

Вариант олимпиадного задания состоит из четырех задач. Оценивание задач происходит по первичным баллам. Максимальное количество баллов, которые могут быть начислены за правильные ответы по каждой задаче, представлены в таблице:

<i>Задача №</i>	<i>Максимальное количество баллов (10-11 класс)</i>
1	2
2	5
3	8
4	10

Подсчет первичной оценки осуществляется путем суммирования первичных баллов, выставленных за ответы на каждый из вопросов. Максимальная сумма первичных баллов по всем правильно решенным задачам равна **25**. Перевод первичных баллов в итоговые осуществляется умножением первичных баллов на коэффициент 4. Максимальная сумма итоговых баллов по всем правильно выполненным задачам равна **100**.

#### **Задание №1.**

Решить систему уравнений относительно  $x, y$ :

$$\begin{cases} 21_x \cdot 30_y = 420_x \\ 10_y \cdot 10_x = 120_y \end{cases}$$

**Цель задачи:** проверить знание систем счисления и умение проводить действия с ними.

#### **Схема решения:**

Для решения подобного рода задач необходимо:

1. преобразовать каждое число в уравнение (из системы счисления  $X, Y$  перейти в десятичную систему счисления);
2. составить систему уравнений;
3. решить составленную систему уравнений;
4. получить либо единственное решение, либо множество решений, либо сделать вывод об отсутствии каких-либо решений в целых числах. При этом надо понимать, что основанием (т.е. решением системы) не может быть целое число, меньшее максимальных значений цифр в первоначальном представлении системы.

#### **Задание №2.**

В файле **in.txt** в одну строчку написано большое натуральное число (более ста знаков, неспособное поместиться в память, выделяемую для целого числа в большинстве реализаций языков программирования) и через запятую второе число - трехзначное. Требуется написать программу поиска остатка от деления первого числа на второе. Результат выполнения программы вывести в файл **out.txt**.

**Пример ввода (сокращенный для удобства):** 6546846343546846462313129, 345

**Вывод:** 344.

**Цель задачи:** проверка знания и/или умения производить "длинную арифметику".

#### **Схема решения:**

Для решения подобного рода задач необходимо:

1. Считывание делимого и делителя;  
Замечание 1: считывание делимого происходит в текстовую переменную (либо в массив, что не всегда является оптимальным); при считывании делимого и делителя необходимо учесть, что их разделяют запятая и пробел.  
Замечание 2: по условию задачи исходные данные вводятся из файла (или с клавиатуры, в случае создания пользовательского интерфейса). При этом указан размер делителя. В каждом варианте олимпиадных задач размер делителя разный. Наиболее универсальным является программный код который учитывает ситуацию когда размера делителя определяется после считывания входных данных.
2. Отделение от делимого части (с левого края), состоящей из  $k$  знаков, и перевод этой части в числовую переменную. Делитель (если это не сделано ранее) так же переводится из текстовой в числовую переменную.  
Замечание 1: после отделения  $k$  знаков, можно удалить их из текстовой переменной тем самым сократив ее длину. Построенный цикл считывания основан на постепенном уменьшении длины строки содержащей делимое.  
Замечание 2: при отделении части делимого иногда можно учитывать, что она должна быть больше чем делитель. Такая проверка может сократить количество шагов.
3. Проводится целочисленное деление и получение остатка от такого деления.
4. Полученный остаток умножается на 10 и к полученному числу прибавляется цифра стоящая на  $k+1$  в первоначальной записи делимого. Дальнейшие действия происходят с полученным числом.  
Замечание: если в п.2 алгоритм учитывает замечание №1 то в п.4 добавляется первый (слева) знак из делимого т.к. после удаления  $k$  символов из строки, следующий ( $k+1$ ) символ перемещается на первую позицию.
5. Последовательное повторение пунктов №3, 4, 5 до тех пор пока делимое (в п.4.) больше чем делитель или не исчерпаются все знаки в делимом.
6. Полученное значение выводится на экран.

Замечание. В п.3. может быть применен альтернативный подход: Пусть имеется делимое  $A$  и делитель  $B$ ; пусть  $C$  является целой частью при делении  $A$  на  $B$ , а  $D$  является остатком от деления т.е.  $A=C*B+D$ . Возможен следующий подход:  $A$  делится на  $B$ ; у полученного числа выделяется целая часть  $C$ ; остаток  $D$  находится по формуле:  $D=A - C*B$ . Дальнейшие действия аналогичны представленным в алгоритме.

### Задание №3.

**Цель задачи:** проверка написания процедур и функций, а так же умения работать с массивами и проверка (в некоторых задачах: проверка умения осуществлять сортировки и выявлять взаимосвязи между элементами).

Дан двумерный массив размером  $N \times N$ , заполненный символами A-Z. Написать программу, осуществляющую поиск в данном массиве строк размером  $\leq N$ , заданных в файле и вывод координат первой буквы каждого слова и указателя направления слова. Строки могут располагаться горизонтально, вертикально и диагонально, и при этом быть написаны в любую сторону. Указатель направления – это направление, в котором читается строка в массиве (С, Ю, З, В, СЗ, ЮВ и т.д.). Если строка присутствует в массиве несколько раз, то необходимо вывести все ее вхождения в массив. Если же строка не встречается в массиве, то явно указать это.

**Пример:**

**Вход :**

**Выход :**

	4	DO	
	Q N O S	(4, 4) Э	
	E A T N	(4, 4) С	
	Z N O O	(4, 4) СЭ	
	A B O D	NOT	
	DO	Не найдено	
	NOT	EAT	
	EAT	(2, 1) В	
	ANTS	ANTS	
		(4, 1) СВ	

### Схема решения:

Для решения подобного рода задач необходимо:

1. Осуществить ввод исходных данных, выделить массив для поиска и слова которые необходимо найти.
2. Взять первую букву слова (которое необходимо найти) и найти ее в заданном массиве (найти все ее положения).
3. Последовательной проверкой по разным направлениям проверить совпадение других букв массива соответствующим буквам искомого слова. В случае если поиск слова завершился удачно необходимо вывести направление слова. В случае не удачи – продолжить поиск по другим направлениям. В случае исчерпания направлений и не удачи в поиске слова – выдать "не найдено".

Замечание 1: необходимо учесть, что один раз выбрав направление, последующую проверку, на соответствие букв искомому слову, следует вести строго в этом направлении.

Замечание 2: проверку по направлениям можно объединить в одну процедуру или функцию. Подобное объединение следует считать наиболее оптимальным (и удобным для описания с помощью комментариев).

Замечание 3: проверку можно упростить если, сравнить длину искомого слова с возможной длиной слова в выбранном направлении (предотвращает выход за пределы массива).

4. Повторить п.2 для нового слова. По исчерпанию слов для поиска – завершить работу программы.

Замечание: возможен иной подход к решению задачи. Для каждой буквы (ячейки) массива составляется множество возможных значений (по всем возможным направлениям) с запоминанием направления для каждого слова; далее производится сортировка массива; окончательный поиск для искомого слова происходит по отсортированному массиву с выводением значения для направления каждого слова. Подобный подход не всегда бывает оптимальным т.к. требует создания дополнительного массива большой размерности. Однако, бывают задачи когда это оправдано т.к. составленный словарь значительно упрощает поиск искомого слова.

### Задание №4.

**Цель задачи:** проверка знания принципов раскраски и обхода лабиринта, а так же принципов написания процедур и функций.

#### Задача №4.

В системе подземных туннелей началось подтопление. Вода двигается во все возможные направления с одинаковой скоростью. В некоторых местах туннелей установлены датчики подтопления. Требуется, имея схематический план туннелей в

файле in.txt, в out.txt вывести номер датчика, который сработает первым.

Формат ввода: двумерный массив из символов (максимум 15x15), схематически отображающий план туннелей, в котором символы 'x' обозначают стены, нули – свободное пространство. 'Т' обозначает источник воды, числа от 1 до n – номера датчиков. Длины коридоров прямо пропорциональны числу нулей, в их представлении.

**Пример:**

**Ввод**

```
x x x x x x x x x x x x x x x
x x x x 5 x x x x x x x x x x
x x x x 0 x x x x x x x 1 x x
x x x x 0 x x x x x x x 0 x x
x x x x 0 x x x x x x x 0 x x
x x x x 0 x x 0 0 0 0 0 0 x x
x x x x 0 x x 0 x x x x 0 x x
x 4 0 0 0 0 x 0 x x x x 0 x x
x x x x x 0 x T x x x x 0 x x
x x x x x 0 0 0 0 0 0 0 0 x x
x x x x x x x x x 0 x x 2 x x
x x x x x x x x x 0 x x x x x
x x x x x x x x x 0 0 3 x x x
x x x x x x x x x x x x x x x
```

**Вывод: 2**

**Схема решения:**

Для решения подобного рода задач необходимо:

1. Считать исходные данные в массив.
2. Найти координаты начальной точки "Т" и обнулить счетчик пройденного пути.
3. Проверка соседних ячеек массива на наличие в них "0" и, в случае его обнаружения, перейти в эту позицию. Текущее значение счетчика пути увеличивается на 1. Действие пункта повторяется до тех пор пока не будет найдена цифра (значение отличное от "0" и "x").

Замечание 1: проверка осуществляется вправо, влево, вверх и вниз от текущей позиции. Проверка элемента по диагонали приведет к грубой ошибке (сокращение пути, не рассмотрение некоторых решений).

Замечание 2: проверку по направлениям можно свести в единую процедуру, входными параметрами которой будет направление обхода т.е. создание универсально процедуры.

4. В случае обнаружения в соседней ячейке цифры (не "0" и не "x"), вычисляется длина пройденного пути.
5. Вычисляется минимальная длина среди всех контрольных точек (обозначенных цифрами) и выводится номер, соответствующий минимальной длине.

Возможен альтернативный подход к решению задачи (раскраска):

1. Рассматриваются два массива. В одном хранится лабиринт, в другом, временном (заполняется "-1"), формируется план обхода лабиринта.
2. Находятся координаты контрольных точек.
3. Находится начало пути, точка "Т" (текущее значение счетчика =0).
4. Проверяются соседние (справа, слева, сверху, снизу от текущей точки) ячейки.

Если в них содержится "0", то во временном массиве в соответствующей ячейке ставится значение счетчика (увеличивается на 1 от текущего для предыдущей точки).

Если в них содержится число, и координаты этой точки не совпадают с координатами контрольных точек (т.е. наткнулись на пересекающийся путь) то если число больше текущего значения счетчика (т.е. предыдущий обход длиннее) то заменяем на текущее, если меньше (т.е. другой обход короче) то оставляем без изменения, прекращаем обход и возвращаемся к другим направлениям.

5. Для каждой контрольной точки во временном массиве исследуются соседние точки (справа, слева, сверху, снизу).

Если среди соседних ячеек найдено только одно положительное число то оно увеличивается на 1 и записывается во временный массив на место контрольной точки.

Если найдено несколько положительных цифр (т.е. существуют несколько подходов к точке) то среди них выбирается минимальное значение, увеличивается на 1 и записывается во временный массив на место контрольной точки.

6. Выбирается минимальное значение, стоящее на местах контрольных точек во временном массиве, и выводится значение для этой точки из основного массива с лабиринтом.

Существуют и другие способы раскраски.

### **Оформление задач по программированию.**

Каждая работа содержит требования к оформлению. За не выполнение этих требований максимальное количество первичных баллов может быть уменьшено по каждому из нижеследующих пунктов:

- за написание программ на языке отличном от C, C++ или Pascal: на 2 балла;
- за отсутствие операций с файлами входных и выходных данных или за отсутствие понятного пользователю интерфейса ввода-вывода исходных данных: на 1 балл;
- за отсутствие комментариев, облегчающих понимание кода программы:

<i>Задача</i>	<i>Количество штрафных баллов (10-11 класс)</i>
1	1*
2	2
3	3
4	4

\* в случае если в качестве решения задачи предлагается программный код.