

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Курганский государственный университет»

Кафедра программного обеспечения
автоматизированных систем

ИЗУЧЕНИЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ C++

Методические рекомендации
для подготовки бакалавров и специалистов
направлений 09.03.03 «Прикладная информатика», 09.03.04 «Программная
инженерия», 10.05.01 «Компьютерная безопасность»,
10.05.03 «Информационная безопасность»

Курган 2021

Кафедра: «Программное обеспечение автоматизированных систем».

Дисциплины: «Информатика и программирование» (09.03.03), «Основы программирования» (09.03.04), «Язык программирования» (10.05.01, 10.05.03).

Составил: канд. пед. наук, доцент А. А. Медведев.

Печатается в соответствии с планом издания, утвержденным методическим советом университета «10» декабря 2020 г.

Утверждены на заседании кафедры «01» июля 2021 г.

Лабораторная работа № 1. Основные управляющие конструкции C++

Цель работы – изучить использование указанных конструкций при обработке числовой информации.

1 Фрагмент теории

Основные типы данных в языке программирования C++ (диапазоны значений, а также количество занимаемых байт переменной каждого типа зависит от компилятора):

- **char** – символьный;
- **short** – короткий целый;
- **int** – целый;
- **long** – длинный целый;
- **float** – вещественный;
- **double** – вещественный, с удвоенной точностью;
- **void** – отсутствие значения.

Операция явного преобразования (приведения) типа имеет две различные формы:

- $(\langle \text{имя_типа} \rangle) \langle \text{операнд} \rangle$. Позволяет преобразовать значение операнда к нужному типу. В качестве операнда используется унарное выражение, которое в простейшем случае может быть переменной, константой или любым выражением, заключенным в круглые скобки. Например, следующие преобразования изменяют длину внутреннего представления целой константы 1, не меняя ее значения:

- ✓ **(long) 1** – внутреннее представление занимает место в памяти как значение типа «длинное целое»;
- ✓ **(char) 1** – внутреннее представление занимает место в памяти как значение символьного типа;

- $\langle \text{имя_типа} \rangle (\langle \text{операнд} \rangle)$. Функциональная форма преобразования типа используется в тех случаях, когда тип имеет простое (несоставное) наименование (обозначение), например:

- ✓ **long (2)** – внутреннее представление занимает место в памяти как значение типа «длинное целое»;
- ✓ **double (2)** – внутреннее представление занимает место в памяти как вещественное значение, с удвоенной точностью.

Однако будет недопустимым выражение: **unsigned long(2)**.

1.1 Условные конструкции

Общий вид условной конструкции следующий:

```
if (<выражение>) { // Блок, если условие истинно
    <оператор1>; [<оператор2>; ... <операторN>;]
}
[else { // Блок, если условие ложно
    <операторN+1>; [<операторN+2>; ... <операторM>;] } ] .
```

Конструкции, расположенные в квадратных скобках, могут отсутствовать. Фигурные скобки могут отсутствовать, если блок содержит только один оператор.

Примеры:

```
if (x > 0) {x = -x; a = 2 * b;} // Полная условная конструкция.  
else { int i = 2; x *= i; a = b / x; }  
if (a < 0) a = -a; // Сокращенная условная конструкция.
```

При использовании блоков (то есть составных операторов с определениями и описаниями) нельзя забывать о локализации определяемых в блоке объектов. Так, например, в примере с полной условной конструкцией «время жизни» переменной **i** ограничено блоком, в котором она описана и используется; вне этого блока эта переменная перестанет существовать.

Помимо рассмотренной условной конструкции имеется еще *переключатель*, общий вид которого следующий:

```
switch (<переключающее выражение>) {  
    case <константное_выражение_1>:  
        <операторы_1>;  
    case <константное_выражение_2>:  
        <операторы_2>;  
    . . . . .  
    case <константное_выражение_n>:  
        <операторы_n>;  
    [ default: <операторы>; ]  
}
```

Управляющая конструкция **switch** передает управление к тому из помеченных с помощью **case** операторов, для которого значение константного выражения совпадает со значением переключающего выражения. Переключающее выражение должно быть целочисленным или его значение приводится к целому. Значения константных выражений, помещаемых за служебными словами **case**, приводятся к типу переключающего выражения. В одном переключателе все константные выражения должны иметь различные значения, но быть одного типа. Любой из операторов, помещенных в фигурных скобках после конструкции **switch**, может быть помечен одной или несколькими метками **case**. Если значение переключающего выражения не совпадает ни с одним из константных выражений, то выполняется переход к оператору, отмеченному меткой **default**. В каждом переключателе должно быть не больше одной метки **default**, однако эта метка может и отсутствовать. В этом случае при несовпадении переключающего выражения ни с одним из константных выражений, помещаемых вслед за **case**, в переключателе не выполняется ни один из операторов.

Метки *не изменяют* последовательности выполнения операторов. Если не предусмотрены переходы или выход из переключателя, то в нем *последовательно выполняются все операторы*, начиная с той метки, на которую передано управление.

Пример: приведенный ниже фрагмент программы выводит названия нечетных целых цифр, не меньших заданной (**ic** – переменная целого типа, содержащая какое-нибудь число):

```

switch (ic)
{ case 0: case 1: cout << "один, ";
  case 2: case 3: cout << "три, ";
  case 4: case 5: cout << "пять, ";
  case 6: case 7: cout << "семь, ";
  case 8: case 9: cout << "девять ";
    break; //Выход из переключателя
  default: cout << "Ошибка! Это не цифра!";
}

```

Приведенный фрагмент также иллюстрирует действие конструкции **break**. С его помощью осуществляется выход из переключателя. Если поместить **break** после вывода каждой цифры, то будет выводиться название только одной нечетной цифры.

1.2 Циклические конструкции

Существуют три конструкции цикла.

Цикл с неизвестным числом повторений. Его общий вид:

```

while (<условие_или_выражение>
      <тело_цикла>;

```

При входе в цикл проверяется условие или вычисляется выражение. Если его значение отлично от нуля, то выполняется тело цикла. Процесс продолжается, пока значение конструкции, стоящей после служебного слова **while** не станет равным 0.

Цикл с постусловием. Его общий вид:

```

do
  <тело_цикла>
while (<выражение_или_условие>;

```

При входе в цикл **do** обязательно выполняется тело цикла. Затем вычисляется выражение или проверяется условие и, если его значение не равно нулю, вновь выполняется тело цикла.

Цикл с известным заранее числом повторений. Его общий вид:

```

for (<инициализация_цикла>; <выражение_условие>;
     <список_выражений>)
  <тело_цикла>;

```

Здесь *инициализация_цикла* – это последовательность определений (описаний) и выражений, разделяемых запятыми. Все выражения, входящие в инициализацию цикла, вычисляются только один раз при входе в цикл. Чаще всего здесь устанавливаются начальные значения счетчиков и параметров цикла. Под *выражением_условием* понимается конструкция, ранее описанная в циклах **while** и **do**: если она равна 0, то выполнение цикла прекращается. В случае отсутствия *выражения_условия* следующий за ним разделитель «точка с запятой» сохраняется. При отсутствии *выражения_условия* предполагается, что его значение всегда истинно. Это справедливо также в случае отсутствия секции инициализация цикла. Выражения из *списка_выражений* вычисляются на каждой итерации после выполнения операторов тела цикла. Тело цикла может быть отдельным оператором, составным оператором или пустым оператором.

Пример: приведем фрагмент программы нахождения суммы квадратов первых **k** натуральных чисел с использованием различных циклических конструкций:

```
int i = 1, s = 0;
while (i <= k) {
    s += i * i;
    i++;
}
```

```
int i = 1, s = 0;
do {
    s += i * i;
    i++;
}
while (i <= k);
```

```
for (int i = 1, s = 0; i <= k; i++)
    s += i * i;
```

В этом примере цикл **for** можно записать таким образом:

```
for (int i = 0, s = 0; i < k; s += ++i * i);
```

Для создания бесконечного цикла нужно подобрать такое *выражение_условие*, которое бы не изменялось в процессе выполнения цикла, оставаясь постоянно истинным, или отсутствовало. Приведем примеры заголовков бесконечных циклов:

```
for (; ;) ;
for (; 1; ) .
```

1.3 Дополнительные конструкции

Перечислим конструкции, которые достаточно часто применяются совместно с циклами.

Оператор continue позволяет перейти к следующей итерации цикла до завершения выполнения всех инструкций внутри цикла.

Оператор break позволяет осуществить выход из цикла или переключателя, т. е. передать управление конструкции, идущей за циклом.

2 Задачи для самостоятельного решения

1.1 *Пифагоровы числа*. Три натуральных числа a , b и c образуют пифагорову тройку, если $c^2 = a^2 + b^2$. Пифагорова тройка называется *основной*, если наибольший общий делитель ее чисел равен единице. Например, 3, 4, 5 – основная тройка, 6, 8, 10 – производная тройка. Найдите все основные пифагоровы тройки, числа в которых не превышают данное число \max .

1.2 Напишите программу нахождения всех решений на отрезке $[N; M]$ уравнения $x^2 + y^2 = z^n$, n – известное натуральное число.

1.3 Треугольники, у которых длины сторон и площадь являются натуральными числами, называются *треугольниками Герона*. Например, площадь треугольника со сторонами 13, 14 и 15 равна 84. Найдите n треугольников Герона.

1.4 Найдите n треугольников Герона, длины сторон каждого из которых являются последовательными числами (например: 13, 14, 15).

1.5 Найдите n треугольников Герона, у которых площадь равна периметру.

1.6 Найти все прямоугольники с целыми длинами сторон данной целочисленной площади. Например, для площади, равной 12, подходят три прямоугольника: $1 * 12$, $2 * 6$, $3 * 4$.

1.7 Напишите программу для нахождения всех различных прямоугольных параллелепипедов, объем которых равен V , а ребра выражены натуральными числами. Параллелепипеды, получающиеся один из другого, если поменять ребра местами, считаются одинаковыми.

1.8 Даны числа \min и \max . Найдите все треугольники с целочисленными длинами сторон от \min до \max включительно.

1.9 Напишите программу, которая определяет, можно ли из четырех отрезков с данными длинами a , b , c и d составить прямоугольник.

1.10 Найдите наименьшее натуральное число, которое можно представить в виде суммы кубов двух натуральных чисел не единственным способом.

1.11 Напишите программу, которая определяет, сколько можно купить быков, коров и телят, платя за быка 10 рублей, за корову – 5 рублей, а за теленка – 50 копеек, если на 100 рублей надо купить 100 голов скота?

1.12 Л. Кэрролл в своем дневнике писал, что он тщетно трудился, пытаясь найти хотя бы три прямоугольных треугольника равной площади, у которых длины сторон были бы выражены натуральными числами. Составьте программу для решения этой задачи, если известно, что такие треугольники существуют.

1.13 Напишите программу, которая находит все прямоугольные треугольники (длины сторон выражаются натуральными числами), площадь которых не превышает данного числа S .

1.14 Числа, запись которых состоит из двух одинаковых последовательностей цифр, называются *симметричными*. Например, 357357 или 17421742. Определите, является ли данное натуральное число симметричным.

1.15 Если сложить все цифры какого-либо натурального числа, затем – все цифры найденной суммы и так далее, то в результате получим однозначное число (цифру), которое называется цифровым корнем данного числа. Например, цифровой корень числа 561 равен 3 ($5 + 6 + 1 = 12$, $1 + 2 = 3$). Найдите числовой корень данного натурального числа.

1.16 В книге n страниц. Найдите количество цифр, необходимое для нумерации всех страниц такой книге.

1.17 Зная количество понадобившихся для нумерации цифр, определить количество страниц в книге.

1.18 Номера троллейбусных билетов представляют собой шестизначные числа. Счастливым считается тот билет, у которого сумма первых цифр равна сумме трех последних цифр. Например, билет 627 294 считается счастливым, так как $6 + 2 + 7 = 2 + 9 + 4 = 15$. Найдите все номера счастливых билетов, такие, что из них можно извлечь натуральный корень какой-либо (превышающей 1) степени. Например, квадратный корень из числа 720801 равен 849.

1.19 Составьте программу для нахождения всех номеров счастливых билетов, у которых сумма первых (или последних) трех цифр, будучи возведенной в какую-либо степень, равна номеру счастливого билета.

1.20 Найдите все n -значные ($2 < n < 9$) числа, которые состоят из разных цифр и являются полными квадратами.

1.21 Найдите наименьшее число, оканчивающееся на 5, такое, что если перенести его последнюю цифру в начало, то число увеличится в пять раз.

1.22 Составьте программу, в которой входное данное – натуральное число n из отрезка $[2; 9]$, а результат – наименьшее число, у которого первая цифра – n и из которого, перенеся первую цифру в конец, получается новое число, в n раз меньшее, нежели искомое.

1.23 Пусть сберегательные банки по бессрочным вкладам выплачивают r % годовых от суммы вклада, присоединяемых к вкладу. Если вкладчик не снимает деньги с вклада, то проценты ежегодно начисляются со все большей суммы. Найдите величину вклада через m лет.

1.24 Составьте программу для проверки, можно ли заданное натуральное число представить в виде: а) произведения двух простых чисел; б) куба какого-либо простого числа.

1.25 Составьте программу для проверки, можно ли заданное натуральное число представить в виде: а) квадрата какого-либо простого числа; б) произведения трех простых чисел.

1.26 Найдите количество прямоугольных треугольников с целочисленными сторонами, меньшими 100.

1.27 Дано n кирпичей. Вы и компьютер ходите поочередно. За ход можно взять 1, 2 или 3 кирпича. Проиграл тот, кому нечего брать. Реализуйте игру с компьютером. Компьютер ходит случайно (без анализа выигрышной стратегии), однако если у него есть ход, который является последним для его выигрыша, то он его совершает.

1.28 Дано число k . Определите, существует ли такое число n , что $1 + 2 + 3 + \dots + n = k$.

1.29 Найдите, сколько точек с целочисленными координатами попадает в круг радиуса r с центром в точке (x, y) .

1.30 Найдите все натуральные числа, не превосходящие 10000, сумма цифр каждого из которых в некоторой степени равна самому числу.

1.31 Найдите n пар простых чисел, которые отличаются друг от друга на 2.

1.32 Назовем автобусный билет несчастливым, если сумма цифр его шестизначного номера делится на 13. Могут ли два идущих подряд билета оказаться несчастливыми? В случае положительного ответа вывести их номера.

1.33 Выведите на экран квадрат из нулей и единиц, причем нули находятся только на диагонали квадрата. Всего в квадрате сто цифр.

1.34 Найдите хотя бы одно натуральное число, которое делится на 11, а при делении на 2, 3, 4, ..., 10 дает в остатке 1.

1.35 Сколько существует четырехзначных чисел, которые в 600 раз больше суммы своих цифр? Вывести эти числа.

1.36 Вывести все пятизначные числа, которые делятся на 2, у которых средняя цифра нечетная, и сумма всех цифр делится на 4.

1.37 Вывести на экран числа от 1000 до 9999, такие, что все цифры различны.

1.38 Начав тренировки, лыжник в первый день пробежал 10 км. Каждый следующий день он увеличивал пробег на 10 % от пробега предыдущего дня. Определите: а) пробег лыжника за второй, третий, ..., десятый день тренировок; б) какой суммарный путь он пробежал за первые 7 дней тренировок; в) суммарный путь за n дней тренировок; г) в какой день ему следует прекратить увеличивать пробег, если он не должен превышать 80 км?

1.39 Вычислите $1 * 2 + 2 * 3 + 3 * 4 + \dots + n * (n+1) + \dots + 2n$.

1.40 Даны целое число k , $1 < k < 180$ и последовательность цифр 10111213...9899, в которой выписаны подряд все двузначные числа. Определить k -ю цифру в этой последовательности.

1.41 Даны два прямоугольника, стороны которых параллельны или перпендикулярны осям координат. Известны координаты левого нижнего угла каждого из них и длины их сторон. Один из прямоугольников назовем первым, другой – вторым. Определить: а) принадлежат ли все точки первого прямоугольника второму; б) принадлежат ли все точки одного из прямоугольников другому; в) пересекаются ли эти прямоугольники.

1.42 Даны целочисленные координаты трех вершин прямоугольника, стороны которого параллельны координатным осям. Найдите координаты его четвертой вершины (после проверки введенных данных на правильность).

1.43 Даны числа h и m , где h – количество часов, m – количество минут некоторого момента времени. Найдите угол между часовой и минутной стрелками в этот момент времени.

1.44 Дано четырехзначное число. Верно ли, что цифры в нем расположены по убыванию или возрастанию? Например, 4311 – нет, 4321 – да, 5567 – нет, 5679 – да, 9871 – да.

1.45 Даны две даты, каждая из которых состоит из трех чисел (день, месяц и год). Определить, какая дата идет первой.

1.46 Дан прямоугольник размером A на B . Сколько квадратов со стороной C можно вырезать из него?

1.47 Будем говорить, что число a лучше числа b , если сумма цифр a больше суммы цифр числа b , а в случае равенства сумм их цифр – если число a меньше числа b . Например, число 124 лучше числа 123, так как у первого из них сумма цифр равна семи, а у второго – шести. Также, число 3 лучше числа 111, так как у них равны суммы цифр, но первое из них меньше.

Дано число n . Найдите такой его делитель (само число n и единица считаются делителями числа n), который лучше любого другого делителя числа n .

1.48 Таблицей умножения назовем таблицу размера n строк на m столбцов, в которой на пересечении i -ой строки и j -го столбца стоит число $i * j$ (строки и столбцы нумеруются с единицы).

В одной из математических школ было решено провести педагогический эксперимент. Для того, чтобы ученикам было проще запоминать таблицу умножения, некоторые числа в ней будут покрашены в красный, некоторые – в синий, а некоторые – в зеленый цвет (оставшиеся числа будут черными).

Процесс покраски чисел можно условно разбить на четыре этапа. На первом этапе все числа красятся в черный цвет. На втором – все четные числа красятся в красный цвет, на третьем – все числа, делящиеся на 3, красятся в зеленый цвет, на четвертом – все числа, делящиеся на 5, красятся в синий цвет.

Директор школы хочет знать, какое количество картриджей для принтеров необходимо закупить для печати таблиц. Поэтому ему необходима информация о том, сколько чисел какого цвета будет в одной раскрашенной таблице умножения n на m . Напишите программу, решающую задачу подсчета соответствующих количеств.

1.49 На шахматной доске 8×8 расположены три фигуры: ферзь, ладья и конь. Требуется определить количество пустых полей доски, которые находятся под боем. Для простоты будем полагать, что фигуры могут «бить» через другие фигуры. Например, в рассмотренной справа ситуации (рисунок 1) будем считать, что ферзь бьет D5 через ладью.

На вход программе задаются координаты расположения трех фигур: ферзя, ладьи и коня соответственно. Каждая координата состоит из одного английского символа (от A до H) и одной цифры (от 1 до 8). Вывести количество пустых полей, которые бьют указанные во входных данных фигуры.

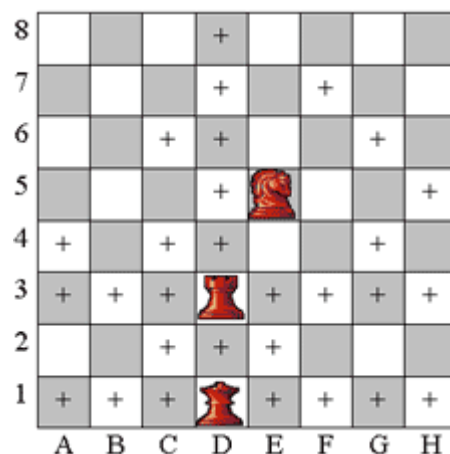


Рисунок 1 – Расположение фигур на шахматной доске

1.50 Простым числом будем называть натуральное число, большее единицы и делящееся только на единицу и на само себя. Выпишем все простые числа в порядке возрастания и i -е в этом порядке число обозначим p_i (число 2 при этом будет иметь номер 1). Так, например, $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_{52} = 239$.

Скажем, что число p_i является *сверхпростым*, если $i = p_k$ для некоторого k . Иными словами, сверхпростое число – это простое число, номер которого в списке простых чисел, упорядоченном по возрастанию, является простым числом.

Дано натуральное число k . Упорядочим все сверхпростые числа по возрастанию. Найдите k -е сверхпростое число в этом порядке.

Лабораторная работа № 2. Указатели и массивы в C++

Цель работы – познакомиться с организацией массивов и ввести понятие указателя.

1 Фрагмент теории

Специальными объектами в программах на языке C++ являются *указатели*. Значениями указателей являются адреса участков памяти, выделенных для объектов конкретных типов. Поэтому в определении и описании указателя все-

гда присутствует обозначение соответствующего ему типа. Эта информация позволяет в дальнейшем с помощью указателя получить доступ к объекту.

Определение указателя на некоторый объект имеет вид:

```
<тип> *<имя указателя>;
```

Таким образом признаком указателя при лексическом разборе определения описания служит символ «*», помещенный перед именем. Например, определение

```
int *i1p, *i2p, *i3p, i;
```

вводит три указателя на объекты целого типа **i1p**, **i2p**, **i3p** и одну переменную **i** целого типа.

При определении указателя в большинстве случаев целесообразно осуществить его *инициализацию*, то есть присвоить ему первоначальное значение. В качестве этого значения может выступать:

- явно заданный адрес участка памяти;
- указатель, уже имеющий значение;
- выражение, позволяющее получать адрес объекта с помощью операции «&».

Если значение константного выражения равно нулю, то это нулевое значение преобразуется к пустому (иначе нулевому) указателю. Примеры:

```
char cc = 'd'; // Символьная переменная (типа char);
char *pc = &cc; // Инициализированный указатель
                // на объект типа char;
char *ptr (null); // Нулевой указатель на объект типа char;
char *p;         // Неинициализированный указатель
                // на объект типа char.
```

Переменная **cc** инициализирована значением символьной константы **'d'**. После определения (с инициализацией) указателя **pc** доступ к значению переменной **cc** возможен как с помощью ее имени, так и с помощью адреса, являющегося значением указателя переменной **pc**. В последнем случае должна применяться операция разыменования «*» (получение значения через указатель). Например, при выполнении строки

```
cout << "\n cc = " << cc << " и *pc = " << *pc;
```

будет выведено

```
cc = d и *pc = d.
```

Присвоив указателю адрес конкретного участка памяти, можно с помощью операции разыменования не только получать, но и изменять содержимое этого участка памяти. Например, оператор присваивания: ***pc = '+'** сделает значением переменной **cc** символ **'+'**.

Операции **new** и **delete** используются для динамического распределения памяти. Операции:

```
<указатель> = new <имя типа>;
<указатель> = new <имя типа>[ (<инициализатор>) ];
```

позволяют выделить и сделать доступным свободный участок в основной памяти, размеры которого соответствуют типу данных, определяемому именем типа. В выделенный участок заносится значение, определяемое инициализатором, который не является обязательным элементом. В случае успешного выполнения

операция **new** возвращает адрес начала выделенного участка памяти. Если участок нужных размеров не может быть выделен (нет памяти), то операция **new** возвращает нулевое значение адреса (**null**).

Необязательный *инициализатор* – это выражение в круглых скобках, значением которого заполняется выделенная область памяти. Указатель, которому присваивается получаемое значение адреса, должен относиться к тому же типу данных, что и *имя_типа* в операции **new**. Примеры:

- оператор **float* f = new float;** определяет указатель **f**, выделяет участок памяти для размещения значения вещественного типа и присваивает адрес его начала указателю **f**;
- операторы **int* ip; ip = new int(15);** определяют указатель **ip**, выделяют участок памяти для размещения значения целого типа, помещают туда значение 15 и присваивают адрес его начала переменной **ip**. В дальнейшем доступ к выделенному участку памяти обеспечивает выражение ***ip**.

В случае отсутствия в операции **new** инициализатора значение, которое заносится в выделенный участок памяти, не определено. Если в качестве имени типа в операции **new** используется массив, то для массива должны быть полностью определены все размерности.

Продолжительность существования выделенного с помощью операции **new** участка памяти – от точки создания до конца программы или до явного его освобождения.

Для явного освобождения выделенного операцией **new** участка памяти используется операция **delete**, общий вид которой следующий:

```
delete <указатель>;
```

где *указатель* адресует освобождаемый участок памяти, ранее выделенный с помощью операции **new**. Например: **delete ip;** освободит участок памяти, связанный с указателем **ip**. Повторное применение операции **delete** к тому же указателю дает неопределенный результат.

Для освобождения памяти, выделенной для массива, используется следующая модификация того же оператора:

```
delete[] <указатель>;
```

где указатель связан с выделенным для массива участком памяти.

Объекты, создаваемые с помощью операции **new**, и удаляемые с помощью операции **delete**, называются объектами с *динамической продолжительностью существования*. В отличие от них, объектам со *статической продолжительностью существования* память выделяется в начале выполнения программы и сохраняется до окончания ее обработки.

При работе с указателями часто используется операция «&» – получение адреса объекта. Для нее существуют следующие ограничения:

- нельзя определять адрес неименованной константы, то есть недопустимы выражения **&3.141593** или **&'?'**;
- нельзя определять адрес значения, получаемого при вычислении скалярных выражений, то есть недопустимы конструкции: **&(44 * x - z)** или **&(a + b) != 12**.

Таким образом, операция «&» применима к объектам, имеющим имя и размещенным в памяти.

Над указателями можно выполнять следующие операции:

- операция разыменования или доступа по адресу «*»;
- преобразование типов (приведение типов);
- присваивание;
- получение (взятие) адреса «&»;
- сложение и вычитание (аддитивные операции);
- инкремент или автоувеличение («++»);
- декремент или автоуменьшение («--»);
- операции отношений (операции сравнения).

С понятием указателя неразрывно связано понятие *массива*.

При определении массива ему выделяется память так же, как массивам других алгоритмических языков. Но как только память для массива выделена, имя массива воспринимается как *константный указатель* того типа, к которому отнесены элементы массива.

Определение одномерного массива имеет вид:

```
<тип_эл-в_массива> <имя_массива> [<кол-во_эл-в_в_массиве>];
```

Нумерация элементов в массиве начинается с нуля! При определении массива может выполняться его инициализация, то есть элементы массива получают конкретные значения. Примеры:

```
char CH[ ] = { 'A', 'B', 'C', 'D' }; // Массив из 4 элементов;  
int pr[6] = {10, 20, 30, 40}; // Массив из 6 элементов.
```

Имя массива является указателем-константой, значением которой служит адрес первого элемента массива (с индексом 0). Таким образом, доступ к первому элементу массива может быть осуществлен так:

```
<имя массива>[0] или *<имя массива>
```

В общем случае доступ к заданному элементу массива можно осуществить двумя способами:

```
<имя массива> [<номер элемента>]
```

или

```
*(<имя массива> + <номер элемента>)
```

Многомерный массив представляет собой массив массивов, то есть массив, элементами которого служат массивы. Определение многомерного массива в общем случае должно содержать сведения о типе, размерности и количествах элементов каждой размерности, например, описание **int array[4][3][6]**; определяет массив, состоящий из четырех элементов, каждый из которых – двумерный массив с размерами 3 на 6. В памяти массив **array** размещается в порядке возрастания самого правого индекса, то есть самый младший адрес имеет элемент **array[0][0][0]**, затем идет элемент **array[0][0][1]** и т. д.

Как и в случае одномерных массивов, доступ к элементам многомерных массивов возможен с помощью индексированных переменных и с помощью указателей. В общем случае для трехмерного массива индексированный элемент **b[i][j][k]** соответствует выражению ***(*(b + i) + j) + k**.

Для понимания конструкций, состоящих из набора звездочек, скобок и имен типов, нужно аккуратно применять синтаксические правила, учитывающие последовательность выполнения операций. Например, определение **int *array[6]**; вводит массив указателей на объекты типа **int**. Имя массива **array**, он состоит из шести элементов, тип каждого **int***. Определение **int (*ptr)[6]**; вводит указатель **ptr** на массив из шести элементов, каждый из которых имеет тип **int**. Возможность создания массивов указателей позволяет экономить память при использовании многомерных массивов.

Напомним, что операция **new** при использовании с массивами имеет следующий формат:

```
new <тип_элементов_массива>.
```

Такая операция позволяет выделить в динамической памяти участок для размещения массива соответствующего типа, но не позволяет его инициализировать. В результате выполнения операция **new** возвратит указатель, значением которого служит адрес первого элемента массива. При выделении динамической памяти для массива его размеры должны быть полностью определены, например:

```
long (*lp)[2][4] ; // Определили указатель  
lp = new long[3][2][4] ; // Выделили память для массива
```

В данном примере использован указатель на объекты в виде двумерных массивов, каждый из которых имеет фиксированные размеры 2 на 4 и содержит элементы типа **long**. В определении указателя следует обратить внимание на круглые скобки, без которых обойтись нельзя. После выполнения приведенных операторов указатель **lp** становится средством доступа к участку динамической памяти с размерами **3 * 2 * 4 * sizeof(long)** байтов. В отличие от имени массива (имени у этого массива из примера нет) указатель **lp** есть переменная, что позволяет изменять его значение и тем самым, например, перемещаться по элементам массива.

Изменять значение указателя на динамический массив нужно осторожно, чтобы «не забыть», где находится начало массива, так как указатель, значение которого определяется при выделении памяти для динамического массива, используется затем для освобождения памяти с помощью операции **delete**. Например, оператор **delete [] lp**; освободит целиком всю память, выделенную для определенного выше трехмерного массива, если **lp** адресует его начало.

В отличие от определения массивов, размещающихся в статической памяти, инициализация динамических массивов не выполняется. Поэтому при выделении памяти для динамических массивов их размеры должны быть полностью определены явно. Кроме того, *только первый (самый левый) размер массива может быть задан с помощью переменной*. Остальные размеры многомерного массива могут быть определены *только с помощью констант*. Это затрудняет работу с многомерными динамическими массивами. Обойти это ограничение позволяет применение массивов указателей.

2 Задачи для самостоятельного решения

2.1 Из заданного множества точек на плоскости выбрать две различные точки так, чтобы количества точек, лежащих по разные стороны прямой, проходящей через эти две точки, различались наименьшим образом.

2.2 Определить радиус и центр окружности, на которой лежит наибольшее число точек заданного на плоскости множества точек.

2.3 Задано множество M точек на плоскости. Определить, верно ли, что для каждой точки $A \in M$ существует точка $B \in M$ ($A \neq B$), такая, что не существует двух точек множества M , лежащих по разные стороны от прямой AB .

2.4 Определить радиус и центр такой окружности, проходящей хотя бы через три различные точки заданного множества точек на плоскости, что минимальна разность количества точек, лежащих внутри и вне окружности.

2.5 Многоугольник (не обязательно выпуклый) задан на плоскости перечислением координат вершин в порядке обхода его границы. Определить площадь многоугольника.

2.6 Задано множество прямых на плоскости (коэффициентами своих уравнений). Подсчитать количество точек пересечения этих прямых.

2.7 Порядок на точках плоскости определим следующим образом: $(x, y) \leq (u, v)$, если либо $x < u$, либо $x = u$ и $y \leq v$. Перечислить точки заданного множества точек на плоскости в соответствии с этим порядком.

2.8 На плоскости задано n множеств по m точек в каждом. Среди точек первого множества найти такую, которая принадлежит наибольшему количеству множеств.

2.9 Выбрать три разные точки заданного на плоскости множества точек, составляющие треугольник наибольшего периметра.

2.10 Из заданного на плоскости множества точек выбрать такие три точки, не лежащие на одной прямой, которые составляют треугольник наименьшей площади.

2.11 Задано множество точек на плоскости. Выбрать из них четыре разные точки, которые являются вершинами квадрата наибольшего периметра.

2.12 Из заданного множества точек на плоскости выбрать три разные точки A, B, C так, чтобы внутри треугольника ABC содержалось максимальное количество точек этого множества.

2.13 Из заданного на плоскости множества точек выбрать три различные точки так, чтобы разность между площадью круга ограниченного окружностью, проходящей через эти три точки, и площадью треугольника с вершинами в этих точках была минимальной.

2.14 Определить радиус и центр окружности минимального радиуса, проходящей хотя бы через три различные точки заданного множества точек на плоскости.

2.15 Найти такую точку заданного на плоскости множества точек, сумма расстояний от которой до остальных минимальна.

2.16 Задано множество точек на плоскости, все точки этого множества попарно различны. Рассмотрим замкнутую ломаную, последовательно проходящую через эти точки. Определить, имеет ли эта ломаная самопересечения.

2.17 На плоскости заданы множество точек A и точка d вне его. Подсчитать количество (неупорядоченных) различных троек точек a, b, c из A , таких, что четырехугольник $abcd$ является параллелограммом.

2.18 Определить радиус и центр окружности, проходящей по крайней мере через три различные точки заданного множества точек на плоскости и содержащей внутри наибольшее количество точек этого множества.

2.19 Выбрать три различные точки из заданного множества точек на плоскости так, чтобы была минимальной разность между количествами точек, лежащих внутри и вне треугольника с вершинами в выбранных точках.

2.20 На плоскости задано множество попарно различных прямых (коэффициентами своих уравнений). Указать среди них ту прямую, которая имеет максимальное число пересечений с остальными прямыми.

2.21 Найти ромб наибольшей площади с вершинами в заданном множестве точек на плоскости.

2.22 Задано множество точек на плоскости, не лежащих на одной прямой. Определить минимальное подмножество точек, после удаления которых остаются точки, лежащие на одной прямой.

2.23 Найти множество всех различных выпуклых четырехугольников с вершинами в заданном множестве точек на плоскости.

2.24 Найти множество всех различных остроугольных треугольников с вершинами в заданном множестве точек на плоскости.

2.25 Среди треугольников с вершинами в заданном множестве точек на плоскости указать такой, стороны которого содержат максимальное число точек заданного множества.

2.26 Найти два треугольника с вершинами в заданном множестве точек на плоскости так, чтобы первый треугольник лежал строго внутри второго.

Лабораторная работа № 3. Строки и векторы в C++

Цель работы – познакомиться с основными конструкциями, используемыми для работы со строковой информацией.

1 Фрагмент теории

Строка (string) – это последовательность символов переменной длины. Прежде чем использовать строки, нужно придать им значения. Следующие примеры иллюстрируют основные способы инициализации строк:

```
string s1; // Инициализация по умолчанию, пустая строка;  
string s2 = "Привет!"; // Инициализация константной строкой;  
string s3 = s2; // Инициализация значением строки;  
string s4 (5, 'я'); // Инициализация повторением символа;  
string s5 ("Ура!"); // Инициализация константной строкой.
```

Перечислим основные операции, выполняемые над строками (таблица 1).

Таблица 1 – Основные операции класса string

Операция	Назначение
<<	Вывод в поток
>>	Чтение из потока, осуществляется до первого пробела или нажатия клавиши Enter
getline(<поток>, <строка>)	Чтение из потока; осуществляется до нажатия клавиши Enter. Второй параметр – прочитанная строка
empty()	Проверка на пустоту: true – строка пуста
size(), length()	Количество символов в строке, длина строки
[<номер>]	Ссылка на символ строки, номер которой задан параметром. Нумерация символов строки начинается с 0
+	Конкатенация строк
=	Копирование строк
==	Проверка на равенство строк: true – строки равны (содержат одинаковые символы). Регистр символов учитывается
!=	Проверка на неравенство строк: true – строки не равны (есть различие хотя бы в одном символе). Регистр символов учитывается
<, <=, >, >=	Сравнение строк. Регистр символов учитывается. Сравняются коды соответствующих символов. Выполняется по правилам сравнения десятичных дробей

В классе **string** имеются методы, предназначенные для работы с отдельными символами. Наиболее употребительные методы из этой группы приведены в таблице 2.

Таблица 2 – Функции для работы с отдельными символами строки (с – символ)

Функция	Результат
isalnum (c)	Возвращает true, если c – буква или цифра
isalpha (c)	Возвращает true, если c – буква
isctrl (c)	Возвращает true, если c – управляющий символ
isdigit (c)	Возвращает true, если c – цифра
isgraph (c)	Возвращает true, если c – печатаемый символ, а не пробел
islower (c)	Возвращает true, если c – символ в нижнем регистре
isprint (c)	Возвращает true, если c – печатаемый символ
ispunct (c)	Возвращает true, если c – знак пунктуации (символ, не являющийся управляющим символом, цифрой, символом или печатаемым отступом)
isspace (c)	Возвращает true, если c – символ отступа (т. е. пробел, табуляция, вертикальная табуляция, возврат каретки, новая строка, прогон страницы)
isupper (c)	Возвращает true, если c – символ в верхнем регистре
isxdigit (c)	Возвращает true, если c – шестнадцатеричная цифра
tolower (c)	Если c – прописная (большая) буква, то возвращает ее в нижнем регистре, иначе возвращает символ неизменным
toupper (c)	Если c – строчная (малая) буква, то возвращает ее в верхнем регистре, иначе возвращает символ неизменным

При разработке приложений часто приходится выполнять достаточно большое количество разнообразных операций над строками. Далее мы перечислим методы класса **string**, которые можно использовать при обработке строк.

1 **Метод substr()** возвращает копию части или всю исходную строку.

Общий вид:

```
<исходная_строка>.substr ([<начальная_позиция>]
[, <количество_символов>] )
```

По умолчанию первый параметр равен 0. Если второй параметр отсутствует, то строка копируется до конца. Примеры:

```
string s("Привет, мир!");  
string s2 = s.substr(0, 6); // s2 = Привет;  
string s3 = s.substr(8); // s3 = мир!  
string s4 = s.substr(8, 4); // s4 = мир!  
string s5 = s.substr(20); // исключение out_of_range.
```

2 **Метод insert()** используется для вставки символов в строку. Общий вид:

```
<исходная_строка>.insert(<начальная_позиция>,  
                          <вставляемые_символы>)
```

Вставляемые символы могут представлять строку **string** или быть C-строкой. Примеры:

```
string s("ароход");  
s = s.insert(0, "П"); // s = Пароход  
string s2("Пароход");  
string s3 = s2.insert(7, " - это хорошо!");  
// s2, s3 = Пароход - это хорошо!  
string s4 = s2.insert(17, " - это хорошо!");  
// исключение out_of_range.
```

3 Частным случаем метода **insert()** является **метод append()**, который вставляет символы, начиная с конца исходной строки. Общий вид:

```
<исходная_строка>.append(<вставляемые_символы>)
```

Вставляемые символы могут представлять строку **string** или быть C-строкой. Пример:

```
string s1("Один");  
string s2("Два");  
string s3("Три");  
string s4 = s1.append(s2.append(s3));  
// s1, s4 = ОдинДваТри; s2 = ДваТри
```

4 **Метод erase()** используется для удаления части строки. Общий вид:

```
<исходная_строка>.erase(<начальная_позиция>  
                        [, <количество_удаляемых_символов>])
```

Если второй параметр отсутствует, то удаляются все символы до конца строки. Примеры:

```
string s2("Пароход - это хорошо!");  
string s3 = s2.erase(7); // s2, s3 = Пароход  
string s("Самолет - это хорошо!");  
string s4 = s.erase(10, 4); // s, s4 = Самолет - хорошо!  
string s5 = s.erase(10, 40); // s, s5 = Самолет -  
string s6 = s.erase(50); // исключение out_of_range.
```

5 Наиболее интересным представляется **метод replace()**, который используется для замены одних символов на другие. Его общий вид:

```
<исходная_строка>.replace(<начальная_позиция>,  
                           <количество_заменяемых_символов>,  
                           <заменяющая_строка> [, <начальная_позиция1>,  
                           <количество_заменяющих_символов>])
```

Здесь первый параметр – позиция в исходной строке, с которой будет производиться замена. Вторым параметром – количество заменяемых символов. Третьим параметром – строка замены. Последние два параметра не являются обяза-

тельными. Четвертый параметр – позиция в строке замены, с которой будут браться заменяющие символы, и последний параметр – их количество. Примеры:

```
string s("Пароход - это хорошо!");
string s1 = s.replace(0, 7, "Сани");
// s, s1 = Сани - это хорошо!
string s2("Самолет - это хорошо!");
string s3 = s2.replace(0, 7, "Санитары", 0, 4);
// s2, s3 = Сани - это хорошо!
string s4("Самолет - это хорошо!");
string s5 = s4.replace(0, 7, "Санитары");
// s4, s5 = Санитары - это хорошо! .
```

В качестве длин заменяемой и заменяющей подстрок можно указать **string::npos**, что означает «до конца строки». Пример:

```
string s4("Самолет - это хорошо!");
string s5 = s4.replace(0, string::npos, "Санитары");
// s4, s5 = Санитары .
```

Вместо строки можно указывать символ, которым будет «затёрта» исходная подстрока. Третий параметр – число символов, которые будут расположены на месте исходной подстроки. Пример:

```
string s2("Самолет - это хорошо!");
string s3 = s2.replace(0, 7, 3, '*');
// s2, s3 = *** - это хорошо! .
```

Все значения позиций и длины должны быть корректными, в противном случае генерируется исключение.

6 Есть достаточно много методов, которые используются для поиска подстроки или символа в строке. Здесь мы рассмотрим только два из них. Первый – это метод **find()**, который осуществляет поиск от начала строки. Вторым – метод **rfind()**, реализующий поиск от конца к началу строки. Их общий вид практически идентичен:

```
<исходная_строка>.[r]find(<символ_или_строка>
[, <начальная_позиция>]); .
```

При использовании метода **find()** *начальная_позиция* – это номер символа, с которого начинается поиск. В методе **rfind()** *начальная_позиция* – это номер символа, до которого осуществляется поиск (напомним, что в этом случае поиск идет с конца строки). Метод возвращает номер первого вхождения искомого символа или строки, или -1 – если символ или строка не найдены. Примеры:

```
string s("Пароход - это хорошо!");
int pos = s.find("хо"); // pos = 4
int pos1 = s.find("хо", 7); // pos1 = 14
int pos2 = s.rfind('x'); // pos2 = 14
int pos3 = s.rfind('x', 7); // pos3 = 4 .
```

7 Очистка всей строки производится с помощью *метода clear()*:

```
<исходная_строка>.clear(); .
```

Конечно, можно присвоить строке пустое значение:

```
string s = ""; .
```

При работе со строками достаточно часто приходится преобразовывать строку в число и число в строку. Рассмотрим функции, которые используются для преобразования строки в число.

Функции

```
(stoi | stol | stoll | stoul) (<исходная_строка>
    [, <позиция_первого_непереводимого_символа>=nullptr
    [, <система_счисления>=10])
```

распознают содержимое исходной строки как целое число (типов **int**, **long**, **long long**, **unsigned long** и **unsigned long long** соответственно – два последних варианта не принимают знак числа), читая её с начала до момента первой ошибки (или конца числа). Второй параметр (если это не нулевой указатель) содержит индекс первого символа за считанным числом. Значение третьего параметра определяет основание системы счисления (от 2 до 36, по умолчанию 10). Если в качестве третьего параметра передать 0, то основание будет распознаваться по префиксу числа (0 – основание равно 8, 0x/0X – 16, другое – 10).

Функции

```
(stof | stod | stold) (<исходная_строка>
    [, <позиция_первого_непереводимого_символа>=nullptr
    [, <система_счисления>=10])
```

распознают содержимое исходной строки как число с плавающей точкой (типов **float**, **double** и **long double** соответственно), читая её с начала до момента первой ошибки (или конца числа). Остальные аргументы аналогичны по смыслу аргументам из предыдущей функции. Числа распознаются в соответствии с правилами записи литералов, кроме того, принимаются строки **INF** или **INFINITY** (без учёта регистра) как значение «бесконечность» и **NAN** как значение «не число».

При разработке приложений может пригодиться функция

```
to_string(<значение_встроенного_типа>),
```

возвращающая строковое представление своего аргумента (если это возможно сделать).

Достаточно часто в качестве динамической структуры данных, используемой для хранения каких-либо значений, применяется переменная класса **vector**. Его можно представить как некоторый контейнер, содержащий набор элементов определенного типа.

Вектор (vector) — это коллекция объектов одинакового типа, каждому из которых присвоен целочисленный индекс, предоставляющий доступ к этому объекту.

Элементами вектора могут быть значения различных типов, в том числе, другие векторы.

Будем придерживаться той схемы изложения, которую мы использовали при изучении строк. Прежде всего, рассмотрим способы определения и инициализации векторов. В этом нам поможет следующий фрагмент программы:

```
// все векторы предназначены для
// размещения целых чисел: <int>
vector<int> v1;           // пустой вектор
vector<int> v2 (v1);     // v2 - копия v1
```

```

vector<int> v3 = v1;    // аналог предыдущего определения
vector<int> v4 (5, -3); // вектор содержит 5 элементов
                        // со значением -3

vector<int> v5;
v5.assign(5, -3);    // вектор содержит 5 элементов
                    // со значением -3

vector<int> v6 (5);    // вектор содержит 5 элементов

```

Начиная со стандарта C++ 11, возможно следующее определение и инициализация вектора:

```

vector<int> v6 = {1, 2, 3}; // вектор содержит
                        // 3 элемента: 1, 2, 3

```

Здесь количество элементов и их значения задаются списком инициализаторов.

Обратите внимание на первое определение. Может показаться, что создание пустого вектора не имеет практического значения. Однако это не так: есть возможность добавления элементов в вектор в процессе выполнения программы. Для этого используется метод **push_back()**. Проиллюстрируем его использование следующим фрагментом:

```

vector<int> v1;    // пустой вектор

for (int i = 0; i < 100; i++)
    v1.push_back(i); // помещение элементов в вектор

// вывод элементов вектора
for (int i = 0; i < 100; cout << v1[i++] <<"\t" );

```

В результате его выполнения будут выведены целые число от 0 до 99 включительно.

Из приведенного фрагмента видно, что доступ к элементам вектора осуществляется также, как к элементам строки. Скажем больше: векторы поддерживают практически все операции, которые мы рассматривали для строк.

Функции-члены **empty()** и **size()** вектора ведут себя так же, как и соответствующие функции класса **string**: **empty()** возвращает значение **true**, если вектор пустой, а функция **size()** – количество элементов вектора.

Векторы, также как и строки, можно сравнивать. Два вектора равны, если у них одинаковое количество элементов и значения соответствующих элементов совпадают. Если у векторов разные размеры, но соответствующие элементы равны, то вектор с меньшим количеством элементов меньше вектора с большим количеством элементов. Если соответствующие элементы векторов различаются, то их отношение определяется по первым различающимся элементам. Отметим, что сравнивать два вектора можно только в том случае, если возможно сравнить элементы этих векторов.

Существует более общий механизм доступа к элементам вектора (строки) – *итераторы (iterator)*.

Как и указатели, итераторы реализуют косвенный доступ к объекту. Для вектора таким объектом является текущий элемент вектора или символ строки. С помощью итератора можно выбрать элемент, а также перемещаться по элементам. Значением итератора может быть либо позиция элемента, либо позиция

последним элементом в контейнере. Все другие значения итератора недопустимы.

Одними из часто используемых методов, связанных с итераторами, являются методы **begin()** и **end()**, которые возвращают итераторы соответственно «показывающие» на первый элемент и позицию, находящуюся за последним элементом. Если вектор (или в общем случае, контейнер) пуст, то **begin()** и **end()** возвращают один и тот же итератор, который показывает на позицию за последним элементом.

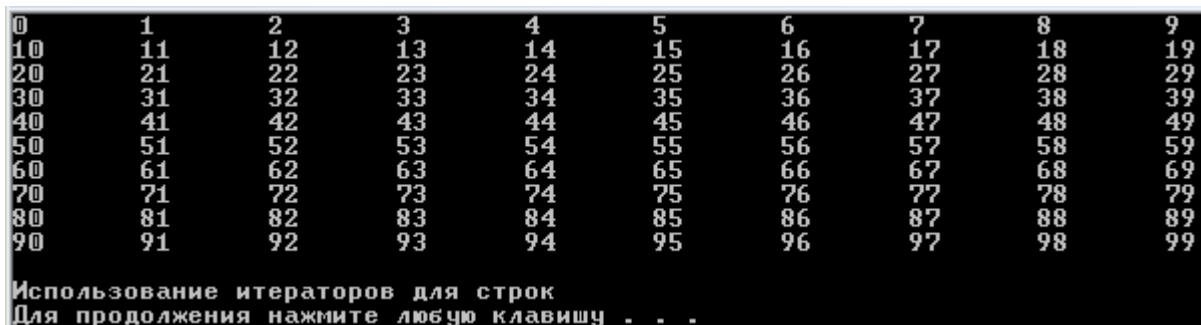
Приведем фрагмент программы, использующий указанные функции.

```
// вывод элементов вектора
vector<int>::iterator iter;
// создали итератор нужного типа

for (iter = v1.begin(); iter !=v1.end(); iter++)
    cout << (*iter) << "\t"; // разыменовали итератор
cout << endl;

// вывод символов строки
string s = "Использование итераторов для строк";
string::iterator iter_str; // создали итератор нужного типа
for (iter_str = s.begin(); iter_str !=s.end(); iter_str++)
    cout << (*iter_str); // разыменовали итератор
cout << endl;
```

Результат работы фрагмента приведен на рисунке 2.



```
0      1      2      3      4      5      6      7      8      9
10     11     12     13     14     15     16     17     18     19
20     21     22     23     24     25     26     27     28     29
30     31     32     33     34     35     36     37     38     39
40     41     42     43     44     45     46     47     48     49
50     51     52     53     54     55     56     57     58     59
60     61     62     63     64     65     66     67     68     69
70     71     72     73     74     75     76     77     78     79
80     81     82     83     84     85     86     87     88     89
90     91     92     93     94     95     96     97     98     99
Использование итераторов для строк
Для продолжения нажмите любую клавишу . . .
```

Рисунок 2 – Вывод содержимого контейнера с использованием итераторов

В данном фрагменте выводится содержимое вектора и строки, используя механизм итераторов. Сначала создается итератор необходимого типа (**iter** – для вектора, **iter_str** – для строки), который в дальнейшем используется в цикле. На каждой итерации цикла итератор указывает на текущий элемент. Для получения доступа к элементу итератор необходимо разыменовать.

Перечислим операции, которые можно выполнять над итераторами (таблица 3).

Таблица 3 – Основные операции над итераторами

Операция	Назначение
iterator + n, iterator - n,	Добавление (вычитание) целочисленного значения к итератору (из итератора). Итератор показывает на элемент, отстоящий от текущего на n позиций вперед (назад) в пределах контейнера
iterator1 - iterator2	Количество позиций между элементами, которые задаются указанными итераторами. Важно: значение iterator1 должно быть больше значения iterator2
<, <=, >, >=	Сравнение итераторов. Значение итератора увеличивается от начала контейнера к концу

Отметим, что при выполнении этих операций не выполняется автоматическая проверка на выход за границы контейнера. За корректность получаемых значений здесь отвечает программист.

Перечислим методы, используемые для вставки и удаления элементов вектора.

1 **Метод insert()** используется для вставки элементов в вектор. Он имеет следующий вид:

```
<вектор>.insert(<позиция_вставки>
                [, <количество_элементов>], <вставляемый_элемент>).
```

Здесь необязательный параметр *количество_элементов* определяет количество копий вставляемого элемента.

Есть еще одна форма этого метода:

```
<вектор>.insert(<позиция_вставки>,
                <начальная_позиция>, <конечная_позиция>).
```

В этом случае в вектор вставляется копия всех элементов интервала, заданного итераторами *начальная_позиция* и *конечная_позиция* (последний элемент не включается).

2 **Метод push_back(<элемент>)** добавляет элемент в конец вектора.

3 Для удаления последнего элемента используется **метод pop_back()**. Отметим, что удаляемый элемент не возвращается.

4 Если требуется удалить несколько элементов вектора, то можно воспользоваться **методом erase()**:

```
<вектор>.erase(<позиция> [, <конечная_позиция>]).
```

Данный метод удаляет элемент, стоящий на указанной позиции, и возвращает позицию следующего элемента. Если задан второй параметр, то удаление элементов происходит из указанного интервала (последний элемент не удаляется).

5 Если требуется изменить размер вектора, то можно воспользоваться **методом resize()**, который имеет следующий вид:

```
<вектор>.resize(<размер> [, <значение>]).
```

Этот метод изменяет размер вектора. Если количество элементов увеличивается, то новые элементы принимают значения по умолчанию или становятся равными значению, заданному в качестве второго параметра.

6 Удалить все элементы вектора можно **методом clear()**.

Что касается эффективности, следует помнить, что вставка и удаление выполняются быстрее, если:

- элементы вставляются или удаляются в конце интервала;
- на момент вызова емкость контейнера достаточно велика, чтобы операция выполнялась без перераспределения памяти;
- группа элементов обрабатывается одним вызовом вместо нескольких последовательных вызовов.

2 Задачи для самостоятельного решения

3.1 Перечислить все слова заданного предложения, которые состоят из тех же букв, что и первое слово предложения.

3.2 В заданном предложении найти пару слов, из которых одно является обращением другого.

3.3 Для каждого из слов заданного предложения указать, сколько раз оно встречается в предложении.

3.4 Из заданного текста выбрать и напечатать те символы, которые встречаются в нем ровно один раз (в том порядке, как они встречаются в тексте).

3.5 В предложении все слова начинаются с различных букв. Напечатать (если можно) слова предложения в таком порядке, чтобы последняя буква каждого слова совпадала с первой буквой следующего слова.

3.6 Найти множество всех слов, которые встречаются в каждом из двух заданных предложений.

3.7 Отредактировать заданное предложение, удаляя из него все слова с нечетными номерами и переворачивая слова с четными номерами.

3.8 Дан текст. Если в нем нет символа «*», то оставить текст без изменения, иначе каждую из малых латинских букв, предшествующих первому вхождению символа «*», заменить на цифру 3.

3.9 Даны два предложения. Найти самое короткое из слов первого предложения, которого нет во втором предложении.

3.10 В заданном предложении указать слово, в котором доля гласных (A, E, I, O, U) максимальна.

3.11 Переставить и распечатать слова заданного предложения в соответствии с ростом доли согласных (B, C, D, F, G, H, K, L, M, N, P, Q, R, S, T, V, W, X, Z) в этих словах.

3.12 Для каждого символа заданного текста указать, сколько раз он встречается в тексте. Сообщение об этом символе должно печататься не более одного раза.

3.13 Отредактировать заданное предложение, удаляя из него все слова, целиком составленные из вхождений не более чем двух букв.

3.14 Задан текст, в котором нет вхождений символов «(» и «)». Выполнить его сжатие, то есть заменить всякую максимальную подпоследовательность, составленную из более чем трех вхождений одного и того же символа, на $(k)s$, где s – повторяемый символ, а $k > 3$ – количество его повторений.

3.15 Отредактировать заданное предложение, заменяя всякое вхождение слова вида $aba1$ на b , где a, b – подслова, $a1$ – обращение слова a .

3.16 Дана строка, состоящая из слов, разделенных символами, которые перечислены во второй строке. Показать все слова.

3.17 Дана строка. Найдите наибольшее количество идущих подряд цифр.

3.18 Даны два слова. Найдите только те символы слов, которые встречаются в обоих словах только один раз.

3.19 Задана последовательность, состоящая только из символов «>», «<» и «-». Требуется найти количество стрел, которые спрятаны в этой последовательности. Стрелы – это подстроки вида «>>-->» и «<<--<<».

3.20 Вам необходимо проверить домашнюю работу Васи Пупкина, в которой он написал равенство. Например, запись вида « $2+3=5$ » является правильной, а « $23*7=421$ » неверная, но корректная. Корректной записью выражения будем называть последовательность: число, операция («+», «-», «*», «/»), число, знак равенства, число. Числом будем считать последовательность из одной или более десятичных цифр, перед которой может стоять один знак минус. В корректной записи выражения нет пробелов.

Если запись не соответствует описанному правилу, то она считается некорректной. Например, записи « $2*=3$ », «173» и « $2+2=a$ » некорректны.

Выведите «ДА», если указанная запись правильна (т. е. равенство представляет собой тождество), «НЕТ» – если корректная, но неверная, и «ОШИБКА», если запись некорректная.

3.21 Уравнение для пятиклассников представляет собой строку длиной 5 символов. Второй символ строки является либо знаком «+» (плюс) либо «-» (минус), четвертый символ – знак «=» (равно). Из первого, третьего и пятого символов ровно два являются цифрами из диапазона от 0 до 9, и один – буквой x , обозначающей неизвестное.

Требуется написать программу, которая позволит решить данное уравнение относительно x .

3.22 Найти все вхождения строки T в строке S . Обе строки состоят только из английских букв. Длины строк могут быть в диапазоне от 1 до 50 000 включительно.

3.23 Мальчик Кирилл написал однажды на листе бумаги строчку, состоящую из больших и маленьких английских букв, а после этого ушел играть в футбол. Когда он вернулся, то обнаружил, что его друг Дима написал под его строкой еще одну строку такой же длины. Дима утверждает, что свою строку он получил циклическим сдвигом строки Кирилла вправо на несколько шагов (циклический сдвиг строки $abcde$ на 2 позиции вправо даст строку $deabc$). Однако Дима известен тем, что может случайно ошибиться в большом количестве вычислений, поэтому Кирилл в растерянности – верить ли Диме? Помогите ему!

По данным строкам выведите минимально возможный размер сдвига вправо или -1 , если Дима ошибся.

3.24 Строка S была записана много раз подряд, после чего из получившейся строки взяли подстроку и дали вам. Ваша задача – определить минимально возможную длину исходной строки S . Строка, которая содержит только прописные английские буквы, длина строки не превышает 50000 символов.

3.25 Будем рассматривать только строки, состоящие из заглавных английских букв. Например, рассмотрим строку AAAABCCCCCDDDD. Длина этой строки равна 14. Поскольку строка состоит только из английских букв, повторяющиеся символы могут быть удалены и заменены числами, определяющими количество повторений. Таким образом, данная строка может быть представлена как 4AB5C4D. Длина такой строки равна 7. Описанный метод мы назовем упаковкой строки.

Напишите программу, которая берет упакованную строку и восстанавливает по ней исходную строку.

3.26 Руны – это древние магические знаки, которые наши предки использовали как буквы. Говорят, что рунные знаки обладают магическими свойствами, а при сложении рун в слова их магическая сила многократно возрастает. Если кузнец изготовит доспехи и начертит там определенные руны в определенном порядке, то доспехи будут наделены необычайными магическими силами.

Для того чтобы стать обладателем таких доспехов, достаточно просто принести кузнецу начертания этих рунных знаков. А вот, чтобы стать обладателем рунного знака, приходилось немало потрудиться. Воины добывали начертания рун других языков и наречий в боях или получали их в качестве наград в благодарность за оказанные услуги.

Но, так или иначе, и в этом деле развелись жулики. По подозрениям ученых, кузнец Игнатус Мошенникус изготавливал благородным воинам фальшивые рунные слова. Из древних преданий ученым стало достоверно известно, что каждая руна записывается из двух, трех или четырех английских букв. Причем первая буква рунного слова всегда записывается как заглавная, а все остальные являются маленькими. Ученые перевели несколько выкованных этим кузнецом рунных слов на английский язык и теперь нуждаются в вашей помощи. Проверьте, является ли приведенное слово рунным.

3.27 В заданной строке, состоящей из малых английских букв, необходимо найти пару самых длинных подстрок, состоящих из одних и тех же букв (возможно, в разном порядке). Например, в строке twotwow это будут подстроки wotwo и otwow.

3.28 Назовем *качеством строки* разность между максимальным и минимальным номерами в алфавите букв, входящих в строку. Например, качество строки ab равно $2 - 1 = 1$, а строки abcz – равно $26 - 1 = 25$.

Дана строка S. Необходимо найти непустую подстроку этой строки, обладающую максимальным качеством, а из всех таких – минимальную по длине.

3.29 Для заданного текста определить максимальную длину содержащейся в нем серии символов, отличных от букв.

3.30 Перечислить все слова заданного предложения, которые состоят из тех же букв, что и последнее слово предложения.

3.31 Шахматную доску будем представлять символьной матрицей, размером 8×8 . Даны натуральные числа n и m ($1 \leq n, m \leq 8$) – номера вертикали и горизонтали, определяющие местоположение ферзя. Соответствующий элемент матрицы нужно положить равным «Ф». Поля, находящиеся под угрозой ферзя,

положить равными символу «*», а остальные поля – символу «0». Вывести результирующую матрицу на экран.

3.32 Шахматную доску будем представлять символьной матрицей размером 8×8 . Даны натуральные числа n и m ($1 \leq n, m \leq 8$) – номера вертикали и горизонтали, определяющие местоположение коня. Соответствующий элемент матрицы нужно положить равным «K». Поля, находящиеся под угрозой коня, положить равными символу «*», а остальные поля – символу «0». Вывести результирующую матрицу на экран.

3.33 Написать программу, которая в заданном тексте меняет все вхождения символа «a» на «b» и «b» на «a» («шайба» → «шбйаб»).

3.34 Написать программу, которая считает в предложении число слов, начинающихся на заданную букву.

3.35 Удалите в заданной строке все последовательности символов «abc», за которыми следует цифра.

3.36 Дан текст. Найдите наибольшее количество подряд идущих пробелов в нем.

3.37 Даны две строки. Определите, можно ли из некоторых символов первой строки составить вторую строку.

3.38 Дан набор строк. Упорядочить его по длине строк.

3.39 Дана строка. Определите общее количество символов «+» и «-» в ней. А также сколько таких символов, после которых следует цифра ноль.

3.40 Палиндромом называется предложение, которое, после удаления из него всех пробелов и знаков препинания, читается одинаково справа налево и слева направо. По заданной строке нужно определить, является ли она палиндромом.

3.41 Запись химической реакции всегда содержит описание нескольких веществ. В свою очередь, описание одного химического вещества – строка, в которой входящие в него атомы химических элементов перечисляются в определенном порядке. При этом последовательности из двух и более одинаковых атомов, идущих подряд, группируются: записывается сокращенное название химического элемента и количество одинаковых элементов подряд. Например, вместо NN пишут N_2 . Обозначения химических элементов состоят из одной или двух английских букв, из которых первая – прописная, а вторая – строчная. В этой задаче не будут рассматриваться более сложные правила. Например, не используются скобки. Вы должны проверить, что заданная последовательность символов подходит под данное выше описание формулы химического вещества. При этом не нужно рассматривать корректность заданной строки, исходя из каких-либо других соображений, даже если они продиктованы здравым смыслом.

Записана непустая последовательность символов, содержащая только цифры и строчные и прописные английские буквы. Гарантируется, что в последовательности перед каждой строчной буквой идет прописная, а все однобуквенные и двухбуквенные подстроки, начинающиеся с прописной буквы – правильные обозначения химических элементов (поэтому здесь даже не приводится их список). Длина последовательности от 1 до 1000 символов.

Выведите слово YES, если данная строка подходит под упрощенное описание формулы химического вещества из условия и NO, если не подходит.

3.42 Для кодирования сообщения используют следующие действия: сообщение записывают, опуская пробелы, в прямоугольник заданной высоты по столбцам, а затем прочитывают строки в заданном порядке.

1	П	Р	И	А
2	Р	А	Р	Н
3	О	М	О	И
4	Г	М	В	Е

А затем, если выбрать порядок строк 3, 1, 2, 4, получают закодированное сообщение «ОМОИПРИАРАРНГМВЕ».

Требуется написать программу, которая по заданным высоте прямоугольника и порядке прочтения строк при кодировке декодирует заданное сообщение.

Задаются: высота прямоугольника H ($2 \leq H \leq 10$); порядок прочтения строк (числа записаны через пробел); закодированное сообщение, длина которого составляет от 1 до 200 символов. Примеры:

Дано:	Результат:
4	ПРОГРАММИРОВАНИЕ
3 1 2 4	
ОМОИПРИАРАРНГМВЕ	
2	физика
2 1	
ииафзк	

3.43 Палиндромом называют строку, читающуюся одинаково с обеих сторон. Задана строка s . Найдите ее наибольшую по длине подстроку, не являющуюся палиндромом. Если все подстроки s являются палиндромами, выведите NO SOLUTION. Примеры:

Дано:	Результат:
abba	abb
abc	abc
aaaaa	NO SOLUTION

3.44 Организаторы финала чемпионата мира по программированию использовали весьма странные клавиатуры – жесткость различных клавиш была различной. Таким образом, на нажатие разных клавиш требовалось различное количество энергии.

Эксперименты, проведенные командой *Dream Team* во время пробного тура, показали следующее. На набор строчной буквы английского алфавита требуется количество энергии, равное сумме цифр ее порядкового номера в алфавите (буквы нумеруются с единицы). На нажатие клавиши «Shift» требуется 10 джоулей энергии (таким образом, набор заглавной буквы английского алфавита требует на 10 джоулей больше, чем набор соответствующей ей строчной буквы), нажатие клавиши «Пробел» требует 4 джоуля энергии. Набор цифры x требует 13 джоулей энергии, набор точки – 5 джоулей, точки с запятой – 7 джоулей, запятой – 2 джоуля. Знак равенства, плюс, минус, одинарная и двойная

кавычка требуют по 3 джоуля энергии. Закрывающая и открывающая круглые скобки требуют по 1 джоулю, а фигурные, квадратные и угловые (т. е. символы «<>» и «>>») – по 8. При этом для всех упомянутых знаков препинания на клавиатуре, используемой на финале, существуют отдельные клавиши, и другой возможности набрать соответствующий символ нет. Нажатие клавиши «Enter» (перевод строки) оказалось настолько легким, что энергозатраты на него можно считать нулевыми.

Ваша задача – написать программу, которая по тексту шаблона вычислит энергозатраты на его набор.

Входные данные – шаблон программы, энергетические затраты на набор которого необходимо вычислить. Он содержит только цифры, пробелы, строчные и заглавные буквы английского алфавита, точки, запятые, знаки равенства, плюсы, точки с запятыми, двойные кавычки ("), одинарные кавычки ('), закрывающие и открывающие круглые, фигурные и квадратные скобки. Его размер не превышает 20000 байт.

3.45 На одной из лекций по информатике студент Петя узнал про новый шифр – простой замены. Он и на самом деле прост: в тексте каждая буква алфавита заменяется некоторой другой буквой того же алфавита (может быть, той же самой).

Петя написал письмо своему другу Васе. Письмо – это текст из нескольких строк, написанный на английском языке, с использованием только строчных английских букв и пробелов. В произвольное место, отдельной строкой Петя вставил ключевую фразу: «**the quick brown fox jumps over the lazy dog**», о которой они с Васей договорились заранее. После чего зашифровал письмо. Известно, что пробелы в письме не шифруются. Получив такое письмо, Вася сумеет его расшифровать и прочесть. Иногда Петя ошибается, и забывает вставить ключевую фразу. Увы, в этом случае прочесть письмо невозможно.

Так как процесс расшифровки трудоемок, Вася просит написать программу, с помощью которой он сможет быстро расшифровывать письмо от Пети.

Входные данные: целое число N – количество строк в письме ($1 \leq N \leq 200$). Далее идет N строк письма (пустые строки отсутствуют, в каждой строке не более 80 символов).

В случае присутствия в тексте ключевой фразы выведите N строк расшифрованного сообщения. Если ключевой фразы нет, следует вывести «No solution». Примеры:

<p>Дано:</p> <p>3</p> <p>vtz ud xnm xugm itr pyu jttk gmv xt otgm xt</p> <p>xnm puk ti xnm fprxq</p> <p>xnm ceuob lrtzv ita hegfd tsmr xnm ypwq ktj</p> <p>frtjrpgguvj otvxmdxd prm iev prmvx xnmq</p> <p>3</p> <p>vtz ud xnm xugm itr pyu jttk gmv xt otgm xt</p> <p>xnm puk ti xnm fprxq</p> <p>xnm fffff lrtzv iia wwwfd tsmr xnm ypwq ktj</p> <p>frtjrpgguvj otvxmdxd prm iev prmvx xnmq</p>	<p>aid of the party</p> <p>the quick brown fox jumps over the lazy dog</p> <p>programming contests are fun arent they</p> <p>No solution</p>
	<p>Результат:</p>

занный способ сжатия. Выведите одну строку из английских строчных букв от «a» до «z» – сжатие заданной бинарной последовательности. Примеры:

Дано:	Результат:
101	ab
101001	abc
00000000000000000000000000000001	y

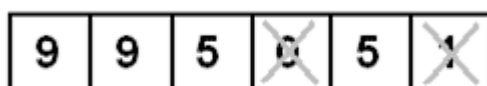
3.48 Основная часть автомобильного государственного регистрационного номера состоит из 6 символов: трех букв и трех цифр. Сначала идет буква, затем 3 цифры и еще 2 буквы заканчивают запись. В качестве цифр могут использоваться любые цифры от 0 до 9, а в качестве букв только прописные буквы, обозначения которых присутствуют как в английском, так и в русском алфавите, т. е. только следующие символы: A, B, C, E, H, K, M, O, P, T, X, Y. Например, «P204BT» – правильный номер, а «X182Yx» и «ABC216» – нет.

Определить, является ли введенный номер верным.

3.49 В честь успешного окончания первой четверти родители Пети и Вовы подарили им один лотерейный билет. Как обычно, ребята всерьез не восприняли данный подарок ввиду довольно скудной вероятности выигрыша, по их мнению. Но каково было удивление братьев, когда они узнали из средств массовой информации, что номер их билета является выигрышным.

После продолжительных минут радости ребята бросились узнавать сумму выигрыша, но, как оказалось, узнать это не так уж просто. Дело в том, что сумма выигрыша отчасти определяется самим владельцем или владельцами (если их несколько) следующим образом. На обратной стороне билета есть определенная секция, стерев слой защитного покрытия, можно увидеть целое положительное число N . После чего, каждый из владельцев билета должен зачеркнуть ровно по одной цифре данного числа N , полученное число и является суммой выигрыша.

Ниже приведен пример с N равным 995051.



Максимальное число, которое может быть получено из данного N посредством вычеркивания двух цифр является 9955. Помогите Пете и Вове с этой непростой, но очень актуальной для них задачей.

На вход программе подается число N ($100 \leq N < 10^{250}$), которое не содержит лидирующих нулей. Вывод: максимальное значение, которое может быть получено из N посредством вычеркивания из него ровно двух цифр.

3.50 Для шифрования слов с ними можно производить множество различных операций. Например, интересна такая операция: первые несколько букв заданного слова приписываются к его концу в обратном порядке, после чего удаляются из начала слова. При этом слово $a_1 a_2 \dots a_k a_{k+1} \dots a_n$ переходит в слово $a_{k+1} \dots a_n a_k a_{k-1} \dots a_1$ (число k выбирается в диапазоне от 0 до n).

Для двух заданных слов требуется определить, можно ли применением описанной операции преобразовать первое слово во второе.

В ответе вывести «Yes», если преобразование возможно, и «No», если нет. В случае положительного ответа во второй строке выведите k – длину перемещаемой части исходного слова k (из всех таких k выберите минимальный).
Примеры:

Дано:	Результат:
wrwdwpw	Yes
wdwpwpw	2
dWOddWd	No
dOdWdWd	

Лабораторная работа № 4. Структуры и объединения в C++.

Структуры

Цель работы – познакомиться с особенностями использования структур для решения практических задач.

1 Фрагмент теории

Структуры – это такие конструкции языка, которые объединяют в данные разных типов, в том числе и подструктуры (такие же структуры, но являющиеся членами главной структуры). Эти конструкции полезны тем, что во многих ситуациях позволяют группировать связанные данные таким образом, что с ними можно работать как с единым целым. Как объявляется структура, покажем на примере объявления данных некоторого человека:

```
struct man
{
    char name[80]; //имя
    char phone_number[80]; //телефон
    int age; //возраст
    int height; //рост
};
```

Так задается шаблон будущего экземпляра структуры. Здесь **man** – имя шаблона. То, что находится в теле, ограниченном фигурными скобками, – это члены структуры-шаблона (под такое объявление компилятор память не выделяет). На основе такого шаблона создается экземпляр структуры, под который память уже выделяется и с которым можно работать в программе. Экземпляры структуры создаются несколькими путями:

- **по шаблону man:**

```
struct man friends[100], others;
```

Здесь созданы два экземпляра структуры: один – это массив структур (каждый элемент такого массива представляет собой структуру шаблона **man**. Можно сказать так: **friends** – это переменная типа **man**), другой – обычный экземпляр по шаблону **man**. В языке C++ ключевое слово **struct** можно опускать;

- **при объявлении шаблона:**

```
struct man
{
    char name[80];
```



```

    char phone_number[80];
    int age;
    int height;
}others;

```

Здесь создан один экземпляр структуры – **others**;

- **с помощью квалификатора типа typedef**, который изменяет имя шаблона и позволяет воспользоваться новым именем в качестве типа данных:

```

typedef struct
{
    char name[80];
    char phone_number[80];
    int age;
    int height; } NewTemp1;

```

Теперь можно писать:

```
NewTemp1 d1, d2[20], *p;
```

(все эти переменные – типа **NewTemp1**). Здесь объявлено три переменных типа **NewTemp1**: экземпляр **d1** структуры шаблона **man**, массив структур **d2[20]** и **p** – указатель на структуру.

Приведем пример вложенной структуры, т. е. такой структуры, которая является членом другой структуры:

```

struct date {
    int day;           // день недели
    int month;        // номер месяца
    int year;         // год
    char monthname[4]; // название месяца
};
struct person {
    char name[30];    // имя
    char adress[70]; // домашний адрес
    long mailcod;    // почтовый код
    float salary;    // заработная плата
    date birthdate;  // дата рождения
    date hiredate;   // дата поступления на работу
}emp[1000], *p;

```

Это типичный пример объявления личной карточки работника (реальная карточка содержит намного больше данных). Здесь объявлен указатель на структуру и массив структур шаблона **person**. Если такой массив заполнить получим данные на 1000 работников.

Отметим, что указатель на структуру – это не экземпляр структуры (экземпляр структуры объявляется как **emp[]**), а указатель, которому в дальнейшем будет присвоен адрес некоторой структуры, с ее элементами можно будет работать через указатель.

Мы видели, что после объявления структуры (а это фактически тип данного, имя которого равно имени структуры) можно объявить некую переменную типа этой структуры или указатель этого типа (указатель на эту структуру).

Если вы объявили переменную типа структуры, то чтобы обратиться к элементам структуры, надо после имени переменной поставить точку, а если

объявили указатель на структуру, то после имени указателя на данную структуру надо поставить стрелку вправо (->). Затем нужно к этим именам приписать имя члена структуры, к которому надо обратиться. Если требуется обратиться к членам вложенной структуры, то следует продолжить операции точкой или стрелкой вправо с именем подструктуры, а затем с именем члена. Примеры обращения к членам экземпляров структуры: **emp[0].name**, **emp[521].salary**, **emp[121].hiredate.year**.

Допустим, **p = &emp[1]**. В этом случае:

- **p->address** – это адрес работника, который содержится в экземпляре структуры **emp[1]**;

- **p->hiredate->year** – год поступления на работу.

Однако существуют некоторые ограничения:

- членом структуры может быть любой тип данных (**int**, **float**, массив, структура), но элемент структуры не может иметь тот же тип, что и сама структура. Например:

```
struct r {int s; r t};
```

Такое объявление структуры неверно, т. к. **t** не может иметь тип **r**. При этом указатель на тот же тип разрешен. Например:

```
struct r {int s; r* t};
```

Такое объявление верно;

- в языке C членом структуры не может быть функция, а указатель на функцию может быть членом структуры. Например:

```
struct r
{
    int s;
    (*comp)(char* a, char* b);
}
```

Такое объявление верно;

- присваивать значения одной структуры другой разрешено только для экземпляров одной структуры. Например, существует структура:

```
struct A { int i; char d } a, a1;
```

и

```
struct B{ int i; char d } b;
```

В этом случае можно выполнить **a = a1**; или **a1 = a**; Но операцию **a = b**; выполнить нельзя, т. к. **a** и **b** считаются относящимися к шаблонам разного типа (у их шаблонов разные имена и этого достаточно, чтобы считать их разными, хотя по структуре они совпадают).

Рассмотрим порядок выполнения некоторых наиболее распространенных операций над элементами структуры на примере следующего описания:

```
struct {
    int x;
    int *y;
} *p; // p - указатель на структуру.
```

Тогда:

- **(++p)->x** – операция увеличивает **p** до доступа к **x**;
- **(p++)->x** – операция увеличивает **p** после доступа к **x** (круглые скобки не обязательны, так как по старшинству раньше будет применена операция «->»);

- ***p->y** – выбирается содержимое объекта, на который указывает **y**;
- ***p->y++** – увеличивается **y** после обработки того, на что он указывает (аналогично ***s++**);
- ***p++->y** – увеличивает **p** после выборки того, на что указывает **y**;
- **(*p).y++** – увеличивает то, на что указывает **y**.

Можно отметить одно очень важное использование структур: создание новых типов данных. Существуют типы данных, гораздо более эффективные при решении определенных задач, чем массивы и структуры. Это очереди, двоичные деревья, множества, таблицы и графы. Многие из этих типов создаются из «связанных» структур. Обычно каждая такая структура содержит один или два типа данных плюс один или два указателя на другие структуры такого же типа. Указатели служат для связи одной структуры с другой и для обеспечения пути, позволяющего вести поиск по всей структуре.

2 Задачи для самостоятельного решения

4.1 Опишите, используя структуру, телефонную книгу. Составьте программу, выдающую список абонентов, имеющих телефонный номер, начинающийся на 46.

4.2 Опишите, используя структуру, каталог книг в библиотеке. Составьте программу, выдающую список книг В. Пикуля, хранящихся в библиотеке.

4.3 Опишите, используя структуру, таблицу дат и событий русской истории. Составьте программу, выдающую список событий XIX века.

4.4 Опишите, используя структуру, таблицу дат и событий русской истории. Составьте программу, выдающую список дат XVIII века.

4.5 Опишите, используя структуру, школьный класс (фамилия и инициалы, дата, месяц и год рождения). Составьте программу, выдающую список учеников, рожденных в мае месяце.

4.6 Опишите, используя структуру, записную книжку. Составьте программу, выдающую список друзей, кому в этом году исполняется 25 лет: фамилия и инициалы, год, месяц и дата рождения.

4.7 Опишите, используя структуру, школьный класс (фамилия и инициалы, дата, месяц и год рождения). Составьте программу, выдающую день рождения класса (среднее арифметическое дат и месяцев).

4.8 Опишите, используя структуру, выборы (фамилия кандидата и количество набранных голосов). Всего избирателей – 2000. Составить программу, определяющую, кто из делегатов прошел или необходимо проводить повторные выборы (должно быть набрано 1/3 голосов от общего количества).

4.9 Опишите, используя структуру, школьную нагрузку (фамилия преподавателя, класс, часы). Составьте программу, определяющую нагрузку каждого преподавателя. Определить, у какого преподавателя самая большая нагрузка и у кого самая низкая.

4.10 После поступления в вуз о студентах собрана информация: фамилия, нуждается ли в общежитии, стаж, работал ли учителем, что окончил, какой язык изучал. Составьте программу, определяющую: 1) сколько человек нужда-

ются в общежитии; 2) списки студентов, проработавших два и более года учителем; 3) списки окончивших педучилище; 4) списки языковых групп.

4.11 Опишите, используя структуру, таблицу соревнований (название команды, количество набранных очков). Выберите команду, занявшую первое место. Упорядочите список команд в зависимости от занятого места.

4.12 При сдаче норм ГТО были получены результаты забега на 100 метров и прыжков в длину. Задайте нормы ГТО по этим видам, определите списки учеников, не выполнивших нормативы, количество учеников, сдавших нормативы, а также списки трех лучших.

4.13 Опишите, используя структуру, вступительные экзамены. Абитуриенты сдавали три экзамена, для поступления необходимо набрать 12 баллов. Определите списки абитуриентов, зачисленных в институт, количество не сдавших экзамены, списки абитуриентов, сдавших три экзамена на 4.

4.14 Опишите, используя структуру, оценки за год. Посчитайте процент и качество успеваемости в классе за год, составьте списки неуспевающих и отличников.

4.15 Опишите, используя структуру, данные на учеников (фамилия, улица, дом, квартира). Составьте программу, определяющую, сколько учеников живет на улице Гоголя, списки учеников, живущих в доме номер 44.

4.16 В библиотеке для каждого заказывающего книгу читателя заполняется карточка: фамилия, дата заказа, дата выдачи книги. Определите: 1) самый маленький срок, за который нашли книгу; 2) сколько заказов было не удовлетворено; 3) кто чаще всего берет книги; 4) кому выдали книги 14.09.21; 5) сколько человек заказывали книги 24.04.21.

4.17 Опишите, используя структуру, сортировку почтовых посылок (город, улица, дом, квартира, кому, ценность). Составьте программу, определяющую: 1) сколько посылок отправлено в г. Нижний Новгород; 2) сколько и куда (список городов) отправлено посылок ценностью выше 100 рублей; 3) есть ли адреса, куда отправлено более 1 посылки, если есть, то сколько и кому.

4.18 Во время сессии несколько студентов не сдали экзамен (фамилия, предмет, группа, дата, оценка). Выведите список сдавших экзамены, сколько должников осталось, кто не сдал геометрию 18.01.21.

4.19 В библиотеке в читательском билете есть данные о человеке (фамилия), записываются данные о книге (автор, название, дата, когда книгу брали читать). Определите, кто брал книгу И. Ефремова «Таис Афинская» 14.04.21, сколько читателей брали книги А. С. Пушкина, кто и какие книги брал 21.04.21.

4.20 В экзаменационной ведомости можно выделить сведения о предмете (наименование, номер группы, дата сдачи экзамена), сведения о человеке (фамилия, номер зачетной книжки, оценка за экзамен). Определите, сколько человек не сдали информатику, выдать их списки: фамилия, номер группы. Определите, сколько групп сдавали экзамены 11.06.21 и какие, составьте списки студентов, сдавших информатику: фамилия, номер группы, оценка.

4.21 Написать программу, осуществляющую приведение подобных членов многочлена с использованием структуры.

4.22 Опишите, используя структуру, расписание (предмет, преподаватель, номер группы, день недели, часы, аудитория). Составьте программу, определяющую: 1) у каких групп совпадают аудитории на занятиях; 2) у кого из преподавателей есть наложения в расписании; 3) какая нагрузка у заданного преподавателя; 4) список групп, у которых ведет занятия данный преподаватель; 5) сколько пар информатики у группы 100, в какие дни и в какое время.

4.23 Опишите, используя структуру, товар (наименование товара, старая цена, новая цена). Составьте программу, определяющую, на какие товары повысились цены и на сколько процентов.

4.24 Опишите, используя структуру, завод (наименование станка, время простоя в месяц, время работы в месяц). Составьте программу, определяющую общее время простоя на заводе, списки станков, не имеющих простоя, относительное время простоя всех и каждого станка (в процентах).

4.25 Приняв способ изображения рационального числа в виде структуры с двумя полями целого типа, написать программу, позволяющую: 1) определить, есть ли среди 50 рациональных чисел равные; 2) вычислить наибольшее из данных рациональных чисел.

4.26 При поступлении на музыкально-педагогический факультет на абитуриентов собираются сведения: фамилия, музыкальный инструмент. Для поступления необходимо сдать экзамен по специальности. Составьте списки для данного экзамена в зависимости от специальности.

4.27 В школе было три 9 класса, в августе каждый классный руководитель имел сведения о своих учениках: фамилия, куда поступал, поступил или нет. Определите, сколько учеников хотели пойти в 10 класс, кто хотел поступать в училище и техникум, кто поступил в училище или техникум, сколько учеников будет учиться в 10 классе, сколько необходимо создать 10 классов и по сколько человек.

4.28 Заполнена анкета на учеников: фамилия, где работают родители. Определите: сколько родителей работают на заводе КМЗ, у кого родители работают в драматическом театре или в филармонии.

4.29 Выберите самую высокую горную вершину из заданного списка.

4.30 Из выбранных двух карт В и К определите бьет одна другую или нет, если А – козырная масть.

4.31 Найдите площади самого большого и самого маленького круга, если заданы координаты центров и радиусы. Центр какого круга находится ближе всех к началу координат.

4.32 Опишите, используя структуру, анкету школьника: (фамилия, возраст). Составьте программу, определяющую возрастные группы в классе и напечатайте их списки.

4.33 На олимпиаде по информатике на школьников заполнялись анкеты: фамилия, номер школы, класс, занятое место. Напечатайте: 1) списки школ, занявших призовые места; 2) какая из школ заняла больше всех призовых мест; 3) списки учеников, занявших первое место, укажите их класс.

4.34 Опишите, используя структуру, время (часы, минуты, секунды). Определите, какое время t_1 или t_2 меньше. Вычислите интервал времени, прошедший от меньшего до большего времени.

4.35 В анкетных данных указываются: фамилия, пол, рост. Определите средний рост женщин, фамилию самого высокого мужчины, есть ли в группе хотя бы два человека одного роста.

4.36 Даны два рациональных числа, опишите их, используя структуру записи: числитель, знаменатель. Приведите их к несократимому виду, найдите их сумму.

4.37 Дана дата (число, месяц, год и день недели). Определите правильность заданной даты; какой день недели приходится на последний день данного месяца.

4.38 Написать программу, позволяющую определить, есть ли в школе в каких-либо классах однофамильцы. Данные об ученике идут в следующем порядке: имя, фамилия, год обучения, буква. Данные о разных учениках идут в некоторой очередности, о которой заранее ничего не известно.

4.39 Опишите, используя структуру, жителей (фамилия, город, улица, дом, квартира). Составьте программу, которая печатает фамилии двух любых жителей из списка, живущих в разных городах по одинаковому адресу.

4.40 Дана анкета: фамилия, пол, число, месяц, год рождения. Выберите самого старшего мужчину, напечатайте все фамилии, начинающиеся с буквы «Б», и даты рождения этих людей.

4.41 Опишите, используя структуру, записную книжку (фамилия, номер телефона). Составьте программу, определяющую: 1) есть ли в записной книжке сведения о знакомом с фамилией на букву "Ф", если есть, то напечатать его фамилию и телефон; 2) есть ли в записной книжке сведения о знакомом с телефоном 43-58-35, если есть, то напечатать его фамилию.

4.42 Опишите, используя структуру, камеры хранения (номер, индекс камеры, заданный буквой). Составьте программу, определяющую, есть ли среди камер камера с номером 99. Упорядочите все камеры по возрастанию номеров.

4.43 В деканате хранится информация о зимней сессии на 1 курсе (фамилия, номер группы, оценка по геометрии, оценка по алгебре, оценка по информатике). Составьте программу, печатающую фамилии студентов, имеющих долг хотя бы по одному предмету, качество успеваемости, то есть процент студентов, сдавших экзамены на «4» и «5», название предмета, который был сдан лучше всего, номера групп в порядке убывания средней успеваемости их студентов.

4.44 В отделе кадров хранится следующая информация о каждом студенте: фамилия, имя, отчество, пол, возраст, курс. Составьте программу, которая печатает номер курса, на котором наибольший процент мужчин, самые распространенные мужские и женские имена, фамилии в алфавитном порядке и инициалы всех студенток, отчество и возраст которых являются одновременно самыми распространенными.

4.45 Дан список класса (фамилия, имя). Напечатайте в алфавитном порядке фамилии учеников, через пробел напишите имя каждого.

4.46 Багаж пассажира характеризуется количеством вещей и общим весом. Найдите: 1) багаж, средний вес одной вещи в котором отличается не более, чем на 0,3 кг от общего среднего веса одной вещи; 2) число пассажиров, имеющих более двух вещей, и число пассажиров, количество вещей которых превосходит среднее число вещей; 3) выясните, имеется ли пассажир, багаж которого состоит из одной вещи весом менее 30 кг; 4) упорядочите сведения о багаже по невозрастанию веса; 5) напечатайте сведения о багаже, общий вес вещей в котором меньше, чем 10 кг.

4.47 В библиотеке хранятся сведения о книгах (фамилия автора, название книги и год издания). Найдите названия книг С. Дангулова, изданных с 1960 года. Определите, имеется ли книга с названием «Информатика», если да, то сообщите сведения об этих книгах.

4.48 В учреждении хранятся записи о сотрудниках: фамилия, инициалы, номер телефона. Найдите номер телефона сотрудника по его фамилии и инициалам.

4.49 Дан перечень дат. Определите год с наименьшим номером, все весенние даты, самую позднюю дату.

4.50 В сводке об экспортируемых товарах указывается: наименование товара, страна, импортирующая товар, объем поставляемой партии в штуках. Напечатайте списки стран, в которые экспортируется данный товар, и общий объем его экспорта.

4.51 В счете указано: название игрушки, стоимость в рублях, возрастные границы. Получите следующие сведения: 1) название игрушек, цена которых не превышает 80 р. и которые подходят детям до пяти лет; 2) цену самого дорогого конструктора; 3) названия наиболее дорогих игрушек, цена которых отличается от цены самой дорогой игрушки не более, чем на 5 р.; 4) название игрушек, которые подходят как детям 4 лет, так и детям 10 лет; 5) можно ли подобрать игрушку, любую, кроме мяча, подходящую ребенку 3 лет, и дополнительно мяч так, чтобы стоимость игрушек не превосходила 150 р.

4.52 Хранятся сведения о лесе: вид дерева, общая численность, численность здоровых деревьев. Составьте программу вычисления: 1) суммарного числа деревьев на контрольном участке; 2) суммарного числа здоровых деревьев; 3) относительную численность (в процентах) больных деревьев; 4) относительную численность (в процентах) различных видов, в том числе больных (в процентах) для каждого вида.

4.53 Написать программу, вычисляющую число дней в месяце для любого года нашего столетия.

Лабораторная работа № 5. Структуры и объединения в C++. Объединения

Цель работы – получить первоначальные навыки создания и использования объединений.

1 Фрагмент теории

Структура данных *объединение* подобна структуре, однако в каждый момент времени может использоваться (является активным) *только один из его компонентов*. Шаблон объединения может задаваться записью вида:

```
union
{
<имя_типа1> <компонента1>;
<имя_типа2> <компонента2>;
.
.
.
<имя_типаN> <компонентаN>;
};
```

Поля структуры размещаются в оперативной памяти одно за другим в той последовательности, в которой перечислены в описании. Поля объединений размещаются, начиная с одного места в памяти и, следовательно, накладываются друг на друга.

Доступ к компонентам объединения осуществляется тем же способом, что и к компонентам структур.

В качестве примера объекта типа **union** рассмотрим объединение **geom_fig**:

```
union {
    int radius; // окружность
    int a[2];   // прямоугольник
    int b[3];   // треугольник
} geom_fig;
```

В этом примере обрабатывается только активный компонент, то есть компонент, который последним получает свое значение. Например, после присваивания значения компоненту **radius** не имеет смысла обращение к массиву **b**.

Объединения применяются для:

- минимизации используемого объема памяти, если в каждый момент времени только один объект из многих является активным;
- интерпретации основного представления объекта одного типа, как если бы этому объекту был присвоен другой тип.

Таким образом, после задания структуры данных «объединение» в программе будет находиться переменная, которая на законных основаниях может хранить «в себе» значения нескольких типов.

Очень часто некоторые объекты программы относятся к одному и тому же классу, отличаясь лишь некоторыми деталями. Например, рассмотрим представление геометрических фигур. Общая информация об этих фигурах может включать такие элементы, как площадь и периметр. Однако соответствующая информация об их размерах может оказаться различной в зависимости от их формы.

В языке C++ можно реализовать тип данных, который называется *переменной структурой* (**variant structure**), которая может быть реализована с использованием комбинации структуры и объединения.

Рассмотрим для примера шаблон структуры **figure**:

```
struct figure {
```



```

int area, perimeter; // Общие компоненты
int type;           // Метка активного компонента
union
{
    int radius; // Окружность
    int a[2];   // Прямоугольник
    int b[3];   // Треугольник
} geom_fig;
};

```

В общем случае каждый объект типа **figure** будет состоять из трех компонентов: **area**, **perimeter** и **type**. Компонент **type** называется *меткой активного компонента (active component tag)*, так как он используется для указания, какой из компонентов объединения **geom_fig** (например, **radius**, **a** или **b**) является активным в данный момент. Такая структура называется *переменной структурой*, потому что ее компоненты меняются в зависимости от значения метки активного компонента.

В общем случае переменные структуры будут состоять из трех частей: набора общих компонентов, метки активного компонента и части с меняющимися компонентами.

Общая форма переменной структуры имеет следующий вид:

```

struct {
    <общие_компоненты>;
    <метка_активного_компонента>;
    union
    {
        <описание_типа1> <компонента1>;
        <описание_типа2> <компонента2>;
        .
        <описание_типаN> <компонентаN>;
    } <идентификатор>;
};

```

2 Задачи для самостоятельного решения

5.1 Две точки заданы на плоскости своими координатами, которые могут быть как декартовыми, так и полярными. Требуется вычислить расстояние между этими двумя точками.

5.2 Составить программу для вычисления объемов геометрических тел: призмы с треугольным основанием и шара. Использовать структуру данных «объединение».

5.3 Написать программу для вычисления площади геометрических фигур: треугольника, квадрата, круга. Использовать структуру данных «объединение».

5.4 Написать программу вычисления длин сторон, углов, площади, радиусов вписанной и описанной окружностей для треугольника, заданного координатами вершин на плоскости в декартовых или полярных координатах.

5.5 В магазине по продаже транспортных средств имеются в продаже велосипеды, автомобили, лодки. Известна цена каждого изделия. Определить: 1) количество двухколесных велосипедов; 2) стоимость всех трехколесных ве-

лосипедов; 3) есть ли среди автомобилей «Жигули»; 4) какова общая стоимость моторных лодок; 5) сколько весельных лодок продается с запасными веслами.

5.6 В магазин поступили серьги и кольца из золота и серебра. На каждый вид изделий завели карточку, в которой указаны название, вес, цена, металл, из которого изготовлена вещь; для колец указан еще и размер. Найти: 1) общую стоимость поступившей партии товара; 2) количество серебряных изделий; 3) общий вес золотых колец; 4) количество серебряных колец 16 размера.

5.7 В продаже имеются женские украшения с драгоценными и поделочными камнями. Про каждое изделие известны название и цена, для изделий с драгоценными камнями известно название камня и его вес. Определить: 1) общий вес всех драгоценных камней; 2) стоимость всех украшений; 3) количество украшений с сапфирами.

5.8 В универмаг поступили женские и мужские плащи. Распечатайте товарные ярлыки этих изделий. Товарный ярлык должен содержать наименование, цену и размер изделия. Размер мужского плаща содержит объем груди и рост, размер женского плаща содержит рост, объем груди и объем бедер. Найти: 1) количество товара; 2) стоимость женских плащей; 3) стоимость женских плащей 46 размера; 4) количество мужских курток, которые дороже N рублей и подходят мужчинам выше 180 см.

5.9 В молочный магазин привезли сметану: 1) развесную с указанием общего веса и цены за 1 кг; 2) в упаковке по 250 и 500 г с указанием стоимости упаковки. Определить: 1) общий вес партии товара; 2) количество упаковок по 500 г; 3) стоимость всех упаковок по 250 г.

5.10 Опишите, используя структуру, телефонную книгу, содержащую номера телефонов городских абонентов, сотовой связи и иногородних, для которых указан еще и код города. Для мобильных телефонов дополнительно указывается оператор (строка). Составьте программу, которая: 1) выдает список абонентов, имеющих телефонный номер, начинающийся на 33; 2) определяет, сколько абонентов живут в Москве; 3) список абонентов МТС.

5.11 Случайно «уронили» два массива. В первом массиве содержалась информация о некоторых датах и событиях русской истории, а во втором – о школьном классе (фамилия и инициалы, дата рождения). Массивы «рассыпались» и «перемешались». Составьте программу, выдающую: 1) список дат XVIII века; 2) список учеников, рожденных в мае месяце.

5.12 При сдаче норм ГТО в 8А классе были получены результаты забега на 100 метров и прыжков в длину, а в 8Б классе – результаты забега на 100 метров, прыжков в высоту и метания гранаты. Определите (задав «собственные» нормы ГТО по этим видам): 1) списки учеников, не выполнивших нормативы; 2) количество учеников 8Б класса, взявших необходимую высоту с первой попытки; 3) длину «общего прыжка» 8А класса.

5.13 Случайно «уронили» два массива. В первом массиве содержалась информация о книгах в библиотеке, а во втором – о некоторых датах и событиях русской истории. Массивы «рассыпались» и «перемешались». Составьте программу, выдающую: 1) список книг В. Пикуля, хранящихся в библиотеке; 2) список дат XIX века.

5.14 Случайно «уронили» два массива. В первом массиве содержалась информация о деревьях (название, количество в штуках), а во втором – о студенческой группе (фамилия и инициалы, пол, дата рождения). Массивы «рассыпались» и «перемешались». Составьте программу, выдающую: 1) информацию о самых молодых разнополых студентах группы; 2) упорядоченный по количеству список деревьев.

5.15 Магазин торгует алкогольными и безалкогольными напитками. Алкогольный напиток характеризуется наименованием, крепостью, ценой, количеством. Безалкогольный – наименованием, емкостью упаковки, ценой, количеством. Определить общее количество всех алкогольных напитков, общую стоимость всех безалкогольных напитков, во сколько раз самый дорогой безалкогольный напиток дешевле самого дорогого алкогольного напитка.

5.16 Создать каталог из журналов и книг: автор, название, год издания, издательство (для книг), журнал и его номер (для журнала). Получить: 1) список журналов, изданных с 2010 года; 2) количество книг заданного автора.

5.17 Сведения о молотках и отвертках, хранящихся на складе, содержат следующие атрибуты: название, количество, стоимость одного инструмента. Дополнительно для молотков хранится информация о длине ручки, а для отвертки – материал, из которого выполнена ручка, и может ли использоваться для определения наличия напряжения в розетке. Определить и сравнить суммарную стоимость каждого типа инструмента. Сколько имеется молотков, длина ручки которых больше 30 см. Сколько на складе отверток с пластиковой ручкой, которые могут использоваться как определители наличия напряжения в сети.

5.18 Имеются сведения о кубиках: длина ребра, его цвет. Дополнительно для кубиков красного цвета указан материал, из которого сделан кубик, а для кубиков синего цвета – имеет ли он сквозное отверстие и каков диаметр этого отверстия. Определить: 1) количество кубиков, окрашенных в другие цвета, кроме красного и синего и максимальный объем такого кубика; 2) есть ли среди синих кубиков кубик, имеющий сквозное отверстие диаметром более 3 см; 3) количество красных деревянных кубиков.

5.19 Составьте список группы спортсменов, участвовавших в соревнованиях по спортивной гимнастике, включающей N человек. Для каждого гимнаста укажите фамилию, имя, название общеобразовательной школы, класс, результаты по следующим видам: брусья, вольные упражнения, прыжки через коня. Дополнительно для юношей: кольца, перекладина, для девушек – бревно.

Составьте программу, которая обеспечивает ввод полученной информации, распечатку ее в виде таблицы, а также распечатайте анкетные данные спортсменов:

- а) показавших лучший результат по каждому виду;
- б) показавших лучший результат по всем видам многоборья,
- в) не получивших ни одного призового места.

5.20 Составьте прайс-лист магазина «Техника», включающий в себя марку предприятия-производителя, страну-производителя и, в зависимости от этих данных, наименования товара, его цену, количество единиц товара на складе.

Составьте программу, которая обеспечивает ввод полученной информации, распечатку ее в виде таблицы. Выведите на экран меню, а затем информацию о товаре в зависимости от запроса покупателя.

5.21 Составьте прайс-лист аптеки, включающий в себя наименование товара, страну-производитель, его цену, его состав, рекомендации врача в зависимости от возраста больного (дозировка, наличие сопутствующих устройств). Информацию о каждом виде товара оформите в программе в виде записи с вариантами. Совокупность записей объедините в массив. Составьте программу, которая обеспечивает ввод полученной информации, распечатку ее в виде таблицы. Выведите на экран меню, а затем информацию о товаре в зависимости от запроса покупателя.

5.22 Составьте банк данных членов своей семьи и (или) ближайших родственников, включающий в себя имя, отчество, степень родства, и в зависимости от введенной информации в поле «Молодой/Старый» придумайте варианты полей (например, хобби, любимый анекдот, количество медалей, количество внуков, любимый напиток, любимая девочка, лучший друг, объем имеющегося наследства). Информацию о каждом родственнике оформите в программе в виде записи с вариантами. Совокупность записей объедините в массив. Составьте программу, которая обеспечивает ввод полученной информации, распечатку ее в виде таблицы. Выведите на экран меню, а затем информацию о родне в зависимости от вашего запроса.

5.23 Составьте банк данных своих одноклассников, включающий в себя фамилию, имя, почтовой и (или) электронный адрес, телефон, а также в зависимости от поля «Друг» наличие соответствующей дополнительной информации по своему усмотрению (например: для девушек – цвет глаз, характер и т. п.; для юношей – хобби и т. п.). Информацию о каждом однокласснике оформите в программе в виде записи с вариантами. Совокупность записей объедините в массив. Составьте программу, которая обеспечивает ввод полученной информации, распечатку ее в виде таблицы. Выведите на экран меню, а затем информацию о друзьях в зависимости от вашего запроса.

5.24 Составьте банк данных кинологов, включающий в себя фамилию и имя владельца собаки, кличку собаки, породу собаки, день, месяц и год рождения собаки, а также в зависимости от породы наличие соответствующей дополнительной информации по своему усмотрению. Информацию о каждом владельце оформите в программе в виде записи с вариантами. Совокупность записей объедините в массив. Составьте программу, которая обеспечивает ввод полученной информации, распечатку ее в виде таблицы. Выведите на экран меню, а затем информацию в зависимости от вашего запроса.

5.25 Составьте банк данных районного отдела милиции, включающий в себя фамилию, имя и отчество нарушителя, дату рождения, и в зависимости от поля «Судимость» наличие соответствующей дополнительной информации по своему усмотрению (например, наличие клички, мера наказания, срок заключения). Информацию о каждом нарушителе оформите в программе в виде записи с вариантами. Совокупность записей объедините в массив. Составьте программу, которая обеспечивает ввод полученной информации, распечатку ее в виде

таблицы. Выведите на экран меню, а затем информацию в зависимости от вашего запроса.

5.26 Составить программу, которая бы считывала и анализировала введенную информацию о пользователях городской телефонной сети. Запись должна иметь поля: фамилию, имя, отчество; номер телефона; адрес; наличие задолженности по оплате.

Предусмотрите в программе варианты полей в зависимости от заполнения поля «Задолженность по оплате» (например, размер долга, отключение от междугородней сети, подсчет пени и другое).

По введенной информации и запросу пользователя предусмотреть в программе вывод предупреждения абонентов, имеющих задолженность, и какое-либо поощрение ответственным плательщикам.

Лабораторная работа № 6. Работа с файлами в C++

Цель работы – познакомиться с основными конструкциями, предназначенными для работы с файлами.

1 Фрагмент теории

Существует достаточно много средств работы с файлом в этом языке программирования. Мы здесь остановимся на некоторых из них.

Класс **ofstream** предназначен для организации работ по *выводу (записи)* в файл с помощью методов этого класса:

- **open()** – открывает файл для записи;
- **is_open()** – возвращает **true**, если файл открыт, и **false** – в противном случае;
- **put()** – записывает в файл один символ;
- **write()** – записывает в файл заданное число символов;
- **seekp()** – перемещает указатель позиционирования в заданное место файла;
- **tellp()** – выдает текущее значение указателя позиционирования;
- **close()** – закрывает файл;
- **rdbuf()** – выдает указатель на буфер вывода (этот буфер находится в структуре, с которой связывается файл при его открытии).

Ниже приведен пример использования класса **ofstream**.

```
#include <fstream>

. . . . .
ofstream FILE; //объявляем переменную FILE типа
                //ofstream (создаем экземпляр класса)
FILE.open("a.txt"); //вызываем метод открытия файла
if ( FILE == NULL ) return 0; //неудачное открытие файла
. . . . .
for( int i = 0; i < 2; i++ )
    FILE << "Строка " << i << endl; //вывод в файл
. . . . .
FILE.close(); //закрытие файла.
```

Класс **ifstream** предназначен для организации работ по *вводу (чтению)* из файла помощью методов этого класса:

- **open()** – открывает файл для чтения;
- **is_open()** – возвращает **true**, если файл открыт, и **false** – в противном случае;
- **get()** – читает из файла один символ;
- **read()** – читает из файла заданное число символов;
- **eof()** – возвращает ненулевое значение, когда файловый указатель достигает конца файла;
- **peek()** – выдает очередной символ потока, но не выбирает его (не сдвигает файловый указатель);
- **seekg()** – перемещает указатель позиционирования в заданное место файла;
- **tellg()** – выдает текущее значение указателя позиционирования;
- **close()** – закрывает файл;
- **rdbuf()** – выдает указатель на буфер вывода (этот буфер находится в структуре, с которой связывается файл при его открытии).

Ниже приведен пример использования этого класса.

```
#include <fstream>

ifstream FILE; //объявляем переменную FILE типа
               //ifstream (создаем экземпляр класса)
FILE.open ("a.txt"); //вызываем метод открытия файла
if ( FILE == NULL ) return 0; //неудачное открытие файла

while(!FILE.eof()) //проверка на признак конца файла
{
    FILE >> p;           //чтение из файла
    cout >> p >> endl;   //вывод прочитанных данных на экран
}

FILE.close(); //закрытие файла.
```

Для работы с текстовыми файлами можно использовать два класса: **StreamReader** и **StreamWriter**.

Для записи в текстовый файл используется класс **StreamWriter**. Некоторые из его конструкторов, которые могут применяться для создания объекта **StreamWriter**:

- **StreamWriter(string path)** – через параметр **path** передается путь к файлу, который будет связан с потоком;
- **StreamWriter(string path, bool append)** – параметр **append** указывает, надо ли добавлять в конец файла данные или же перезаписывать файл. Если равно **true**, то новые данные добавляются в конец файла. Если равно **false**, то файл перезаписывается заново.

Свою функциональность **StreamWriter** реализует через следующие методы:

- **int Close()** – закрывает записываемый файл и освобождает все ресурсы;

- **void Flush()** – записывает в файл оставшиеся в буфере данные и очищает буфер;

- **void Write(string value)** – записывает в файл данные простейших типов, как **int**, **double**, **char**, **string** и т. д. Соответственно имеет ряд перегруженных версий для записи данных элементарных типов, например: **Write(char value)**, **Write(int value)**, **Write(double value)** и т. д.;

- **void WriteLine(string value)** – также записывает данные, только после записи добавляет в файл символ окончания строки.

Класс **StreamReader** позволяет нам легко считывать весь текст или отдельные строки из текстового файла.

Один из конструкторов класса **StreamReader**:

StreamReader(string path) – через параметр **path** передается путь к считываемому файлу.

Среди методов **StreamReader** можно выделить следующие:

- **void Close()** – закрывает считываемый файл и освобождает все ресурсы;
- **int Peek()** – возвращает следующий доступный символ, если символов больше нет, то возвращает -1;

- **int Read()** – считывает и возвращает следующий символ в численном представлении. Имеет перегруженную версию: **Read(char array[], int index, int count)**, где **array[]** – массив, куда считываются символы, **index** – индекс в массиве **array[]**, начиная с которого записываются считываемые символы, и **count** – максимальное количество считываемых символов;

- **string ReadLine()** – считывает одну строку из файла;

- **string ReadToEnd()** – считывает весь текст из файла.

2 Задачи для самостоятельного решения

6.1 Дан файл *f*, компоненты которого являются действительными числами. Найти:

а) сумму компонентов файла *f*;

б) произведение компонентов файла *f*;

в) сумму квадратов компонентов файла *f*;

г) модуль суммы и квадрат произведения компонентов файла *f*;

д) последний компонент файла.

6.2 Дан файл *f*, компоненты которого являются действительными числами. Найти:

а) наибольшее из значений компонентов;

б) наименьшее из значений компонентов с четными номерами;

в) наибольшее из значений модулей компонентов с нечетными номерами;

г) разность первого и последнего компонентов файла.

6.3 Дан файл *f*, компоненты которого являются целыми числами. Найти количество квадратов простых чисел среди компонентов.

6.4 Дан символьный файл *f*. Получить копию файла в файле *g*.

6.5 Даны символьные файлы f_1 и f_2 . Переписать с сохранением порядка следования компоненты файла f_1 в файл f_2 , а компоненты файла f_2 – в файл f_1 . Использовать вспомогательный файл h .

6.6 Дан символьный файл f . В файле не менее двух компонентов. Определить, являются ли два первых символа файла цифрами. Если да, то установить, является ли число, образованное этими цифрами, четным.

6.7 Дан файл f , компоненты которого являются целыми числами. Получить в файле g все компоненты файла f :

- а) являющимися четными числами;
- б) делящиеся на 3 и не делящиеся на 7;
- в) являющимися точными квадратами.

6.8 Вычислить по схеме Горнера значение многочлена с рациональными коэффициентами для данного рационального значения переменной. Считать, что числители и знаменатели коэффициентов записаны в файле f в следующем порядке: вначале числитель и знаменатель старшего коэффициента и т. д., в последнюю очередь числитель и знаменатель свободного члена.

6.9 Дан файл f , компоненты которого являются целыми числами. Записать в файл g все четные числа из файла f , а в файл h – все нечетные. Порядок следования чисел сохранить.

6.10 Дан символьный файл f . Записать в файл g компоненты файла f в обратном порядке.

6.11 Даны символьные файлы f и g . Записать в файл h сначала компоненты файла f , затем – компоненты файла g с сохранением порядка.

6.12 Дан файл f , компоненты которого являются целыми числами. Получить файл g , образованный из файла f исключением повторных вхождений одного и того же числа.

6.13 Дан файл f , компоненты которого являются целыми числами. Все компоненты файла не равны нулю. Файл f содержит столько же отрицательных чисел, сколько и положительных. Используя вспомогательный файл h , переписать компоненты файла f в файл g так, чтобы в файле g :

- а) не было двух соседних чисел с одинаковым знаком;
- б) сначала шли положительные, затем отрицательные числа;
- в) числа шли в следующем порядке: два положительных, два отрицательных, два положительных, два отрицательных и т. д. (предполагается, что число компонентов в файле f делится на 4).

6.14 Дан файл f , компоненты которого являются целыми числами. Все компоненты файла не равны нулю. Числа в файле идут в следующем порядке: десять положительных, десять отрицательных, десять положительных, десять отрицательных и т. д. Переписать компоненты файла f в файл g так, чтобы в файле g числа шли в следующем порядке:

- а) пять положительных, пять отрицательных, пять положительных, пять отрицательных и т. д.;
- б) двадцать положительных, двадцать отрицательных, двадцать положительных, двадцать отрицательных и т. д. (предполагается, что число компонентов в файле f делится на 40).

6.15 Дан файл f , компоненты которого являются целыми числами. Число компонентов файла делится на 100. Записать в файл g наибольшее значение первых ста компонентов файла f , затем – следующих ста компонентов и т. д.

6.16 Дан файл f , компоненты которого являются целыми числами. Записать в файл g наибольшее значение первых ста компонентов файла f , затем – следующих ста компонентов и т. д. Если в последней группе окажется менее ста компонентов, то последний компонент файла g должен быть равен наибольшему из компонентов файла f , образующих последнюю (неполную) группу.

6.17 Дан символьный файл f . Добавить в его конец символы «e», «n», «d» (если это необходимо, использовать вспомогательный файл g).

6.18 Дан символьный файл f .

а) Подсчитать число вхождений в файл сочетаний «ab».

б) Определить входит ли в файл сочетание «abcdefgh».

в) Подсчитать число вхождений в файл каждой из букв a, b, c, d, e, f и вывести результат в виде таблицы $a - N_a, b - N_b, c - N_c, d - N_d, e - N_e, f - N_f$, где $N_a, N_b, N_c, N_d, N_e, N_f$ – количество вхождений соответствующих букв.

6.19 Даны символьные файлы f и g . Определить, совпадают ли компоненты файла f с компонентами файла g . Если нет, то получить номер первого компонента, в котором файлы f и g отличаются между собой. В случае, когда один из файлов имеет n компонентов ($n \geq 0$) и повторяет начало другого (более длинного) файла, ответом должно быть число $n+1$.

6.20 Даны символьные файлы f и g . Записать в файл h все начальные совпадающие компоненты файлов f и g .

6.21 Дан символьный файл f . Записать в файл g с сохранением порядка следования те символы файла f :

а) которым в этом файле предшествует буква «a»;

б) вслед за которыми в этом файле идет буква «a».

6.22 Дан символьный файл f . Группы символов, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Удалить из файла все однобуквенные слова и лишние пробелы. Результат записать в файл g .

6.23 Дан символьный файл f . Найти самое длинное слово (см. предыдущую задачу) среди слов, вторая буква которых есть «e»; если слов с наибольшей длиной несколько, то найти последнее. Если таких слов нет вообще, то сообщить об этом. Решить эту задачу:

а) полагая, что слова состоят не более чем из 10 символов;

б) без ограничения на длину слов.

6.24 Дан символьный файл f . Считая, что количество символов в слове не превосходит двадцати:

а) определить, сколько в файле f имеется слов, состоящих из одного, двух, трех и т. д. символов;

б) получить гистограмму (столбчатую диаграмму) длин всех слов файла f ;

в) определить количество слов в файле f .

6.25 Дан символьный файл *f*. Предполагается, что длина одного слова не превосходит десяти и что число слов делится на 100. Подготовить файл для печати слов в две колонки по пятьдесят строк на странице. Слова должны быть размещены в файле *f1* в следующем порядке: 1-е слово, 51-е слово, 2-е слово, 52-е слово, ..., 50-е слово, 100-е слово, затем (следующая страница) 101-е слово, 151-е слово, ..., 150-е слово, 200-е слово и т. д.

6.26 Дан символьный файл *f*, содержащий сведения о сотрудниках учреждения, записанные по следующему образцу: фамилия имя отчество фамилия имя отчество ...

Записать эти сведения в файле *g*, используя образцы:

а) имя отчество фамилия имя отчество фамилия ...;

б) фамилия отчество имя фамилия отчество имя ...

6.27 Даны два символьных файла *f1* и *f2*. Файл *f1* содержит произвольный текст. Слова в тексте разделены пробелами и знаками препинания. Файл *f2* содержит не более 40 слов, которые разделены запятыми. Эти слова образуют пару: первое слово считается заменяемым, а второе – заменяющим. Найти в файле *f1* все заменяемые слова и заменить их на заменяющие. Результат поместить в файл *g*.

6.28 Сведения об автомобиле состоят из его марки, номера и фамилии владельца. Дан файл *f*, содержащий сведения о нескольких автомобилях. Найти:

а) фамилии владельцев и номера автомобилей данной марки;

б) количество автомобилей данной марки.

6.29 Дан файл *f*, содержащий различные даты. Каждая дата – это число, месяц, год. Найти:

а) год с наименьшим номером;

б) все весенние даты;

в) самую позднюю дату.

6.30 Дан файл *f*, содержащий сведения об экспортируемых товарах: указывается наименование товара, страна, импортирующая товар, и объем поставляемой партии в штуках. Найти страны, в которые экспортируется данный товар, и общий объем его экспорта.

6.31 Дан файл, содержащий целые числа. Первый и центральный элементы этого файла поменять местами (новый файл не заводить).

6.32 Дан файл, содержащий целые числа. Последний и центральный элементы этого файла поменять местами (новый файл не заводить).

6.33 Вывести на экран самую длинную строку текстового файла.

6.34 Вывести из текстового файла все слова палиндромы.

6.35 Дан целочисленный файл *f*. Все отрицательные элементы этого файла заменить нулями (новый файл не заводить).

6.36 В текстовый файл построчно записаны фамилия и имя учащихся класса и его оценка за контрольную. Вывести на экран всех учащихся, чья оценка меньше 3 баллов и посчитать средний балл по классу.

Примерное содержание файла:

Иванов О. 4

Петров И. 3
Дмитриев Н. 2
Смирнова О. 4
Керченских В. 5
Котов Д. 2
Бирюкова Н. 1
Данилов П. 3
Аранских В. 5
Лемонов Ю. 2
Олегова К. 4

6.37 Подсчитать в текстовом файле количество слов.

6.38 Дан файл, содержащий текстовые строки. Удалить из него все числа.

6.39 Даны два файла со словами, разделенными пробелами. Создать новый файл, содержащий: а) строки, которые встречаются только в первом файле; б) строки, которые встречаются в обоих файлах; в) строки, которые встречаются в каждом файле более двух раз.

6.40 Даны два файла, состоящие из предложений. Создать третий файл, содержащий все предложения, которые есть хотя бы в одном из файлов. Повторы не добавлять в третий файл.

6.41 Дан файл с html-кодом. Показать все ссылки из этого кода.

6.42 Дан файл с текстом. Показывать содержимое файла по страницам с навигацией. На каждую страницу помещается заданное пользователем количество символов.

6.43 Сделать тестирующую программу. Программа должна состоять из четырёх страниц по два вопроса на каждой. Варианты ответов обозначить радикальными переключателями – по три варианта на каждый вопрос. Список номеров правильных ответов хранится в текстовом файле, номера разделены пробелами. На пятой странице программы выводится количество правильных ответов и содержится кнопка «Пройти заново», которая возвращает пользователя на первую страницу.

6.44 Пользователь загружает текстовый файл со списком ссылок. Добавить в базу (другой файл) из этого файла только те ссылки, которых нет ни в базе, ни в файле с запрещенными ссылками.

6.45 Дано целое число $K > 0$ и строковый файл. Создать два новых файла: строковый, содержащий первые K символов каждой строки исходного файла, и символьный, содержащий K -й символ каждой строки (если длина строки меньше K , то в строковый файл записывается вся строка, а в символьный файл записывается пробел).

6.46 Дан текстовый файл, содержащий более трех строк. Удалить из него последние три строки.

6.47 Дано имя файла и целые положительные числа N и K . Создать текстовый файл с указанным именем и записать в него N строк, каждая из которых состоит из K символов «*» (звездочка).

6.48 Дан файл вещественных чисел. Заменить в нем все элементы на их квадраты.

6.49 Напишите программу, которая заменяет одно слово в текстовом файле на другое. Словом называется последовательность непробельных символов, ограниченная пробелами или границами строки. Слово-образец может начинаться как с заглавной, так и со строчной буквы; замена должна быть соответствующая. Если слово-образец совпадает с частью какого-то слова, замена не выполняется. После слова может стоять знак препинания из следующего набора: «.,;!?». Исходный текст записан в файле input.txt, обработанный текст нужно вывести в файл output.txt.

6.50 Прямая на плоскости задается уравнением $ax + by + c = 0$, где a и b одновременно не равны нулю. Будем рассматривать только прямые, для которых коэффициенты a , b , c – целые числа. Пусть f – файл, содержащий коэффициенты нескольких прямых (не менее трех). Переписать из файла f в файл g коэффициенты тех прямых, которые:

- а) параллельны первой из прямых, заданной в файле f ;
- б) указаны в а), но дополнительно требуется, чтобы все прямые были различны;
- в) пересекают первую из прямых, заданных в файле f ;
- г) указаны в в), но дополнительно требуется, чтобы среди прямых не было параллельных.

Лабораторная работа № 7. Построение графиков в C++

Цель работы – получить первоначальные навыки использования компонента **Chart**.

1 Фрагмент теории

Компонент **Chart** позволяет строить различные диаграммы и графики. Он является контейнером объектов **Series** – серий данных, характеризующихся различными стилями отображения. Каждый компонент может включать несколько серий. Например, при отображении графика каждая серия будет соответствовать одной кривой на графике. Очевидно, что в этом случае серии можно наложить друг на друга. В случае же диаграмм, например круговых, наложение затруднит понимание представленных данных. Однако в последнем случае можно задать для одного компонента **Chart** несколько серий одинаковых данных с разным типом диаграммы. Тогда, делая в каждый момент времени активной одну из них, можно предоставить пользователю выбор типа диаграммы, отображающей интересующие его данные.

Компонент **Chart** имеет множество свойств, методов, событий. Более подробную информацию и примеры его использования можно получить на сайте «Информатика и программирование: шаг за шагом», в разделе «Microsoft Visual C++ 2010. Язык C/C++», начиная с 225 шага (http://it.kgsu.ru/C++MSVS2010/c++vs2010_225.html).

2 Задачи для самостоятельного решения

Построить графики функций, самостоятельно выбрать удобные параметры настройки.

$$7.1 \quad t = \frac{2 \cos(x - \frac{\pi}{6})}{0,5 + \sin^2 y} \left(1 + \frac{z^2}{3 - z^2/5}\right).$$

$$7.2 \quad u = \frac{\sqrt[3]{8 + |x - y|^2 + 1}}{x^2 + y^2 + 2} - e^{|x-y|} (tg^2 z + 1)^x.$$

$$7.3 \quad v = \frac{1 + \sin^2(x+y)}{\left|x - \frac{2y}{1+x^2y^2}\right|} x^{|y|} + \cos^2\left(\operatorname{arctg} \frac{1}{z}\right).$$

$$7.4 \quad w = |\cos x - \cos y|^{(1+2\sin^2 y)} \left(1 + z + \frac{z^2}{2} + \frac{z^3}{3} + \frac{z^4}{4}\right).$$

$$7.5 \quad \alpha = \ln\left(y^{-\sqrt{|x|}}\right) \left(x - \frac{y}{2}\right) + \sin^2 \operatorname{arctg}(z).$$

$$7.6 \quad \beta = \sqrt{10(\sqrt[3]{x} + x^{y+2})} (\arcsin^2 z - |x - y|).$$

$$7.7 \quad \gamma = 5 \operatorname{arctg}(x) - \frac{1}{4} \arccos(x) \frac{x + 3|x - y| + x^2}{|x - y|z + x^2}.$$

$$7.8 \quad \varphi = \frac{e^{|x-y|} |x - y|^{x+y}}{\operatorname{arctg}(x) + \operatorname{arctg}(z)} + \sqrt[3]{x^6 + \ln^2 y}.$$

$$7.9 \quad \psi = \left|x^{\frac{y}{x}} - \sqrt[3]{\frac{y}{x}}\right| + (y - x) \frac{\cos y - \frac{z}{(y-x)}}{1 + (y-x)^2}.$$

$$7.10 \quad a = 2^{-x} \sqrt{x + 4\sqrt{|y|}} \sqrt[3]{e^{x-1/\sin z}}.$$

$$7.11 \quad b = y^{\sqrt[3]{|x|}} + \cos^3(y) \frac{|x - y| \left(1 + \frac{\sin^2 z}{\sqrt{x + y}}\right)}{e^{|x-y|} + \frac{x}{2}}.$$

$$7.12 \quad c = 2^{(y^x)} + (3^x)^y - \frac{y \left(\operatorname{arctg} z - \frac{\pi}{6}\right)}{|x| + \frac{1}{y^2 + 1}}.$$

$$7.13 \quad f = \frac{\sqrt[4]{y + \sqrt[3]{x-1}}}{|x - y| (\sin^2 z + tgz)}.$$

$$7.14 \quad g = \frac{y^{x+1}}{\sqrt[3]{|y-2| + 3}} + \frac{x + \frac{y}{2}}{2|x + y|} (x+1)^{-1/\sin z}.$$

$$7.15 \quad h = \frac{x^{y+1} + e^{y-1}}{1 + x|y - tgz|} (1 + |y - x|) + \frac{|y - x|^2}{2} - \frac{|y - x|^3}{3}.$$

$$7.16 \quad q = b - 2f.$$

$$7.17 \quad p = ta.$$

$$7.18 \quad k = g + \frac{c}{2}.$$

$$7.19 \quad l = f + \frac{1}{f}.$$

$$7.20 \quad m = v - |u|.$$

$$7.21 \quad n = \ln(a + \arcsin(a)).$$

$$7.22 \quad o = 1 - ah.$$

$$7.23 \quad d = \varphi - \frac{1}{t}.$$

$$7.24 \quad s = \sin(|h|).$$

$$7.25 \quad r = \sqrt{|\gamma - 1|}.$$

РАСПРЕДЕЛЕНИЕ ЗАДАЧ ПО ВАРИАНТАМ

При решении задач, если сказано: «дана строка», «дано число» и т. п., это означает, что эти значения вводятся с клавиатуры.

При решении задач с файлами, фраза «дан файл» подразумевает, что в программе нужно организовать ввод элементов файла, после чего вывести его содержимое, а сам файл закрыть. При решении задачи **заранее неизвестно** количество элементов в файле; считается, что оно может быть очень большим.

Замечания.

1-й раздел: для одной задачи создать консольное приложение, а для второй – Windows Forms.

2-й раздел: консольное приложение, для размещения элементов массива резервировать память в «куче».

3-й раздел: для одной задачи создать консольное приложение, а для второй – Windows Forms.

4-й раздел: для одной задачи создать консольное приложение, а для второй – Windows Forms.

5-й раздел: Windows Forms.

6-й раздел: для одной задачи создать консольное приложение, а для второй – Windows Forms.

7-й раздел: Windows Forms.

ВАРИАНТ 1: 1.1, 1.28, 2.7, 3.1, 3.26, 4.1, 4.29, 5.1, 6.1, 6.48, 7.15.
ВАРИАНТ 2: 1.2, 1.29, 2.24, 3.2, 3.29, 4.2, 4.30, 5.2, 6.2, 6.47, 7.14.
ВАРИАНТ 3: 1.3, 1.31, 2.22, 3.3, 3.30, 4.4, 4.27, 5.3, 6.10, 6.46, 7.12.
ВАРИАНТ 4: 1.4, 1.39, 2.20, 3.4, 3.31, 4.5, 4.26, 5.4, 6.3, 6.38, 7.11.
ВАРИАНТ 5: 1.5, 1.32, 2.18, 3.5, 3.32, 4.6, 4.28, 5.6, 6.11, 6.37, 7.4.
ВАРИАНТ 6: 1.6, 1.33, 2.16, 3.6, 3.33, 4.7, 4.31, 5.7, 6.7, 6.33, 7.2.
ВАРИАНТ 7: 1.7, 1.34, 2.14, 3.7, 3.34, 4.3, 4.32, 5.5, 6.4, 6.32, 7.1.
ВАРИАНТ 8: 1.8, 1.35, 2.12, 3.8, 3.35, 4.10, 4.33, 5.8, 6.6, 6.35, 7.5.
ВАРИАНТ 9: 1.9, 1.36, 2.9, 3.9, 3.36, 4.11, 4.34, 5.9, 6.8, 6.45, 7.3.
ВАРИАНТ 10: 1.10, 1.37, 2.10, 3.10, 3.37, 4.12, 4.35, 5.10, 6.9, 6.36, 7.13.
ВАРИАНТ 11: 1.11, 1.30, 2.6, 3.11, 3.38, 4.13, 4.36, 5.11, 6.5, 6.34, 7.10.
ВАРИАНТ 12: 1.12, 1.27, 2.4, 3.12, 3.39, 4.14, 4.38, 5.12, 6.12, 6.31, 7.6.
ВАРИАНТ 13: 1.13, 1.38, 2.2, 3.13, 3.41, 4.15, 4.39, 5.13, 6.15, 6.30, 7.7.
ВАРИАНТ 14: 1.14, 1.49, 2.1, 3.14, 3.44, 4.9, 4.40, 5.15, 6.14, 6.29, 7.8.
ВАРИАНТ 15: 1.15, 1.50, 2.3, 3.15, 3.45, 4.8, 4.41, 5.14, 6.13, 6.28, 7.9.
ВАРИАНТ 16: 1.16, 1.48, 2.5, 3.16, 3.48, 4.16, 4.42, 5.16, 6.21, 6.27, 7.16.
ВАРИАНТ 17: 1.17, 1.47, 2.9, 3.17, 3.27, 4.17, 4.43, 5.17, 6.22, 6.39, 7.17.
ВАРИАНТ 18: 1.18, 1.45, 2.11, 3.18, 3.28, 4.20, 4.44, 5.18, 6.16, 6.26, 7.18.
ВАРИАНТ 19: 1.19, 1.46, 2.13, 3.19, 3.40, 4.22, 4.45, 5.19, 6.17, 6.40, 7.19.
ВАРИАНТ 20: 1.20, 1.44, 2.15, 3.20, 3.42, 4.23, 4.47, 5.20, 6.18, 6.41, 7.20.
ВАРИАНТ 21: 1.21, 1.43, 2.17, 3.21, 3.43, 4.21, 4.48, 5.21, 6.19, 6.42, 7.21.
ВАРИАНТ 22: 1.22, 1.42, 2.19, 3.22, 3.46, 4.19, 4.49, 5.22, 6.20, 6.43, 7.22.
ВАРИАНТ 23: 1.23, 1.41, 2.21, 3.23, 3.47, 4.18, 4.50, 5.23, 6.23, 6.44, 7.23.
ВАРИАНТ 24: 1.24, 1.26, 2.23, 3.24, 3.49, 4.24, 4.46, 5.24, 6.24, 6.49, 7.24.
ВАРИАНТ 25: 1.25, 1.40, 2.25, 3.25, 3.50, 4.25, 4.37, 5.25, 6.25, 6.50, 7.25.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Среда программирования Microsoft Visual C++ 2010. Язык C/C++ // Информатика и программирование: шаг за шагом : офиц. сайт. – URL: <http://it.kgsu.ru/C++MSVS2010/oglav.html> (дата обращения: 01.07.21).

2 Язык программирования C++. Начала // Информатика и программирование: шаг за шагом : офиц. сайт. – URL: <http://it.kgsu.ru/C++/oglav.html> (дата обращения: 01.07.21).

3 Чтение и запись текстовых файлов. StreamReader и StreamWriter // METAINT.COM: сайт о программировании : офиц. сайт. – URL: <https://metanit.com/sharp/tutorial/5.5.php> (дата обращения: 01.07.21).

Медведев Аркадий Андреевич

ИЗУЧЕНИЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ C++

Методические рекомендации
для подготовки бакалавров и специалистов
направлений 09.03.03 «Прикладная информатика», 09.03.04 «Программная
инженерия», 10.05.01 «Компьютерная безопасность»,
10.05.03 «Информационная безопасность»

Редактор В. С. Никифорова

Подписано в печать 23.08.21	Формат 60x84 1/16	Бумага 80 г/м ²
Печать цифровая	Усл. печ. л. 3,5	Уч.-изд. л. 3,5
Заказ № 92	Тираж 25	

Библиотечно-издательский центр КГУ.
640020, г. Курган, ул. Советская, 63/4.
Курганский государственный университет.