеки.
5. Для заданной точности ε и трех вещественных чисел *x*нач*, x*кон(*x*нач*<x*кон)и *h* найти сумму ряда



при значениях *x*= *x*нач, *x*нач*+h,* *x*нач*+2h…x*кон. Составить программу таким образом, чтобы результат выводился в виде таблицы (обязательно использовать функции форматного вывода). Для указанных значений аргумента сравнить полученные результаты со значениями функции , вычисленными с помощью математической библиотеки.

1. Для заданной точности ε и трех вещественных чисел *x*нач*, x*кон(*x*нач*<x*кон)и *h* найти сумму ряда



при значениях *x*= *x*нач, *x*нач*+h,* *x*нач*+2h…x*кон. Составить программу таким образом, чтобы результат выводился в виде таблицы (обязательно использовать функции форматного вывода). Для указанных значений аргумента сравнить полученные результаты со значениями функции , вычисленными с помощью математической библиотеки.

1. Для заданной точности ε, целого числа *m*>1 и трех вещественных чисел *x*нач*, x*кон** (*x*нач*<x*кон) и *h* найти сумму ряда



при значениях *x*= *x*нач, *x*нач*+h,* *x*нач*+*2*h…x*кон. Составить программу таким образом, чтобы результат выводился в виде таблицы (обязательно использовать функции форматного вывода). Для указанных значений аргумента сравнить полученные результаты со значениями функции , вычисленными с помощью математической библиотеки.

1. Для заданной точности ε > 0 и трех вещественных чисел *h, x*нач*, x*кон** (*x*нач*<x*кон) найти сумму ряда



при значениях *x* = *x*нач, *x*нач*+h,* *x*нач*+*2*h…x*кон. Составить программу таким образом, чтобы результат выводился в виде таблицы (обязательно использовать функции форматного вывода). Для указанных значений аргумента сравнить полученные результаты со значениями функции , вычисленными с помощью математической библиотеки.

1. Для заданной точности ε, целого числа *m*>1 и трех вещественных чисел *x*нач*, x*кон** (*x*нач*<x*кон) и *h* найти сумму ряда



при значениях *x*=*x*нач, *x*нач*+h,* *x*нач*+*2*h…x*кон. Составить программу таким образом, чтобы результат выводился в виде таблицы (обязательно использовать функции форматного вывода). Для указанных значений аргумента сравнить полученные результаты со значениями функции , вычисленными с помощью математической библиотеки.

1. Дано действительное число *x*. Написать программу, определяющую, с какой точностью будет вычислена сумма ряда

,

если известно количество слагаемых *N*, входящих в сумму.

1. Дано действительное число *x*. Написать программу, определяющую, с какой точностью будет вычислена сумма ряда

,

если известно количество слагаемых *N*, входящих в сумму.

1. Дано действительное число *x*. Написать программу, определяющую, с какой точностью будет вычислена сумма ряда

,

если известно количество слагаемых *N*, входящих в сумму.

### 4. Рекуррентные последовательности

**Требования к написанию и оформлению кода программы**

Рекуррентный алгоритм задает способ вычисления членов последовательности описывающей? функцию при помощи рекуррентных формул. Следующий член последовательности вычисляют как функцию от предыдущего:

, где .

Возможен более сложный случай, когда очередной член последовательности зависит от двух (и более) предыдущих значений элементов последовательности:

 , где ,

 , где  и так далее.

Для каждой задачи алгоритм функции вычисления требуемого значения разработайте в трех вариантах с использованием цикла

* while**;**
* do....while;
* for.

Функция должна иметь возвращаемое значение и список необходимых параметров.

*Примечание:* последовательность Фибоначчи определяется следующим образом:  и .

**Задачи**

1. Дано натуральное число *N*. Вычислить значение натурального логарифма:

.

1. Дано натуральное число *N*. Вычислить значение следующего выражения:

.

1. Дано натуральное число *N*. Вычислить значение синуса:

.

1. Дано натуральное число *N*. Вычислить значение косинуса:

.

1. Вычислить значение натурального логарифма:

,

где *N* –заданное натуральное число.

1. Вычислить значение следующего выражения:

*Y*=.

1. Дано натуральное число *n* и вещественное число *a*. Составить программу для вычисления:

.

1. Составить программу вычисления *N* скобок:

.

1. Составить программу вычисления суммы (последнее слагаемое имеет глубину вычисления синуса, равную *n*): .
2. Составить программу вычисления  ( ), если  и  для *k*=2,3,… Вещественные числа *b*, *c*, *d*, *r*, *q* вводятся пользователем.
3. Составить программу вычисления выражения для *x* ≠0:



1. Даны действительные числа . Последовательность чисел  образована по закону  Составить программу вычисления , где *k* – наименьшее целое число, удовлетворяющее двум условиям: *k*> *N* и .
2. Составить программу вычисления  (*n* ≥ 4), если , и , *i*=4,5,…
3. Дано . Элементы  вычисляются по формуле



1. Составить программу нахождения первого члена последовательности чисел  для которого выполнено условие 
2. Составить программу вычисления выражения, учитывая, что *ai*=2*i*–1 при *i*=1,2,3…*n*:

.

1. Составить программу вычисления  (*n*≥3), если  и



1. Заданы действительные числа *k*, *b*, *c*, *d*, натуральное число *n* (*n*≥2). Пусть  и  для *k*=2,3…*n*. Составить программу вычисления .
2. Дано натуральное число *N*. Составить программу, определяющую процент четных элементов в последовательности Фибоначчи, состоящей из *N* членов.
3. Дано натуральное число *N* и целое число *M*. Составить программу, выводящую на экран только те элементы последовательности Фибоначчи до *n*-ого, которые не кратны *M*.
4. Дано натуральное число *n*. Составить программу, выводящую на экран цифры *n*-ого числа Фибоначчи.
5. Даны два натуральных числа *n* и *m*. Составить программу, вычисляющую сумму *n*-ого и *m*-ого чисел Фибоначчи.
6. Дано целое число *M* (*M*>100). Составить программу, вычисляющую количество элементов в последовательности Фибоначчи, не превосходящих *M*.
7. Дано натуральное число *n*. Составить программу, выводящую на экран все элементы последовательности Фибоначчи до *n*-ого, являющиеся простыми числами.
8. Дано целое число *c* (0≤*c*≤9) и натуральное число *n*. Составить программу, определяющую, сколько раз цифра *c* встречается в записи первых *n* чисел Фибоначчи.

### 5. Целые числа

**Требования к написанию и оформлению кода программы**

Алгоритм решения задачи должен быть разделен, как минимум, на две функции:

* функцию, которая организует вычисления, определенные в задаче вашего варианта; при необходимости алгоритм решения этой задачи разделите на функции, имеющие более простой алгоритм и решающие отдельные подзадачи общей задачи вашего варианта;
* функцию main(), которая содержит операции ввода исходных данных, обращается к функции вычисления и выводит результат на экран.

В функции main() предусмотрите возможность многократного решения задачи с различными исходными данными.

**Задачи**

1. В значении целой переменной произвольной разрядности определить количество цифр, совпадающих со старшей цифрой числа.
2. Определить, состоит ли значение целой переменной из одинаковых цифр.
3. Проверить, является ли целое число *Х* палиндромом.
4. Построить число - инверсию заданной целой переменной .
5. Определить, составляют ли цифры числа упорядоченную по возрастанию последовательность.
6. Дано вещественное число. Вычислить суммы цифр его целой и дробной части.
7. Вводятся последовательно цифры. Количество вводимых чисел заранее не известно. Из этих цифр составить число таким образом, чтобы цифры числа были расположены в порядке, обратном вводу. Предусмотреть проверку правильности ввода цифровой информации.
8. Вводятся последовательно цифры. Количество вводимых чисел заранее не известно. Из этих цифр составить число таким образом, чтобы цифры числа были расположены в порядке ввода. Предусмотреть проверку правильности ввода информации
9. Для целого числа *N*, вводимого с клавиатуры, переставить первую и последнюю цифры и выдать его на печать.
10. Дана целая переменная произвольной разрядности. Необходимо вывести на экран цифры этого числа в столбец, начиная со старшего разряда, если число отрицательное, в противном случае – начиная с младшего разряда. Кроме того, вывести на экран сообщение: четная или нет сумма цифр данного числа.
11. В заданном диапазоне натуральных шестизначных чисел (числа – номера билетов) найти и вывести «счастливые билеты».
12. В числовые переменные вводятся два натуральных числа *Х* и *N*. Необходимо найти ближайшее к *Х* натуральное число, кратное *N* (если таких чисел два – найти наименьшее из них).
13. Проверить, являются ли дружественными два заданных натуральных числа. Два числа называют дружественными, если каждое из них равно сумме всех делителей другого, кроме самого этого числа.
14. Натуральное число из *N* цифр является числом Армстронга, если сумма его цифр, возведенных в *N*-ю степень, равна самому числу (как, например, 153=13+53+33). Получить все числа Армстронга из трех и четырех цифр.
15. Дано целое число *X* произвольной разрядности. Определить, сколько раз в числе встретилась каждая десятичная цифра.
16. В числовую переменную последовательно вводятся целые числа, не равные нулю. Количество вводимых чисел заранее не известно. Требуется найти сумму тех введенных чисел, в которых встречается *k* раз цифра *N* (*k* и саму цифру *N* также вводят с клавиатуры).
17. Напечатать первые *n* натуральных чисел, которые при удалении последней цифры уменьшаются в целое число раз. Вывести на экран эти числа с указанием, во сколько раз они уменьшаются.
18. Вывести на экран все «автоморфные» целые числа из интервала от *m* до *n*. Автоморфным называется число, которое содержится в младших разрядах квадрата этого числа. Примеры автоморфных чисел: 52 =25 и 252=625.
19. Найти все натуральные *n*-значные числа, не превосходящие заданного числа *m* , которые делятся на каждую из своих цифр. Если таких чисел нет, выдать соответствующее сообщение. Произвести проверку вводимой информации.
20. Напечатать *m* натуральных *n*-значных чисел, у каждого из которых все цифры являются разными. Вычислить и напечатать сумму этих чисел.
21. Среди всех натуральных *n*-значных чисел выбрать и вывести на экран числа-палиндромы. Палиндром – число, которое имеет одинаковое чтение слева направо и справа налево.
22. Ввести с клавиатуры границы диапазона *n* и *m* натуральных чисел (*n*<*m*). Из чисел, входящих в этот диапазон, необходимо напечатать только те, цифры которых являются соседними в натуральном ряду. Подсчитать количество этих чисел и определить, сколько среди них четных и нечетных. Предусмотреть проверку правильности ввода информации.
23. Ввести число, содержащее нечетное количество цифр. Из введенного числа сформировать число удалением средней цифры и вывести результат. Предусмотреть проверку правильности ввода чисел с нечетным количеством цифр.
24. Ввести число, содержащее нечетное количество цифр. Из введенного числа сформировать число удалением всех его средних цифр, кроме первой и последней, и вывести результат. Предусмотреть проверку правильности ввода чисел с нечетным количеством цифр.
25. Среди последовательности вводимых натуральных чисел определить количество чисел, содержащих цифру *N*. Количество вводимых чисел заранее не известно. Предусмотреть проверку правильности ввода информации.
26. Дана последовательность из *N* натуральных чисел. Определить, сколько раз в числах этой последовательности встречалась каждая из цифр. Числа последовательно вводятся в одну и ту же переменную.
27. Дана последовательность натуральных чисел. Найти наибольшую цифру в каждом члене последовательности. Числа последовательно вводятся в одну и ту же переменную.
28. Для всех *N*-значных чисел вывести те, цифры которых составляют арифметическую прогрессию. Подсчитать количество этих чисел.
29. Для каждого числа *m*, вводимого с клавиатуры, добавить цифру *N* (*N* = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9) к первой цифре этого числа. Например, добавив 2 к первой цифре числа 49 получим 69.
30. Вывести на экран все «совершенные» целые числа из интервала от *m* до *n*. Совершенным называется натуральное число, равное сумме всех своих делителей (исключая само число). Пример совершенного числа 28= 1 + 2 + 4 + 7 + 14.
31. В интервале от *a* до *b* найти все сверхпростые числа. Сверхпростым называется число, если оно простое, и число, полученное из данного посредством записи цифр данного числа в обратном порядке, – тоже простое, например, 13 и 31 – сверхпростые числа.
32. Дано натуральное число *N*. Получить все пифагоровы тройки натуральных чисел, каждое из которых не превосходит *N*, т.е. все такие тройки натуральных чисел *a*, *b*, *c*, для которых *a*2+*b*2=*c*2 (*a* ≤*b* ≤ *c* ≤ *N*).
33. Найти наибольший общий делитель двух заданных чисел.
34. В числовую переменную поочередно вводятся положительные числа, отличные от нуля. Количество чисел заранее не известно. Определить количество чисел, в которых нет двух одинаковых цифр.
35. В числовую переменную поочередно вводятся числа, отличные от нуля. Количество чисел заранее не известно. Среди элементов, расположенных после последнего отрицательного, определить наибольшее.
36. В числовую переменную поочередно вводятся числа, отличные от нуля. Количество чисел заранее не известно. Среди элементов, расположенных до последнего отрицательного, определить наименьшее.
37. В числовую переменную поочередно вводятся числа, отличные от нуля. Количество чисел заранее не известно. Среди элементов, расположенных после первого отрицательного, определить количество четных.
38. В числовую переменную поочередно вводятся числа, отличные от нуля. Количество чисел заранее не известно. Определить, составляют ли введенные числа упорядоченную по возрастанию последовательность.
39. Ввести числа, отличные от нуля, количество которых заранее не известно. Найти первое максимальное и последнее минимальное из этих чисел и количество чисел между ними в введенной последовательности.

### 6. Системы счисления

**Требования к написанию и оформлению кода программы**

Алгоритм решения задачи должен быть разделен, как минимум, на две функции:

* функцию, которая организует перевод чисел из одной системы в другую;
* функцию main(), которая содержит операции ввода исходных данных, обращается к функции вычисления и выводит результат на экран.

В функции main() предусмотрите возможность многократного решения задачи с различными исходными данными.

**Задачи**

1. Дано целое десятичное число. Проверить, преобладают ли единицы в двоичном представлении этого числа.
2. Среди всех *n*-значных (*n* задается с клавиатуры) целых десятичных чисел найти числа, запись которых в шестнадцатеричной системе счисления представляет собой палиндром, и подсчитать количество таких чисел. Палиндром – число, которое имеет одинаковое чтение слева направо и справа налево.
3. Среди простых чисел, не превосходящих заданного натурального числа *N*, найти такое, запись которого в двоичной системе счисления содержит максимальное число единиц. Если таких чисел несколько, найти минимальное из них.
4. Найти все такие простые числа, не превосходящие заданного натурального числа *N*, запись которых в двоичной системе счисления представляет чередующуюся последовательность единиц и нулей, начинающуюся с единицы.
5. Найти все простые числа, не превосходящие натурального числа *N*, двоичная запись которых представляет собой палиндром. *N* задано, и не должно превосходить 1000. Палиндром – число, которое имеет одинаковое чтение слева направо и справа налево.
6. В числовую переменную вводится отличное от нуля вещественное десятичное число, меньшее нуля. Необходимо вывести на экран представление этого числа в двоичной системе счисления. Перевод дробной части числа ограничить пятью знаками десятичного числа. Предусмотреть проверку правильности ввода информации.
7. В числовую переменную вводится отличное от нуля вещественное десятичное число. Необходимо вывести на экран представление этого числа в восьмеричной системе счисления. Перевод дробной части числа ограничить семью знаками. Предусмотреть проверку правильности ввода информации
8. Среди всех *n*-значных целых чисел найти числа, запись которых в восьмеричной системе счисления представляет собой палиндром, и подсчитать количество таких чисел. Палиндром – число, которое имеет одинаковое чтение слева направо и справа налево.
9. Последовательно вводятся двоичные цифры числа. Вывести это число в восьмеричном формате.
10. Последовательно вводятся двоичные цифры числа. Вывести это число в шестнадцатеричном формате.
11. Ввести шестнадцатеричное число. Получить его значение в двоичной системе счисления.
12. Ввести восьмеричное число. Получить его значение в двоичной системе счисления.
13. Ввести восьмеричное число. Получить его значение в шестнадцатеричной системе счисления.
14. Ввести два десятичных числа. Получить их значения в двоичной системе счисления и определить, в каком из полученных чисел наибольшая цепочка подряд идущих нулей.
15. Ввести два десятичных числа. Получить их значения в троичной системе счисления и определить, являются ли полученные числа зеркальным отображением друг друга.
16. Вычислить сумму двух чисел. Числа представлены в системе с основанием *p*.
17. Вычислить разность двух чисел. Числа представлены в системе с основанием *p*.

### 7. Массивы

**Требования к написанию и оформлению кода программы**

Основная функция main() должна реализовывать последовательный вызов трех функций:

* ввод исходных данных (чтение *n* элементов массива со стандартного устройства ввода или из файла; заполнение элементов массива псевдослучайными числами);
* функция, обеспечивающая непосредственное решение поставленной задачи; при необходимости задачу вашего варианта разбейте на функции, каждая из которых может реализовать отдельную независимую процедуру, и из которых можно составить задачу обработки массивов *как одну функцию*;
* функция вывода *n* элементов массива на экран или в файл;
* вывод результата.

Необходимо определить наиболее оптимальную систему передачи информации из одной функции в другую.

В функции main() предусмотрите возможность многократного решения задачи с различными исходными данными.

Программу разработайте в двух вариантах, используя

* + *статическое распределение памяти* для массивов (предусмотрите некоторое максимально возможное количество обрабатываемых элементов в массиве);
  + *динамическое распределение памяти* для массивов.

Особое внимание при написании программы уделить экономии памяти. Дублирующих и вспомогательных массивов, по возможности, стараться избегать.

**Задачи**

1. Значения минимального и максимального элементов могут встречаться в массиве неоднократно. Определить, каких элементов больше, равных максимальному или минимальному элементу.
2. Среди элементов массива, расположенных после наибольшего, определить сумму и количество простых чисел.
3. Определить сумму элементов, расположенных между первым максимальным и последним минимальным элементами массива.
4. Дан массив целых чисел. Найти наибольшее среди чисел, кратных заданному числу *k*, и напечатать номера всех его вхождений в массив.
5. Дан массив целых чисел. Найти наименьшее среди чисел, начинающихся с единицы, и напечатать номера всех таких чисел.
6. Назовем элемент массива особым, если он больше суммы элементов, расположенных до него. Определить количество особых элементов массива.
7. Назовем элемент массива особым, если он больше суммы элементов, расположенных после него. Определить количество особых элементов массива.
8. Найти третий по величине элемент одномерного массива.
9. Определить, сколько нулей стоит после второго по величине элемента массива.
10. Даны целые числа *а*1...*аn*. Пусть *m* и *М –* наибольшее и наименьшее из них. Сформировать массив из целых чисел от *m* до *М*, не входящих в исходную последовательность *а*1...*аn*. Числа в новом массиве расположить в порядке убывания.
11. Определить, сколько элементов массива располагается между самым большим и самым маленьким элементами, являющимися простыми числами.
12. Проверить, является ли заданный одномерный массив вещественных чисел упорядоченным по возрастанию.
13. Проверить, является ли заданный одномерный массив вещественных чисел арифметической или геометрической прогрессией.
14. Проверить является ли заданный одномерный массив вещественных чисел симметричным относительно середины.
15. Проверить, состоит ли заданный одномерный массив целых чисел только из простых чисел.
16. Инвертировать заданный одномерный массив целых чисел, т.е. поменять порядок следования его элементов на обратный.
17. Дан массив вещественных чисел *а*1...*аn*. Требуется получить новый массив *b*1...*bn*, где каждое *bi* – среднее арифметическое всех членов последовательности *а*1...*аn* кроме *аi* (*i*=1,2, …, *n*).
18. Объединить два заданных массива **A** и **B** одинакового размера в один. Порядок следования: первый элемент массива **A**, первый элемент массива **B**, второй элемент массива **A**, второй элемент массива **B** и т.д.
19. Дан массив четного размера. Разделить его элементы на две последовательности, записав в первую все элементы с четными индексами, а во вторую – с нечетными.
20. Определить сумму и количество четных элементов массива, предшествующих первому отрицательному в массиве.
21. Вычислить сумму элементов массива, расположенных между первым и последним отрицательными элементами. Предусмотреть все варианты ответов.
22. Выписать в отдельный массив элементы, расположенные между первым и последним нулевыми элементами исходного массива. Если выполнить такую задачу невозможно, вывести соответствующее сообщение.
23. Инвертировать фрагмент исходного массива, расположенный между первым и вторым отрицательными числами, если такие есть.
24. Даны два упорядоченных по возрастанию массива. Объединить их в один массив, упорядоченный также по возрастанию.
25. Дан массив **А**, упорядоченный по возрастанию, и массив **В**, упорядоченный по убыванию. Объединить их в один массив, упорядоченный по убыванию.
26. Вычислить сумму всех элементов одномерного массива, являющихся палиндромами. Если таких элементов нет, вывести соответствующее сообщение.
27. В заданном одномерном массиве вещественных чисел посчитать сумму элементов, в записи целой части которых присутствует цифра *k*.
28. Дано целое число *k* и массив целых чисел. Выполнить циклический сдвиг элементов массива на *k* позиций вправо (влево). Предусмотреть диалоговый режим для организации направления сдвига.
29. Выполнить циклический сдвиг исходного массива влево, так чтобы наибольший элемент стал первым в массиве.
30. Массив разбивается первым наибольшим и первым наименьшим элементами на три части. В центральной части осуществить циклический сдвиг на один элемент влево.
31. Исключить из массива первый наибольший элемент.
32. Первый отрицательный элемент массива сдвинуть в начало массива.
33. Последний четный элемент массива сдвинуть в конец массива.
34. Переставить элементы массива так, чтобы вначале расположились отрицательные элементы, затем нулевые, далее положительные. Взаимный порядок следования отрицательных элементов не меняется (аналогично для положительных).
35. Исключить первую подпоследовательность (два или более расположенных подряд) отрицательных элементов.
36. Вставить копию первого элемента массива после наибольшего.
37. Перенести в начало массива наименьший элемент, а в конец массива наибольший, оставив остальные элементы в исходном взаимном порядке.
38. Из массива **А** исключить все отрицательные числа (каждое неотрицательное число сдвигается один раз).
39. Дан массив целых чисел, каждое из которых отлично от нуля. Если в последовательности отрицательные и положительные члены чередуются (+,-,+,-,+,-... или -,+,-,+,-,+,...), то в новый массив переписать исходную последовательность. Иначе получить новый массив, включив в него все отрицательные члены последовательности, задав обратный порядок их следования.
40. Определить самую длинную подпоследовательность, упорядоченную по возрастанию элементов.
41. Поменять местами первую и последнюю подпоследовательности отрицательных чисел.
42. Массив состоит из 0 и 1. Определить самую длинную подпоследовательность из одинаковых чисел, а также сообщить, какие это числа.
43. Среди элементов, расположенных после последней подпоследовательности отрицательных чисел, определить наибольшее.
44. Самую длинную серию отрицательных чисел заданного массива **А** переместить в начало, а самую короткую в конец массива.
45. Даны целые числа *m* и *n* и массив действительных чисел **X**[*n*]. Найти целое число *i* (1 ≤ *i* ≤ *n*– *m*), для которого сумма *x*[*i*]+ ... +*x*[*i*+*m*] ближе всего к нулю.
46. Задан целочисленный массив **A**[*n*]. Найти отрезок массива максимальной длины, в котором первое число равно последнему, второе – предпоследнему и т.д. Напечатать длину этого отрезка.
47. В новый массив выписать неповторяющиеся элементы массива.
48. Определить моду массива, т.е. число, которое встречается в массиве чаще остальных.
49. Напечатать все различные элементы массива, а также указать, сколько раз каждый из них повторяется в массиве.
50. Напечатать общие элементы двух заданных массивов.
51. Из заданного массива удалить повторяющиеся элементы. Массив сжать.
52. Проверить, состоит ли массив только из различных чисел?
53. Из заданного массива удалить элементы, являющиеся простыми числами. Массив сжать.
54. Напечатать (по одному разу) те элементы массива **A**, которые не входят в массив **B**.
55. Определить, сколько элементов массива являются числами Фибоначчи.
56. Два числа являются дружественными, если каждое из них равно сумме делителей другого. Определить количество дружественных чисел в массиве.
57. Массив **A**[*n*] содержит целые положительные числа произвольной разрядности. Напечатать числа, у которых цифры составляют возрастающую последовательность; определить количество таких чисел и проверить, преобладают ли они в массиве.
58. Напечатать все совершенные числа массива. Совершенное число равно сумме своих делителей, включая единицу, например, 28=1+2+4+7+14.
59. Проверить, есть ли в массиве парные простые числа. Парными простыми числами называют два простых числа, разность между которыми равна 2, например, 3 и 5, 11 и 13, 17 и 19.
60. Элементы массива упорядочены по возрастанию. Элемент, введенный с клавиатуры, вставить в массив так, чтобы не нарушить его упорядоченность.
61. Дано два массива разных размеров. Проверить, входит ли более короткий массив в более длинный.
62. Составить функцию, которая, получив два массива, определяет, равны ли они. Используя эту функцию, решить следующую задачу: если три массива равны между собой, создать четвертый массив, все элементы которого равны утроенным элементам исходных, в противном случае создать массив, склеив все три исходных массива в порядке 1, 2 и 3.
63. Составить функцию, которая для упорядоченного массива чисел, определяет с помощью метода «бинарного поиска», является ли некоторое заданное число элементом этого массива. Используя эту функцию, решить следующую задачу: для упорядоченного массива и заданного числа создать новый массив путем добавления в него этого числа в начало массива, если это число не является его элементом. Если массив не подвергнут изменению, выдать об этом соответствующее сообщение.
64. Даны два целочисленных массива. Определить их симметрическую разность, т.е. такой массив, в который включены те элементы исходных массивов, которые входят в первый, но не входят во второй, и наоборот, входят во второй, но не входят в первый.
65. Составить функцию, которая определяет, является ли заданное натуральное число простым. Используя эту функцию решить следующую задачу. Задан массив из *m* натуральных чисел. Сформировать из него новый массив, поместив в него из исходного массива только простые числа.
66. В одну и ту же переменную последовательно вводятся *m* действительных чисел, среди которых могут встретиться одинаковые числа. Сформировать массив, в который поместить только разные числа. Запомнить их порядковые номера при вводе. Новый массив вывести на экран с их старыми номерами.
67. Дана исходная последовательность натуральных чисел. Проанализировать каждое число исходной последовательности и в новую последовательность включить только те числа, которые имеют четное число разрядов и которые в новой последовательности записать в порядке, обратном их следованию в исходной последовательности.
68. В массив заданной размерности *n* ввести произвольные числа. Создать новый массив, поменяв местами столько равноудаленных от *k*-го элемента элементов массива, сколько возможно (*k* задано). Напечатать модифицированный массив.
69. В одномерный массив размерности *N* ввести произвольные числа. Ввести число *k*. Если в исходном массиве чисел больше чем *k*, то выполнить кольцевой сдвиг массива так, чтобы его первый максимальный элемент оказался на *k*-ом месте. Кольцевой сдвиг массива выполняется всеми его элементами так, что элементы «выталкиваемые» из массива в одну сторону, занимают освободившееся место с другой стороны. Если *k* больше или равно числу элементов в исходном массиве, то создать новый массив, добавив в начало исходного массива число *k*. Напечатать модифицированный массив.
70. В массив заданной размерности *n* ввести произвольные числа. Не изменяя состояния исходного массива создать новый массив, в который поместить номера элементов исходного массива, соответствующие порядку убывания значений его элементов.
71. Дан массив чисел *a***1**, *a*2, …*a*2*N*. Получить из него новый массив **B** по правилу: *b*1=*a*1+*aN*+1, *b*2=*a*2+*aN*+2 и т.д. Отсортировать полученный массив **B** по убыванию.
72. Задан целочисленный массив **A**[*n*] ненулевых чисел. Отсортировать отрицательные числа по возрастанию, положительные по убыванию, оставив отрицательные на местах, принадлежащих отрицательным, а положительные – на местах, принадлежащих положительным.
73. Дан массив *n* действительных чисел. Если исходный массив имеет равное количество положительных и отрицательных чисел, создать новый массив, упорядочив исходный массив следующим образом: первыми идут два максимальных элемента, затем два минимальных; далее два максимальных и два минимальных из оставшихся и т.д. Если количество положительных и отрицательных элементов различно, то выдать сообщение о том, что новый массив не будет создан.
74. Дано: список абитуриентов, их средний бал при поступлении. Известно, что зачислить можно *n* человек. Напечатать проходной балл, а также список зачисленных в порядке убывания среднего балла.
75. Дано: список политиков, их рейтинг (количество полученных голосов) в прошлом году, рейтинг текущего года. Список упорядочен по рейтингу прошлого года. Напечатать список политиков в порядке убывания рейтинга этого года, а также список политиков повысивших свой рейтинг.
76. В массиве **А** выполнить упорядочение по возрастанию самой длинной серии отрицательных чисел.
77. Даны два массива – **А** и **В**. Переставить элементы первого так, чтобы его максимальный элемент находился на месте расположения максимального из второго, а каждый очередной по убыванию элемент из первого располагался на месте, соответствующем расположению очередного по убыванию из второго. Вывести преобразованный массив
78. В массив заданной размерности *n* ввести произвольные числа. Ввести число *k* (*k* < *n*). В исходном массиве изменить порядок следования элементов в нем на обратный порядок отдельно до и отдельно после *k*-го элемента массива.
79. Дан массив действительных чисел, среди которых заведомо есть как отрицательные числа, так и неотрицательные. Получить *а*1·*x*1 +...+*аs*·*xs*, где *а*1...*аp* – отрицательные члены исходной последовательности, взятые в порядке их следования, *x*1...*xq* – неотрицательные члены, взятые в обратном порядке, *s*=min(*p*,*q*).
80. Пусть дан массив *а*1...*аn*. Создать новый массив, переставив *а*1...*аn*так, чтобы вначале в массиве шла группа элементов, больших того элемента, который в исходном массиве располагается на первом месте, а затем – сам этот элемент, потом – группа элементов, меньших или равных ему.
81. Даны действительные числа *а*1...*аn*, *b*1...*bm*, (*а*1≤*а*2≤...≤*аn*). Получить натуральные числа *k*1...*km*, такие, что *ki*– это решение задачи поиска места *bi* среди *а*1...*аn*, (*i*=1,...,*m*). Применить алгоритм деления пополам. В функции ввода последовательности чисел предусмотреть контроль ввода в порядке возрастания их значений.
82. Задан массив действительных чисел. Если массив не упорядочен в порядке возрастания, то упорядочить его, используя сортировку «простой выбор», в противном случае создать новый массив, удалив из исходного массива все числа, стоящие на четном месте.
83. Задан массив действительных чисел. Фрагмент массива, расположенный между первым и последним отрицательными числами, упорядочить по возрастанию, используя метод «быстрая сортировка».
84. Фрагмент массива, расположенный после наибольшего элемента, упорядочить по возрастанию, используя метод «простая вставка».
85. Фрагмент массива, расположенный между наибольшим и наименьшим элементами, упорядочить по возрастанию, используя «шейкерную сортировку».
86. Задан массив действительных чисел. Если элементы массива образуют знакопеременный ряд, то упорядочить его, используя «пирамидальную сортировку», в противном случае упорядочить, используя «сортировку Шелла».

### 8. Матрицы

**Требования к написанию и оформлению кода программы**

Для решения задачи разработайте следующие функции:

* функцию ввода *N*×*M* элементов числовой матрицы, используя ранее разработанную вами функцию ввода числового вектора**;**
* функцию обработки числовой матрицы по правилу, определяемому в задаче вашего варианта;
* функцию вывода числовой матрицы на экран в общепринятом виде;
* функцию main(), которая организует вызов разработанных вами функций для решения поставленной задачи; реализуйте в ней возможность многократного решения задачи с различными исходными данными.

Программу разработайте в двух вариантах:

* используя статическое распределение памяти для хранения исходной числовой матрицы;
* используя динамическое распределение памяти для хранения исходной числовой матрицы (при этом размеры матрицы вводите с клавиатуры или из файла).

В каждой из программ продемонстрируйте умение использовать доступ к элементам вектора указателей на вектор с помощью как индексов, так и указателей (часть функций сделайте с индексами, часть – с указателями). Предусмотрите вывод на экран числовых матриц, как вновь введенных, так и сформированных, с целью наилучшего контроля полученных результатов.

**Задачи**

1. Дана матрица размером *n*×*m*. Найти наибольшее среди всех элементов тех столбцов матрицы, сумма элементов в которых отрицательна.
2. Дана матрица размером *n*×*m*. Найти наименьшее среди всех элементов тех строк матрицы, которые начинаются с отрицательного элемента.
3. Дана матрица размером *n*×*n*. В каждом столбце поменять местами элемент, стоящий на пересечении с главной диагональю, и первый наибольший.
4. Дана матрица размером *n*×*n*. В каждой строке матрицы поменять местами наименьший элемент и элемент, стоящий на пересечении с побочной диагональю.
5. Дана матрица размером *n*×*m*. Назовем особым элемент текущей строки, если он больше суммы элементов ему предшествующих в строке. Определить строку с наибольшим количеством особых элементов.
6. Дана матрица размером *n*×*m*. Назовем особым элемент текущего столбца, если он равен сумме остальных элементов этого столбца. Определить столбец с наибольшим количеством особых элементов.
7. Дана матрица размером *n*×*m*. Назовем особым элемент текущей строки, если он больше всех элементов ему предшествующих в строке. Определить номера строк, содержащих не менее трех особых элементов.
8. Дана целочисленная матрица размером *n*×*m*. Назовем особой строку, если все ее элементы кратны пяти. Определить количество особых строк.
9. Дана матрица размером *n*×*m*. Назовем особым столбец, если его элементы образуют упорядоченную (по возрастанию или убыванию) последовательность. Определить количество особых столбцов.
10. Дана целочисленная матрица размером *n*×*m*. Определить в ней количество элементов, являющихся простыми числами.
11. Дана матрица размером *n*×*n*. Среди элементов, расположенных выше главной диагонали, определить количество таких, которые больше любого элемента ниже главной диагонали.
12. Дана матрица размером *n*×*n*. Среди элементов, расположенных ниже побочной диагонали, определить количество таких, которые меньше любого элемента выше побочной диагонали.
13. Дана целочисленная матрица размером *n*×*m*. Поменять местами строки с наибольшим и наименьшим количеством четных элементов.
14. Найти строки с наибольшей и наименьшей суммой и строки с наибольшим и наименьшим произведением элементов и сформировать из них результирующую матрицу.
15. Дана матрица размером *n*×*m*. Назовем матрицу особой, если суммы элементов строк образуют возрастающую последовательность. Определить, является ли заданная матрица особой.
16. Дана матрица размером *n*×*m*. Назовем характеристикой столбца сумму его положительных элементов. Проверить, образуют ли характеристики столбцов строго убывающую последовательность.
17. Дана матрица размером *n*×*m*. Исключить из нее первую упорядоченную по возрастанию элементов строку.
18. В матрице размером *n*×*m* удалить столбец и строку, на пересечении которых находится наибольший по модулю элемент.
19. Дана матрица размером *n*×*m*. Инвертировать каждую строку матрицы.
20. Дана матрица размером *n*×*m*. Инвертировать каждый столбец матрицы.
21. Дана матрица размером *n*×*m*. Каждый из элементов матрицы заменить суммой элементов, расположенных ниже и правее его.
22. Дана матрица размером *n*×*m*. Каждый из элементов матрицы заменить суммой элементов, расположенных выше и левее.
23. Дана матрица размером *n*×*n*. Найти такие *k*, что *k*-я строка матрицы совпадает с ее *k*-ым столбцом.
24. Дана матрица размером *n*×*n*. Определить, является ли эта матрица магическим квадратом, т.е. такой, в которой суммы элементов во всех строках и столбцах одинаковы.
25. Дана квадратная числовая матрица. Переставить столбцы таким образом, чтобы суммы элементов столбцов образовали возрастающую последовательность.
26. Ввести числовую квадратную матрицу размером *n*×*n*. Перестановкой строк и столбцов упорядочить по возрастанию элементы главной диагонали матрицы.
27. Дана целочисленная матрица **A** размером *n*×*m*. Назовем элемент *aij* особым, если сумма элементов в *i*-строке равна нулю, а сумма элементов в *j*-столбце меньше нуля. Найти количество особых элементов в матрице.
28. Дана целочисленная матрица **A** размером *n*×*m* и целочисленный массив **B** размером *k*. Определить количество элементов матрицы, совпадающих хотя бы с одним из элементов массива **B**.
29. Даны две квадратные матрицы размером *n*×*n*. Проверить, совпадают ли они.
30. Дана вещественная матрица размером *n*×*n*, все элементы которой различны. Транспонировать элементы матрицы относительно главной диагонали (т.е. поменять местами элементы, расположенные симметрично относительно этой диагонали).
31. Дана вещественная матрица размером *n*×*n*, все элементы которой различны. Транспонировать элементы матрицы относительно побочной диагонали (т.е. поменять местами элементы, расположенные симметрично относительно этой диагонали).
32. Дана целочисленная матрица размером *n*×*n*. Проверить, является ли она симметричной относительно главной диагонали.
33. Дана целочисленная матрица размером *n*×*n*. Проверить, является ли она симметричной относительно побочной диагонали.
34. Дана вещественная матрица **А** размером *n*×*n*. Обходя матрицу **А** по строкам, из ее различных элементов сформировать одномерный массив **В** (т.е. повторяющиеся элементы матрицы **А** в массив **В** записывать один раз).
35. Дана матрица размером *n*×*m*. В каждой строке первый наибольший элемент сместить в начало строки.
36. Дана матрица размером *n*×*m*. В каждой строке последний отрицательный элемент сместить в конец строки.
37. Дана целочисленная матрица размером *n*×*m*. В каждой строке исключить все числа, кратные заданному числу *k*.
38. Дана матрица размером *n*×*m*. В каждом столбце сместить в начало столбца все числа, кратные заданному числу *k*.
39. Дана матрица размером *n*×*n*. В каждой строке выполнить циклический сдвиг так, чтобы в ее начале оказался первый наибольший элемент строки.
40. Дана матрица размером *n*×*m*. В каждом столбце выполнить циклический сдвиг таким образом, чтобы первым в столбце оказался первый отрицательный элемент.
41. Дана матрица размером *n*×*n*. В вектор **B** поместить наибольшие из элементов, расположенных на диагоналях, параллельных побочной диагонали. Обход матрицы начать с нижнего правого угла.
42. Дана матрица размером *n*×*n*. В вектор **B** поместить наибольшие из элементов, расположенных на диагоналях, параллельных главной диагонали. Обход матрицы начать с нижнего левого угла.
43. Дана матрица размером *n*×*m*. Определить номер первой, упорядоченной по возрастанию, строки.
44. Дана матрица размером *n*×*n*. Вывести элементы матрицы в порядке обхода матрицы по спирали. Обход матрицы начать с левого верхнего угла.
45. Дана матрица размером *n*×*m*. Найти наибольшее среди неповторяющихся чисел этой матрицы.
46. Дана матрица размером *n*×*n* с нормой меньше единицы. Вычислить первое *k*, при котором норма заданной матрицы, возведенной в степень *k*, становится меньше заданного числа ε.
47. Ввести числовую квадратную матрицу размерности *n*×*n*. Найти суммы элементов на периметрах всех вложенных квадратных подматриц, центр которых совпадает с центром исходной матрицы.
48. Введенная квадратная числовая матрица размером *n*×*n* разделена диагоналями, проведенными из углов, на четыре треугольника. Поменять местами элементы верхнего и нижнего треугольников, симметричные относительно горизонтальной оси симметрии матрицы. Диагональные элементы оставить на прежних местах.
49. Введенная квадратная числовая матрица размером *n*×*n* разделена диагоналями, проведенными из углов, на четыре треугольника. Поменять местами элементы левого и правого треугольников, симметричные относительно вертикальной оси симметрии матрицы. Диагональные элементы оставить на прежних местах.
50. Дана матрица размером *n*×*m*. Определить количество чисел, которые встречаются во всех строках матрицы. Перечислить эти числа.
51. Ввести числовую квадратную матрицу размером *n*×*n*, каждый элемент которой может быть равен нулю или единице. Найти в данной матрице количество квадратов размером *k*×*k*, состоящих только из нулей и только из 1. Числа *n* и *k* заданы.
52. Проверить, есть ли в матрице одинаковые элементы.
53. Составить программу, которая получает матрицу **Х** размерностью *n*×*m,* и если в ней есть строки и столбцы, все элементы которых равны нулю, то программа уплотняет исходную матрицу влево и вверх, удаляя эти строки и столбцы. Затем формируется новая матрица, в которой строки и столбцы исходной матрицы меняются местами. Программа выдает на экран исходную и сформированную матрицу, сообщив, уплотнялась ли исходная матрица.
54. Составить программу, которая получает матрицу **A** размерностью *n*×*m* и формирует новую матрицу, поместив в нее только те строки из исходной матрицы, которые образуют монотонную последовательность.
55. Составить программу, которая получает матрицу **B** размерностью *n*×*m* и формирует новую матрицу, включив в нее только те строки, элементы которых не образуют арифметической прогрессии. Если таких строк не найдено, то новая матрица не формируется, и об этом выдается сообщение.
56. Дана матрица **B** размерностью *n*×*n*. Проверить свойство **В**=(**В***Т*)*Т*, *Т* означает операцию транспонирования.
57. Составить программу, которая получает матрицу **Х** размерностью *n*×*m* и находит минимальный и максимальный элементы матрицы. Если оба элемента находятся под главной диагональю, то формируется новая матрица, из которой удалены строки и столбцы, на пересечении которых находятся эти элементы, в противном случае новая матрица не формируется и на экран выдается сообщение об этом.
58. Составить программу, которая получает матрицу **Х** размерностью *n*×*m* и целое число *k*. Программа формирует новую матрицу, осуществив циклический сдвиг всех строк исходной матрицы вверх *k* раз.
59. Для заданного натурального числа *n* (*n* не превышает 9) построить матрицу следующего вида:

n n … n n n n n … n n  
n n-1 … n-1 n-1 n-1 n-1 n-1 … n-1 n  
… … … … … … … … … … …  
n n-1 … 2 2 2 2 2 … n-1 n  
n n-1 … 2 1 1 1 2 … n-1 n  
n n-1 … 2 1 0 1 2 … n-1 n  
n n-1 … 2 1 1 1 2 … n-1 n  
n n-1 … 2 2 2 2 2 … n-1 n  
… … … … … … … … … … …  
n n-1 … n-1 n-1 n-1 n-1 n-1 … n-1 n  
n n … n n n n n … n n

1. Задана числовая матрица **A** размером *n*×*m*. Некоторый элемент этой матрицы назовем седловой точкой, если он является одновременно наименьшим в своей строке и наибольшим в своем столбце. Определить количество седловых точек в матрице.
2. Дана целочисленная матрица размером *n*×*m*. Определить сумму ее элементов, превосходящих всех своих ближайших соседей.
3. Дана матрица размером *n*×m. Выполнить поворот матрицы на 90 градусов по часовой стрелке.
4. Дана матрица размером n×*n*. Выполнить поворот матрицы на 90 градусов против часовой стрелки.
5. Из матрицы **A** с размерами 3*n*×3*m* сформировать матрицу **B** с размерами *N*×*М*. Каждый элемент *bij* матрицы **B** вычисляется как определитель третьего порядка, составленный из элементов матрицы **A**, лежащих на пересечении строк (3*i*-2), (3*i*-1), (3*i*) и столбцов (3*j*-2), (3*j*-1), (3*j*).
6. Сформировать матрицу **B**, элементами которой являются средние арифметические значения соседних элементов исходной матрицы **A**, т.е. элемент *bij* равен . При этом для крайних элементов считать соседними элементы на другом краю (например, для элемента *a*0,3 соседним будет *an*-1,3).
7. Переставить строки и столбцы квадратной матрицы так, чтобы алгебраические суммы произведений элементов *i*-го столбца и *i*-строки  убывали при изменении *i* от 1 до *n*.
8. Считая, что элементами матрицы *n*×*m* являются координаты *n* точек *m*-мерного евклидового пространства, определить диаметр этого множества точек и найти две точки, наименее удаленные друг от друга.
9. Считая, что элементами матрицы *n*×*m* являются координаты *n* точек *m*-мерного евклидового пространства, определить минимальный гиперпараллелепипед, содержащий все *m* точек. Вычислить объем найденного гиперпараллелепипеда и его проекции на оси координат.
10. Матрица **A** содержит координаты *n* точек на плоскости, а матрица **B** задает *m* прямоугольников (координатами левого верхнего и нижнего правого углов). Для каждого прямоугольника подсчитать число точек, находящихся строго внутри прямоугольника.
11. Вычислить различные нормы прямоугольной матрицы:  
    1) *m*-норма |*a*| = ; 2) *l*-норма |*a*| = ;   
    3) *k*-норма |*a*| = .
12. Исходная матрица содержит координаты *n* точек на плоскости, определяющих замкнутую ломаную линию. Определить, имеет ли эта ломаная самопересечения.
13. Исходная матрица содержит координаты *n* точек в 3-мерном пространстве. Определить, лежат ли эти точки в одной плоскости.
14. Исходный трехмерный массив задает *n* двумерных матриц одинаковой размерности. Транспонировать каждую из *n* матриц.
15. Целочисленную матрицу отсортировать следующим образом: элементы с четными значениями должны быть упорядочены внутри строк по убыванию внутри строк, элементы с нечетными значениями – по возрастанию. Четные и нечетные элементы не должны меняться местами друг с другом.
16. Найти в исходной матрице ненулевой элемент, в строке и столбце которого максимальное число нулей. Переставить строки и столбцы так, чтобы этот элемент оказался в верхнем левом углу.
17. На основе исходной прямоугольной матрицы получить два вектора: вектор, сформированный путем суммирования элементов строк матрицы; вектор, полученный путем умножения матрицы на первый вектор.
18. По заданной квадратной матрице вычислить обратную матрицу. Проверить правильность вычисления путём умножения результата и исходной матрицы (должна получиться единичная матрица).
19. Матрица *n*×2 задает координаты *n* точек плоскости (*n* – четное). Найти такую прямую, по разные стороны от которой лежат по *n*/2 точек.
20. Матрица *n*×2 задает координаты *n* точек плоскости. Найти ширину этого множества точек (минимальное расстояние между такими двумя параллельным прямыми, что все точки лежат между ними).
21. Исходная квадратная матрица состоит из 0 и 1. Вычислить определитель матрицы.
22. По исходной матрице сформировать новую, имеющую столько же строк, но хранящую только ненулевые элементы. Элементом новой матрицы является пара (*j*,*aij*). Выполнить обратное преобразование.
23. По исходной матрице сформировать новую, хранящую только ненулевые элементы. Элементами новой матрицы являются тройки (*i*, *j*, *aij*). Выполнить обратное преобразование.
24. Даны две квадратные матрицы одинаковых размеров. Проверить, можно ли получить одну из другой, выполнив зеркальное отражение относительно горизонтальной оси симметрии матрицы.
25. Даны две квадратные матрицы одинаковых размеров. Проверить, можно ли получить одну из другой, выполнив зеркальное отражение относительно вертикальной оси симметрии матрицы.
26. Во вводимой числовой матрице размером *n*×*m* определить количество прямоугольных рамок (замкнутых прямоугольных контуров) толщиной в одну клетку, состоящих из одинаковых элементов.
27. Двигаясь змейкой от одного из углов введенной числовой квадратной матрицы, состоящей из *n* строк и столбцов (*n* – заданное число), по линиям, параллельным главной диагонали, обойти все элементы матрицы и распечатать их в порядке обхода. Главная диагональ соединяет левый верхний и правый нижний углы матрицы.
28. Целочисленную матрицу отсортировать так, чтобы элементы строк образовали единую упорядоченную по возрастанию последовательность.

### 9. Строки

**Требования к написанию и оформлению кода программы**

Алгоритм решения задачи должен быть разделен на функции. Эти функции разработать двумя способами, так чтобы доступ к нужному символу в строке (движение по строке) осуществлять путем изменения

* смещения относительно начального адреса строки (*индекса*);
* значения *указателя* (адреса) этого символа.

Эти функции при необходимости могут быть разделены на ряд составляющих функций.

Функция main() должна содержать

* меню, которое позволяет сделать выбор режима работы со строкой (через индекс или указатель);
* возможность многократного выполнения программы.

*Примечание:* предложение – строка, содержащая слова. Слово – подпоследовательность символов, с одной или с обеих сторон ограниченная одним или несколькими пробелами (первое и последнее слова ограничены пробелами, возможно, с одной стороны). Предложение не содержит знаков препинания и строчных букв.

**Задачи**

1. Дана строка символов. Внести в текст изменения. Каждую малую латинскую букву заменить стоящей перед ней в алфавите. Букву *a* заменять буквой *z*.
2. Дано предложение. Каждую цепочку пробелов, ограничивающих слова, сократить до одного пробела.
3. Дано предложение. Распечатать все симметричные слова нечетной длины.
4. Дано предложение. Исключить из него все слова, начинающиеся и оканчивающиеся на одну и ту же букву.
5. Дано предложение. Распечатать самое длинное слово, не содержащее буквы *a*.
6. Дано предложение. Определить, сколько слов имеют длину такую же, как у слова с заданным номером *k*.
7. Дано предложение. Напечатать все гласные, которые входят в каждое слово с четным номером и не входят ни в одно слово с нечетным номером.
8. Дано предложение. Напечатать все малые английские буквы, которые не входят ни в одно слово.
9. Дано предложение. Распечатать самое короткое слово, полностью расположенное во второй половине предложения.
10. Дано предложение. Исключить все слова, начинающиеся с префикса «наи».
11. Дано предложение. В словах с окончанием «ed»,сменить его на «ing».
12. Даны два предложения. Напечатать те слова из первого, которые входят и во второе предложение.
13. Дано предложение. Распечатать в столбик все слова в порядке убывания их длин.
14. Дано предложение. Распечатать в столбик все слова в алфавитном порядке.
15. Даны два предложения. Проверить, есть ли во втором слова, являющиеся инверсией какого-либо слова из первого предложения.
16. Дано предложение. Определить количество слов, в которых первая буква повторяется хотя бы один раз.
17. Дано предложение. Поменять местами самое длинное и самое короткое слова предложения. Если таковых несколько, то менять местами первое самое длинное и первое самое короткое.
18. Дано предложение. Определить максимальное количество слов, начинающихся с одной и той же буквы.
19. Дано предложение. Слова предложения содержат либо последовательность букв, либо последовательность не более чем из пяти цифр. Вычислить сумму чисел, содержащихся в тексте.

**Разработать и использовать указанные функции в контексте поставленной задачи.**

1. Функция char\***subStr**(char\*str, char с, int& len) получает строку str, символ c и возвращает адрес самой длинной подстроки, начинающейся и заканчивающейся заданным символом, а также длину этой подстроки len. Если такой подстроки нет в строке, то функция возвращает NULL. Программа, используя функцию **subStr** в каждой из *m* введенных строк, передавая в нее заданный символ, ищет соответствующую подстроку и зеркально ее переворачивает. На экран выводятся только преобразованные строки.
2. Функция char\* **delBigSymb**(char\* str)получает строку str и удаляет прописные (заглавные) символы, если они дублируют рядом стоящий строчный символ. Функция возвращает адрес преобразованной строки str или NULL, если в строке не удалялось ни одного прописного символа. Программа со стандартного устройства ввода считывает фразу, состоящую из произвольного количестваслов, и выдает на экран преобразованную фразу, в которой все слова отредактированы функцией **delBigSymb**. Программа подсчитывает количество слов, не подвергнутых редактированию.
3. Функция int **pos**(char\* str, char\* Substr, int n) получает строку str, строку Substr и позицию n, с которой начинается поиск подстроки Substr в строке str. Функция, найдя первую подстроку Substr и ее позицию в этой строке str, переворачивает эту подстроку наоборот. Функция возвращает найденную позицию подстроки, или если подстрока в строке str не найдена, то функция возвращает ноль*.* Программа, используя функцию **pos,** в каждой из *m* введенных строк, ищет заданную пользователем подстроку и зеркально переворачивает каждую такую найденную подстроку. На экран выводятся только преобразованные строки.
4. Функция char\* **isNum** (char\* str, int& len) получает исходную строку str и ищет в ней первую подстроку, содержащую только цифровые символы. Если такая подстрока найдена, функция возвращает через параметр по ссылке ее длину, а ее адрес через возвращаемое значение. Значение NULL будет результатом функции, если подстрока не найдена. В каждой из *m* введенных строк, используя **isNum,**исключаем первую подстроку, содержащую только цифровые символы. На экран выводятся преобразованные строки.
5. Функция char\* **copyLeft**(char\*str, char с, int n)получает строку str, символ c, число n и добавляет к строке str слева *n* повторений символа c*.* Функция возвращает адрес str. Программа со стандартного устройства ввода считывает фразу, состоящую из произвольного количестваслов, считывает символ и выдает на экран преобразованную фразу, в которой все слова отредактированы функцией **copyLeft**, где n – количество слов в фразе.
6. Функция char\* **DelOne**(char\* str)получает строку strи удаляет из нее первый символ, код которого оказался смежным с кодом следующего символа (смежными назовем числа, значения которых отличаются на единицу). Функция возвращает адрес удаленного символа, если можно удалить такой символ, и NULL, если нельзя. Функция char\* **DelAll**(char\* str)получает строку strи удаляет из нее все символы, коды которых оказались смежными с кодом следующих за ними символов. Функция возвращает адрес строки, если можно удалить хотя бы один такой символ, и NULL, если нельзя. Функция **DelAll** должна использовать функцию **DelOne*.*** Программа со стандартного устройства ввода считывает фразу, состоящую из произвольного количестваслов, и, используя функцию **DelAll*,*** выдает на экран преобразованную фразу, в которой оставлены только измененные слова исходной.
7. Функция char\* **left**(char\* str, int n) получает строку str, число n и оставляет в строке n первых символов. Функцияchar\* **right**(char\* str, int n) получает строку str, число n и оставляет в строке n последних символов. Если длина строки str меньше n, выделяется вся строка. Если полученная строка не подверглась изменению, то функция возвращает NULL, в противном случае возвращает адрес преобразованной строки str. Преобразовать заданную строку, оставив в ней первые *k* и последние *n* символов.
8. Функция int **inStr**(char\*str1, char\*str2, int n) получает строку str1, строку str2, позицию n и осуществляет поиск строки str2 в строке str1, начиная с позиции n строки str1. Если поиск окончился успешно, она возвращает позицию символа в строке str1, начиная с которой обнаружено вхождение в нее строки str2, в противном случае возвращает ноль. Программа подсчитывает с помощью функции **inStr** суммарное количество вхождений заданной пользователем подстроки в *n*словах введенной им фразы.
9. Функция char\* **dispose**(char\*str, int n, int Len) получает строку str*,* начальную позицию символа n, с которого начинается удаление подстроки, длиной Len*,* из строки str*.* Если позиция символа n выходит за границы строки str, то функция возвращает NULL***.*** Если длина удаляемой подстроки n больше оставшейся части строки str, то удаляется оставшаяся часть строки str. Функция возвращает адрес str, если удаление было осуществлено. Используя функцию **dispose***,* исключить из заданного предложения все слова, у которых совпадают первая и последняя буквы.
10. Функция char\* **small**(char\* str) получает строку str и «сжимает» строку, удалив из нее все, кроме одного, одинаковые символы, следующие подряд. Если таких символов в строке не найдено, то функция возвращает NULL, в противном случае – адрес «сжатой» строки str*.* Программа со стандартного устройства ввода считывает фразу, состоящую из *n*слов, и выдает на экран преобразованную фразу, в которой слова «сжаты» с помощью функции **small**. Программа подсчитывает количество непреобразованных слов в исходной фразе.
11. Функция char\* **detl**(char\* str, int n)получает строку str, позицию символа вставки nв ней и вставляет в эту позицию символ, стоящий на этой позиции (дублирует символ). Функция возвращает адрес преобразованной строки str или NULL, если позиция вставки nвыходит за длину строки str*.* Программа со стандартного устройства ввода считывает фразу, состоящую из произвольного количестваслов, и символ *с*. В каждом слове выполнить сдваивание символа *с*, если он входит в это слово, но не является начальным или конечным.
12. Функция char\* **UpDownStr**(char\* str, int n) получает строку str и преобразует каждую n-ую подстроку латинских букв в подстроку тех же букв, но в противоположном регистре, преобразуя каждый символ латинской буквы строки strв этот же символ, но в другом регистре. Функция возвращает адрес строки str, если преобразования были осуществлены над всеми n-ыми подстроками строки str, и NULL, если ни один символ в строке не был подвергнут преобразованию. Программа со стандартного устройства ввода считывает число *n* и *n*слов и, используя функцию **UpDownStr,** преобразует слова, выдавая на экран только те из них, в которых удалось сделать преобразования.

### 10. Записи

**Требования к написанию и оформлению кода программы**

Оформить работу со структурой и (или) массивом структур в отдельном файле заголовков «ИмяСтруктуры.h». Для передачи в основную программу должны быть доступны только константы (если необходимо), разработанный пользовательский тип данных и функции, решающие поставленную задачу. Никакие глобальные переменные не должны фигурировать в созданном файле заголовков.

Выполнить указанную в задании обработку массивов данных и результаты вывести на экран

* в текстовом режиме в отдельном окне, используя цветовую палитру;
* в графическом режиме в виде цветной диаграммы, дополнив ее текстовыми комментариями.

Варианты диаграммы указаны в заданиях.

Подготовить файлы (в текстовом и бинарном форматах) для ввода исходной информации. Количество записей в файле не должно быть меньше 30-ти.

**Задачи**

● Дан массив записей. Каждая запись содержит следующие поля:

* название субъекта РФ;
* площадь;
* уровень безработицы;
* стоимость потребительской корзины;
* средняя зарплата по региону.

1. Напечатать список субъектов федерации, в которых уровень зарплаты выше средней по РФ, а уровень безработицы ниже средней по РФ. Вывести вертикальную диаграмму уровней безработицы в субъектах федерации.
2. Напечатать список субъектов федерации, в которых уровень зарплаты ниже средней по РФ, а стоимость потребительской корзины выше средней ее стоимости по РФ. Вывести круговую диаграмму уровней безработицы в субъектах федерации.
3. Напечатать список субъектов федерации с максимальным уровнем безработицы. Вывести горизонтальную диаграмму уровней безработицы в субъектах федерации.
4. Напечатать список названий субъектов федерации в порядке убывания средней зарплаты. Вывести круговую диаграмму уровня средней зарплаты по субъектам федерации.
5. Напечатать список субъектов федерации в порядке возрастания стоимости потребительской корзины. Вывести горизонтальную диаграмму уровня средней зарплаты по субъектам федерации.
6. Напечатать список субъектов федерации с максимальной средней зарплатой. Напечатать вертикальную диаграмму уровней средней зарплаты в субъектах федерации.

● Дан массив записей о студентах первого курса. Каждая запись содержит следующие поля:

* фамилия студента;
* номер группы;
* пол;
* массив оценок за последнюю сессию.

Записи отсортированы по группам. Информация о количестве студентов в каждой группе не зафиксирована.

1. Для каждой группы вывести на экран информацию о процентном соотношении студентов успевающих на «отлично», «хорошо», «удовлетворительно», «неудовлетворительно». Для каждой группы представить результаты в текстовом варианте и в виде горизонтальной диаграммы.
2. Определить лучшую группу на курсе и вывести на экран информацию о процентном соотношении студентов этой группы, успевающих на «отлично», «хорошо», «удовлетворительно», «неудовлетворительно». Если таковых групп несколько, то рассмотреть первую из них.
3. Для каждой группы составить список студентов с успеваемостью выше средней на курсе. Построить диаграмму распределения лучших студентов по группам.
4. Из списка курса исключить группу с минимальной успеваемостью по программированию. Выдать на экран новый список курса. Кроме того, построить диаграмму распределения по группам студентов, успевающих на «отлично» по программированию.
5. Исключить из списка курса всех, кто имеет три и более двойки. Напечатать новый список курса. Построить вертикальную диаграмму распределения количества исключенных по группам.
6. Для каждой группы составить список студенток с успеваемостью «хорошо» и «отлично». Построить круговую диаграмму распределения таких студенток по группам.
7. В каждой группе расположить студентов в порядке убывания их успеваемости. Построить горизонтальную диаграмму распределения студентов-хорошистов по группам.
8. В каждой группе исключить двух студентов с минимальной успеваемостью. Построить вертикальную диаграмму распределения по группам количества студентов, имеющих «отлично» по математике.
9. Создать новую группу на курсе и перевести в нее всех отличников. Построить вертикальную диаграмму, описывающую процентное распределение переведенных студентов по прежним группам (новую группу расположить в начале массива).
10. Во входном потоке имеется информация о студентах, дополнительно зачисленных на курс. Их количество соответствует количеству групп. Включить новых студентов по одному в каждую группу в конец списка группы. Построить круговую диаграмму процентного распределения общего количества студентов по группам.
11. Образовать новую группу программистов и перевести в нее всех, у кого по программированию «отлично». Построить вертикальную диаграмму распределения по группам количества студентов, имеющих «отлично» по программированию.

● Дан массив записей. Каждая запись содержит следующие поля:

* фамилия автовладельца;
* название марки автомобиля;
* год выпуска;
* сумма уплаченного налога за последний год.

Записи отсортированы по маркам. Информация о количестве автомобилей каждой марки не зафиксирована.

1. Составить список владельцев, не заплативших налог. Информацию сгруппировать по маркам автомобилей. Построить круговую диаграмму процентного соотношения автомобилей, представленных в списке.
2. Составить список марок автомобилей, представленных в массиве, и указать количество автомашин каждой марки. Построить горизонтальную диаграмму процентного соотношения автомобилей, представленных в списке.
3. Составить список владельцев самых старых автомобилей. Такую информацию выдать по каждой марке автомобиля. Построить вертикальную диаграмму процентного соотношения автомобилей, представленных в списке.
4. Составить список владельцев автомобилей выпуска 2010 г. Информацию сгруппировать по маркам автомобилей. Построить круговую диаграмму процентного соотношения марок автомобилей выпуска 2010 г.
5. Для каждой марки автомобиля составить список владельцев в порядке возрастания возраста автомобилей. Построить горизонтальную диаграмму процентного соотношения автовладельцев, заплативших и не заплативших налог. Такую диаграмму представить для каждой марки.
6. Составить список владельцев автомобиля Renault, не заплативших налог. Построить горизонтальную диаграмму процентного соотношения марок автомобилей выпуска года, предшествующего данному.
7. Для каждой марки авто составить список владельцев автомобилей выпуска этого и предыдущего года. Построить круговую диаграмму процентного соотношения марок новых автомобилей.

● Дан массив записей. Каждая запись содержит следующие поля:

* название пункта назначения рейса;
* номер рейса;
* тип самолета;
* время вылета;
* цена билета.

Записи отсортированы по времени вылета. Информация о количестве типов самолетов не зафиксирована.

1. Вывести на дисплей всю информацию (номера рейсов, типы самолетов и цены билетов) о самолетах, вылетающих в пункт назначения, название которого введено с клавиатуры (если таких рейсов нет, вывести соответствующее сообщение). Построить вертикальную диаграмму процентного соотношения типов самолетов, представленных в списке.
2. Вывести на дисплей всю информацию (пункты назначения, номера рейсов, типы самолетов и цены билетов) о самолетах, вылетающих в пункты назначения в ночное время с 00.00 до 05.00. Информацию расположить по времени вылета. Построить круговую диаграмму процентного соотношения типов самолетов, представленных в списке.
3. С клавиатуры вводится информация о новом рейсе. Вставить в массив новую запись, не нарушив упорядоченность массива по времени вылета. Построить горизонтальную диаграмму процентного соотношения типов самолетов, представленных в списке.

### 11. Рекурсия

**Требования к написанию и оформлению кода программы**

При решении поставленных задач, основной алгоритм необходимо оформить в виде рекурсивной функции. Вывести на экран всю необходимую информацию об исходных данных и результатах работы программы.

**Задачи**

1. Найти сумму элементов одномерного массива. В бинарном файле по строкам записана матрица размером *n*×*m* (значения *n* и *m* указаны первыми значениями в начале файла). Для каждой строки матрицы с помощью рекурсивной функции определить сумму ее элементов.
2. Проверить, является ли массив симметричным относительно середины. В бинарном файле по строкам записана матрица размером *n*×*m* (значения *n* и *m* указаны первыми значениями в начале файла). С помощью рекурсивной функции посчитать количество симметричных строк.
3. Преобразовать последовательность положительных и отрицательных чисел таким образом, чтобы в начале этой последовательности оказались все положительные, а затем все отрицательные числа. В бинарном файле по строкам записана матрица размером *n*×*m* (значения *n* и *m* указаны первыми значениями в начале файла). Каждую строку матрицы преобразовать с помощью рекурсивной функции.
4. Преобразовать число в строку. Дан бинарный файл вещественных чисел, на его основе создать текстовый файл, переписав в него числа из бинарного файла через пробел, преобразовывая числа в строку с помощью рекурсивной функции.
5. Определить, является ли введенная строка палиндромом. Дан текстовый файл, определить, сколько в нем строк-палиндромов, используя рекурсивную функцию.
6. Инвертировать введенную строку. Написать программу, которая с помощью этой рекурсивной функции переворачивает в обратном порядке каждую строку текстового файла.
7. Дано натуральное число *n*. Вывести все числа от единицы до *n*.
8. Даны два целых числа *a* и *b*. Вывести все числа от *a* до *b* включительно в порядке возрастания, если *a* < *b*, или в порядке убывания в противном случае.
9. Даны два целых числа *m* ≥ 0 и *n* ≥ 0. Вычислите для *n* и *m* функцию Аккермана



1. Дано натуральное число *n*. Вывести true, если число *n* является точной степенью двойки, или слово false в противном случае (операцией возведения в степень пользоваться нельзя!).
2. Дано натуральное число *n*. Вычислить сумму его цифр. При решении задачи нельзя использовать строки, списки, массивы и циклы.
3. Дано натуральное число *n*. Вывести все его цифры по одной в обратном порядке, разделяя их пробелами или новыми строками. При решении этой задачи нельзя использовать строки, списки, массивы и циклы. Разрешена только рекурсия и целочисленная арифметика.
4. Дано натуральное число *n*. Вывести все его цифры по одной в обычном порядке, разделяя их пробелами или новыми строками. При решении этой задачи нельзя использовать строки, списки, массивы и циклы. Разрешена только рекурсия и целочисленная арифметика.
5. Дано натуральное число *n* > 1. Вывести все простые множители этого числа в порядке неубывания с учетом кратности. Алгоритм должен иметь сложность *O*(log *n*).
6. Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом ноль. Вывести все нечетные числа из этой последовательности, сохраняя их порядок (в этой задаче нельзя использовать глобальные переменные и передавать какие-либо параметры в рекурсивную функцию). Функция получает данные, считывая их с клавиатуры, и не возвращает значение, а сразу же выводит результат на экран. Основная программа должна состоять только из вызова этой функции.
7. Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом ноль. Определить наибольшее значение числа в этой последовательности. В задаче нельзя использовать глобальные переменные и передавать какие-либо параметры в рекурсивную функцию. Функция получает данные, считывая их с клавиатуры, и возвращает единственное значение – максимум считанной последовательности. Гарантируется, что последовательность содержит хотя бы одно число (кроме нуля).
8. Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом ноль. Определить значение второго по величине элемента в этой последовательности, т.е. элемента, который будет наибольшим, если из последовательности удалить наибольший элемент. В задаче нельзя использовать глобальные переменные. Функция получает данные, считывая их с клавиатуры, а не получая их в виде параметра. Гарантируется, что последовательность содержит хотя бы два числа (кроме нуля).
9. Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом ноль. Определить, какое количество элементов этой последовательности равно ее наибольшему элементу. Гарантируется, что последовательность содержит хотя бы одно число (кроме нуля).
10. Дано число *n*, десятичная запись которого не содержит нулей. Получить число, записанное теми же цифрами, но в противоположном порядке. При решении задачи нельзя использовать циклы, строки, списки, массивы; разрешаются только рекурсия и целочисленная арифметика. Функция должна возвращать целое число, являющееся результатом работы программы; выводить число по одной цифре нельзя.
11. Составить программу с рекурсивной функцией, которая получает наибольший делитель двух целых чисел.
12. Составить программу с рекурсивной функцией, которая сортирует одномерный массив методом выбора.
13. Составить программу с рекурсивной функцией, которая получает из десятичного числа его представление в *p*-ой системе счисления (*p*<10).
14. Составить программу с рекурсивной функцией, которая вычисляет по следующей формуле:  и  при условии 0 < *m* < *n*.
15. Составить программу с рекурсивной функцией ROOT(f,a,b,eps), которая методом деления отрезка пополам находит с точностью eps корень уравнения *f*(*x*)=0 на отрезке [*a*,*b*]. Cчитать, что eps>0, *a* < *b*, *f*(*a*)·*f*(*b*) < 0 и *f*(*x*) –непрерывная и монотонная функция на отрезке [*a*,*b*].
16. Составить программу, которая формирует фрактал «Ветка». Параметры процедуры: int x, int y, float Len, float Angle, где *x*, *y* – координаты начала отрезка; *Len* – длина отрезка; *Angle* – угол наклона отрезка к оси *OX.*

Алгоритм: из начальных координат строится отрезок заданной длины, из конца которого рисуется левая и правая ветки длиной в 1.5 раза меньше под заданным углом. Далее рисунок повторяется для каждого из вновь построенных отрезков. Для этого

* рекурсивная процедура пересчитывает новые координаты *x*, *y* по заданным длине и углу;
* рисуется отрезок от начального значения координат до их нового значения;
* вызывается рекурсивно процедура для рисования левой и правой ветки с новыми параметрами (новые координаты *x*, *y*; длина, уменьшенная в 1.5 раза, и угол, уменьшенный (увеличенный) на 30 градусов);
* выход из процедуры осуществить, если длина станет меньше одного пикселя.

Начальные параметры: 100, 400, 160, 40\**radian*, где *radian*=*pi*/180.

1. Составить программу, которая формирует фрактал «Ветка мимозы». Параметры процедуры: int *x,* int *y,* int *Len,* float *Angle*, где *x, y* – координаты начала отрезка, *Len* – длина отрезка, *Angle* – угол наклона отрезка к оси *OX* .

Алгоритм: из начальных координат строится отрезок заданной длины, из конца которого рисуется левая и правая ветки уменьшенной длины под заданным углом. Далее рисунок повторяется для каждого из вновь построенных отрезков. Для этого

* рекурсивная процедура пересчитывает новые координаты *x*, *y* по заданным длине и углу;
* рисуется отрезок от начального значения координат до их нового значения;
* длина отрезка уменьшается на случайную величину (например, random(Len) / 2 );
* подсчитывается добавка к углу, выбранная случайным образом (например, 0.1+random);
* вызывается рекурсивно процедура для рисования левой и правой ветки с новыми параметрами (новые координаты *x*, *y*; длина и угол, уменьшенный (увеличенный на полученную добавку);
* выход из процедуры осуществить, если длина станет меньше пяти пикселей.

Начальные параметры: 220, 340, 60, 0.8.

1. Составить программу, которая формирует фрактал «Салфетка Серпинского». Параметры процедуры: int *x*1*,* int *y*1*,* int *x*2*,* int *y*2*,* int *x*3*,* int *y*3*,* int *n*, где *x*1, *y*1, *x*2, *y*2, *x*3, *y*3 – координаты угловых точек треугольника; *n* – количество вложенности процедуры.

Алгоритм: из начальных координат строятся отрезки. В полученном треугольнике вычисляются средние точки и через них проводятся отрезки. Далее рисунок повторяется для каждого из вновь построенных треугольников при углах первоначального треугольника. Для этого

* рисуется с помощью линий треугольник по заданным точкам;
* пересчитываются новые координаты для средних точек каждой стороны треугольника;
* вызываются рекурсивно три процедуры для рисования трех меньших треугольников с пересчитанными координатами точек (при этом *n* уменьшается на единицу);
* выход из процедуры осуществить, если n ≤ 0.

Начальные параметры: 100, 300, 200, 120, 300, 300, 7.

1. Составить программу, которая формирует фрактал «Кривая дракона». Параметры процедуры: int *x*1*,* int *y*1*,* int *x*2*,* int *y*2*,* int *n*, где *x*1, *y*1 и *x*2, *y*2 – координаты начальной и конечной точек линии; *n* – количество вложенности процедуры.

Алгоритм: на отрезке, как на гипотенузе, строится прямой угол. На полученных сторонах прямого угла тоже строятся прямые углы, но один развернут в правую сторону, а другой – в левую. Далее процесс повторяется *n* раз. Для этого

* если *n* > 0, то пересчитываются координаты для средней точки треугольника с прямым углом (*xn* = (*x*2+*x*1)/2 + (*y*2–*y*1)/2 и *yn* = (*y*1+*y*2)/2 – (*x*2–*x*1)/2) и вызываются рекурсивно две процедуры для рисования двух новых отрезков из точек с первоначальными координатами в точки с пересчитанными координатами (при этом *n* уменьшается на единицу);
* иначе рисуем линию для первоначальных координат точек.

Начальные параметры: 200, 300, 500, 300, 14.

1. Составить программу, которая формирует фрактал «Снежинка». Параметры процедуры: int *x,* int *y,* int *r*, где *x* и *y* – координаты центра; *r* – радиус снежинки.

Алгоритм: рисунок представляет собой десятиконечную снежинку, каждый конец которой в свою очередь представляет собой множество подобных десятиконечных снежинок, уменьшающихся в размере по направлению к центру. Если радиус снежинки меньше пикселя, то рисуется лишь точка с заданными координатами. В противном случае

* в процедуре изображаются 10 ветвей снежинки в цикле по углу с шагом 36 градусов (т.е. с количеством проходов, равным 10);
* внутри этого цикла рисуется собственно сама ветвь, которая состоит из восьми подобных снежинок; для этого вызывается внутренний цикл, в котором пересчитываются координаты центра новой снежинки, и вызывается рекурсивно процедура с пересчитанными координатами центра и радиусом, уменьшенным в пять раз;
* по мере приближения к центру радиус уменьшается в 1.5 раза.

Начальные параметры: 320, 240, 160.

1. Составить программу, которая формирует фрактал «Снежинка игольчатая». Параметры процедуры: int *x,* int *y,* int *Len*, где *x*, *y* – координаты центра снежинки; *Len* – длина отрезков.

Алгоритм: из начальных координат строятся шесть отрезков, расположенных друг к другу под углом 60 градусов. Далее рисунок повторяется для каждого из вновь построенных отрезков с уменьшением первоначальной длины. Для этого в цикле с количеством проходов, равным шести,

* пересчитываются новые координаты для конца отрезка по известным координатам центра, длине отрезка и углу, увеличенному на 60 градусов;
* рисуется этот отрезок;
* вызывается рекурсивно процедура для рисования меньшей снежинки с пересчитанными координатами центра (при этом длина уменьшается в три раза);
* выход из процедуры осуществить, если длина < 1.

Начальные параметры: 320, 240, 100.

1. Составить программу, которая формирует фрактал «Планеты и спутники». Параметры процедуры: int *x,* int *y,* int *r,* int *n*, где *x*, *y* – координаты центра планеты; *r* – ее радиус; *n* – количество вложенности процедуры.

Алгоритм: из начальных координат строится окружность заданного радиуса, вычисляется радиус орбиты, который в 2.5 раза больше радиуса планеты, в цикле с количеством проходов, равным шести (по количеству спутников),

* вычисляются новые координаты центра через полученный радиус орбиты и угол поворота, который меняется на π/3;
* вызывается рекурсивно процедура для рисования новой планеты с пересчитанными координатами центра (при этом радиус уменьшается в три раза и *n* уменьшается на единицу);
* выход из процедуры осуществить, если *n* ≤ 0.

Начальные параметры: 300, 200, 60, 5.

1. Составить программу, которая формирует фрактал «Модель легкого». Параметры процедуры: float *x,* float *y,* float *w,* float *h,* int *vertical*, где *x*, *y* – координаты центра ветви; *w* – ширина ветви; *h* – высота ветви; *vertical* равно единице, если ветвь вертикальная, и нулю, если горизонтальная.

Алгоритм: из центра ветви строится прямоугольник высотой *h* и шириной *w*. Под прямым углом рисуется прямоугольник меньшей толщины и ширины. Далее рисунок повторяется для каждой из двух сторон вновь построенного прямоугольника с уменьшением ширины и высоты. Для этого

* рисуется закрашенный прямоугольник с учетом координат, высоты и ширины ветви;
* если ширина или высота меньше 0.15, то выйти из процедуры;
* если ветвь вертикальная, то длина ветви = 0.7*h*, а толщина равна 0.66*w,* и вызываются рекурсивно две процедуры для построения верхнего и нижнего прямоугольников, при этом *vertical* равно нулю;
* в противном случае наоборот – длина равна 0.7*w*, толщина равна 0.66*h* и вызываются рекурсивно две процедуры для правого и левого прямоугольников с параметром *vertical*, равным единице;
* в основной программе нарисовать первоначальный закрашенный прямоугольник с начальными параметрами.

Начальные параметры: 320, 240, 120, 5, 0.

1. Составить программу, которая формирует фрактал «Уменьшающиеся квадраты». Параметры процедуры: int *x,* int *y,* int *r,* int*n*, где *x*, *y* – координаты верхней левой точки квадрата; *r* – сторона квадрата; *n* – количество вложенности процедуры.

Алгоритм: из начальных координат строится квадрат со стороной *r*. На каждой из сторон квадрата строится новый квадрат со стороной в два раза меньше *r*. Далее рисунок повторяется для каждого из вновь построенного квадрата. Для этого

* рисуется квадрат по заданным параметрам;
* пересчитываются новые координаты для левого (правого) и верхнего (нижнего) квадратов;
* вызываются рекурсивно четыре процедуры для рисования четырех меньших квадратов на сторонах старого квадрата (при этом *n* уменьшается на единицу);
* выход из процедуры осуществить, если *n* ≤ 0.

Начальные параметры: 200,200,100,6.

1. Дано *n* различных натуральных чисел (*n* ≤ 5). Получить все перестановки этих чисел.
2. На шахматной доске расставить восемь ферзей так, чтобы они не били друг друга. Написать программу, которая печатает a) одну из таких расстановок; б) все такие расстановки.

### 12. Организация работы с файлами в С/С++

**Требования к написанию и оформлению кода программы**

Основная функция должна быть организована в виде вызова функций, каждая и которых решает свою подзадачу. Файлы для работы должны быть подготовлены заранее и быть достаточно объемными и информативными.

Решения задач должны быть представлено в двух вариантах с использованием

* функций библиотеки stdio.h;
* потоковых классов библиотеки fstream.h.

**Задачи**

1. Дан текстовый файл. Количество символов в каждой строке произвольное. Скопировать в новый файл все символы, с разбивкой по *N* символов в строке.
2. С консоли ввести строку. Определить, сколько раз эта строка встречается в файле.
3. Строка файла содержит предложения. В новый файл перенести предложения, не содержащие слов-палиндромов.
4. Текстовый файл содержит вещественные числа (в строке произвольное количество чисел). В новый файл вывести среднеарифметические значения строк файла по *n* чисел в строке.
5. Текстовый файл содержит целые числа (в строке произвольное количество чисел). В новый файл вывести строки с максимальной суммой чисел, содержащихся в строке.

● Структура с именем *Student* содержит следующие поля:

*Name* – фамилия и инициалы;

*Course* – номер курса;

*Number* – номер зачетки;

*Marks* – успеваемость (массив из пяти элементов).

○ Дан текстовый файл, содержащий структуры типа *Student*.

1. Вывести в новый текстовый файл информацию о студентах первого курса со средним баллом больше 4.0.
2. Вывести в новый текстовый файл информацию о студентах первого курса, имеющих хотя бы одну двойку.
3. Вывести в новый текстовый файл информацию об отличниках второго курса.
4. Вывести в новый текстовый файл информацию о студентах первого курса со средним баллом больше среднего балла на этом курсе.
5. Даны два текстовых файла, содержащих структуры типа *Student*. Оба файла упорядочены в алфавитном порядке фамилий. Объединить эти файлы, создав третий файл, тоже упорядоченный в алфавитном порядке фамилий.

○ Дан бинарный (типизированный) файл, содержащий структуры типа *Student.*

1. Исключить из файла записи о студентах, чьи фамилии вводятся с клавиатуры.
2. Исключить из файла записи о студентах, имеющих хотя бы одну двойку.
3. Добавить в начало файла *n* новых записей.
4. В упорядоченный по убыванию среднего балла файл внести записи о новых студентах, не нарушая упорядоченности файла.

● Структура с именем *Worker* содержит следующие поля:

*Name* – фамилия и инициалы работника;

*Post* – название занимаемой должности;

*Year* – год поступления на работу;

*Salary* – массив ежемесячных зарплат за год.

○ Дан текстовый файл, содержащий структуры типа *Worker*.

1. Вывести в новый текстовый файл информацию о сотрудниках, чей стаж работы превышает значение, введенное с клавиатуры; при этом среднемесячная зарплата не превосходит 30 000 руб.
2. Вывести в новый текстовый файл информацию о сотрудниках, чья зарплата повышалась из месяца в месяц.
3. Вывести в новый текстовый файл информацию о сотрудниках, чья среднемесячная зарплата превосходит среднемесячную на предприятии за последний год.
4. Вывести в новый бинарный (типизированный) файл информации о работниках, чья зарплата в каждом месяце превышала среднемесячную на предприятии за последний год.
5. Вывести в новый текстовый файл информацию о сотрудниках с максимальной среднемесячной зарплатой.
6. Добавить в конец файла записи о вновь поступивших на работу.

● Структура с именем *Goods* содержит следующие поля:

*Name* – название товара;

*Shop* – название магазина, в котором продается товар;

*Sale* – суммы выручки за товар в каждом месяце прошлого года.

○ Дан текстовый файл, содержащий структуры типа *Goods*.

1. Вывести в новый текстовый файл информацию о тех товарах, за которые получили максимальную годовую выручку.
2. Вывести в новый текстовый файл информацию о том, в каких магазинах получили максимальную годовую выручку за товар, название которого введено с клавиатуры.
3. Вывести в новый текстовый файл информацию о годовой выручке по каждому из магазинов.
4. Вывести в новый текстовый файл информацию о том, в каких магазинах выручка за товар, название которого введено с клавиатуры, повышалась из месяца в месяц.

● Текстовый (бинарный) файл содержит информацию о произвольном количестве массивов.Для каждого массива вначале из файла записан размер массива, затем значение элементов массива.

1. В новый текстовый (или бинарный) файл переписать все массивы, в которых преобладают простые числа.
2. В новый текстовый (или бинарный) файл переписать все массивы, предварительно преобразовав каждый по следующему алгоритму: выполнить циклический сдвиг массива влево таким образом, чтобы первый отрицательный элемент стал первым в массиве.
3. В новый текстовый (или бинарный) файл переписать все массивы, в которых отрицательные элементы упорядочены по возрастанию.
4. Исключить из файла все неупорядоченные массивы.
5. В новый текстовый (или бинарный) файл переписать все массивы, предварительно упорядочив каждый по возрастанию элементов.

● Текстовый (бинарный) файл содержит информацию о произвольном количестве матриц. Для каждой матрицы вначале из файла записаны размеры матрицы, затем построчно значения ее элементов.

1. В новый файл перенести матрицы, у которых все элементы (в порядке обхода строк) образуют упорядоченную по возрастанию последовательность.
2. В новый файл перенести матрицы, у которых все элементы (в порядке обхода столбцов) образуют упорядоченную по возрастанию последовательность.
3. Перенести каждую матрицу в новый файл, предварительно упорядочив каждую ее строку в порядке возрастания элементов.
4. В новый файл перенести матрицы, с нормой меньше единицы. Вычислить норму по формуле  (норма Фробениуса).
5. Дан бинарный файл квадратных матриц. Исключить из файла все матрицы, содержащие хотя бы одну нулевую строку или нулевой столбец.

● Разработать и использовать указанные функции в контексте поставленной задачи.

1. Описать функцию getInt(Name, k) целого типа, возвращающую *k*-й элемент текстового файла целых чисел с именем *Name* (числа нумеруются от нуля). Если файл не существует или не содержит *k*-го элемента, то функция возвращает ноль. С помощью этой функции вывести пять элементов данного файла с указанными номерами.
2. Описать функцию *getLine*(*Name, k*), возвращающую *k*-ю строку текстового файла с именем *Name* (строки нумеруются от нуля и разделяются символами возврата каретки и перехода на новую строку). Если файл не существует или не содержит *k*-й строки, то функция возвращает пустую строку. С помощью этой функции вывести пять строк данного файла с указанными номерами.
3. Описать функцию *TextSize*(*Name*) целого типа, возвращающую число строк (строки нумеруются от нуля и разделяются символами возврата каретки и перехода на новую строку) в текстовом файле с именем *Name*. Если файл не существует, то функция возвращает (–1). С помощью этой функции определить размер трех файлов с данными именами.
4. Описать функцию *InvertIntFile(Name)*, меняющую порядок следования элементов файла целого типа с именем *Name* на противоположный. Если файл не существует или содержит менее двух элементов, то процедура не выполняет никаких действий. Обработать с помощью этой процедуры три файла с данными именами.
5. Описать функцию *SplitIntFile*(*Name*0*, k, Name*1*, Name*2), помещающую первые *k* ≥ 0 элементов существующего файла целых чисел с именем *Name*0 в файл *Name*1, а остальные элементы — в файл *Name*2 (прежнее содержимое результирующих файлов стирается). Один из результирующих файлов может оказаться пустым. Применить эту функцию к файлу *Name*0, используя указанные значения *Name*1, *Name*2 и *k*.
6. Описать функцию *SplitText*(*Name*0*, k, Name*1*, Name*2), копирующую первые *k* ≥ 0 строк существующего текстового файла с именем *Name*0 в файл *Name*1, а остальные элементы — в файл *Name*2 (прежнее содержимое результирующих файлов стирается). Один из результирующих файлов может оказаться пустым. Применить эту функцию к файлу *Name*0, используя указанные значения *Name*1, *Name*2 и *k*.
7. Описать функцию *ConcatFile*(*NameA, NameB, NameAB*), позволяющую объединить содержимое двух двоичных файлов *NameA* и *NameB* одного и того же типа в новом файле *NameAB*. Применить эту функцию к парам исходных файлов *Name*1–*Name*2, *Name*1–*Name*3 и *Name*2–*Name*3, создав файлы с именами *Name*12, *Name*13, *Name*23.

### 13. Динамические линейные списки

**Требования к написанию и оформлению кода программы**

Работу с динамическим списком организовать в виде отдельной библиотеки. Основная функция должна содержать в себе только вызов функций для работы со списком.

Если в задании не указан тип списка, необходимо выбрать его оптимальное представление.

Если в условии сказано создать новый список, то этот список должен располагаться в другой области памяти по сравнению с исходным. И наоборот, если требуется изменить список или часть его, то список должен располагаться в той же памяти, где находился исходно.

**Задачи**

1. На вход подается программа на языке С. Используя стек, проверить корректность расположения фигурных скобок в программе.
2. Построить стек. Элемент, расположенный на глубине *n*, перевести на верхушку стека.
3. Построить стек. Верхушечный элемент «утопить» на глубину *n*.
4. Построить двунаправленный список. Исключить все повторения наибольшего элемента.
5. Построить двунаправленный список. Затем *m* его элементов, начиная с *k*-го элемента, выделить в новый список. Напечатать оба списка.
6. Построить двунаправленный список *L*, включив в него по одному разу элементы, которые входят и в список *L*1 и в список *L*2. Напечатать все списки.
7. Многочлен *P*(*x*)=*an*·*xn*+*an*-1·*xn*-1+…+*a*1·*x*+*a*0 с вещественными коэффициентами представить в виде списка (звено такого списка содержит коэффициент *ai* и *i* – показатель степени *x*), причем, если *ai*=0, то соответствующее звено в список не включается. Описать операцию сложения двух многочленов. Результат (многочлен) напечатать в традиционном виде.
8. Построить двунаправленный список чисел. Выполнить проверку упорядоченности списка. Если он не упорядочен, то упорядочить его по возрастанию элементов.
9. Построить список. Затем *m* копий наибольшего элемента поместить в начало списка. Напечатать список до и после преобразования.
10. Построить двунаправленный список строк. Затем выделить из него в новый список все симметричные строки. Напечатать оба списка.
11. Построить список. Самую длинную серию одинаковых чисел выделить в отдельный список.
12. Построить два двунаправленных списка чисел, упорядоченных по возрастанию. Затем объединить их в один упорядоченный по возрастанию список. Напечатать все списки.
13. Построить однонаправленный список чисел. Затем последнюю серию отрицательных чисел расположить после первого элемента. Напечатать список до и после преобразования.
14. Построить двунаправленный список *L*, включив в него по одному разу элементы, которые входят в список *L*1, но не входят в список *L*2. Напечатать все списки.
15. Построить однонаправленный список чисел. Затем изменить его. Из каждой группы подряд идущих равных элементов оставить только один. Напечатать список до и после преобразования.
16. Построить два однонаправленных списка чисел. Один из них упорядочен по возрастанию, другой – по убыванию. Объединить их в один упорядоченный по возрастанию список. Напечатать все списки.
17. Построить однонаправленный список чисел. Затем исключить из него все повторы чисел, оставить каждое число в списке в единственном экземпляре. Напечатать список до и после преобразования.
18. Построить два двунаправленных списка чисел *L*1 и *L*2. Проверить, входит ли список *L*2 в список *L*1.
19. Построить двунаправленный список строк. Затем исключить из него все строки, являющиеся инверсией первой строки. Напечатать список до и после преобразования.
20. Построить однонаправленный список чисел. Затем в нем все отрицательные числа перенести в конец списка. Взаимный порядок следования среди отрицательных чисел не изменять. Напечатать список до и после преобразования.
21. Построить двунаправленный список строк. Затем все строки-палиндромы перенести в начало списка. Напечатать список до и после преобразования.
22. Построить однонаправленный список чисел. Выполнить переворот списка. Напечатать список до и после преобразования.
23. Построить кольцевой однонаправленный список фамилий ребят, участвующих в считалке. Начав отсчет от первого, удалять каждого *k*-го. Определить порядок удаления ребят из круга.
24. Многочлен *P*(*x*)=*an*·*xn*+*an*-1·*xn*-1+…+*a*1·*x*+*a*0 с вещественными коэффициентами представить в виде списка (звено такого списка содержит коэффициент *ai* и *i* – показатель степени *x*), причем если *ai*=0 , то соответствующее звено в список не включается. Написать программу, рассчитывающую производную многочлена. Результат (многочлен) напечатать в традиционном виде.
25. Многочлен *P*(*x*)=*an*·*xn*+*an*-1·*xn*-1+…+*a*1·*x*+*a*0 с вещественными коэффициентами представить в виде двунаправленного списка (звено такого списка содержит коэффициент *ai* и *i* – показатель степени *x*), причем если *ai*=0 , то соответствующее звено в список не включается. Написать программу, вычисляющую значение многочлена для заданного значения *x* (за основу алгоритма принять схему Горнера).
26. Два многоразрядных числа представить в виде двунаправленных списков. Выполнить сложение этих чисел.
27. Два многоразрядных числа представить в виде двух двунаправленных списков. Выполнить вычитание этих чисел.
28. Два многоразрядных числа представить в виде двух двунаправленных списков. Выполнить умножение этих чисел.
29. Построить двунаправленный список *L*. Преобразовать список, исключив из него часть, расположенную между наибольшим и наименьшим элементами.

### 14. Деревья

**Требования к написанию и оформлению кода программы**

Работу с деревьями организовать в виде отдельной библиотеки. Основная функция должна содержать в себе только вызов функций для работы с деревьями.

**Задачи**

1. Создать идеально сбалансированное дерево. Напечатать его. Определить количество отрицательных ключей в дереве.
2. Создать идеально сбалансированное дерево. Напечатать ключи дерева в префиксном, инфиксном и постфиксном порядках.
3. Создать дерево поиска. Напечатать его. Проверить, не является ли оно одновременно и идеально сбалансированным.
4. Создать идеально сбалансированное дерево. Напечатать его. Определить значение наибольшего ключа в дереве.
5. Создать идеально сбалансированное дерево. Напечатать его. Проверить, не является ли оно одновременно и деревом поиска.
6. Создать идеально сбалансированное дерево. Напечатать его. Определить высоту дерева.
7. Создать идеально сбалансированное дерево. Напечатать его. Поменять местами наибольший и наименьший ключи в дереве.
8. Создать идеально сбалансированное дерево. Напечатать его. Определить, в каком из поддеревьев больше сумма ключей – в левом или правом.
9. Создать дерево поиска. Напечатать его. Определить, в каком из поддеревьев больше четных ключей – в левом или правом.
10. Создать дерево поиска. Напечатать его. Определить ключ с наибольшей частотой появления в дереве.
11. Создать дерево поиска. Напечатать его. Создать его копию.
12. Создать идеально сбалансированное дерево. Напечатать его. Скопировать дерево в стек.
13. Создать стек. Скопировать его в дерево поиска.
14. Создать два дерева. Проверить, совпадают ли они.
15. Создать сбалансированное дерево. Напечатать его. Определить высоту дерева.
16. Создать идеально сбалансированное дерево. Напечатать его. Проверить, есть ли в дереве хотя бы два одинаковых ключа.
17. Создать идеально сбалансированное дерево. Напечатать его. Скопировать дерево в двусвязный список.
18. Создать дерево поиска. Напечатать его. Определить число ключей на *k*-ом уровне дерева.
19. Создать бинарное дерево. Определить в нем количество листьев.
20. Создать бинарное дерево. Определить количество нетерминальных вершин.
21. Создать бинарное дерево. Определить количество вершин, имеющих ровно два поддерева.
22. Создать бинарное дерево. Определить, все ли листья находятся на одном уровне.
23. Уровень корня дерева равен нулю. Уровень любого другого узла всегда больше на единицу, чем уровень узла, ссылающегося на данный. Глубиной (или высотой) дерева называется уровень того узла, чей уровень максимален. Длина внутреннего пути по дереву есть общее число ребер, находящихся в дереве. Написать программу, которая строит двоичное дерево и определяет уровень указанного узла этого дерева, а также глубину дерева и длину его внутреннего пути.
24. Написать программу, которая уничтожает все узлы двоичного дерева.
25. Написать программу, которая в каждый узел дерева дополнительно записывает количество потомков этого узла.
26. Написать программу, которая создает новое двоичное дерево, являющееся зеркальным отображением заданного (т.е. все левые поддеревья становятся правыми и наоборот).

### Содержание

[1. Линейные алгоритмы 3](#_Toc353188872)

[2. Условный оператор 5](#_Toc353188873)

[3. Ряды 7](#_Toc353188874)

[4. Рекуррентные последовательности 14](#_Toc353188875)

[5. Целые числа 18](#_Toc353188876)

[6. Системы счисления 22](#_Toc353188877)

[7. Массивы 24](#_Toc353188878)

[8. Матрицы 32](#_Toc353188879)

[9. Строки 41](#_Toc353188880)

[10. Записи. 46](#_Toc353188881)

[11. Рекурсия. 50](#_Toc353188882)

[12. Организация работы с файлами в С/С++ 58](#_Toc353188883)

[13. Динамические линейные списки 63](#_Toc353188884)

[14. Деревья 66](#_Toc353188885)

|  |  |  |  |
| --- | --- | --- | --- |
| Редактор З.И. Сныкова | | | |
| Компьютерная верстка А.А. Трийченко | | | |
| **ЛР N 020713 от 27.04.1998** |  |  | |
| Подписано к печати 17.06.2013 | Формат бумаги 60×84/16 | | |
| Печать ризограф | Бумага SvetoCopy | | Печ. л. 4 |
| Заказ № | Тираж 67 экз. | | Цена договорная |
| Отдел множительной техники ИАТЭ | | | |
| 249040, г. Обнинск, Студгородок, 1. | | | |