# А.А.Быков

# Сборник задач по программированию с решениями

## Общие положения

Экзамен по курсу «Программирование на языке C++» проводится в терминальном классе с установленным программным обеспечением, Microsoft Visual Studio или Borland C++ Builder по выбору экзаменуемого. Экзаменационный билет содержит три задания.

1) [1] Простое задание, для выполнения которого достаточно знакомства с основными конструкциями языка C++, в том числе иметь понятие об основных типах объектов, знать арифметические, логические и условные операторы if, операторы цикла for, одномерные и двумерные массивы с постоянной размерностью. Время выполнения: 30 минут.

2) [2] Задание средней сложности, для выполнения которого достаточно знания всех конструкций языка C++, включая одномерные и двумерные массивы с переменными границами (операторы new и delete), логические операторы, оператор switch, операторы цикла while и do, умение работать с текстовыми переменными и константами, знать основные стандартные функции для работы со строками, уметь выводить информацию в файл и считывать информацию из файла с помощью операторов << и >>. Время выполнения: 60 минут.
3) [3] Сложное задание, для выполнения которого требуется свободное владение всеми конструкциями языка в рамках программы первого семестра обучения, а также умение составить простейшую модель объекта. Время выполнения: 90 минут.

Общая продолжительность экзамена составляет не более 180 минут в терминальном классе. Студент, выполнивший первое задание, получает оценку не ниже 3. Студент, выполнивший второе задание, получает оценку не ниже 4. Студент, выполнивший третье задание, получает оценку 5. Во время выполнения заданий студент может использовать любые пособия, а также свои записи лекций и практических занятий. Можно использовать также любые файлы (в том числе C++ файлы), которые заранее должны быть сохранены на локальном диске в рабочей директории студента. В случае необходимости студент может обратиться за помощью к преподавателю. Преподаватель помогает студенту исправить ошибки в коде. В этом случае оценка 5 не ставится. Если курсант обращается за помощью несколько раз, не ставится оценка выше 3.

Список типовых задач объявляется заранее не позже чем за месяц до экзамена. Типовые задания разбираются на лекциях и отрабатываются на практических занятиях. Первое (простое) задание на экзамене может незначительно отличаться от типового задания из данного списка. Второе (средней сложности) задание также выбирается из данного списка и видоизменяется так, чтобы ход его решения ненамного отличался от типового. Например, если в типовом задании сказано, что оценка каждого курсанта потока за каждую неделю обучения в течение семестра хранится в двумерном массиве и требуется найти среднюю оценку каждого курсанта за весть семестр, то на экзамене может быть поставлена задача: найти среднюю оценку всего потока за каждую неделю обучения.





Третье задание (сложное) будет похоже на одно из отработанных на лекциях или ПЗ заданий, но будет содержать новые элементы, которые студент должен разработать самостоятельно.

Задания из этого списка следует выполнять в среде Microsoft Visual Studio или Borland C++ Builder. При создании проекта указывайте опции “Win32 application” и “Console”. Вывод в файл осуществляйте с помощью оператора myofs<<, ввод из файла – с помощью оператора myifs>>, где myofs – объект типа ofstream, связанный с файлом с помощью оператора myofs.open(“имя файла”), myifs – объект типа ifstream, связанный с файлом с помощью оператора myifs.open(“имя файла”). После завершения работы с файлом не забудьте его закрыть, например, myofs.close(). Ввод с клавиатуры (при необходимости) осуществляйте с помощью оператора cin>>…, а вывод на экран cout<<…

В тех задачах, в которых форма отчета не определена явно, предполагается, что результаты направляются в файл protocol.txt.

Следует иметь в виду, что разделение задач по темам является условным, так как для решения некоторых задач придется обращаться к разным разделам курса программирования.

Задачи с решениями помечены символом +, например [+2].

Исходные тексты можно найти по адресу boombook@narod.ru

# Часть 1, C++ без классов

## 1. Простейшие функции

### Задачи

1. [+1] Определите функцию double f(double x, double y), которая вычисляет и возвращает длину гипотенузы прямоуголь-

ного треугольника, две другие стороны x и y которого известны.

2. [+1] Напишите функцию double f(double x1, double y1, double x2, double y2), которая вычисляет расстояние между двумя точками (xl, yl) и (x2, y2).
3. [+1] Напишите функцию int f(int m2, int m1, int m0), которая вычисляет и возвращает натуральное число, первая (сотни), вторая (десятки) и третья (единицы) цифры которого равны соответственно m2, m1, m0.
4. [+1] Напишите функцию int f(int m2, int m1, int m0), в которой заданы значения всех трех параметров по умолчанию. Эта функция вычисляет и возвращает натуральное число, первая (сотни), вторая (десятки) и третья (единицы) цифры которого равны соответственно m2, m1, m0. Требуется определить функцию так, чтобы после выполнения оператора x=f() значение x равнялось 567.
5. [+1] Напишите несколько функций с одним именем int f(…) и с разными наборами параметров. Продемонстрируйте работу перегруженных функций.
6. [+1] Операторы `int u=f(3, 4); double v=f(20.5, 10.5); myofs<<"u="<<u<<" v="<<v<<endl;` дают в результате `u=7 v=10` (т.е. соответственно сумму и произведение своих параметров). Напишите функции f(…,…).
7. [1] Напишите функцию f(double& a, double& b, double c, double q), которая возвращает катеты прямоугольного треугольника, гипотенуза которого равна c, а острый угол q (градусов).
8. [1] Напишите функцию f(double* a, double* b, double c, double q), которая возвращает катеты прямоугольного треугольника, гипотенуза которого равна c, а острый угол q (градусов).





9. [+1] Напишите функцию swap(int* a, int* b), которая изменяет значения параметров a и b так, что новое значение a равно старому значению b и наоборот.
10. [+1] Напишите функцию swap(int& a, int& b), которая изменяет значения параметров a и b так, что новое значение a равно старому значению b и наоборот.
11. [1] Напишите функцию swap(int* a, int* b, int* c), которая изменяет значения параметров по правилу a→b→c→a.
12. [1] Напишите функцию swap(int& a, int& b, int& c), которая изменяет значения параметров по правилу a→b→c→a.
13. [1] Напишите функцию int f(int& a, int& b, int x, int y), которая присваивает объекту a значение, равное x*y, а объекту b значение x/y. Если значение y было равно нулю, функция должна возвратить 0, иначе 1.
14. [1] Напишите функцию int f(int x, int y), которая возвращает 0, если значения x и y оба равны нулю, `12/x`, если y равен 0, `12/y`, если x равен 0, иначе `144/(x*y)`.
15. [1] Напишите функцию int f(int x, int y), которая возвращает `x-y`, если x больше y, иначе возвращает `y-x`.
16. [1] Напишите функцию double f(double x, double y), которая возвращает `x/y`, если x больше y, иначе возвращает `y/x`. Предполагается, что значения параметров больше нуля.
17. [1] Напишите функцию double f(double x, double y, double z), которая возвращает m*n/k, где k есть меньшее из чисел x, y, z, а m и n есть среднее и большее из этих чисел. Предполагается, что значения параметров больше нуля.
18. [1] Напишите функцию double f(double x, double y, double z), которая возвращает наибольшее из значений `|x-y|`, `|y-z|`, `|z-x|`.
19. [1] Напишите функцию int f(int a, int b, int c), которая возвращает наименьшее из значений a, b, c.

20. [1] Напишите функцию int f(int a, int b, int c), которая возвращает наибольшее из значений a, b, c.
21. [1] Напишите функцию int f(int* a, int* b, int* c), которая возвращает наименьшее из значений объектов a, b, c.
22. [1] Напишите функцию bool f(int x, int y, int z), которая возвращает `true`, если $x^2 + y^2 = z^2$, иначе возвращает `false`.
23. [1] Напишите функцию bool f(int x, int y), которая возвращает `true`, если x делится нацело на y, или наоборот, y делится нацело на x, иначе возвращает `false`. Предполагается, что значения параметров больше нуля.
24. [1] Положительные числа x, y, z могут быть сторонами треугольника, если большее из них меньше суммы двух других. Напишите функцию bool f(int x, int y, int z), которая возвращает `true`, если числа x, y, z могут быть сторонами треугольника, иначе возвращает `false`. Предполагается, что значения параметров больше нуля.

# Решения

## Вычисление гипотенузы треугольника

[1] Определите функцию double f(double x, double y), которая вычисляет и возвращает длину гипотенузы прямоугольного треугольника, когда две другие стороны x и y известны.

**Решение.**

```
double f(double x, double y)
{
  // Вычислим гипотенузу прямоугольного треугольника:
  return sqrt(x*x+y*y);
}
{
  myofs << " Вычислим гипотенузу, " << endl;
  double x = 2+rand()%20;
  double y = 2+rand()%20;
  double z = f(x,y);
  myofs << " x=" << x << ", y=" << y <<
    ", Гипотенуза(x,y)=" << z << endl;
```





```
}
```

**Результат в файле протокола:**

```
Вычислим гипотенузу, x=8, y=3, Гипотенуза(x,y)=8.544
```

## Вычисление расстояния между двумя точками

25. [+1] Напишите функцию double f(double x1, double y1, double x2, double y2), которая вычисляет расстояние между двумя точками (xl, yl) и (x2, y2).

**Решение.**

```
double f(double x1, double y1, double x2, double y2)
{
  return sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
}
{
  myofs << "Вычислим расстояние, " << endl;
  double R = f(3.0, 7.0, 7.0, 10.0);
  myofs << " x1-x2=" << x1-x2 << ", y1-y2=" << y1-y2
    << ", Расстояние R=" << R << endl;
}
```

**Результат в файле протокола:**

```
 Вычислим расстояние, x1-x2=-4, y1-y2=-3, R=5
```

## Перегруженные функции

[+1] Напишите несколько функций с одним именем int f(…) и с разными наборами параметров. Продемонстрируйте работу перегруженных функций.

**Решение.**

```
int f32(int x){return x;}
double f32(double x){return x*x;}
int f32(double x, int y){return x*y;}
char* f32(char* x){return x;}
int f32(int x, int y){return x+y;}
{
  myofs << "Перегруженные функции" << endl;
  myofs << "Знач f32(7) равно " << f32(7) << endl;
  myofs << "Знач f32(7.0) равно " << f32(7.0) << endl;
  myofs << "f32(строка)"<<f32("Sample f32")<< endl;
  myofs << "Знач f32(7,8) равно " << f32(7, 8) << endl;
```

```
  myofs << "Знач f32(7) равно " << f32(7) << endl;
  myofs << "Знач f32(8.0,7) равно " << f32(8.0, 7);
  myofs << endl << endl;
}
```

**Результат в файле протокола:**

```
Перегруженные функции
 Значение f32(7) равно 7
 Значение f32(7.0) равно 49.000000
 Значение f32(строка) равно Sample f32
 Значение f32(7, 8) равно 15
 Значение f32(7) равно 7
 Значение f32(8.0, 7) равно 56
```

### Функция swap(…) с передачей указателей

### Функция swap(…) с передачей параметров по ссылке

[+1] Напишите функцию swap(int* a, int* b), которая изменяет значения параметров a и b так, что новое значение a равно старому значению b и наоборот.

[+1] Напишите функцию swap(int& a, int& b), которая изменяет значения параметров a и b так, что новое значение a равно старому значению b и наоборот.

**Решение.**

```
/* Мы представим вариант с перегруженными функциями.
Если требуется определить несколько функций, которые
выполняют одну и ту же операцию с параметрами разных
типов, то перегруженные функции дают оптимальное
решение. */
int dswp(double* x, double* y);
int dswp(double& x, double& y);
int main(){
  {
  myofs << "Функция swap(double*, double*)" << endl;
  double x = 123.456, y=987.654;
  myofs << "Начальные  x=" << x << "  y=" << y << endl;
  dswp(&x, &y);
  myofs<<"После dswp(&x, &y) x="<<x<<"  y="<<y<<endl;
  }
  {
  myofs << "Функция swap(double&, double&)" << endl;
```





```
  double x = 123.456, y=987.654;
  myofs << "Начальные  x=" << x << "  y=" << y << endl;
  dswp(x, y);
  myofs<<"После dswp(x, y) x="<<x<<"  y="<<y<<endl;
  }
  return 0;
}
int dswp(double* x, double* y)
{
  double z=*x;
  *x=*y;
  *y=z; return 1;
}
int dswp(double& x, double& y){
  double z=x;
  x=y;
  y=z; return 0;
}
```

**Результат в файле протокола:**

```
Функция swap(double*, double*)
Начальные значения        x=123.456000  y=987.654000
После вызова dswp(&x, &y) x=987.654000  y=123.456000
Функция swap(double&, double&)
Начальные значения        x=123.456000  y=987.654000
После вызова dswp(x, y)   x=987.654000  y=123.456000
```

## 2. Рекурсивные функции

## З а д а ч и

26. [2] Напишите нерекурсивную функцию int f(int n), которая находит число Фибоначчи $f_n$. Числа Фибоначчи определяются соотношениями $f_1 = 1$, $f_2 = 1$, $f_n = f_{n-1} + f_{n-2}$.

27. [+2] Напишите рекурсивную функцию int f(int n), которая находит число Фибоначчи $f_n$. Числа Фибоначчи определяются соотношениями $f_1 = 1$, $f_2 = 1$, $f_n = f_{n-1} + f_{n-2}$.

## Р е ш е н и я

### Рекурсивные функции

[2] Напишите рекурсивную функцию int f(int n), которая находит число Фибоначчи $f_n$. Числа Фибоначчи определяются соотношениями $f_1 = 1,\ f_2 = 1,\ f_n = f_{n-1} + f_{n-2}$.

**Решение.**

```
int f35(int x)
{
  counter++;
  if(x<=1) return 1;
  if(x==2) return 1;
  return f35(x-1) + f35(x-2);
}
{
  myofs << "Упражнение 5. Рекурсия" << endl;
  counter=0; int m3=f35(3);
  myofs << "f35(3)=" <<  m3 <<"co="<<counter << endl;
  counter=0; int m4=f35(4);
  myofs << "f35(4)=" <<  m4 <<"co="<<counter << endl;
  counter=0; int m5=f35(5);
  myofs << "f35(5)=" <<  m5 <<"co="<< counter << endl;
  counter=0; int m6=f35(6);
  myofs << "f35(6)=" <<  m6 <<"co="<< counter << endl;
  counter=0; int m7=f35(7);
  myofs << "f35(7)=" <<  m7 <<"co="<< counter << endl;
  counter=0; int m10=f35(10);
  myofs << "f35(10)=" << m10<<"co="<< counter << endl;
  counter=0; int m20=f35(20);
  myofs << "f35(20)=" << m20<<"co="<< counter << endl;
  counter=0; int m30=f35(30);
  myofs << "f35(30)=" << m30<<"co=" << counter << endl;
  counter=0; int m35=f35(35);
  myofs << "f35(35)=" << m35<<"co=" <<counter << endl;
}
```

**Результат в файле протокола:**

```
Рекурсия
 Значение f35(3) равно 2  counter= 3
 Значение f35(4) равно 3  counter= 5
```





```
Значение f35(5) равно 5  counter= 9
Значение f35(6) равно 8  counter= 15
Значение f35(7) равно 13  counter= 25
Значение f35(10) равно 55  counter= 109
Значение f35(20) равно 6765  counter= 13529
Значение f35(30) равно 832040  counter= 1664079
Значение f35(35) равно 9227465  counter= 18454929
```

## 3. Десятичные целые числа

## З а д а ч и

28. [+1] Напишите функцию int f(int h, int m, int s), которая принимает три целых аргумента (часы h, минуты m и секунды s) и возвращает количество секунд, прошедших с начала дня.
29. [1] Напишите функцию int f(int m, int d), которая принимает два целых аргумента (месяц m и день d) и возвращает количество дней, прошедших с начала года. Считаем, что в каждом месяце 30 дней.
30. [1] Напишите функцию int f(int m, int d), которая принимает два целых аргумента (месяц m и день d) и возвращает количество дней, прошедших с начала года. Считаем, что в каждом месяце 30 или 31дней, а год не високосный.
31. [+1] Напишите функцию f(int& m1, int& m0, int N), которая возвращает первую и последнюю цифры двузначного натурального числа N.
32. [+1] Напишите функцию f(int* m1, int* m0, int N), которая возвращает первую и последнюю цифры двузначного натурального числа N.
33. [1] Напишите функцию int f(int m1, int m2, int m3), которая находит наименьшее число из заданного набора трех целых чисел. Используйте условный оператор if.
34. [1] Напишите функцию int f(int m1, int m2, int m3) с аргументами по умолчанию, которая находит сумму одного, двух или трех целых чисел (находящихся в пределах от 1 до 999). Используйте условный оператор if.

35. [1] Напишите функцию int f(int m1, int m2, int m3) с аргументами по умолчанию, которая находит наименьшее из одного, двух или трех целых чисел (находящихся в пределах от 1 до 999). Используйте условный оператор if.
36. [1] Напишите функцию bool even(int n), определяет четность числа n и возвращает true для четного числа и false для нечетного числа.
37. [1] Напишите функцию f(int m, int n), которая определяет для пары целых чисел m и n, кратно ли второе число первому.
38. [1] Напишите функцию int f(m, n), которая вычисляет и возвращает сумму всех натуральных чисел от m до n включительно.
39. [1] Напишите функцию int f(int& S, int M, int N), которая суммирует последовательность целых чисел от M до N включительно. Функция должна возвратить 1 при условии M<=N и 0 в противном случае, и при условии M<=N находить и возвращать сумму по ссылке.
40. [1] Напишите функцию int f(m, n), которая вычисляет и возвращает произведение всех натуральных чисел от m до n включительно.
41. [1] Напишите функцию int fact(int m), которая вычисляет и возвращает факториал положительного числа m, т.е. произведение всех натуральных чисел, меньших или равных m.
42. [+2] Напишите функцию f(int& m2, int& m1, int& m0, int N), которая возвращает первую, вторую и третью цифры трехзначного натурального числа N.
43. [+2] Напишите функцию f(int& m4, int& m3, int& m2, int& m1, int& m0, int N), которая возвращает все цифры пятизначного натурального числа N.
44. [+2] Напишите функцию f(int* arr, long N), которая возвращает все цифры пятизначного натурального числа N.
45. [+2] Напишите функцию f(int& h, int& m, int& s, int sec), которая принимает количество секунд, прошедших с начала





дня, и возвращает три целых переменных (часы, минуты и секунды).

46. [+2] Напишите функцию int f(int x), которая принимает целое значение (не больше пяти знаков) и возвращает число с обратным порядком цифр. Например, принимается число 7631, возвращается число 1367.
47. [+2] [1]Палиндром – число или текст, который одинаково читается слева направо и справа налево. Например, каждое из следующих пятизначных целых чисел является палиндромом: 12321, 55555, 45554 и 11611. Текст–палиндром: **Ешь немытого ты меньше.** Напишите функцию bool f(int N), которая определяет, является ли пятизначное целое число N палиндромом.
48. [+2] Напишите функцию int f(int* arr, int M, int N), которая составляет таблицу палиндромов в пределах от M до N и возвращает их число.
49. [2] Наибольший общий делитель (НОД) двух целых чисел — это наибольшее целое, на которое без остатка делится на каждое из двух чисел. Напишите функцию int f(int m, int n), которая возвращает НОД двух целых чисел m и n.
50. [2] Наименьшее общее кратное (НОК) двух целых чисел — это наименьшее целое, которое без остатка делится на каждое из двух чисел. Напишите функцию int f(int m, int n), которая возвращает НОК двух целых чисел m и n.
51. [2] Говорят, что целое число является совершенным числом, если его сомножители, включая 1 (но не само число) в сумме дают это число. Например, 6 – совершенное число, так как 6=1+2+3. Напишите функцию bool f(int n), которая опре-

[1] Коллекцию палиндромов можно посмотреть здесь. [doc] [pdf] http://sheba.spb.ru/links-audioknigi.htm
Степень благозвучности палиндромов иногда оставляет желать лучшего, поэтому читатетеля с недостаточно крепкими нервами прошу по этой ссылке не переходить.

деляет, является ли ее параметр n совершенным числом. Используйте эту функцию в программе, которая определяет и печатает все совершенные числа в диапазоне от 1 до 1000.

52. [2] Простое число делится нацело только на 1 и на само себя. Напишите функцию bool f(int n), которая возвращает true, если n – простое число и false в противном случае. Составьте таблицу и подсчитайте количество простых чисел от 2 до 1000.

# Решения

## Вычисление времени в секундах

[+1] Напишите функцию int f(int h, int m, int s), которая принимает три целых аргумента (часы h, минуты m и секунды s) и возвращает количество секунд, прошедших с начала дня.

**Решение.**

```
int f08(int h, int m, int s)
{
  // Сформируем время из часов, минут, секунд:
  return (h*60+m)*60+s;
}
int main()
{
  myofs << "Вычисляем время." << endl;
  int h = rand()%24;
  int m = rand()%60;
  int s = rand()%60;
  int tim = f08(h, m, s);
  myofs << " h=" << h << ", m=" << m
        << ", s=" << s
        <<", f08(h,m,s)=" << tim << "."
        << endl << endl;
  return 0;
}
```

**Результат в файле протокола:**

```
Упражнение 08.0108 Вычисляем время.
 h=6, m=7, s=14, f08(h,m,s)=22034.
```





## Функция с параметрами по умолчанию

[+1] Напишите функцию int f(int m2, int m1, int m0), в которой заданы значения всех трех параметров по умолчанию. Эта функция вычисляет и возвращает натуральное число, первая (сотни), вторая (десятки) и третья (единицы) цифры которого равны соответственно m2, m1, m0.

**Решение.**

```
int f31(int x=5, int y=6, int z=7)
  {return 100*x+10*y+z;}
int main(){
  myofs << "Функция с параметрами по умолчанию";
  myofs << "Значение f31() равно " << f31() << endl;
  myofs << "Значение f31(1) равно " << f31(1) << endl;
  myofs << "Значение f31(1,2) равно "<<f31(1,2)<< endl;
  myofs << "Значение f31(1,2,3) равно "
        << f31(1, 2, 3) << endl << endl;
  return 0;
}
```

**Результат в файле протокола:**

```
Функция с параметрами по умолчанию
 Значение f31() равно 567
 Значение f31(1) равно 167
 Значение f31(1, 2) равно 127
 Значение f31(1, 2, 3) равно 123
```

## Вычисление значения натурального числа.

[+1] Напишите функцию int f(int m2, int m1, int m0), которая вычисляет и возвращает натуральное число, первая (сотни), вторая (десятки) и третья (единицы) цифры которого равны m2, m1, m0.

**Решение.**

```
int f02(int x2, int x1, int x0)
{
  // Сформируем трехзначное число из его цифр:
  return x2*100+x1*10+x0;
}
{
  myofs << "Вычисляем трехзначное число"
        << endl;
```

```
  int x2 = rand()%10;
  int x1 = rand()%10;
  int x0 = rand()%10;
  int m = f02(x2, x1, x0);
  myofs << " x2=" << x2 << ", x1=" << x1
    << ", x0=" << x0 << ", f02(x2,x1,x0)="
    << m << "." << endl << endl;
}
```

**Результат в файле протокола:**

```
Вычисляем трехзначное число.
x2=6, x1=7, x0=9, f02(x2,x1,x0)=679.
```

## Вычисление цифр двузначного числа через ссылки

[+1] Напишите функцию f(int& m1, int& m0, int N), которая возвращает первую и последнюю цифры двузначного натурального числа N.

**Решение.**

```
int f03(int& x1, int& x0, int x)
{
  // Расщепляем двузначное число на цифры:
  if(x<0) return -1;
  if(x>99) return 1;
  x0 = x%10;
  x1 = (x-x0)/10;
  return 0;
}
{
  myofs << "Расщепляем двузначное число на цифры." <<
endl;
  int x = 10+rand()%90;
  int x1, x0;
  int m = f03(x1, x0, x);
  myofs << " x=" << x << ", x1=" << x1 <<
     ", x0=" << x0 <<", f03(x2,x1,x)=" << m
     << "." << endl << endl;
}
```

**Результат в файле протокола:**

```
Расщепляем двузначное число на цифры.
 x=53, x1=5, x0=3, f03(x2,x1,x)=0.
```





## Вычисление цифр двузначного числа через указатели

[+1] Напишите функцию f(int* m1, int* m0, int N), которая возвращает первую и последнюю цифры двузначного натурального числа N.

**Решение.**

```
int f04(int* x1, int* x0, int x)
{
  // Расщепляем двузначное число на цифры:
  if(x<0) return -1;
  if(x>99) return 1;
  *x0 = x%10;
  *x1 = (x-*x0)/10;
  return 0;
}
{
  myofs << "Расщепляем двузначное число на цифры." <<
endl;
  int x = 10+rand()%90;
  int x1, x0;
  int m = f04(&x1, &x0, x);
  myofs << " x=" << x << ", x1=" << x1 <<
    ", x0=" << x0 <<", f04(x1,x0,x)=" << m
    << "." << endl << endl;
}
```

**Результат в файле протокола:**

```
Расщепляем двузначное число на цифры.
x=92, x1=9, x0=2, f04(x1,x0,x)=0.
```

## Выделение цифры многозначного числа

[+2] Напишите функцию f(int& m2, int& m1, int& m0, int N), которая возвращает первую, вторую и третью цифры трехзначного натурального числа N.

[+2] Напишите функцию f(int& m4, int& m3, int& m2, int& m1, int& m0, int N), которая возвращает все цифры пятизначного натурального числа N.

[+2] Напишите функцию f(int* arr, long N), которая возвращает все цифры пятизначного натурального числа N.

[+2] Напишите функцию f(int& h, int& m, int& s, int sec), которая принимает количество секунд, прошедших с начала дня, и возвращает три целых переменных (часы, минуты и секунды).

**Решение.**

```
int f05(int& x2, int& x1, int& x0, int x)
{
  // Расщепляем трехзначное число на цифры:
  if(x<0) return -1;
  if(x>999) return 1;
  x0 = x%10;
  x1 = ((x-x0)/10)%10;
  x2 = (x-10*x1-x0)/100;
  return 0;
}
int f06(int& x4, int& x3, int& x2, int& x1, int& x0,
int x)
{
  // Расщепляем пятизначное число на цифры:
  if(x<0) return -1;
  if(x>99999) return 1;
  x0 = x%10;
  x1 = ((x-x0)/10)%10;
  x2 = ((x-10*x1-x0)/100)%10;
  x3 = ((x-100*x2-10*x1-x0)/1000)%10;
  x4 = ((x-1000*x3-100*x2-10*x1-x0)/10000)%10;
  return 0;
}
int f07(int* arr, int x)
{
  // Расщепляем пятизначное число на цифры:
  if(x<0) return -1;
  if(x>99999) return 1;
  int x0 = x%10;
  int x1 = ((x-x0)/10)%10;
  int x2 = ((x-10*x1-x0)/100)%10;
  int x3 = ((x-100*x2-10*x1-x0)/1000)%10;
  int x4 =
    ((x-1000*x3-100*x2-10*x1-x0)/10000)%10;
  arr[0]=x4;
  arr[1]=x3;
  arr[2]=x2;
  arr[3]=x1;
```





```
  arr[4]=x0;
  return 0;
}
int f09(int& h, int& m, int& s, int x)
{
// Расщепляем время на часы, минуты, секунды:
  if(x<0) return -1;
  if(x>23*60*60+59*60+59) return 1;
  s = x%60;
  m = ((x-s)/60)%60;
  h = (x-60*m-s)/3600;
  return 0;
}
{
  myofs << "Расщепляем трехзначное число
           на цифры." << endl;
  int x = 100+rand()%900;
  int x2=0, x1=0, x0=0;
  int m = f05(x2, x1, x0, x);
  myofs << " x=" << x << ", x2=" << x2
    << ", x1=" << x1 << ", x0=" << x0
    << ", f05(x2,x1,x0,x)="
    << m << "." << endl << endl;
}
{
  myofs << "Расщепляем пятизначное число
           на цифры." << endl;
  int x = 10000*(1+rand()%9) + rand()%10000;
  int x4=0, x3=0, x2=0, x1=0, x0=0;
  int m = f06(x4, x3, x2, x1, x0, x);
  myofs << " x=" << x << ", x4=" << x4
    << ", x3=" << x3 << ", x2=" << x2
    << ", x1=" << x1 << ", x0=" << x0
    << ", f06(x4,x3,x2,x1,x0,x)="
    << m << "." << endl << endl;
}
{
  myofs << "Расщепляем пятизначное число
           на цифры." << endl;
  int x = 10000*(1+rand()%9) + rand()%10000;
  int arr[5];
  for(int n=0; n<5; n++) arr[n]=0;
  int m = f07(arr, x);
```

```
  myofs << " x=" << x;
  for(int n=0; n<5; n++)
    myofs << ", x" << n << "=" << arr[n];
    myofs << ", f07(arr,x)=" << m << "."
        << endl << endl;
}
{
  myofs << "Расщепляем время на часы, минуты,
            секунды." << endl;
  int tim =
    (rand()%24)*60*60 + (rand()%60)*60 +
     rand()%60;
  int h=0, m=0, s=0;
  int xx = f09(h, m, s, tim);
  myofs << " tim=" << tim << ", h=" << h
    << ", m=" << m << ", s=" << s
    << ", f09(h,m,s,tim)=" << xx << "." << endl;
  int tim1 = f08(h, m, s);
  myofs << " h=" << h << ", m=" << m
    << ", s=" << s <<", f08(h,m,s)="
    << tim1 << "." << endl << endl;
}
```

**Результат в файле протокола:**

```
Расщепляем трехзначное число на цифры.
  x=145, x2=1, x1=4, x0=5, f05(x2,x1,x0,x)=0.

Расщепляем пятизначное число на цифры.
  x=54370,      x4=5,      x3=4,      x2=3,      x1=7,      x0=0,
f06(x4,x3,x2,x1,x0,x)=0.

Расщепляем пятизначное число на цифры.
  x=47029, x0=4, x1=7, x2=0, x3=2, x4=9, f07(arr,x)=0.
Расщепляем время на часы, минуты, секунды.
  tim=28309, h=7, m=51, s=49, f09(h,m,s,tim)=0.
  h=7, m=51, s=49, f08(h,m,s)=28309.
```

## Изменение цифр десятичных чисел

53. [+2] Напишите функцию int f(int x), которая принимает целое значение (не больше пяти знаков) и возвращает число с обратным порядком цифр. Например, принимается число 7631, возвращается число 1367.





**Решение.**

```
int f10(int x)
{
  // Переставляем цифры пятизначного числа:
  if(x<10000) return -1;
  if(x>99999) return 1;
  int x0 = x%10;
  int x1 = ((x-x0)/10)%10;
  int x2 = ((x-10*x1-x0)/100)%10;
  int x3 = ((x-100*x2-10*x1-x0)/1000)%10;
  int x4 =
    ((x-1000*x3-100*x2-10*x1-x0)/10000)%10;
  return (((x0*10+x1)*10+x2)*10+x3)*10+x4;
}
bool f11(int x)
{
  // Проверяем, является ли заданное число x палиндро-
мом:
  int y = f10(x);
  if(x==y) return true;
  return false;
}
int f12(int* arr, int N1, int N2)
{
  // Найдем количество палиндромов в педелах от N1 до
N2:
  int m=0;
  for(int n=N1; n<N2; n++)
  if(f11(n))
  {
    arr[m]=n;
    m++;
  }
  return m;
}
{
  myofs << "Зеркально отражаем
            пятизначное число." << endl;
  int x = 10000*(1+rand()%9) + rand()%10000;
  int y = f10(x);
  myofs << " x=" << x
        << ", y=" << y << endl << endl;
```

```
}
```

### Проверка пятизначных палиндромов

[+2] Палиндром – число или текст, который одинаково читается слева направо и справа налево. Например, каждое из следующих пятизначных целых чисел является палиндромом: 12321, 55555, 45554 и 11611. Напишите функцию bool f(int N), которая определяет, является ли пятизначное целое число N палиндромом.

### Поиск пятизначных палиндромов

[+2] Напишите функцию int f(int* arr, int M, int N), которая составляет таблицу палиндромов в пределах от M до N и возвращает их число.

**Решение.**

```
{
  myofs << "Проверяем пятизначный палиндром."
        << endl;
  int x = 10000*(1+rand()%9) + rand()%10000;
  int y = f10(x);
  myofs << " x= " << x << ", y= "
        << y << ", f11(x)= " << f11(x)
        << endl;
  int x1 = 5445;
  int y1 = 5445;
  myofs << " x1=" << x1 << ", y1=" << y1
  << ", f11(x1)=" << f11(x1) << endl << endl;
  int x2 = 12321;
  int y2 = 12321;
  myofs << " x2=" << x2 << ", y2=" << y2
  << ", f11(x2)=" << f11(x2) << endl << endl;
}
```

**Решение.**

```
{
  myofs << "Ищем пятизначные палиндромы."
        << endl;
  const int N = 100000;
  int arr[N];
  int M = f12(arr, 80000, 85000);
```





```
  myofs << " range N= " << N << ", count M= "
        << M << endl;
  for(int m=0, s=0; m<M; m++)
  {
    myofs << " [" << setw(2) << m << "]="
          << setw(5) << arr[m] << " ";
    s+=(5+int(log10(double(arr[m]))));
    if(s>72)
    {
      myofs << endl;
      s=0;
    }
  }
  myofs << endl << endl;
}
```

**Результат в файле протокола:**

```
Зеркально отражаем пятизначное число.
   x=17767, y=76771
Проверяем пятизначный палиндром.
   x= 12045, y= 54021, f11(x)= 0
   x1=5445, y1=5445, f11(x1)=0
   x2=12321, y2=12321, f11(x2)=1
Ищем пятизначные палиндромы.
   range N= 100000, count M= 50
[ 0]=80008 [ 1]=80108 [ 2]=80208 [ 3]=80308 [ 4]=80408
[ 5]=80508 [ 6]=80608 [ 7]=80708 [ 8]=80808 [ 9]=80908
[10]=81018 [11]=81118  [12]=81218 [13]=81318 [14]=81418
[15]=81518 [16]=81618 [17]=81718 [18]=81818 [19]=81918
[20]=82028 [21]=82128 [22]=82228 [23]=82328 [24]=82428
[25]=82528 [26]=82628 [27]=82728 [28]=82828 [29]=82928
[30]=83038 [31]=83138 [32]=83238 [33]=83338 [34]=83438
[35]=83538 [36]=83638 [37]=83738 [38]=83838 [39]=83938
[40]=84048 [41]=84148 [42]=84248 [43]=84348 [44]=84448
[45]=84548 [46]=84648 [47]=84748 [48]=84848 [49]=84948
```

## 4. Условные операторы и циклы

## З а д а ч и

54. [1] Напишите программу, которая находит наименьшее число из заданного набора трех целых чисел. Используйте условный оператор if.

55. [+1] Напишите функцию, которая находит наименьшее число из заданного набора трех целых чисел. Используйте условный оператор if.
56. [1] Напишите программу, которая с помощью функции rand() ставит оценки в пределах от 2 до 5. В файл протокола направляйте слово “Неуд” при появлении 2, «Удовл» при появлении 3, и т.д. Используйте оператор switch.
57. [1] Напишите функцию int f(int M, int N), которая вычисляет и возвращает сумму всех нечетных целых чисел в пределах от M до N включительно. Используйте оператор for.
58. [1] Напишите функцию int f(int N, int n), которая вычисляет и возвращает наименьшее из чисел, больших или равных N, которое делится нацело на n. Используйте оператор while.
59. [2] Напишите функцию int f(int* arr, int n), которая суммирует последовательность целых чисел из массив int arr*=new int[N]. Полагайте, что первое прочитанное целое число указывает N, количество целых чисел, которые далее будут введены. Например, входная последовательность может иметь вид 5 123 234 345 467 543, где 5 показывает, что будет введено последовательно 5 чисел, которые надо суммировать.
60. [1] Напишите функцию int fmina(int* arr, int N), которая находит наименьшее из N целых чисел, которые находятся в массиве arr.
61. [2] Множество троек целых положительных значений сторон прямоугольного треугольника называется тройками Пифагора. Эти три стороны должны удовлетворять соотношению, по которому сумма квадратов двух сторон (катетов) равна квадрату третьей стороны (гипотенузы). Найдите все тройки Пифагора, в которых все стороны не больше 500. Используйте трижды вложенные циклы for.
62. [+2] Известно, что сейф открывается при правильном вводе кода из 3 цифр 0…9. Задайте код и затем откройте сейф, используя метод перебора с помощью нескольких операторов цикла for.





# Р е ш е н и я

## Вычисление наименьшего из трех чисел.

[+1] Напишите функцию, которая находит наименьшее число из заданного набора трех целых чисел. Используйте условный оператор if.

**Решение.**

```
int fmin(int x1, int x2, int x3)
{
  // Найдем наименьшее:
  int x=x1;
  if(x2<x) x=x2;
  if(x3<x) x=x3;
  return x;
}
{
  myofs << "Найдем наименьшее из трех чисел." << endl;
  int a1 = fmin(3, 5, 7);
  int a2 = fmin(7, 3, 5);
  int a3 = fmin(5, 7, 3);
  myofs << " fmin(3, 5, 7)=" << a1 << endl;
  myofs << " fmin(7, 3, 5)=" << a2 << endl;
  myofs << " fmin(5, 7, 3)=" << a3 << endl << endl;
}
```

**Результат в файле протокола:**

```
Упражнение +++1728. Найдем наименьшее из трех чисел.
 fmin(3, 5, 7)=3
 fmin(7, 3, 5)=3
 fmin(5, 7, 3)=3
```

## Подбор кода сейфа.

[+2] Известно, что сейф открывается при правильном вводе кода из 3 цифр 0…9. Задайте код и затем откройте сейф, используя метод перебора с помощью нескольких операторов цикла for.

**Решение.**

```
int mysafe(int& x1, int& x2, int& x3)
{
  // Зададим код сейфа:
  x1=7; x2=3; x3=8;
  return 100*x1+10*x2+x3;
}
{
  myofs<<"Сейф методом регулярного перебора."<<endl;
  int count=0;
  int x1, y1, z1, x2, y2, z2;
  mysafe(x1, y1, z1);
  bool b=false;
  for(int x=0; x<10; x++)
  {
    if(b) break;
    for(int y=0; y<10; y++)
    {
      if(b) break;
      for(int z=0; z<10; z++)
      {
        if(x==x1&&y==y1&&z==z1)
        {
          x2=x; y2=y; z2=z;
          b=true;
          break;
        }
        count++;
      }
    }
  }
  myofs << " код=" << x2 << y2 << z2 << ",
    потребовалось " << count << " испытаний." << endl;
  myofs<<"Сейф методом случайного перебора."<<endl;
  count=0;
  int x, y, z;
  while(true)
  {
    x=rand()%10;
    y=rand()%10;
    z=rand()%10;
    if(x==x1&&y==y1&&z==z1)
    {
      break;
```





```
    }
    count++;
  }
  myofs << " код=" << x << y << z << ",
    потребовалось " << count << " испытаний." << endl;
}
```

**Результат в файле протокола:**

```
Откроем сейф методом регулярного перебора.
 код=738, потребовалось 738 испытаний.
Откроем сейф методом случайного перебора.
 код=738, потребовалось 3026 испытаний.
```

## 5. Простейшие операторы ввода из файла

### З а д а ч и

63. [+1] Прочитайте из файла последовательность целых чисел, разделенных пробелами. Результат сохраните в массиве и направьте в файл протокола.
64. [+1] Прочитайте из файла последовательность символов из набора {а…я}, не разделенных пробелами. Результат сохраните в массиве char buf[128] и направьте в файл протокола.
65. [+1] Прочитайте из файла последовательность символов, разделенных пробелами. Результат сохраните в массиве и направьте в файл протокола.
66. Отображение информации
67. [1] Напишите функцию f(int x), которая отображает у левого края экрана строку из x звездочек.
68. [1] Напишите функцию sq(int rows, int cols), которая отображает у левого края экрана прямоугольник из звездочек, размеры которого указаны целыми параметрами rows и cols.
69. [1] Напишите функцию, которая отображает у левого края экрана сплошной прямоугольник из заданных символов, сторона которого указана целыми параметрами rows и cols.
70. [2] Напишите функцию int f(int M, int N), которая выводит в файл шаблон шахматной доски (M рядов по N колонок сим-

волов *) и возвращает значение M*N. Числа M и N прочитайте из файла data.txt.

71. [+2] Напишите программу, которая читает из файла несколько целых чисел (каждое между 1 и 72). Для каждого просчитанного числа ваша программа должна напечатать строку, содержащую соответствующее число звездочек. Например, если ваша программа прочла число 7, она должна напечатать *******.

72. [+2] Напишите программу, которая читает из файла несколько целых чисел (каждое между 1 и 72). Для каждого просчитанного числа ваша программа должна напечатать столбец, содержащий соответствующее число символов *. Например, если ваша программа прочла числа 6 4 4 3 5 4 3 2 4 и т.д, она должна напечатать

```
o
o   o   o   oo  o
ooo oo  oo ooo  o  o
ooooooo ooooooo oooo
oooooooooooooooooooo
oooooooooooooooooooo
```

# Р е ш е н и я

## Ввод чисел из файла.

[+1] Прочитайте из файла последовательность целых чисел, разделенных пробелами. Результат сохраните в массиве и направьте в файл протокола.

**Решение.**

```
{
  myofs << "Ввод массива целого типа."
        << endl;
  myifs.open("data01.txt");
  const int M=128;
  int N;
  int x[M];
  myifs >> N;
  if(N>M-1)
    myofs << " Слишком много чисел: "
          << N << endl ;
  else{
    for(int n=0; n<N; n++)
      myifs >> x[n];
    myofs << " Контрольная выдача:" << endl;
    for(int n=0; n<N; n++)
      myofs << setw(3) << x[n];
  }
  myofs << endl << endl;
  myifs.close();
}
```

**Файл data01.txt содержит следующий текст:**

```
15 3 17 48 52 46 58 59 64 57 52 43 37 23 12 8
```

**Результат в файле протокола:**

```
Ввод массива целого типа.
Контрольная выдача
3 17 48 52 46 58 59 64 57 52 43 37 23 12  8
```

## Ввод отдельных символов из файла.

[+1] Прочитайте из файла последовательность символов из набора {а…я}, не разделенных пробелами. Результат сохраните в массиве char buf[128] и направьте в файл протокола.

**Решение.**

```
{
  myofs << "Ввод одного слова." << endl;
  myifs.open("data03.txt");
  char buf[128];
  myifs >> buf;
  myifs.close();
  myofs << " Контрольная выдача." << endl;
  myofs << buf << endl << endl;
}
```

**Файл data03.txt содержит следующий текст:**

```
длина7удава7от7носа7до7хвоста7равна33попугая
```

**Результат в файле протокола:**

```
Ввод одного слова.
Контрольная выдача
длина7удава7от7носа7до7хвоста7равна33попугая
```

### Ввод символов из файла.

[+1] Прочитайте из файла последовательность символов, разделенных пробелами. Результат сохраните в массиве и направьте в файл протокола.

**Решение.**

```
{
  myofs << "Ввод символов." << endl;
  myifs.open("data05.txt");
  char buf[128];
  int count=0;
  for(int n=0; n<16; n++)
  {
    myifs >> buf[n];
    count++;
  }
  buf[count]=0;
  myifs.close();
  myofs << " Контрольная выдача." << endl;
  myofs << buf << endl << endl;
}
```

**Файл data05.txt содержит следующий текст:**

```
x 3 a 8 [ * 2 @ > \ ~ ё Ъ s 2 8
```

**Результат в файле протокола:**

```
Ввод символов.
Контрольная выдача
x3a8[*2@>\~ёЪs28
```

## 6. Случайные числа

## З а д а ч и

73. [+1] ***Генерирование набора случайных чисел.*** Получите заданное количество N (например, 20) случайных целых чисел в диапазоне от 0 до M-1 (например, M=20). Найдите их сумму.

74. [+2] ***Генерирование набора случайных чисел.*** Получите заданное количество N (например, 100) случайных целых чисел в диапазоне от 0 до M-1 (например, M=200). Обратите





внимание на то, что среди этих чисел попадаются пары равных чисел (а также тройки, четверки и т.д.). Найдите количество пар равных чисел.

75. [+2] ***Генерирование набора различных случайных чисел.*** Получите заданное количество N (например, 20) различных случайных целых чисел в диапазоне от 0 до N-1. Найдите их сумму.

76. [2] Получите заданное количество N (например, 100) случайных целых чисел в диапазоне от 0 до M-1 (например, M=10). Найдите наибольшее количество равных чисел (например, набор 1 6 5 4 3 6 7 2 6 5 4 3 8 1 6 3 5 4 2 6 содержит пять чисел равных 6).

77. [2] ***Бросание монеты.*** Напишите программу, моделирующую бросание монеты с помощью генератора случайных чисел rand(). Для каждого броска монеты программа должна печатать в файл Орел или Решка. Смоделируйте с помощью этой программы бросание 100 раз и подсчитайте, сколько раз появилась каждая сторона монеты.

78. [2] ***Бросание пары монет.*** Напишите программу, моделирующую многократное (N=100 раз) бросание пары монет с помощью генератора случайных чисел rand(). Подсчитайте, сколько раз появилась пара OO, пара OP, пара PP.

79. [2] ***Бросание трех монет.*** Напишите программу, моделирующую многократное (N=100 раз) бросание трех монет с помощью генератора случайных чисел rand(). Подсчитайте, сколько раз появились тройки OOO, OOP, OPP и PPP.

80. [2] ***Бросание игральной кости.*** Напишите программу, моделирующую бросание игральной кости с помощью генератора случайных чисел rand(). Для каждого броска кости программа должна печатать в файл текст I, II, III, IV, V, или VI. Смоделируйте с помощью этой программы бросание N раз и подсчитайте, сколько раз появилась каждая сторона кости.

81. [2] ***Бросание нескольких игральных костей.*** Методом многократного (N=10000) моделирования с помощью генератора случайных чисел rand() найдите вероятность того, что при бросании одновременно четырех костей сумма выпавших очков будет равна 21.

82. [+3] ***Одновременное бросание 4 игральных костей.*** Методом многократного (N=100000 испытаний) моделирования с помощью генератора случайных чисел найдите вероятность того, что при бросании одновременно четырех костей сумма выпавших очков будет равна 4, 5, …, 24 очка. Результаты представьте в виде гистограммы.

83. [2] Напишите программу, моделирующую многократный выбор кости домино с помощью генератора случайных чисел rand(). Всего имеется 28 костей, на которых написаны пары чисел 00, 10, …, 65, 66. Для каждого броска кости программа должна печатать в файл соответствующий текст. Смоделируйте с помощью этой программы бросание N=10000 раз и подсчитайте, сколько раз появилась заданная кость (например, 43).

84. [2] Напишите программу, моделирующую многократный выбор двух костей домино из полного набора с помощью генератора случайных чисел rand(). Всего имеется 28 костей, на которых написаны пары чисел 00, 10, …, 65, 66. Смоделируйте с помощью этой программы бросание пары костей N=10000 раз и подсчитайте, сколько раз появилась пара, допускающая создание цепочки, например, 34 и 46.

85. [2] В барабане револьвера имеется N=6 позиций, из которых заняты M=2. Смоделируйте многократную «игру в рулетку», в ходе которой случайным образом выбирается позиция барабана и затем производится нажатие на курок. Методом случайного (по–науке, стохастического) моделирования найдите вероятность того, что выстрел не произойдет после K=10 испытаний.





86. [1] Известно, что сейф открывается при правильном вводе буквенного кода. Задайте код и затем откройте сейф, используя генератор случайных чисел. Сколько испытаний пришлось сделать, если длина кода была 7 букв?

# Решения

## Генерирование случайных чисел

[+1] Получите заданное количество N (например, 20) случайных целых чисел в диапазоне от 0 до M-1 (например, M=20). Обратите внимание на то, что среди этих чисел попадаются равные пары. Найдите их сумму.

**Решение.**

```
int main(){
  myofs << "Генерируем набор случайных чисел." << endl;
  const int N = 20, M=20;
  int S=0, x[N];
  for(int n=0; n<N; n++)
    x[n]=rand()%M;
  for(int n=0; n<N; n++)
  {
    S+=x[n];
    myofs << setw(3) << x[n] << " ";
    if(n%25==24) myofs << endl;
  }
  myofs << endl << " Количество=" << N
    << ", Сумма= " << setw(4) << S << endl;
}
```

**Результат в файле протокола:**

```
Генерируем набор 20 случайных чисел.
12   9  13  18   3  17   0   5   5  14  10  19   5  17
10  10   0   7   0   0
 Количество=20, Сумма=  174
```

## Исследование набора случайных чисел

[+2] Получите заданное количество N (например, 100) случайных целых чисел в диапазоне от 0 до M-1 (например, M=200). Найдите количество пар равных чисел.

```
{
```

```
  myofs << "Тема: случайные числа." << endl;
  myofs << "Генерируем набор случайных чисел,
           ищем пары." << endl;
  const int N = 48, M=192; int count=0;
  int x[N];
  int y[3][N];
  for(int n=0; n<N; n++) x[n]=rand()%M;
  for(int n=0; n<N; n++)
  {
    myofs << setw(3) << x[n] << " ";
    if(n%12==11) myofs << endl;
  }
  myofs << endl;
  // Подсчитаем пары равных
  for(int n=0; n<N-1; n++)
    for(int k=n+1; k<N; k++)
    {
      if(x[n]==x[k])
      {
        y[0][count]=n;
        y[1][count]=k;
        y[2][count]=x[n];
        count++;
      }
    }
  myofs << " в наборе из N= "
        << N << " случайных чисел имеется "
        << count << " пар равных." << endl;
  for(int n=0; n<count; n++)
  {
    myofs << setw(2) << n+1 << ") ";
    myofs << " n1=" << setw(2) << y[0][n];
    myofs << " n2=" << setw(2) << y[1][n];
    myofs << " n=" << setw(3) << y[2][n];
    myofs << endl;
  }
  myofs << endl;
}
```

**Результат в файле протокола:**

```
Генерируем набор 48 случайных чисел.
80 125  46  94 177   0 170 152  61 113  87
84 131  53  85  66   4  16 163  50  30 190
```





```
37 141 172 191 165 131 168  50  89   4  82
165  36 115  22 108  94  87 169 176  24
20 116 138  27  27
 В наборе из N= 48 чисел имеется 7 пар равных.
 1)  n1= 3 n2=38 n= 94
 2)  n1=10 n2=39 n= 87
 3)  n1=12 n2=27 n=131
 4)  n1=16 n2=31 n=  4
 5)  n1=19 n2=29 n= 50
 6)  n1=26 n2=33 n=165
 7)  n1=46 n2=47 n= 27
```

## Генерирование набора различных случайных чисел

[+2] Получите заданное количество N (например, 20) различных случайных целых чисел в диапазоне от 0 до N-1. Найдите их сумму.

**Решение.**

```
{
  myofs << "Генерируем набор разных
           случайных чисел." << endl;
  const int N = 20, M=20; int count=0;
  int S=0, y, x[N];
  bool b;
  x[0]= rand()%M;
  for(int n=1; n<N; n++)
  {
    while(true)
    {
      y=rand()%M;
      b=false;
      for(int m=0; m<n; m++)
      {
        if(y==x[m])
        {
          b=true;
          break;
        }
      }
      if(!b)break;
    }
    x[n]=y;
```

```
  }
  myofs << endl;
  for(int n=0; n<N; n++)
  {
    S+=x[n];
    myofs << setw(2) << x[n] << " ";
  }
  myofs << endl << " Количество=" << N
    << ", Сумма= " << setw(4) << S << endl;
  myofs << endl;
}
```

**Результат в файле протокола:**

```
Генерируем набор разных случайных чисел.
2   9 11 10 17 19  0  6  4  3
15 16 18 12 13  8 14  5  1  7
 Количество=20, Сумма=  190
```

### Одновременное бросание 4 игральных костей

[+3] ***Одновременное бросание 4 игральных костей.*** Методом многократного (N=100000 испытаний) моделирования с помощью генератора случайных чисел найдите вероятность того, что при бросании одновременно четырех костей сумма выпавших очков будет равна 4, 5, …, 24 очка. Результаты представьте в виде гистограммы.

**Решение.**

```
{
  myofs << "Тема: Случайные числа." << endl;
  myofs <<"Одновременное бросание 4 игральных костей";
  const int N=400, K=4, M=K*6+1, J=100;
  int x[N];
  for(int n=0; n<N; n++)
  {
    x[n]=0;
    for(int k=0; k<K; k++)
      x[n]+=(1+rand()%6);
  }
  int z[M];
  for(int m=0; m<M; m++)
    z[m]=0;
```





```
  for(int n=0; n<N; n++)
    if(x[n]>=0 && x[n]<M) z[x[n]]++;
  for(int m=K; m<M; m++)
  {
    myofs << setw(2)<<m<<" z="<<setw(3)<<z[m]<<" ";
    for(int j=0; j<z[m]; j++)
      myofs << "*";
    myofs << endl;
  }
  myofs << endl << endl;
}
```

**Результат в файле протокола:**

```
Одновременное бросание 4 игральных костей
 4 z=  0
 5 z=  2 **
 6 z=  4 ****
 7 z=  6 ******
 8 z=  8 ********
 9 z= 20 ********************
10 z= 17 *****************
11 z= 35 ***********************************
12 z= 47 ***********************************************
13 z= 42 ******************************************
14 z= 45 *********************************************
15 z= 45 *********************************************
16 z= 33 *********************************
17 z= 36 ************************************
18 z= 24 ************************
19 z= 12 ************
20 z= 12 ************
21 z=  7 *******
22 z=  3 ***
23 z=  0 *
24 z=  2 **
```

## 7. Случайное блуждание

## З а д а ч и

87. [+2] Паук находится на прямой в точке с координатой x=50. Каждую секунду он делает шаг влево или вправо с равной вероятностью. Смоделируйте движение паука с помощью генератора случайных чисел. Координату x(t) сохраните в одномерном массиве. Напечатайте траекторию паука в файл

протокола в виде таблицы, которая содержит в строке с номером t символ x в точке, где паук находился в этот момент времени. В остальных позициях этой строки поставьте символ символ o.

88. [+2] Паук находится на плоскости в точке с координатами x=50 и y=50. Каждую секунду он делает шаг влево, вправо, вниз или вверх с равной вероятностью. Смоделируйте движение паука с помощью генератора случайных чисел. Координаты x(t) и y(t) сохраните в одномерных массивах. Напечатайте траекторию паука в файл протокола в виде таблицы, которая содержит в клеточке с координатами x и y символ +, если паук там побывал, и символ -, если он там не побывал.

# Решения

## Случайное блуждание по прямой

[+2] Паук находится на прямой в точке с координатой x=50. Каждую секунду он делает шаг влево или вправо с равной вероятностью. Смоделируйте движение паука с помощью генератора случайных чисел. Координату x(t) сохраните в одномерном массиве. Напечатайте траекторию паука в файл протокола в виде таблицы, которая содержит в строке с номером t символ x в точке, где паук находился в этот момент времени. В остальных позициях этой строки поставьте символ символ o.

**Решение.**

```
{
  myofs << " Случайное блуждание на прямой." << endl;
  const int N=61, M=15, J=8;
  char buf[N+1];
  int r, x[M];
  x[0] = N/2;
  for(int m=1; m<M; m++)
  {
    r=0;
    for(int j=0; j<J; j++)
      r += (rand()%2)*2-1;
    x[m]=x[m-1]+r;
```





```
  }
  for(int m=0; m<M; m++)
  {
    for(int n=0; n<N; n++)
      buf[n]=' ';
    buf[x[m]]='*';
    buf[N]='\0';
    myofs << buf << endl;
  }
  myofs << endl;
}
```

**Результат в файле протокола:**

```
Случайное блуждание на прямой.
                                *
                               *
                              *
                               *
                              *
                               *
                        *
                     *
                    *
                   *
                   *
                 *
                   *
                   *
                   *
```

## Случайное блуждание по плоскости

[+2] Паук находится на плоскости в точке с координатами x=50 и y=50. Каждую секунду он делает шаг влево, вправо, вниз или вверх с равной вероятностью. Смоделируйте движение паука с помощью генератора случайных чисел. Координаты x(t) и y(t) сохраните в одномерных массивах. Напечатайте траекторию паука в файл протокола в виде таблицы, которая содержит в клеточке с координатами x и y символ +, если паук там побывал, и символ -, если он там не побывал.

**Решение.**

```
{
  myofs << " Случайное блуждание
```

```
            по плоскости." << endl;
  const int N=61, K=1550, J=1;
  char buf[N+1];
  int xy[N][N];
  int x=N/2;
  int y=N/2;
  for(int m=0; m<N; m++)
    for(int n=0; n<N; n++)
      xy[m][n]=0;
  xy[x][y]=1;
  for(int k=1; k<K; k++)
  {
    for(int j=0; j<J; j++)
      x+=rand()%3-1;
    for(int j=0; j<J; j++)
      y+=rand()%3-1;
    if(x<0)x=0;
    if(x>N-1)x=N-1;
    if(y<0)y=0;
    if(y>N-1)y=N-1;
    xy[x][y]+=1;
  }
  for(int m=0; m<N; m++)
  {
    for(int n=0; n<N; n++)
      switch(xy[m][n])
      {
      case 0: buf[n]='.'; break;
      case 1: buf[n]='+'; break;
      case 2: buf[n]='*'; break;
      case 3: buf[n]='#'; break;
      default: buf[n]='@';
    }
    buf[N]='\0';
    myofs << buf << endl;
  }
  myofs << endl;
}
```

**Результат в файле протокола:**

```
Случайное блуждание по плоскости.
.............................................
.............**++.............................
............+.*@#@............................
```





```
.............*#@@@+...........................
.............*@@@+...........................
............@@@#.............................
............#+*@.+.....................+*.....
............+**+..*.....................++@.....
............+*+....++.............*.+##@*.....
............*.....+*+...........*@@*+*@@@*.....
............*....+*............+*@+###@@+......
...........+++...+....++.........+@@.*@+*@#.....
...........##....*.*.@*+......+***+#+..+@@@+++*
...........+.+.*.++#.@@*+.....+**.+......++++*.
............+@.....*@++..*...+@#*@+.........+.
.........+......*....+#**+@..+.##@**+#.........
.........*.....++****+.@@#*+.++*....#..........
.........*+....*+.++#.@@*#+.+...*..............
......+.++*.........+++*##+....+++..............
...+@**+.+**+........**#@@.+..+@++.+............
...*@.....#.++.......+#+@+#+...#@+*.+...........
....@*...+.+*++.......++++@*...++#++.+..........
.....@*.++*+*@.*.......+*#+.....*+*...+.........
....#.#+...+.+*@....+.+*@*+.+..++@#++++........
...+.+++....++#@#*.+*@*@@+#*++..#@*+@#.........
.....*.......**.*++++*.@###@++***.*.*#@#.+......
......*......+*+.*#.++.+*###@@@@*...@**+.......
.............+#+.*@@++...*.@#@@@@*..@@+........
............@@*.@@@@*..+#*@+@@@@+..#..........
............++..@*@##+###+*@#@@**+.............
...........+...+*@#@+*@.@@*@@#@*+.+*...........
.........*.++..+###@@@#*+##.@@**+**.*..........
........+.++#+.#@###@@@*++++#@##+#+*............
........+.+.+*#@@+*@*@@++#@@#+*..+.............
........*+.#+@*@@@*@**@+*++**+..................
.......*@*@**@@@@#++*++..+++....................
.......**+**@#@@@@+#+@.*........................
.......+.###@.*@@@...++++*.......................
.......*#@*@.+@@@#.....+++.......................
.....+@++..*#**#.................................
......+..........................................
.................................................
```

## 8. Случайные процессы

### З а д а ч и

89. [3] Комплекс состоит из M одинаковых объектов, каждый из которых в начале находится в исправном состоянии. При каждом включении сгорает некоторое количество объектов из

числа исправных. Число сгорающих объектов – случайное число в пределах от 1 до K. Сколько циклов включения переживет комплекс, если для обеспечения жизнедеятельности достаточно иметь N исправных объектов, если M=100, K=10, N=7.

90. [+3] Комплекс состоит из M одинаковых объектов, каждый из которых в начале находится в исправном состоянии. Противник производит одна за другой атаки, в ходе каждой из которых он запускает K боеголовок, причем цели – объекты нашего комплекса – выбираются случайным образом. Противник не знает, какие объекты уже поражены и может обстрелять пораженный объект еще раз. Сколько атак переживет комплекс, если для обеспечения жизнедеятельности достаточно иметь N исправных объектов, если M=16, K=3, N=4. Результаты представьте в наглядной форме.

91. [+3] ***Задача случайного обстрела без прогноза.*** Комплекс состоит из N одинаковых объектов, каждый из которых в начале находится в исправном состоянии. Противник производит одна за другой атаки, в ходе каждой из которых он запускает K боеголовок, причем цели – объекты нашего комплекса – выбираются случайным образом. После каждой атаки исправные объекты перемещаются в другие позиции, которые противнику неизвестны. Сколько атак переживет комплекс, если для обеспечения жизнедеятельности достаточно иметь J исправных объектов, если N=16, K=3, J=4. Результаты представьте в наглядной форме.

# Р е ш е н и я

## Задача случайного обстрела без прогноза

[+3] Комплекс состоит из M одинаковых объектов, каждый из которых в начале находится в исправном состоянии. Противник производит одна за другой атаки, в ходе каждой из которых он запускает K боеголовок, причем цели – объекты нашего комплекса – выбираются случайным образом. Противник не знает,





какие объекты уже поражены и может обстрелять пораженный объект еще раз. Сколько атак переживет комплекс, если для обеспечения жизнедеятельности достаточно иметь N исправных объектов, причем M=16, K=3, N=4. Результаты представьте в наглядной форме.

**Решение.**

```
{
  myofs << " Модель случайного размещения." << endl;
  myofs << " 1 исправный объект,"
  << " * уничтоженный объект," << " . промах." << endl;
  const int N = 64, M=12, J=1;
  char buf[N+1];
  int count, k;
  int x[N], z[M];
  for(int n=0; n<N; n++) x[n]=1;
  while(true)
  {
    z[0]=rand()%N;
    for(int m=1; m<M; m++)
    {
      bool b=true;
      while(b)
      {
        k = rand()%N;
        b=false;
        for(int j=0; j<m; j++)
        {
          if(k==z[j])
          {
            b=true;
            break;
          }
        }
      }
      z[m]=k;
    }
    for(int n=0; n<N; n++)
    {
      if(x[n]==1) buf[n]='1';
      else buf[n] = ' ';
      buf[N] = '\0';
```

```
    }
    for(int m=0; m<M; m++)
      if(x[z[m]]==1)
        buf[z[m]] = '*';
      else buf[z[m]] = '.';
    myofs << buf << endl;
    for(int m=0; m<M; m++)
      x[z[m]]=0;
    count=0;
    for (int n=0; n<N; n++) count+=x[n];
    if(count<J) break;
  }
}
```

**Результат в файле протокола:**

```
Тема: случайные числа, Модель случайного размещения.
 1 исправный объект, * уничтоженный объект, . промах.
11111111111111111111111111111111111111111111111111111111111111111111111111
11*111* 11*11 1*** 111*11 111 . 1 11.*111111111*1111111 1 *1111
1*.111  11.11 1   .111.11 *11 . 1.11. 111111111 1*11111 1  *1*1
1 .111  1* 11 *    111.11  11. .1 11 .1111***11 1 *1111 1   * 1
1. 1**  1  11      111 *1. 1*   * *1  1111   11 1  *111 *    .*
1  *.. .1  *1  .  .111  1  1       1. 1111.  11.1   111     ..
1 .     1.  1     .111. 1 .1       1. 11**  .1* 1   111 .     .
1       1   *. ... 11*  1  1   .  .1  11     *  1   1*1. .
1   .   *    .    .11   1  1  ...  1  11      . 1 . * *.
*       .   .      11   1  1    . .1  11       *   .     . ....
  ..    .     . .  .11   1  1    .  1  11  . .         . .     .
  . .  .        .. 1*.  *  1.      *  11         .    .
.          ..  .  1  . .  1      .   11.  .      . .          .
      .     .     1      .1  .  .    **   .. .     .   .
.     ..          1   . . 1 . .    .            .   .    .   .
   .          .   .1       *          .        .    ....    . .
. .     .   .     1 .         .. .   . .             .    .
  .   .     . ..  .*      . . .                     . .
```

## 9. Одномерные массивы постоянного размера

## З а д а ч и

92. [2] Создайте одномерный массив целого типа заданного размера M. Задайте значения элементов с помощью генератора случайных чисел. Распечатайте массив в файле протокола.





93. [+2] Напишите функцию int sumi(int* arr, int N), которая находит сумму элементов массива int arr заданного размера N. Создайте одномерный массив типа int заданного размера N и инициализируйте его набором N случайных чисел в диапазоне 2…5. Примените функцию sumi(…) для анализа массива. Массив и результаты его анализа направьте в файл протокола.

94. [+2] Напишите функцию int sumi(int* arr, int N), которая находит сумму элементов массива int arr заданного размера N. Используя текстовый редактор, сформируйте файл data.txt, который содержит число N и затем набор из N целых чисел, разделяемых символом пробела. Создайте одномерный массив типа int постоянного размера и введите в него набор N чисел из файла data.txt. Примените функцию sumi(…) для анализа введенного из файла массива. Введенный массив и результаты его анализа направьте в файл протокола.

95. [2] Напишите функцию f(int& minval, int& maxval, int* arr, int N), которая находит и возвращает наименьшее и наибольшее значение из набора целых чисел arr заданного размера N. Используя текстовый редактор, сформируйте файл data.txt, который содержит число N и набор целых чисел, разделяемых символом пробела. Создайте одномерный массив типа int постоянного размера и введите в него набор N чисел из файла data.txt. Примените функцию f(…) для анализа введенного из файла массива. Введенный массив и результаты его анализа направьте в файл протокола.

96. [2] Напишите функцию f(double* minval, double* maxval, double* arr, int N), которая находит и возвращает наименьшее и наибольшее значение из набора double чисел arr заданного размера N. Используя текстовый редактор, сформируйте файл data.txt, который содержит число N и набор double чисел, разделяемых символом пробела. Создайте одномерный массив типа double постоянного размера и введите в него набор N чисел из файла data.txt. Примените функцию f(…) для анализа

введенного из файла массива. Введенный массив и результаты его анализа направьте в файл протокола.

# Решения

## Вычисление суммы элементов одномерного массива

[+2] Напишите функцию int sumi(int* arr, int N), которая находит сумму элементов массива int arr заданного размера N. Создайте одномерный массив типа int заданного размера N и инициализируйте его набором N случайных чисел в диапазоне 2…5. Примените функцию sumi(…) для анализа массива. Массив и результаты его анализа направьте в файл протокола.

**Решение.**

```
{
  myofs << "Вычисление суммы элементов массива."
    << endl;
  const int N=16;
  int x[N];
  for(int n=0; n<N; n++)
    x[n]=2+rand()%4;
  myofs << " Массив целых чисел [" << N << "]" << endl;
  for(int n=0; n<N; n++)
    myofs << setw(3) << x[n];
  myofs << endl;
  int s = sumi(x, N);
  myofs << " Сумма=" << s << endl << endl;
}
```

## Ввод из файла и анализ одномерного массива

[+2] Напишите функцию int sumi(int* arr, int N), которая находит сумму элементов массива int arr заданного размера N. Используя текстовый редактор, сформируйте файл data.txt, который содержит число N и затем набор из N целых чисел, разделяемых символом пробела. Создайте одномерный массив типа int постоянного размера и введите в него набор N чисел из файла data.txt. Примените функцию sumi(…) для анализа введенного из





файла массива. Введенный массив и результаты его анализа направьте в файл протокола.

**Решение.**

```
{
  myofs << "Ввод и вычисление суммы элементов массива."
<< endl;
  myifs.open("data02.txt");
  int N;
  myifs >> N;
  int x[128];
  for(int n=0; n<N; n++)
    myifs >> x[n];
  myifs.close();
  myofs << " Контрольная выдача [" << N << "]" << endl;
  for(int n=0; n<N; n++)
    myofs << setw(3) << x[n];
  myofs << endl;
  int s = sumi(x, N);
  myofs << " Сумма=" << s << endl << endl;
}
```

**Результат в файле протокола:**

```
Вычисление суммы элементов массива.
 Массив целых чисел [16]
  5  2  5  3  2  4  4  5  4  4  5  4  5  5  5  4
 Сумма=66
```

**Файл data02.txt содержит следующий текст:**

```
32
13 17 16 12 16 18 19 14 9 7 8 17 18 19  9 12
 5  7 18 21 21 18 19 14 7 3 5  7 18 19 19 22

Вычисление суммы элементов массива.
 Контрольная выдача введенного массива целых чисел [32]
13 17 16 12 16 18 19 14  9  7  8 17 18 19  9 12  5  7
18 21 21 18 19 14  7  3  5  7 18 19 19 22
 Сумма=447
```

# 10. Сортировка одномерного массива

## З а д а ч и

97. [+2] Напишите функцию sorti(int* arr, int N), которая сортирует по возрастанию массив int arr заданного размера N. Создайте одномерный массив типа int, содержащий набор случайных чисел. Примените функцию sorti(…) для сортировки. Исходный массив и результат сортировки направьте в файл протокола.

98. [+3] Напишите функцию sortif(int* arr, int N, PIFII f), которая сортирует по возрастанию массив int arr заданного размера N с помощью функции сравнения int compare(int m, int n). Тип PIFII (Pointer to Integer Function with arguments Int and Int) определите с помощью оператора typedef. Создайте одномерный массив типа int, содержащий набор случайных чисел. Примените функцию sortif(…) для сортировки. Исходный массив и результат сортировки направьте в файл протокола.

99. [+3] Напишите функцию sort2(int* arr, char* txt[], int N, PIFII f), которая одновременно сортирует по возрастанию массив int arr заданного размера N с помощью функции сравнения int compare(int m, int n) и массив текстового типа. Это может быть полезно, например, если нам нужно рассортировать список учебного потока, содержащий номера (указанные беспорядочно) и фамилии. Тип PIFII (Pointer to Integer Function with arguments Int and Int) определите с помощью оператора typedef. Результаты сортировки направьте в файл протокола.

100. [+3] Напишите функцию sortxt(char* txt[], int N, PIFPCPC f), которая сортирует массив char* arr[] с помощью функции сравнения int compare(char* t1, char* t2). Тип PIFPCPC (Pointer to Integer Function with arguments Char* and Char*) определите с помощью оператора typedef. Результаты сортировки направьте в файл протокола.





101. [2] Напишите функцию sortd(double* arr, int N), которая сортирует по возрастанию массив double arr заданного размера N. Используя текстовый редактор, сформируйте файл data.txt, который содержит число N и набор из N целых чисел, разделяемых символом пробела. Создайте одномерный массив типа double постоянного размера и введите в него набор N чисел из файла data.txt. Примените функцию sortd(…) для сортировки введенного из файла массива. Введенный массив и результаты его анализа направьте в файл протокола.

102. [2] Напишите функцию int f(int* arr, int N), которая вычисляет и возвращает количество пар равных друг другу элементов массива int arr заданного размера N. Используя текстовый редактор, сформируйте файл data.txt, который содержит число N и затем набор из N целых чисел, разделяемых символом пробела. Создайте одномерный массив типа int постоянного размера и введите в него набор N чисел из файла data.txt. Примените функцию f(…) для анализа введенного из файла массива. Введенный массив и результаты его анализа направьте в файл протокола.

103. [2] Напишите функцию int f(int m, int* arr, int N), которая вычисляет и возвращает количество элементов массива int arr заданного размера N, равных m. Используя текстовый редактор, сформируйте файл data.txt, который содержит число N и затем набор из N целых чисел, разделяемых символом пробела. Создайте одномерный массив типа int постоянного размера и введите в него набор N чисел из файла data.txt. Примените функцию f(…) для анализа введенного из файла массива. Введенный массив и результаты его анализа направьте в файл протокола.

## Р е ш е н и я

### Сортировка с помощью функции сравнения.

[+3] Напишите функцию sortif(int* arr, int N, PIFII f), которая сортирует по возрастанию массив int arr заданного размера N с

помощью функции сравнения int compare(int m, int n). Тип PIFII (Pointer to Integer Function with arguments Int and Int) определите с помощью оператора typedef. Создайте одномерный массив типа int, содержащий набор случайных чисел. Примените функцию sortif(…) для сортировки. Исходный массив и результат сортировки направьте в файл протокола.

**Решение.**

```
typedef int PIFII(int, int);
int fsiia(int, int);
int fsiib(int, int);
int fout(int* arr, int N, char* comment, int wd1, int
wd2);
int srt(int* arr, int N, PIFII f);
{
    myofs << "Сортируем массив." << endl;
    int N=32;
    int* nn = new int[N];
    for(int n=0; n<N; n++) nn[n]=rand()%64;
    fout(nn, N, "Исходный массив", 2, 2);
    srt(nn, N, fsiia);
    fout(nn, N, "Сортированный массив", 2, 2);
    srt(nn, N, fsiib);
    fout(nn, N, "По-другому сортированный массив", 2,
2);
    delete[] nn;
}
int fsiia(int n1, int n2){ // обычная сортировка:
  if(n1<n2) return -1;
  if(n1>n2) return 1;
  return 0;
}
int fsiib(int n1, int n2){
  // сравниваем сумму всех цифр
  // трехзначного числа:
  int n1a = n1%10;
  int n1b = ((n1-n1a)%100)/10;
  int n1c = (n1-n1a-10*n1b)/100;
  int n2a = n2%10;
  int n2b = ((n2-n2a)%100)/10;
  int n2c = (n2-n2a-10*n2b)/100;
  if(n1a+n1b+n1c>n2a+n2b+n2c) return 1;
```





```
  if(n1b+n1c>n2b+n2c) return 1;
  if(n1c>n2c) return 1;
  if(n1a+n1b+n1c<n2a+n2b+n2c) return -1;
  if(n1b+n1c<n2b+n2c) return -1;
  if(n1c<n2c) return -1;
  return 0;
}
int srt(int* arr, int N, PIFII f){
  int tmp, count=0;
  for(int n=0; n<N-1; n++)
    for(int m=n+1; m<N; m++){
      if(f(arr[n], arr[m])<1) continue;
      tmp=arr[n];
      arr[n]=arr[m];
      arr[m]=tmp;
      count++;
  }
  return count;
}
int fout(int* arr, int N, char* comment, int wd1=2, int
wd2=4){
  int addl=true;
  myofs << comment << endl;
  for(int n=0; n<N; n++){
    myofs << " [" << setw(wd1) << n << "]=" <<
setw(wd2) << arr[n];
    addl=true;
    if((n+1)%8==0){ myofs << endl; addl=false; }
  }
  if(addl) myofs << endl;
  return 0;
}
```

**Результат в файле протокола:**

```
Сортируем массив.
Исходный массив
[ 0]=21 [ 1]= 2 [ 2]=29 [ 3]=21 [ 4]=63 [ 5]=63 [ 6]=57
[ 7]= 9 [ 8]=19 [ 9]=58 [10]=38 [11]=45 [12]=38 [13]= 4
[14]=61 [15]= 7 [16]=54 [17]=52 [18]=10 [19]= 5 [20]=14
[21]=58 [22]=17 [23]=44 [24]=42 [25]=48 [26]=14 [27]=18
[28]=28 [29]=50 [30]= 1 [31]=13
Сортированный массив
[ 0]= 1 [ 1]= 2 [ 2]= 4 [ 3]= 5 [ 4]= 7 [ 5]= 9 [ 6]=10
[ 7]=13 [ 8]=14 [ 9]=14 [10]=17 [11]=18 [12]=19 [13]=21
```

```
[14]=21 [15]=28 [16]=29 [17]=38 [18]=38 [19]=42 [20]=44
[21]=45 [22]=48 [23]=50 [24]=52 [25]=54 [26]=57 [27]=58
[28]=58 [29]=61 [30]=63 [31]=63
По-другому сортированный массив
[ 0]= 1 [ 1]=10 [ 2]= 2 [ 3]=21 [ 4]=21 [ 5]= 4 [ 6]=13
[ 7]= 5 [ 8]=14 [ 9]=14 [10]=50 [11]=42 [12]= 7 [13]=52
[14]=61 [15]=17 [16]=44 [17]= 9 [18]=18 [19]=45 [20]=54
[21]=63 [22]=63 [23]=19 [24]=28 [25]=29 [26]=38 [27]=38
[28]=48 [29]=57 [30]=58 [31]=58
```

## Сортировка двух одномерных массивов разного типа одновременно

[+3] Напишите функцию sort2(int* arr, char* txt[], int N, PIFII f), которая одновременно сортирует по возрастанию массив int arr заданного размера N с помощью функции сравнения int compare(int m, int n) и массив текстового типа. Это может быть полезно, например, если нам нужно рассортировать список учебного потока, содержащий номера (указанные беспорядочно) и фамилии. Тип PIFII (Pointer to Integer Function with arguments Int and Int) определите с помощью оператора typedef. Результаты сортировки направьте в файл протокола.

**Решение.**

```
typedef bool PIFII(int, int);
int srt(int* ia, char* arr[], int N, PIFII f);
int fout(char* arr[], int N, char* comment, int wd1,
int wd2);
char* nam[10]={"Иванов", "Петров", "Сидоров", "Никола-
ев", "Гагарин", "Волков", "Толстой", "Гоголь", "Турге-
нев", "Булгаков"};
char* name[10]={"Петр", "Сергей", "Иван", "Николай",
"Александр", "Алексей", "Григорий", "Константин", "Ви-
талий", "Валентин"};
char* names[10]={"Иванович", "Петрович", "Александро-
вич", "Николаевич", "Григорьевич", "Витальевич", "Ва-
лентинович", "Константинович", "Васильевич", "Алексее-
вич"};
int fout(int* ia, char* arr[], int N, char* comment,
int wd1=12, int wd2=128)
{
  int addl=true;
```





```
  int count=0;
  myofs << endl << comment << endl;
  for(int n=0; n<N; n++){
    addl=true;
    myofs << " [" << setw(2) << ia[n]
          << "]=" << arr[n];
    count += (6+strlen(arr[n]));
    if(count>wd2)
    {
      myofs << endl;
      count=0;
      addl=false;
    }
  }
  if(addl) myofs << endl;
  return 0;
}
{
myofs << "Одновременная сортировка нескольких масси-
вов." << endl;
    const int N=16; int wd=3, n1, n2, n3;
    int* nn = new int[N];
    char* txt[N];
    char buf[128];
    for(int n=0; n<N; n++){
      strcpy(buf, nam[rand()%10]);
      strcat(buf, " ");
      strcat(buf, name[rand()%10]);
      strcat(buf, " ");
      strcat(buf, names[rand()%10]);
      int k = 1 + strlen(buf);
      txt[n] = new char[k];
      strcpy(txt[n], buf);
    }
    for(int n=0; n<N; n++) nn[n]=n;
    for(int n=0; n<1000; n++){
      n1=rand()%N;
      n2=rand()%N;
      n3=nn[n1]; nn[n1]=nn[n2]; nn[n2]=n3;
    }
    fout(nn, txt, N, "Исходный массив", 3, 4);
    srt(nn, txt, N, fsiia);
    fout(nn, txt, N, "Сортированный массив", 3, 4);
```

```
    delete[] nn;
    for(int n=0; n<N; n++) delete[] txt[n];
}
```

**Результат в файле протокола:**

```
Одновременная сортировка нескольких массивов.
Исходный массив
 [ 1]=Сидоров Григорий Витальевич
 [14]=Николаев Валентин Александрович
 [ 8]=Гагарин Григорий Петрович
 [11]=Тургенев Иван Петрович
 [12]=Петров Валентин Константинович
 [ 6]=Толстой Иван Алексеевич
 [13]=Волков Иван Иванович
 [10]=Иванов Николай Алексеевич
 [ 3]=Петров Виталий Петрович
 [ 9]=Булгаков Алексей Николаевич
 [ 7]=Сидоров Алексей Александрович
 [ 2]=Волков Виталий Валентинович
 [ 5]=Гоголь Константин Александрович
 [15]=Сидоров Валентин Григорьевич
 [ 0]=Петров Валентин Валентинович
 [ 4]=Булгаков Виталий Александрович
Сортированный массив
 [ 0]=Петров Валентин Валентинович
 [ 1]=Сидоров Григорий Витальевич
 [ 2]=Волков Виталий Валентинович
 [ 3]=Петров Виталий Петрович
 [ 4]=Булгаков Виталий Александрович
 [ 5]=Гоголь Константин Александрович
 [ 6]=Толстой Иван Алексеевич
 [ 7]=Сидоров Алексей Александрович
 [ 8]=Гагарин Григорий Петрович
 [ 9]=Булгаков Алексей Николаевич
 [10]=Иванов Николай Алексеевич
 [11]=Тургенев Иван Петрович
 [12]=Петров Валентин Константинович
 [13]=Волков Иван Иванович
 [14]=Николаев Валентин Александрович
 [15]=Сидоров Валентин Григорьевич
```

## Сортировка одномерного текстового массива

[+3] Напишите функцию sortxt(char* txt[], int N, PIFPCPC f), которая сортирует массив char* arr[] с помощью функции сравнения int compare(char* t1, char* t2). Тип PIFPCPC (Pointer to Integer Function with arguments Char* and Char*) определите с помощью оператора typedef. Результаты сортировки направьте в файл протокола.





**Решение.**

```
int srt(char* arr[], int N, PIFPCPC f);
int fscca(char*, char*);
int fsccb(char*, char*);
{
  myofs << endl << "0757 Сортируем текстовый  массив."
<< endl;
  const int N=64;
  char* txt[N];
  char buf[128];
  for(int n=0; n<N; n++){
    int k = 1 + rand()%8;
    txt[n] = new char[k+1];
    for(int j=0; j<k; j++)
      buf[j]=char(65+rand()%16);
    buf[k]='\0';
    strcpy(txt[n], buf);
  }
  fout(txt, N, "Исходный массив", 12, 72);
  srt(txt, N, fscca);
  fout(txt, N, "Сортированный массив", 3, 72);
  srt(txt, N, fsccb);
  fout(txt,  N,  "По-другому  сортированный  массив",  3,
72);
  for(int n=0; n<N; n++) delete[] txt[n];
}
int srt(char* arr[], int N, PIFCC f)
{
  char buf[128], count=0;
  for(int n=0; n<N-1; n++)
    for(int m=n+1; m<N; m++){
      if(f(arr[n], arr[m])<=0)continue;
      strcpy(buf, arr[n]);
      delete[] arr[n];
      arr[n]=new char[strlen(arr[m])+1];
      strcpy(arr[n], arr[m]);
      delete[] arr[m];
      arr[m]=new char[strlen(buf)+1];
      strcpy(arr[m], buf);
      count++;
  }
  return count;
```

```
}
int fscca(char* txt1, char* txt2){
  if(strlen(txt1)<strlen(txt2)) return -1;
  if(strlen(txt1)==strlen(txt2)) return 0;
  return 1;
}
int fsccb(char* txt1, char* txt2){
  if(int(txt1[0])<int(txt2[0])) return -1;
  if(int(txt1[0])>int(txt2[0])) return 1;
  if(strlen(txt1)<strlen(txt2)) return -1;
  return 0;
}
```

**Результат в файле протокола:**

```
Исходный массив
 JBOINFE DFB EDBDKE NAGEIHHE KC PE DIBN GCBOLF ODF HKNAPODN
 FCLN IJ DC FPF HPPI KMCN NEMLGOA HBG NEJCKGKE FDII JPJGBKH EJ
 B JPKDOM KNAH NDNLOJMH DDKN JPA AK EP ILAFK GKDP HKLKG CBBHIK
 HKBHC IKEDB DFL NOCO FAOF INAAFMG OD BMFKNP P ODDIOJ OG
EPAHBCB
 FINGBKF DPCPH HCNF PNA GINA BGIDNNG GO AG CBCH HENEJO AHM B
JEAD
 CDEFHB LA DIJJLPIN MMFBFALG E

Сортированный массив
B P B E EJ KC AK EP OD PE OG GO AG IJ LA DC JPA PNA FPF HBG
AHM DFL DFB ODF HPPI KNAH DDKN HCNF KMCN GINA FCLN FDII CBCH
GKDP DIBN JEAD NOCO FAOF IKEDB ILAFK HKLKG DPCPH HKBHC JPKDOM
HENEJO GCBOLF BMFKNP EDBDKE CDEFHB ODDIOJ CBBHIK NEMLGOA
INAAFMG JPJGBKH
BGIDNNG EPAHBCB FINGBKF JBOINFE NAGEIHHE HKNAPODN NEJCKGKE
DIJJLPIN MMFBFALG NDNLOJMH

По-другому сортированный массив
AK AG AHM B B BMFKNP BGIDNNG CDEFHB CBBHIK CBCH DIBN DPCPH DFL
DFB DDKN DC DIJJLPIN E EP EJ EPAHBCB EDBDKE FDII FPF FCLN
FINGBKF FAOF GINA GO GKDP GCBOLF HBG HKBHC HENEJO HCNF HPPI
HKNAPODN HKLKG IKEDB ILAFK INAAFMG IJ JPKDOM JPA JBOINFE
JPJGBKH JEAD KNAH KMCN KC LA MMFBFALG NOCO NEJCKGKE NEMLGOA
NAGEIHHE NDNLOJMH OG ODDIOJ OD ODF P PNA PE
```





## 11. Динамические одномерные массивы

### З а д а ч и

104. [2] Используя текстовый редактор, сформируйте файл data.txt, который содержит набор чисел типа double, разделяемых символом пробела. Договоримся о том, что, первое число в этом файле натуральное и оно задает количество последующих чисел типа double. Прочитайте это значение. Создайте одномерный массив переменной размерности типа double нужного размера с помощью оператора new. Введите в этот массив набор чисел из файла data.txt. Напишите функцию, которая находит и возвращает наименьшее и наибольшее значение из набора чисел типа double заданного размера. Примените эту функцию для анализа введенного из файла массива. Введенный массив и результаты его анализа направьте в файл протокола.

105. [3] Используя текстовый редактор, сформируйте файл data.txt, который содержит набор чисел типа int, разделяемых символом пробела. Договоримся о том, что, первое число в этом файле натуральное и оно задает количество последующих чисел типа int. Прочитайте это значение. Создайте одномерный массив переменной размерности типа int нужного размера с помощью оператора new. Введите в него набор чисел из файла data.txt. Напишите функцию, которая формирует гистограмму массива чисел типа int заданного размера. Примените эту функцию для анализа введенного из файла массива. Введенный массив и результаты его анализа направьте в файл протокола.

## 12. Гистограмма одномерного массива

### З а д а ч и

106. [2] На курсе N=20 студентов, пронумерованных номерами от 1 до N. В начальный момент времени каждый из них имел знания, которые можно выразить целым числом в пределах от

1 до 49. Сгенерируйте набор случайных чисел в указанном диапазоне. Сформируйте в файле протокола диаграмму, которая отражает знания студентов в наглядной форме, например, примерно так:

```
01 oooooo
02 ooooooooo
03 ooooooooooooooooo
04 ooo
05 oooooooooooooooooooo
06 ooooooooooo
07 oooooooooooooo
08 oooooooooooo
```

(в этом примере 8 студентов и оценки от 2 до 20, такая диаграмма называется гистограммой).

107. [2] На курсе N=25 студентов, пронумерованных номерами от 1 до N. В начальный момент времени каждый из них имел знания, которые можно выразить целым числом в пределах от 1 до 49. Сгенерируйте набор случайных чисел в указанном диапазоне. Сформируйте в файле протокола диаграмму, которая отражает знания студентов в наглядной форме, например, примерно так:

```
05                  o
04 o    o    o    oo  o
03 ooo oo   oo ooo  o  o
02 ooooooo ooooooo oooo
01 oooooooooooooooooooo
00 oooooooooooooooooooo
```

108. [2] Напишите функцию int f(double dmin, double dmax, int* arr, int N), которая вычисляет и возвращает количество элементов массива double arr заданного размера N, значения которых находятся строго в пределах от dmin до dmax. Используя текстовый редактор, сформируйте файл data.txt, который содержит число N и затем набор из N double чисел, разделяемых символом пробела. Создайте одномерный массив типа double постоянного размера и введите в него набор N чисел из файла data.txt. Примените функцию f(…) для анализа введенного из файла массива. Введенный массив и результаты его анализа направьте в файл протокола.





109. [2] Напишите функцию int f(int dmin, int dmax, int* arr, int N), которая вычисляет и возвращает количество элементов массива int arr заданного размера N, значения которых находятся в пределах от dmin до dmax включительно. Если некоторый элемент массива arr был меньше dmin, сделайте его равным dmin. Если некоторый элемент массива arr был больше dmax, сделайте его равным dmax. Используя текстовый редактор, сформируйте файл data.txt, который содержит число N и затем набор из N целых чисел, разделяемых символом пробела. Создайте одномерный массив типа int постоянного размера и введите в него набор N чисел из файла data.txt. Примените функцию f(…) для анализа введенного из файла массива. Введенный массив и результаты его анализа направьте в файл протокола.

# Решения

## Простая гистограмма одномерного массива

[+2] Напишите программу, которая читает из файла несколько целых чисел (каждое между 1 и 72). Для каждого просчитанного числа ваша программа должна напечатать строку, содержащую соответствующее число звездочек. Например, если ваша программа прочла число 7, она должна напечатать *******.

**Решение.**

```
{
  myofs << "Простая гистограмма." << endl;
  myifs.open("data04.txt");
  int N;
  myifs >> N;
  int* x = new int[N];
  for(int n=0; n<N; n++)
    myifs >> x[n];
  myifs.close();
  myofs << " Контрольная выдача." << endl;
  for(int n=0; n<N; n++) myofs << setw(3)
```

```
      << x[n]; myofs << endl;
  myofs << " Гистограмма." << endl;
  for(int n=0; n<N; n++)
  {
  myofs << "x[" << setw(2) << n << "]="
        << setw(3) << x[n] << " ";
    for(int m=0; m<x[n]; m++)
      myofs << "*";
      myofs << endl;
  }
  delete[] x;
  myofs << endl;
}
```

**Файл data04.txt содержит следующий текст:**

```
15 3 17 48 52 46 58 59 64 57 52 43 37 23 12 8
```

**Результат в файле протокола:**

```
Простая гистограмма.
Контрольная выдача введенного массива целых чисел.
  3 17 48 52 46 58 59 64 57 52 43 37 23 12  8
 Гистограмма:
x[ 0]=  3 ***
x[ 1]= 17 ****************
x[ 2]= 48 ************************************
x[ 3]= 52 ****************************************
x[ 4]= 46 **********************************
x[ 5]= 58 ****************************************
x[ 6]= 59 *****************************************
x[ 7]= 64 ************************
x[ 8]= 57 ***************************************
x[ 9]= 52 ****************************************
x[10]= 43 *********************************
x[11]= 37 *************************************
x[12]= 23 ***********************
x[13]= 12 ************
x[14]=  8 ********
```

## Сложная гистограмма одномерного массива

[+2] Напишите программу, которая читает из файла несколько целых чисел (каждое между 1 и 72). Для каждого просчитанного числа ваша программа должна напечатать столбец, содержащий





соответствующее число символов *. Например, если ваша программа прочла числа
6 4 4 3 5 4 3 2 4 и т.д,
она должна напечатать

```
o
o   o   o   oo  o
ooo oo  oo ooo  o  o
ooooooo ooooooo oooo
oooooooooooooooooooo
oooooooooooooooooooo
```

**Решение.**

```
{
  myofs << "Сложная гистограмма." << endl;
  myifs.open("data02.txt");
  int N; const int M=24, K=128;
  myifs >> N;
  int* x = new int[N];
  for(int n=0; n<N; n++)
  {
    myifs >> x[n];
    if(x[n]>=M)x[n]=M-1;
  }
  myifs.close();
  myofs << " Контрольная выдача[" << N << "]" << endl;
  for(int n=0; n<N; n++)
    myofs << setw(3) << x[n];
  myofs << endl;
  int y[M][K];
  for(int n=0; n<N; n++)
  {
    for(int m=0; m<M; m++)
      y[m][n]= 0;
    for(int m=0; m<x[n]; m++)
      y[m][n]= 1;
  }
  myofs << " Гистограмма." << endl;
  char buf[K+1];
  for(int m=0; m<M; m++)
  {
    for(int n=0; n<N; n++)
    {
```

```
      if(y[m][n]==1) buf[n]=' ';
      else buf[n]='*';
    }
    buf[N]='\0';
    myofs << setw(2) << M-m << " ";
    myofs << buf << endl;
  }
  myofs << endl << endl;
  delete[] x;
}
```

**Файл data02.txt содержит следующий текст:**

```
32
13 17 16 12 16 18 19 14 9 7 8 17 18 19  9 12
 5  7 18 21 21 18 19 14 7 3 5  7 18 19 19 22
```

**Результат в файле протокола:**

```
Сложная гистограмма.
Контрольная выдача введенного массива целых чисел [32]
13 17 16 12 16 18 19 14 9 7 8 17 18 19  9 12
 5  7 18 21 21 18 19 14 7 3 5  7 18 19 19 22

Гистограмма.
22
21                          *
20                          *
19                 *        **
18                 *        **
17           *     **      ****
16           **    **      ****
15          ***   * **     ****
14          ***   * **     ****
13          ***   * **     ****
12    *     ***   ****     ****
11 *  *     ***   ****     ****
10 *  *    ****   ****    *****
 9 *  *    ****   ****    *****
 8 * ***   ****   ****    *****
 7 *****   *****  ****    *****
 6 ****** ****** *****  * ******
 5 ******************  **********
 4 ******************  **********
 3 ******************************
 2 ******************************
 1 ******************************
```





# 13. Двумерные массивы с постоянной размерностью

## З а д а ч и

110. [2] Создайте двумерный массив целого типа заданного размера, M=12 строк и N=24 столбцов. Задайте значения элементов с помощью генератора случайных чисел в пределах от 2 до 5. Распечатайте массив по строкам в файле протокола.

111. [2] Создайте двумерный массив целого типа заданного размера, M=12 строк и N=24 столбцов. Задайте значения элементов с помощью генератора случайных чисел. Найдите сумму всех элементов массива и среднее значение величины элемента массива. Распечатайте массив по строкам в файле протокола.

112. [+2] Создайте двумерный массив целого типа заданного размера, M=12 строк и N=24 столбцов. Задайте значения элементов с помощью генератора случайных чисел в пределах от 2 до 5. Найдите сумму всех элементов в каждой строке. Массив и результаты его анализа распечатайте в файле протокола.

113. [2] Создайте двумерный массив целого типа заданного размера, M строк и N столбцов. Задайте значения элементов с помощью генератора случайных чисел в пределах от 0 до 99. Найдите сумму всех элементов в каждом столбце. Массив и результаты его анализа направьте в файл протокола.

114. [2] Создайте двумерный массив целого типа заданного размера, M=12 строк и N=24 столбцов. Каждый элемент этого массива может быть равен 0 или 1. Задайте значения элементов с помощью генератора случайных чисел. Найдите а) отрезок строки этого массива, содержащий только 1, имеющий наибольшую длину. b) отрезок столбца этого массива, содержащий только 1, имеющий наибольшую длину. c) Наибольший прямоугольный участок данного массива, содержащий только 1. Распечатайте массив по строкам в файле протокола.

115. [2] На первом курсе M=20 студентов. Каждый из них за год получает N=30 оценок в пределах от 2 до 5. Сформируйте целый массив нужного размера. Задайте оценки с помощью генератора случайных чисел. Найдите среднюю оценку всего курса за каждый день обучения. Массив и результаты его анализа направьте в файл протокола.

116. [2] На первом курсе M студентов. Каждый из них за год получает N оценок в пределах от 2 до 5. Сформируйте целый массив нужного размера. Задайте оценки с помощью генератора случайных чисел. Найдите среднюю оценку каждого студента за весь учебный год. Массив и результаты его анализа направьте в файл протокола.

117. [2] На первом курсе M=40 студентов. Каждый из них за год получает N=20 оценок в пределах от 1 до 100. Лучшим считается студент, имеющий наибольшую сумму трех лучших оценок за весь курс. Сформируйте целый массив нужного размера. Задайте оценки с помощью генератора случайных чисел. Найдите лучшего и худшего студентов.

118. [2] На первом курсе M=25 студентов. Каждый из них за год получает N=30 оценок в пределах от 1 до 100. Лучшим считается студент, имеющий наибольшую сумму трех следующих подряд оценок. Сформируйте целый массив нужного размера. Задайте оценки с помощью генератора случайных чисел. Найдите лучшего и худшего студентов.

119. [2] На первом курсе M=30 студентов. Каждый из них раз в неделю получает оценку по программированию в пределах от 2 до 5, в году N=45 недель. Лучшим считается студент, который получил наибольшее количество оценок «4» и «5». Сформируйте массив нужного размера. Задайте оценки с помощью генератора случайных чисел. Найдите лучшего студента.

120. [2] На первом курсе M=30 студентов. Каждый из них раз в неделю получает оценку по программированию в пределах от 2 до 5, в году N=38 недель. Получивший две двойки подряд





отчисляется. Сколько студентов закончили курс? Сформируйте двумерный массив целого типа нужного размера. Задайте оценки с помощью генератора случайных чисел. Напечатайте массив и результаты его анализа в файл в наглядной форме.

121. [2] На первом курсе M=50 студентов. Каждый из них в понедельник получает оценку по программированию, а во вторник – оценку по физике в пределах от 2 до 5 каждая, всего в году N=35 недель. Лучшим считается студент, который получил наибольшее количество пар отличных оценок. Сформируйте два целых массива нужного размера. Задайте оценки с помощью генератора случайных чисел. Найдите лучшего и худшего студентов.

122. [+2] ***Одновременный анализ нескольких двумерных массивов.*** На первом курсе M=40 студентов. Каждый из них в понедельник получает оценку по программированию, во вторник – оценку по математике, в среду – по физике в пределах от 2 до 5 каждая, всего в году N=35 недель. Лучшим считается студент, который наибольшее количество недель продержался без троек (т.е. получал не ниже «4»). Сформируйте три целых массива нужного размера. Задайте оценки с помощью генератора случайных чисел. Найдите лучшего студента.

# Решения

## Сумма элементов в строке двумерного массива

[+2] Создайте двумерный массив целого типа заданного размера, M строк и N столбцов. Задайте значения элементов с помощью генератора случайных чисел в пределах от 2 до 5. Найдите сумму всех элементов в каждой строке. Массив и результаты его анализа направьте в файл протокола.

**Решение.**

```
{
  myofs << "Анализ двумерного массива." << endl;
  const int M=12, N=24;
```

```
  int xy[M][N], sum[M];
  double s, r[M];
  for(int m=0; m<M; m++)
    for(int n=0; n<N; n++)
      xy[m][n]=2+rand()%4;
  for(int m=0; m<M; m++)
  {
    s=0;
    for(int n=0; n<N; n++)
      s+=xy[m][n];
    sum[m]=s;
    r[m]=double(s)/double(N);
  }
  myofs << "   ";
  for(int n=0; n<N; n++)
    if(n%2==0) myofs << setw(4) << n;
  myofs << endl << "     ";
  for(int n=0; n<N; n++)
    if(n%2==1) myofs << setw(4) << n;
  myofs << endl;
  for(int m=0; m<M; m++)
  {
    myofs << "[" << setw(2) << m << "] ";
    for(int n=0; n<N; n++)
      myofs << setw(1) << xy[m][n] << " ";
    myofs   <<   "      "   <<   setw(3)   <<   sum[m]   <<
setprecision(2) << r[m] << endl;
  }
  myofs << endl;
}
```

**Результат в файле протокола:**

```
Анализ двумерного массива.

     0   2   4   6   8  10  12  14  16
       1   3   5   7   9  11  13  15  17
[ 0] 5 5 4 2 4 4 4 5 3 3 2 3 2 5 4 2 2 2   61 3.4
[ 1] 4 5 5 2 3 3 5 2 4 5 3 5 5 3 4 2 5 2   67 3.7
[ 2] 4 2 2 4 2 5 5 5 4 4 4 5 3 2 3 2 2 2   60 3.3
[ 3] 2 4 3 3 2 5 2 3 2 5 5 5 4 2 3 3 4 5   62 3.4
[ 4] 5 3 4 2 2 4 5 4 2 2 5 4 5 2 5 2 5 4   65 3.6
[ 5] 5 4 5 2 5 3 4 2 2 3 5 3 2 3 4 4 4 5   65 3.6
[ 6] 5 4 4 3 3 5 5 2 4 4 4 5 3 4 4 5 2 2   68 3.8
```





```
[ 7] 2 4 5 5 3 4 3 5 3 2 3 2 4 2 2 2 4 5   60 3.3
[ 8] 3 2 4 4 2 4 5 2 5 5 3 5 4 3 4 5 5 5   70 3.9
[ 9] 2 2 3 2 4 2 2 3 5 2 3 4 2 2 4 4 5 2   53 2.9
[10] 3 4 3 4 3 5 4 3 3 5 3 4 2 4 3 3 3 3   62 3.4
[11] 4 4 2 4 3 4 5 3 5 2 4 4 5 5 3 3 4 4   68 3.8
```

## Одновременный анализ нескольких двумерных массивов

[+2] ***Одновременный анализ нескольких двумерных массивов.***

На первом курсе M=40 студентов. Каждый из них в понедельник получает оценку по программированию, во вторник – оценку по математике, в среду – по физике в пределах от 2 до 5 каждая, всего в году N=35 недель. Лучшим считается студент, который наибольшее количество недель продержался без троек (т.е. получал не ниже «4»). Сформируйте три целых массива нужного размера. Задайте оценки с помощью генератора случайных чисел. Найдите лучшего студента.

**Решение.**

```
{
  myofs<<"Анализ нескольких двумерных таблиц."<<endl;
  const int M=20, N=13, K=3;
  bool b;
  // Для оценок M студетнов за N дней по K предметам:
  int x[M][N][K], y[M];
  for(int m=0; m<M; m++)
    for(int n=0; n<N; n++)
      for(int k=0; k<K; k++)
        x[m][n][k]=2+rand()%4;
  // Результат каждого студента:
  for(int m=0; m<M; m++)
  {
    y[m]=0;
    for(int n=0; n<N; n++)
    {
      b=true;
      for(int k=0; k<K; k++)
        if(x[m][n][k]<4) b=false;
      if(b) y[m]++;
    }
```

```
  }
  // Найдем лучший результат:
  int best=0;
  for(int m=0; m<M; m++)
    if(y[m]>y[best])
      best=m;
  myofs<<" Контрольная выдача массива оценок."<<endl;
  for(int m=0; m<M; m++)
  {
    myofs << "[" << setw(2) << m << "]";
    for(int n=0; n<N; n++)
    {
      myofs << "-";
      for(int k=0; k<K; k++)
        myofs << x[m][n][k];
    }
    myofs << endl;
  }
  myofs << " Гистограмма оценок." << endl;
  for(int m=0; m<M; m++)
  {
    myofs << "[" << setw(2) << m << "]";
    myofs << "--" << setw(2) << y[m] << " ";
    for(int j=0; j<y[m]; j++)
      myofs << "*";
    myofs << endl;
  }
  myofs << " Лучший результат:" << y[best]
        << "; достигнут студентами " << endl;
  for(int m=0; m<M; m++)
    if(y[m]==y[best])
      myofs << setw(3) << m;
  myofs << endl << endl;
}
```

**Результат в файле протокола:**

```
Анализ нескольких двумерных таблиц.
 Контрольная выдача массива оценок (обрезана справа).
[ 0]-232-535-233-442-454-223-332-523-322-335-224-422-332
[ 1]-242-234-343-342-225-235-543-433-533-225-545-545-543
[ 2]-452-455-524-425-355-332-352-522-345-334-542-232-532
[ 3]-234-455-444-242-233-233-423-343-345-235-335-355-253
[ 4]-325-535-445-353-235-223-552-353-244-523-443-334-525
[ 5]-533-432-324-552-445-224-332-355-244-445-334-324-225
```





```
[ 6]-535-525-453-455-323-345-245-332-522-522-522-432-542
[ 7]-445-434-342-554-544-234-444-525-525-454-522-553-424
[ 8]-425-243-434-324-525-543-443-554-323-445-244-234-432
[ 9]-435-545-335-334-433-352-552-454-543-435-543-424-334
[10]-252-334-323-432-433-522-443-334-553-245-452-543-424
[11]-335-222-445-234-245-545-444-454-235-343-525-253-553
[12]-543-434-335-355-354-535-255-255-554-252-554-325-235
[13]-352-443-454-552-534-425-325-224-325-525-353-243-252
[14]-525-445-533-452-334-443-332-443-354-345-533-353-223
[15]-233-333-244-533-455-334-333-533-243-352-242-235-534
[16]-545-255-525-433-255-333-234-242-233-445-245-223-322
[17]-432-442-554-434-534-333-422-324-522-443-245-553-532
[18]-444-222-253-345-424-525-555-425-433-354-552-253-523
[19]-544-544-252-332-333-445-453-244-542-434-245-255-234
Гистограмма оценок.
[ 0]-- 5 *****
[ 1]-- 6 ******
[ 2]-- 3 ***
[ 3]-- 1 *
[ 4]-- 2 **
[ 5]-- 2 **
[ 6]-- 3 ***
[ 7]-- 6 ******
[ 8]-- 5 *****
[ 9]-- 5 *****
[10]-- 6 ******
[11]-- 4 ****
[12]-- 5 *****
[13]-- 6 ******
[14]-- 4 ****
[15]-- 3 ***
[16]-- 2 **
[17]-- 6 ******
[18]-- 4 ****
[19]-- 5 *****
 Лучший результат:6; достигнут студентами
  1  7 10 13 17
```

## 14. Двумерные массивы с переменной размерностью

### З а д а ч и

123. [+3] Задайте значения M и N, например, 7 и 9. Создайте одномерный массив указателей на int* размера M. Инициализируйте каждый указатель с помощью оператора new так,

чтобы обеспечить возможность хранения массива из M строк и N столбцов. Задайте значения элементов полученного двумерного массива с помощью генератора случайных чисел. Распечатайте массив в файле протокола.

124. [3] Создайте двумерный массив переменной размерности, в котором имеется 20 стром и 10 столбцов. Задайте значения элементов равными всем целым числам от 0 до 199. Предполагается, что номер строки равен числу десятков, а номер столбца равен числу единиц в сохраняемом числе.

125. [3] Создайте с помощью операторов new двумерный массив, в первой строке которого 1 элемент, во второй строке 2 элемента, и так далее, вплоть до 10 строки, в которой 10 элементов. Задайте их значения. Напечатайте массив в виде таблицы треугольной формы.

126. [+3] Создайте с помощью операторов new двумерный массив, в нулевой строке которого 1 элемент, в первой строке 2 элемента, во второй строке 3 элементов, и так далее, вплоть до 10 строки, в которой 11 элементов. В нулевую строку занесите число 1, в первую числа 1, и 1, во вторую 1, 2, 1, в третью 1, 3, 3, 1, в четвертую 1, 4, 6, 4, 1, в пятую 1, 5, 10, 10, 5, 1, и т.д. Напечатайте массив в виде таблицы треугольной формы.

127. [3] Задано число число M>0 и набор целых чисел $N_0$,…, $N_{M-1}$, все положительны. Задан набор из M числовых последовательностей, длина каждой из которых равна $N_m$. Создайте указатель на int*, инициализируйте его с помощью оператора new. Инициализируйте M указателей на int. Сохраните указанные последовательности в полученном массиве. Для задания всех указанных чисел используйте генератор случайных чисел.

128. [3] Источник информации генерирует и посылает нам слова (не больше M=100 штук), состоящие из символов стандартного набора, длина которых может быть любой в пределах от 1 до K=20 символов. Вы должны принять и сохранить эти слова в двумерном массиве. Длину каждого поступающего слова задайте с помощью генератора случайных чисел. Последовательность символов, образующую очередное слово, также задайте генератором случайных чисел.





# Решения

## Динамические двумерные массивы

[+3] Задайте значения M и N, например, 7 и 9. Создайте одномерный массив указателей типа int* размера M. Инициализируйте каждый указатель с помощью оператора new int[N]. Задайте значения элементов полученного двумерного массива с помощью генератора случайных чисел. Распечатайте массив по строкам в файле протокола.

**Решение.**

```
{
  myofs << "Динамический двумерный массив." << endl;
  const int M=16; const int N=64;
  myofs << " Первый этап: создадим одномерный массив
указателей." << endl;
  int** aa = new int*[M];
  myofs << " Второй этап: инициализируем каждый из них
указателей new." << endl;
  for(int m=0; m<M; m++) aa[m]=new int[N];
  for(int m=0; m<M; m++)
    for(int n=0; n<N; n++)
      aa[m][n]=999.0/double((m-M/2)*(m-M/2)+0.2*(n-
N/2)*(n-N/2)+100.0);
  myofs << " Массив aa можно вывести с помощью вложен-
ных операторов цикла," << endl;
  for(int m=0; m<M; m++){
    myofs << "[" << setw(2) << m << "] ";
    for(int n=0; n<N; n++)
      myofs << aa[m][n];
      myofs << endl;
  }
  myofs << endl;
```

```
  myofs << " Сначала разрушаем каждый из указываемых
обьектов." << endl;
  for(int m=0; m<M; m++){
    if(aa[m]) delete[]aa[m];
    aa[m]=NULL;
  }
  myofs << " После этого разрушаем массив указателей."
<< endl << endl;
  if(aa){ delete[]aa; aa=NULL; }
}
```

**Результат в файле протокола:**

```
Создание динамического двумерного массива.
 Первый этап: создадим одномерный массив указателей.
 Второй этап: инициализируем каждый из них указателей с
помощью оператора new.
 Массив aa можно вывести в файл с помощью вложенных операторов
цикла,

22233333333344444445555555556666666555555555544444433333333322
22333333333444444455555556666666666666666666555555544444433333332
23333333344444455555566666667777777777766666665555544444433333333
33333333444445555566666777777777777777776666665555544444433333333
33333334444455555666677777888888888888777776666555554444433333333
33333344444555556666777788888999999999988888777666655554444433333
33333444445555666677788889999999999999888877766665555444443333
33333444445555666677788889999999999999888877766665555444443333
33333444445555666677788889999999999999888877766665555444443333
33333444445555666677788889999999999999888877766665555444443333
33333444445555666677788889999999999999888877766665555444443333
33333344444555556666777788888999999999888887777666655554444433333
33333334444455555666677777888888888888877777666655555444443333333
33333333444444555556666666777777777777666666655555444444333333333
22333333334444444555555666666666666666666655555544444443333333332

Разрушаем также в два этапа этапа.
Сначала разрушаем каждый из указываемых обьектов.
После этого разрушаем массив указателей.
```

## Двумерный массив треугольной формы

[+3] Создайте с помощью операторов new двумерный массив, в нулевой строке которого 1 элемент, в первой строке 2 элемента, во второй строке 3 элементов, и так далее, вплоть до 10 строки, в которой 11 элементов. В нулевую строку занесите число 1, в первую числа 1, и 1, во вторую 1, 2, 1, в третью 1, 3, 3, 1, в четвертую 1, 4, 6, 4, 1, в пятую 1, 5, 10, 10, 5, 1, и т.д. Напечатайте массив в виде таблицы треугольной формы.





**Решение.**

```
{
  myofs << "Двумерный треугольный массив." << endl;
  int M=8;
  int** x = new int*[M];
  for(int m=0; m<M; m++) x[m]=new int[m+1];
  x[0][0]=1;
  x[1][0]=1; x[1][1]=1;
  for(int m=2; m<M; m++)
  {
    x[m][0]=1; x[m][m]=1;
    for(int n=1; n<m; n++)
      x[m][n]=x[m-1][n]+x[m-1][n-1];
  }
  for(int m=0; m<M; m++){
    myofs << "[" << setw(2) << m << "] ";
    for(int n=0; n<=m; n++)
      myofs << x[m][n] << " ";
      myofs << endl;
  }
  myofs << endl;
  for(int m=0; m<M; m++){
    if(x[m]) delete[]x[m];
    x[m]=NULL;
  }
  if(x){ delete[]x; x=NULL; }
}
```

**Результат в файле протокола:**

```
Двумерный треугольный массив.
[ 0] 1
[ 1] 1 1
[ 2] 1 2 1
[ 3] 1 3 3 1
[ 4] 1 4 6 4 1
[ 5] 1 5 10 10 5 1
[ 6] 1 6 15 20 15 6 1
[ 7] 1 7 21 35 35 21 7 1
[ 8] 1 8 28 56 70 56 28 8 1
```

# 15. Символьные массивы

## З а д а ч и

129. [+1] Создайте массив типа char с помощью оператора new. Инициализируйте этот массив с помощью генератора случайных чисел. Найдите длину строки, находящейся в этом массиве.

130. [1] Создайте массив типа char заданного размера, например, char x[128]. Инициализируйте этот массив с помощью текстовой константы, например, “Компьютер”. Найдите длину строки, находящейся в этом массиве (количество байт).

131. [2] Создайте массив типа char заданного размера, например, char x[128]. Инициализируйте этот массив с помощью текстовой константы, например, «Компьютер». Создайте второй массив той же размерности, например, char y[128]. Используя оператор цикла, перенесите во второй массив текст из первого массива в обратном порядке, т.е. «ретюьпмоК».

132. [2] Возьмите произвольное слово, например, «ток». Создайте символьный массив, содержащий буквы этого слова. Переставьте буквы в случайном порядке с помощью генератора случайных чисел rand(). Позаботьтесь о том, чтобы генератор случайных чисел выдавал последовательность разных чисел для новой позиции символа. Полученное новое слово занесите в файл протокола. Проделайте это 10 раз и прочитайте файл протокола. Найдите осмысленные слова.

133. [+3] Возьмите любое слово, например, «корова». Используя генератор случайных чисел, переставьте его буквы в случайном порядке. Делайте это до тех пор, пока полученное слово не совпадет с начальным словом, прочитанным справа налево. Сколько пришлось ждать, если в исходном слове было 6 букв? Есть ли разница между словами, состоящими из разных букв, типа «огурец» и словами, в которых две буквы совпадают?





134. [+3] ***Статический массив указателей на строки.*** Предположим, что из некоторого источника (например, от генератора случайных чисел) поступают одно за другим M слов разной длины. Создайте массив указателей char* txt[M] с помощью операторов new и обеспечьте сохранение входного потока слов.

135. [+3] [2]***Динамический массив указателей на строки***. Предположим, что из некоторого источника (например, от генератора случайных чисел) поступают одно за другим M слов разной длины. Создайте массив указателей char** txt с помощью операторов new и обеспечьте сохранение входного потока слов.

## Р е ш е н и я

### Статический массив указателей на строки

[+3] Предположим, что из некоторого источника (например, от генератора случайных чисел) поступают одно за другим слова разной длины. Создайте массив указателей типа char* [] с помощью операторов new и обеспечьте сохранение входного потока слов. Распечатайте массив в файле протокола.

**Решение.**

```
{
  myofs << "Массив для хранения текста." << endl;
  const int M=16;
  char* txt[M];
  int len[M];
  for(int m=0; m<M; m++)
  {
    int N=rand()%32+16;
    len[m]=N;
    txt[m]=new char[N+1];
    for (int n=0; n<N; n++)
      txt[m][n] = char(32+rand()%64);
```

[2] От предыдущей задачи отличается типом создаваемого массива указателей, динамический вместо статического.

```
    txt[m][N] = '\0';
  }
  for(int m=0; m<M; m++)
    myofs << "[" << setw(2) << m << "]["
      << setw(2) << len[m] << "]= " << txt[m] << endl;
  for(int m=0; m<M; m++)
    delete[]txt[m];
}
```

**Результат в файле протокола:**

```
Конструируем массив для хранения текста.
[ 0][42]= ZSQK/:!G5@O>HYL0<[&>?QQ_9J&T77,^I6W.((N)\I
[ 1][33]= ]]A[096$W5%*N71R6$:ISE2P7/,7>#6(<
[ 2][35]= )(4&>ZK^$E<8OVL4R)AV0DOK/$SP/=CP:<%
[ 3][28]= \O@@5'QDL=>0&SPCX<\FY^B,[7Z=
[ 4][44]= ^,38K_%AZ5"QL5 _O(VQ?G&Q,5?#^*B6V/#8WQS/%P\F
[ 5][29]= WL$%1UDU94@VRQ.:Y&0-X$G:=R(QJ
[ 6][40]= RGM;^^SIXGY7M%C)%TP33*N=K<J,T$K?^B@65A6:
[ 7][38]= \N9! /Q038/'U88-&"'Z(;8QGVFPVB0E\B@#O=
[ 8][31]= 6_7#47@[T?,M#%B-T]91W& 2.O\V<;W
[ 9][28]= _!7W@VM/)OO$V"I$NLO?486IY>DA
[10][36]= +$/U@^S6=G;S_55\'6&(]B:LU->-1!IR'L"=
[11][20]= V^<1VI;5EG\<3;JW8(JC
[12][30]= WR.]8_77KB6FA:<"_H\'$?*&+[B[SS
[13][28]=  \2?_T"$M4J]VEZ%SF#YCR-89>C'
[14][34]= AR7*W2+P'D56#=A-/A4QF46AZM.S;,LE:-
[15][29]= 6N=:V7,3[!D*5JY2E'":JDGT[,;<\
```

## Динамический массив указателей на строки

[+3] Предположим, что из некоторого источника (например, от генератора случайных чисел) поступают одно за другим M слов разной длины. Создайте массив указателей типа char** txt с помощью операторов new и обеспечьте сохранение входного потока слов.

**Решение.**

```
{
  myofs << "Конструируем динамический массив для хране-
ния текста." << endl;
  int M=6;
  char** txt=new char*[M];
  int* len = new int[M];
  for(int m=0; m<M; m++)
```





```
  {
    int N=rand()%32+16;
    len[m]=N;
    txt[m]=new char[N+1];
    for (int n=0; n<N; n++)
      txt[m][n] = char(-64+rand()%63);
    txt[m][N] = '\0';
  }
  for(int m=0; m<M; m++)
    myofs << "[" << setw(1) << m << "][" << setw(2) <<
len[m] << "]= " << txt[m] << endl;
  for(int m=0; m<M; m++)
    delete[] txt[m];
  delete[] txt;
  delete[] len;
}
```

**Результат в файле протокола:**

```
Конструируем динамический массив для хранения текста.
[0][18]= мэцЦдОЪЙуъДпТзЦмыХ
[1][31]= мАмдЛЬаБСШИЗКцПеЕЩОЯтУДМрИМиилЖ
[2][29]= гШЬЗЩИНЙпГдЛуЭиЫЬлЦБАчхсьЯшоЦ
[3][27]= НЧАЦАйСТЛААЯАцоЩТунеГзПЫЗко
[4][38]= ГлВьаишВмЛКЕчЩБщСйгкБМТгчЧвПНЗщСАХшхЖЯ
[5][32]= ЗиеврЪзХэШМюрКМЕРБЫМухЦечмЫРъЩрь
```

## Символьные массивы

[+3] Возьмите любое слово, например, «корова». Используя генератор случайных чисел, переставьте его буквы в случайном порядке. Делайте это до тех пор, пока полученное слово не совпадет с начальным словом, прочитанным справа налево. Сколько пришлось ждать, если в исходном слове было 6 букв? Есть ли разница между словами, состоящими из разных букв, типа «огурец» и словами, в которых две буквы совпадают?

**Решение.**

Решим задачу для случая, когда работа завершается при повторении слова в обычном порядке.

```
{
  myofs << "Работа с отдельными символами строки.";
  const int M = 64;
  char s[M], t[M];
```

```
  int z[M];
  strcpy(s, "корова");
  myofs << s << endl;
  int k, count=0, N=strlen(s);
  while(true)
  {
    z[0]=rand()%N;
    for(int m=1; m<N; m++)
// Создадим массив различных случайных чисел:
    {
      bool b=true;
      while(b)
      {
        k = rand()%N;
        b=false;
        for(int j=0; j<m; j++)
        {
          if(k==z[j])
          {
            b=true;
            break;
          }
        }
      }
      z[m]=k;
    }
    for(int n=0; n<N; n++)
      t[n]=s[z[n]];
    t[N]='\0';
    myofs << "[" << setw(3) << count << "]";
    for(int k=0; k<N; k++) myofs << z[k];
    myofs << " " << t;
    if(count%5==4) myofs << endl;
    count++;
    if(count>10000)break;
    if(strcmp(t,s)==0)break;
  }
  myofs << endl << endl;
}
```

**Результат в файле протокола:**

```
Тема: Строки. Работа с отдельными символами строки.
корова
```





```
[ 0] 435021 воакро [ 1] 430215 вокроа [ 2] 521340 ароовк
[ 3] 025134 краоов [ 4] 104532 окваор [ 5] 024531 крваоо
[ 6] 214305 ровока [ 7] 152034 оарков [ 8] 130542 оокавр
[ 9] 031524 кооарв [10] 503421 аковро [11] 310425 ооквра
[12] 412035 воркоа [13] 140532 овкаор [14] 402513 вкраоо
[15] 124530 орваок [16] 452130 ваоок [17] 423105 вроока
[18] 134520 ооварк [19] 104352 оквоар [20] 302415 окрвоа
[21] 203541 ркоаво [22] 231504 рооакв [23] 023541 кроаво
[24] 504132 аквоор [25] 423015 врокоа [26] 514320 аоворк
[27] 012345 корова
```

## 16. Форматированный ввод и вывод

### З а д а ч и

136. [+2] Создайте файл data.txt, содержащий значения некоторых величин, снабженных метками, например,

длина 137 высота 76 глубина 45 температура 36

Предполагается, что метки известны составителю кода, однако порядок следования меток во входном файле произволен. Прочитайте значения из файла с помощью меток, которые требуется распознать. Выдайте значения в файл протокола, снабдив соответствующими комментариями. Смоделируйте возможности ошибок, 1) появление значения с неизвестной системе меткой, например, склонение 452. 2) отсутствие значения величины с известной меткой, например, система ожидает ввода чего-то типа «влажность 75», но из-за искажения входного файла там написано нечто другое, например, «вражность 75».

137. [3] [3]В рамках условий предыдущей задачи создайте код, который распознает метку даже в том случае, если один из символов введен ошибочно.

138. [3] [4]Некоторый исполнительный механизм включается в том случае, если введен верный код, который состоит из последовательности символов латинского алфавита заданной длины, например, БВГДЖЗ. Каждый символ передается по

[3] Добавлена в версии 2.
[4] Добавлена в версии 2.

радиосвязи с помощью кодового слова, например, Баня Вилка Галка Домино Железо Замок. Каждая буква алфавита кодируется одним словом. Во время приема и записи кодовых слов может быть допущена ошибка. Создайте код, обеспечивающий проверку кода и включение исполнительного механизма в том случае, если при приеме не более чем одного кодового слова допущена ошибка не белее чем в одной букве.

139. Прочитайте из файла последовательность символов {a…z}, не разделенных пробелами. Результат сохраните в массиве char buf[128] и направьте в файл протокола. Подсчитайте количество появлений каждой буквы.

140. Прочитайте из файла фразу, состоящую из нескольких слов, разделенных пробелами. Результат направьте в файл протокола.

141. [+3] Вам дан файл book.txt размера порядка 250 kb, содержащий текст учебника программирования на языке C++. Слова в файле разделены пробелами. Создайте текстовый массив соответствующей структуры, прочитайте заданный текст из файла и сохраните в массиве каждое прочитанное слово. Выдайте прочитанный текст в файл протокола, причем в каждой строке файла протокола печатайте не более 72 символов, включая пробел.

142. В рамках условия предыдущей задачи сформатируйте выводимый в файл протокола текст так, чтобы длина каждой строки была равна ровно 72 символа. При необходимости добавляйте между выводимыми словами более одного пробела.

143. В рамках условия предыдущей задачи подсчитайте количество каждого из заданных слов-образцов в прочитанном файле. Например, найдите количество слов “array”, “operator”, “function” и т.д.

144. В рамках условия предыдущей задачи составьте словарь данного Вам текста. Создайте текстовый массив-словарь нужной структуры, в котором сохраняйте каждое новое слово и массив целого типа для накопления числа повторений. При





чтении из файла книги каждого нового слова проверяйте, имеется ли оно в словаре. Если имеется, добавьте единицу в соответствующий счетчик. Если не имеется, добавьте это слово в словарь. Подсчитайте, сколько раз появилось в тексте каждое занесенное в словарь слово.

145. В рамках условия предыдущей задачи подсчитайте количество появлений каждой буквы латинского (русского) алфавита в заданном Вам тексте. Найдите среднюю длину слова.

## Р е ш е н и я

### Форматированный ввод и вывод.

[+2] Создайте файл data.txt, содержащий значения некоторых величин, снабженных метками, например,

длина 137 высота 76 глубина 45 температура 36 конец

Предполагается, что метки известны составителю кода. Слово конец означает завершение ввода. Прочитайте значения из файла и выдайте их в файл протокола, снабдив соответствующими комментариями. Смоделируйте возможности ошибок, 1) появление значения с неизвестной системе меткой, например, склонение 452. 2) отсутствие значения величины с известной меткой, например, система ожидает ввода чего-то типа влажность 75, но из-за искажения входного файла там написано нечто другое, например, вражность 75.

**Решение.**

```
{
    myofs << "Тема: Ввод и вывод." << endl;
    myofs << "Упражнение +++13.1237. Ввод данных с мет-
ками." << endl;
    myifs.open("data06.txt");
    char buf[128];
    char* s[]=
{"глубина", "высота", "температура", "длина", "вяз-
кость", "сонливость"};
    int t, x[128], y[128], count=0;
    while(true)
    {
```

```
      myifs >> buf;
      if(strcmp(buf, "конец")==0) break;
      if(strlen(buf)<1) break;
      myifs >> t;
      for(int m=0; m<6; m++)
      {
        if(strcmp(buf, s[m])==0)
        {
          x[count]=t;
          y[count]=m;
          count++;
          break;
        }
      }
    }
    myifs.close();
    myofs << " Контрольная выдача, распознано "
          << count << " значений" << endl;
    for(int n=0; n<count; n++)
      myofs << s[y[n]] << " равна " << x[n] << endl;
    myofs << endl;
}
```

**Файл data06.txt содержит следующий текст:**

```
длина 34 ширина 17 высота 24 глубина 48
температура 36 влажность 38  конец
```

**Результат в файле протокола:**

```
Тема: Ввод и вывод.
Упражнение 13.1237. Ввод данных с метками.
 Контрольная выдача введенного текста, распознано 4
значений
длина равна 34
высота равна 24
глубина равна 48
температура равна 36
```

## Форматированный ввод и вывод.

[+3] Вам дан файл book.txt размера порядка 250 kb, содержащий текст учебника программирования на языке C++. Слова в файле разделены пробелами. Создайте текстовый массив соответствующей структуры, прочитайте заданный текст из файла и со-





храните в массиве каждое прочитанное слово. Выдайте прочитанный текст в файл протокола, причем в каждой строке файла протокола печатайте не более 72 символов, включая пробел.

**Решение.**

```
{
  myofs << "Чтение, анализ и форматирование документа"
<< endl;
  const int L=5;
  char sample[L][100];
  int sc[L];
  for(int n=0; n<L; n++) sc[n]=0;
  ifstream ifs;
  ifs.open("data08.txt");
  for(int n=0; n<L; n++)
    ifs >> sample[n];
  ifs.close();
  for(int n=0; n<L; n++)
    myofs << " sample[" << n << "] " << sample[n] <<
endl;
  const int N=100000;
  char* t[N];
  for(int n=0; n<N; n++)
    t[n]=NULL;
  char buf[10000];
  ifs.open("data09.txt");
  int wordcounter=0, k;
  while(true){
    ifs >> buf;
    if(strcmp(buf, "конец")==0) break;
    for(int n=0; n<L; n++){
      if(strcmp(sample[n], buf)==0) sc[n]++;
    }
    k=strlen(buf);
    if(k<=0) break;
    t[wordcounter] = new char[k+1];
    strcpy(t[wordcounter], buf);
    wordcounter++;
    if(wordcounter>=N) break;
  }
  ifs.close();
  myofs << " Прочитали файл data09.txt и составили сло-
варь." << endl;
```

```
  myofs << " Частотный словарь:" << endl;
  for(int n=0; n<L; n++){
    myofs << " sample[" << n << "] " << setw(12) <<
sample[n] << " ";
    myofs << setw(6) << sc[n] << endl;
  }
  myofs << " Форматированный текст (заимствован у Срау-
струпа)" << endl;
  int charcounter=0;
  for(int n=0; n<wordcounter; n++){
    myofs << t[n] << " ";
    charcounter += strlen(t[n]);
    if(charcounter>72){
      myofs << endl;
      charcounter=0;
    }
  }
  for(int n=0; n<N; n++)
  if(!t[n]) delete[]t[n];
}
```

**Файл data08.txt содержит следующий текст:**

```
файлов файл заголовочных чтение открытие
```

**Файл data09.txt содержит следующий текст:**

```
<текст опущен, он может быть любым.
Подберите только образцы, встречающиеся в вашем тексте>
конец // это признак конца ввода
```

**Результат в файле протокола:**

```
Чтение, анализ и форматирование документа
Прочитали файл data09.txt и составили словарь.
 Частотный словарь прочитанного текста с заданными образцами:
 sample[0]       файлов      5
 sample[1]         файл      4
 sample[2] заголовочных      4
 sample[3]       чтение      2
 sample[4]     открытие      2
Форматированный текст (заимствован у Срауструпа)
```

```
Разбиение программы в расчете на один заголовочный файл больше
подходит для небольших программ, отдельные части которых
не имеют самостоятельного назначения. Для таких программ
допустимо, что по заголовочному файлу нельзя определить,
чьи описания там находятся и по какой причине. Здесь могут
помочь только комментарии. Возможно альтернативное решение:
пусть каждая часть программы имеет свой заголовочный файл,
```





```
в котором определяются средства, предоставляемые другим
частям. Теперь для каждого файла .c будет свой файл .h, определяющий,
что может предоставить первый. Каждый файл .c будет включать
как свой файл .h, так и некоторые другие файлы .h, исходя из
своих потребностей. Попробуем использовать такую организацию
программы для калькулятора. Заметим, что функция error()
нужна практически во всех функциях программы, а сама использует
только <iostream.h>. Такая ситуация типична для функций,
обрабатывающих ошибки. Какое число заголовочных файлов
следует использовать для данной программы зависит от многих
факторов. Большинство их определяется способом обработки
файлов именно в вашей системе, а не собственно в C++. Например,
если ваш редактор не может работать одновременно с несколькими
файлами, диалоговая обработка нескольких заголовочных
файлов затрудняется. Другой пример: может оказаться, что
открытие и чтение 10 файлов по 50 строк каждый занимает существенно
больше времени, чем открытие и чтение одного файла из 500 строк.
В результате придется хорошенько подумать, прежде чем разбивать
небольшую программу, используя множественные заголовочные
файлы. Предостережение: обычно можно управиться с множеством,
состоящим примерно из 10 заголовочных файлов (плюс стандартные
заголовочные файлы). Если же вы будете разбивать программу
на минимальные логические единицы с заголовочными файлами
(например, создавая для каждой структуры свой заголовочный
файл), то можете очень легко получить неуправляемое множество
из сотен заголовочных файлов.
```

# 17. Анализ данных

## З а д а ч и

146. [1] Напишите функцию int f(int m) которая принимает оценку по 100-бальной шкале и возвращает оценку по 5-бальной шкале, например, 5, если m=90—100, 4, если x=80-89, 3, если m=70-79, 2, если m=60-69, 1, если m меньше 60.

147. [2] Сгенерируйте N случайных чисел в диапазоне 1…100, найдите их среднее значение. Получите оценки по 5-бальной шкале и найдите их среднее.

148. [3] Напишите программу, которая поможет слушателям начальной школы изучить умножение. Используйте rand() для выработки двух положительных целых чисел в диапазоне 1…5. Программа должна печатать вопрос типа Сколько будет 2*3? Затем учащийся печатает ответ. Ваша программа проверяет этот ответ. Если он правильный, напечатайте «Очень хорошо!» и затем задайте следующий вопрос на умножение. Ес-

ли ответ неправильный, напечатайте «Нет. Повторите, пожалуйста, снова.» и затем задавайте тот же самый вопрос повторно до получения правильного ответа. Ответы испытуемого сгенерируйте с помощью генератора случайных чисел rand(). Все результаты направьте в файл.

149. [3] Напишите программу, которая поможет слушателям начальной школы изучить сложение, умножение, вычитание. Используйте rand() для выработки двух положительных целых чисел в диапазоне 1…5. Используйте rand() для вбора одной из трех упомянутых операций. Программа должна печатать вопрос типа Сколько будет 2*3? Затем учащийся печатает ответ. Ваша программа проверяет этот ответ. Если он правильный, напечатайте «Очень хорошо!» и затем задайте следующий вопрос на умножение. Если ответ неправильный, напечатайте «Нет. Повторите, пожалуйста, снова.» и затем задавайте тот же самый вопрос повторно до получения правильного ответа. Ответы испытуемого сгенерируйте с помощью генератора случайных чисел rand(). Все результаты направьте в файл.

150. [3] Модифицируйте предыдущую программу так, чтобы для каждого правильного или неправильного ответов печатались разнообразные комментарии типа: Отклики на правильные ответы «Очень хорошо!», «Отлично!», «Чудесная работа!», «Браво», «Cool», «Продолжайте работать так же хорошо!». Отклики на неправильные ответы: «Нет. Попытайтесь, пожалуйста, снова.», «Неверно. Попытайтесь еще раэ.», «Не опускайте руки!», «Нет. Продолжайте ваши попытки.». Используя генератор случайных чисел, выбирайте подходящую реплику для каждого ответа. Используйте структуру switch для представления отклика.

151. [+3] ***Контрольно–обучающая система.*** Напишите интерактивный учебник биологии. Он должен обеспечивать выдачу в случайном порядке пяти вопросов типа Что такое курица (крокодил, береза, муравей, щука)? и принимать ответ типа





курица это 1) рыба, 2) насекомое, 3) птица, 4) земноводное, 5) растение. После опроса поставьте испытуемому оценку.

152. [3] Напишите программу, которая играет в игру «Угадай число» следующим образом. Ваша программа выбирает случайное число, которое должно быть отгадано, в диапазоне от 1 до 100. Затем программа печатает: «Мое число между 1 и 100. Пожалуйста, напечатайте вашу первую догадку.» Затем игрок печатает свою первую догадку. Программа отвечает одним из следующих вариантов: 1) «Отлично! Вы отгадали число!. 2) «Слишком мало. Попытайтесь снова.» 3) «Слишком много. Попытайтесь снова.» Программа должна работать до получения верного ответа. Ответы испытуемого генерируйте с помощью функции rand().

153. [3] Напишите интерпретатор. Предположим, что в файле data.txt находится последовательность слов и чисел типа 13 плюс 4 минус 2 умножить 3 разделить 4 возвести 2 и т.д. Интерпретатор должен прочитать и распознать словесные команды и выполнить необходимые операции. Результаты в протокол.

154. [3] Напишите автопилот. Предположим, что в файле data.txt находится последовательность слов и чисел типа скорость 3 вперед 5 направо 45 вперед 7 направо 90 скорость 5 пушка налево 30 скорость 7 ракета 3 и т.д. Интерпретатор должен прочитать и распознать словесные команды и выполнить необходимые операции. Результаты в протокол.

# Р е ш е н и я

## Контрольно–обучающая система

[+3] Напишите интерактивный учебник биологии. Он должен обеспечивать выдачу в случайном порядке пяти вопросов типа
Что такое курица (крокодил, береза, муравей, щука)?
и принимать ответ типа
курица это 1) рыба, 2) насекомое, 3) птица, 4) земноводное, 5) растение.

После опроса поставьте испытуемому оценку.

**Решение:**

```
{
  myofs << "Обучающая программа. Изучаем биологию." <<
endl;
  char* a[]={"mammal (млекопитающее)", "bird (птица)",
"fish (рыба)", "reptile (земноводное)"};
  char*  b[]={"cow  (корова)",  "chicken  (курица)",
"sturgeon (осетр)", "crocodile (крокодил)"};
  int num=5;
  for(int m=0; m<num; m++)
  {
    int n=0, mb=0, k=1;
    mb = rand()%4;
    cout << b[mb] << " is (это есть) " << endl << endl;
    for (int n=0; n<4; n++)
      cout << n+1 << "  " << a[n] << endl;
    cout << endl << "Enter the key (Введите код): ";
    cin >> k;
    cout << endl;
    if(k-1==mb)
    {
      cout << "Good! Once more? ";
      cout  <<  "(Правильно!  Попробуем  еще  раз?)"  <<
endl;
    }
    else
    {
      cout << "Bad (Неверно). " << b[mb] << " is (это
есть) " << a[mb] << "." << endl;
    }
  }
}
```

**Результат в файле протокола:**

```
Обучающая программа. Изучаем биологию.

cow (корова) is (это есть)
1  mammal (млекопитающее)
2  bird (птица)
3  fish (рыба)
4  reptile (земноводное)
Enter the key (Введите код): 1
```





```
(Правильно! Попробуем еще раз?)

chicken (курица) is (это есть)
1  mammal (млекопитающее)
2  bird (птица)
3  fish (рыба)
4  reptile (земноводное)
Enter the key (Введите код): 3
Bad (Неверно).

и т.д.
```

# 18. Текстовые задачи

## З а д а ч и

155. [2] ***Анализ последовательности данных из файла.*** Торговый дом продает пять различных продуктов, розничная цена которых: продукт 1 — $2.98, продукт 2 — $4.50, продукт 3 — $9.98, продукт 4 — $4.49 и продукт 5 — $6.87. Напишите программу, которая читает из файла последовательность пар чисел, означающих номер продукта и количество, проданное за день. Ваша программа должна использовать оператор switch, который помогает определить розничную цену каждого продукта. Программа должна рассчитать и вывести на экран общую розничную стоимость всех проданных за неделю продуктов.

156. [+2] ***Анализ управляющей последовательности из файла.*** Вероятность поразить цель ракетой типа 1 равна 67%, ракетой типа 2 равна 53% и ракетой типа 3 равна 38%. Напишите программу, которая читает из файла последовательность пяти чисел, каждое из которых может быть равно 1, 2 или 3 и означающих тип выпущенной ракеты в очереди из пяти ракет. Ваша программа должна использовать оператор switch, который помогает определить тип очередной ракеты. Программа должна рассчитать и вывести в файл протокола вероятность поражения цели данной очередью.

157. [+2] [5]***Задача массового обслуживания.*** Станция технического обслуживания имеет M=10 рабочих позиций, на каждой из которых может быть размещен один обслуживаемый объект, обслуживание которого занимает ровно 1 час. На вход поступает поток неисправных объектов. Количество объектов, поступающих за 1 час, является случайной величиной в пределах от N1=4 до N2=15 включительно. В начале первого часа все позиции были свободны. Моделируйте работу станции на протяжении N=100000 часов. Найдите среднее значение времени ожидания и среднюю длину очереди. Результаты представьте в наглядной форме.

## Решения

### Анализ управляющей последовательности из файла

[+2] Вероятность поразить цель ракетой типа 1 равна 67%, ракетой типа 2 равна 53% и ракетой типа 3 равна 38%. Напишите программу, которая читает из файла последовательность пяти чисел, каждое из которых может быть равно 1, 2 или 3 и означающих тип выпущенной ракеты в очереди из пяти ракет. Ваша программа должна использовать оператор switch, который помогает определить тип очередной ракеты. Программа должна рассчитать и вывести в файл протокола вероятность поражения цели данной очередью.

**Решение.**

```
  {
    myofs << "Тема: Текстовая задача."
          << endl;
    myofs << " 7.9752. Моделирование последовательности
операций." << endl;
    myifs.open("data07.txt");
    char buf[128];
    char* type[]={"тополь", "кирпич",
      "буратино", "бегемот", "тигр",
      "дым", "что-то"};
    int t, p, x[128], y[128], z[128], count=0, full=0;
```

[5] Добавлена в версии 2.





```
    double Q=100;
    while(true)
    {
      myifs >> buf;
      if(strcmp(buf, "конец")==0) break;
      if(strlen(buf)<1) break;
      myifs >> p >> t;
      for(int m=0; m<6; m++)
      {
        x[full]=1;
        y[full]=0;
        z[full]=6;
        if(strcmp(buf, type[m])==0)
        {
          x[full]=t;
          y[full]=p;
          z[full]=m;
          count++;
          break;
        }
      }
      full++;
    }
    myifs.close();
    myofs << " Введено " << full
<< " , распознано " << count << " значений" << endl;
myofs << " В начале летело " << Q << " целей." << endl;
    for(int n=0; n<full; n++)
    {
      for (int j=0; j<x[n]; j++)
        Q*=(1.0-double(y[n])/100.0);
      myofs << " Запускаем " << setw(12)
            << type[z[n]] << " очередью из "
            << x[n] << " штук. ";
      myofs <<" Вероятность поражения "
          << setw(2) << y[n] << " процентов.";
      myofs << " Осталось " << Q << " целей";
      myofs << endl;
    }
    myofs << "Вероятность поражения цели данной серией
равна "
          << 100.0-Q << " процентов." << endl << endl;
  }
```

**Файл data07.txt содержит следующий текст:**

```
буратино 17 2 кирпич 28 3 тополь 55 1 мышка 13 7 бегемот 3 8
конец
```

**Результат в файле протокола:**

```
Моделирование последовательности операций.
 Введено 5 значений, распознано 4 значений
 В начале летело 100 целей.
буратино 2 штук.  17 проц.  Осталось 68.89   целей
кирпич   3 штук.  28 проц.  Осталось 25.7131 целей
тополь   1 штук.  55 проц.  Осталось 11.5709 целей
что-то   1 штук.   0 проц.  Осталось 11.5709 целей
бегемот  8 штук.   3 проц.  Осталось  9.0686 целей
Вероятность поражения цели данной серией равна 90.9314
процентов.
```

## Задача массового обслуживания

[+2] Станция технического обслуживания имеет M=10 рабочих позиций, на каждой из которых может быть размещен один обслуживаемый объект, обслуживание которого занимает ровно 1 час. На вход поступает поток неисправных объектов. Количество объектов, поступающих за 1 час, является случайной величиной в пределах от N1=4 до N2=15 включительно. В начале первого часа все позиции были свободны. Моделируйте работу станции на протяжении N=100000 часов. Найдите среднее значение времени ожидания и среднюю длину очереди. Результаты представьте в наглядной форме.

**Решение.**

```
{
  myofs << "Модель массового обслуживания." << endl;
  myofs << " * Обслуживаемый оъект,"
    << " # Ожидающий обьект." << endl;
  int N = 100000; // Время наблюдения;
  int N0=N/10; // Начало накопления результатов;
  int M=10; // Производительность станции;
  int N1=4, N2=15; // Наименьшее и наибольшее
// значения интенсивности входящего потока;
  int K=10; // Столько точек выдадим на печать;
  int s=0, count=0; // Для накопления результата;
  int* x = new int[N];
  x[0]=0; // Начальное состояние станции;
```





```
  for(int n=1; n<N; n++)
  {
    x[n]=x[n-1] + N1+rand()%(N2-N1+1);
    if(n>N0)
    {
      if(x[n]>M) s+=(x[n]-M);
      count++;
    }
    if(n%(N/K)==(N/K-1))
    {
      myofs << "Шаг=" << setw(5) << n << " ";
      for(int m=0; m<x[n]; m++)
      {
        if(m<M) myofs << "*";
        else myofs << "#";
      }
      myofs << endl;
    }
    x[n]-=M;
    if(x[n]<0)x[n]=0;
  }
  myofs << " Среднее время ожидания="
    << double(s)/double(count) << endl << endl;
  delete[] x;
}
```

**Результат в файле протокола:**

```
Модель случайного обслуживания.
Обозначения: * Обслуживаемый оъект, # Ожидающий обьект.
Шаг= 9999 **********#####
Шаг=19999 ****
Шаг=29999 **********########################
Шаг=39999 **********
Шаг=49999 **********############################
Шаг=59999 ********
Шаг=69999 **********############
Шаг=79999 ******
Шаг=89999 *****
Шаг=99999 **********########
```

Среднее время ожидания=10.047

# 19. Пример экзаменационного билета

## З а д а ч и

### 1. Вычисление наименьшего из трех чисел.

[+1] Напишите функцию, которая находит наименьшее число из заданного набора трех целых чисел. Используйте условный оператор if.

**Решение.**

```
int fmin(int x1, int x2, int x3)
{
  // Найдем наименьшее:
  int x=x1;
  if(x2<x) x=x2;
  if(x3<x) x=x3;
  return x;
}
{
  myofs << "Найдем наименьшее из трех чисел." << endl;
  int a1 = fmin(3, 5, 7);
  int a2 = fmin(7, 3, 5);
  int a3 = fmin(5, 7, 3);
  myofs << " fmin(3, 5, 7)=" << a1 << endl;
  myofs << " fmin(7, 3, 5)=" << a2 << endl;
  myofs << " fmin(5, 7, 3)=" << a3 << endl << endl;
}
```

**Результат в файле протокола:**

```
Упражнение +++1728. Найдем наименьшее из трех чисел.
 fmin(3, 5, 7)=3
 fmin(7, 3, 5)=3
 fmin(5, 7, 3)=3
```

### 2. Анализ нескольких двумерных массивов

[+2] ***Одновременный анализ нескольких двумерных массивов.***

На первом курсе M=40 студентов. Каждый из них в понедельник получает оценку по программированию, во вторник – оценку по математике, в среду – по физике в пределах от 2 до 5 каждая, всего в году N=35 недель. Лучшим считается студент, который





наибольшее количество недель продержался без троек (т.е. получал не ниже «4»). Сформируйте три целых массива нужного размера. Задайте оценки с помощью генератора случайных чисел. Найдите лучшего студента.

**Решение.**

```
{
  myofs<<"Анализ нескольких двумерных таблиц."<<endl;
  const int M=20, N=13, K=3;
  bool b;
  // Для оценок M студетнов за N дней по K предметам:
  int x[M][N][K], y[M];
  for(int m=0; m<M; m++)
    for(int n=0; n<N; n++)
      for(int k=0; k<K; k++)
        x[m][n][k]=2+rand()%4;
  // Результат каждого студента:
  for(int m=0; m<M; m++)
  {
    y[m]=0;
    for(int n=0; n<N; n++)
    {
      b=true;
      for(int k=0; k<K; k++)
        if(x[m][n][k]<4) b=false;
      if(b) y[m]++;
    }
  }
  // Найдем лучший результат:
  int best=0;
  for(int m=0; m<M; m++)
    if(y[m]>y[best])
      best=m;
  myofs<<" Контрольная выдача массива оценок."<<endl;
  for(int m=0; m<M; m++)
  {
    myofs << "[" << setw(2) << m << "]";
    for(int n=0; n<N; n++)
    {
      myofs << "-";
      for(int k=0; k<K; k++)
        myofs << x[m][n][k];
```

```
    }
    myofs << endl;
  }
  myofs << " Гистограмма оценок." << endl;
  for(int m=0; m<M; m++)
  {
    myofs << "[" << setw(2) << m << "]";
    myofs << "--" << setw(2) << y[m] << " ";
    for(int j=0; j<y[m]; j++)
      myofs << "*";
    myofs << endl;
  }
  myofs << " Лучший результат:" << y[best]
        << "; достигнут студентами " << endl;
  for(int m=0; m<M; m++)
    if(y[m]==y[best])
      myofs << setw(3) << m;
  myofs << endl << endl;
}
```

**Результат в файле протокола:**

```
Анализ нескольких двумерных таблиц.
 Контрольная выдача массива оценок (обрезана справа).
[ 0]-232-535-233-442-454-223-332-523-322-335-224-422-332
[ 1]-242-234-343-342-225-235-543-433-533-225-545-545-543
[ 2]-452-455-524-425-355-332-352-522-345-334-542-232-532
[ 3]-234-455-444-242-233-233-423-343-345-235-335-355-253
[ 4]-325-535-445-353-235-223-552-353-244-523-443-334-525
[ 5]-533-432-324-552-445-224-332-355-244-445-334-324-225
[ 6]-535-525-453-455-323-345-245-332-522-522-522-432-542
[ 7]-445-434-342-554-544-234-444-525-525-454-522-553-424
[ 8]-425-243-434-324-525-543-443-554-323-445-244-234-432
[ 9]-435-545-335-334-433-352-552-454-543-435-543-424-334
[10]-252-334-323-432-433-522-443-334-553-245-452-543-424
[11]-335-222-445-234-245-545-444-454-235-343-525-253-553
[12]-543-434-335-355-354-535-255-255-554-252-554-325-235
[13]-352-443-454-552-534-425-325-224-325-525-353-243-252
[14]-525-445-533-452-334-443-332-443-354-345-533-353-223
[15]-233-333-244-533-455-334-333-533-243-352-242-235-534
[16]-545-255-525-433-255-333-234-242-233-445-245-223-322
[17]-432-442-554-434-534-333-422-324-522-443-245-553-532
[18]-444-222-253-345-424-525-555-425-433-354-552-253-523
[19]-544-544-252-332-333-445-453-244-542-434-245-255-234
Гистограмма оценок.
[ 0]-- 5 *****
[ 1]-- 6 ******
```





```
[ 2]-- 3 ***
[ 3]-- 1 *
[ 4]-- 2 **
[ 5]-- 2 **
[ 6]-- 3 ***
[ 7]-- 6 ******
[ 8]-- 5 *****
[ 9]-- 5 *****
[10]-- 6 ******
[11]-- 4 ****
[12]-- 5 *****
[13]-- 6 ******
[14]-- 4 ****
[15]-- 3 ***
[16]-- 2 **
[17]-- 6 ******
[18]-- 4 ****
[19]-- 5 *****
 Лучший результат:6; достигнут студентами
  1   7 10 13 17
```

## 3. Задача случайного обстрела без прогноза

[+3] Комплекс состоит из M одинаковых объектов, каждый из которых в начале находится в исправном состоянии. Противник производит одна за другой атаки, в ходе каждой из которых он запускает K боеголовок, причем цели – объекты нашего комплекса – выбираются случайным образом. Противник не знает, какие объекты уже поражены и может обстрелять пораженный объект еще раз. Сколько атак переживет комплекс, если для обеспечения жизнедеятельности достаточно иметь N исправных объектов, причем M=16, K=3, N=4. Результаты представьте в наглядной форме.

**Решение.**

```
{
  myofs << " Модель случайного размещения." << endl;
  myofs << " 1 исправный объект,"
  << " * уничтоженный объект," << " . промах." << endl;
  const int N = 64, M=12, J=1;
  char buf[N+1];
  int count, k;
  int x[N], z[M];
```

```
  for(int n=0; n<N; n++) x[n]=1;
  while(true)
  {
    z[0]=rand()%N;
    for(int m=1; m<M; m++)
    {
      bool b=true;
      while(b)
      {
        k = rand()%N;
        b=false;
        for(int j=0; j<m; j++)
        {
          if(k==z[j])
          {
            b=true;
            break;
          }
        }
      }
      z[m]=k;
    }
    for(int n=0; n<N; n++)
    {
      if(x[n]==1) buf[n]='1';
      else buf[n] = ' ';
      buf[N] = '\0';
    }
    for(int m=0; m<M; m++)
      if(x[z[m]]==1)
        buf[z[m]] = '*';
      else buf[z[m]] = '.';
    myofs << buf << endl;
    for(int m=0; m<M; m++)
      x[z[m]]=0;
    count=0;
    for (int n=0; n<N; n++) count+=x[n];
    if(count<J) break;
  }
}
```

**Результат в файле протокола:**

```
Тема: случайные числа, Модель случайного размещения.
```





```
 1 исправный объект, * уничтоженный объект, . промах.
11111111111111111111111111111111111111111111111111111111111111111111
11*111* 11*11 1*** 111*11 111 . 1 11.*111111111*1111111 1 *1111
1*.111  11.11 1   .111.11 *11 . 1.11. 111111111 1*11111 1  *1*1
1 .111  1* 11 *    111.11  11. .1 11 .1111***11 1 *1111 1   * 1
1. 1**  1  11      111 *1. 1*   * *1  1111   11 1  *111 *    .*
1  *.. .1  *1  .  .111  1  1       1. 1111.  11.1   111     ..
1 .     1.  1     .111. 1 .1       1. 11**  .1* 1   111 .     .
1       1   *. ... 11*  1  1   .  .1  11     *  1   1*1. .
1   .   *    .    .11   1  1  ...  1  11      . 1 . * *.
*       .   .      11   1  1    . .1  11        *   .     . ....
  ..   .     . .  .11   1  1    .  1  11  . .         . .     .
  . .  .        .. 1*.  *  1.      *  11        .    .
.           ..  .  1  . .  1      .   11.  .      . .          .
      .     .     1      .1  .  .    **   .. .     .   .
.     ..          1   . . 1 . .    .           .   .   .   .
   .          .   .1       *        .        .    ....    . .
. .      .   .     1 .          .. .   . .             .    .
  .   .     . ..  .*      . . .                        . .
```