Грацианова Татьяна Юрьевна
Кандидат физико-математических наук, преподаватель подготовительных курсов факультета Вычислительной математики и кибернетики МГУ им. М.В. Ломоносова.

Как подружить задачки (советы девятиклассникам)

Ты в 9 классе, и тебе предстоит сдавать экзамен по информатике. А там 20 задачек. Вроде и не очень сложные, но все такие разные. Первые 10 выучишь, а потом, пока остальные 10 учишь, эти первые забудешь. Как быть? А надо, чтобы они перестали для тебя быть такими разными, надо найти способ их объединить, какие-то «подружить» между собой. Это возможно, ведь все задачки по одному предмету.

Возьмём задачку 16 – одну из самых сложных из первой части. Вот как она формулируется в демо-версии 2017 года.

Автомат получает на вход трёхзначное десятичное число. По полученному числу строится новое десятичное число по следующим правилам.

1. Вычисляются два числа – сумма старшего и среднего разрядов, а также сумма среднего и младшего разрядов заданного числа.

2. Полученные два числа записываются друг за другом в порядке невозрастания (без разделителей).

Далее на примере показывается, как по этому алгоритму из числа 277 получается 149.

Подумаем, с кем, а точнее, с чем, можно «подружить» эту задачку. Это алгоритм, а на запись алгоритмов у нас есть задачки с алгоритмическим языком. В них надо сказать, как работает программа, посчитать, что она напечатает, или написать свою программу на алгоритмическом языке.

Вот и попробуем написать свою программу для этого алгоритма на языке Паскаль.

Всё ли мы здесь умеем делать? Посмотрим на текст теперь с этой точки зрения.

«Автомат получает на вход трёхзначное число» – мы умеем вводить числа (оператор ввода Read или Readln). Мы даже можем проверить, является ли оно трёхзначным (трёхзначное число должно быть больше 99 и меньше 1000). Если программе

по ошибке введут не трёхзначное число, она ничего считать не должна. А что программа должна делать с правильным числом? Читаем дальше: надо вычислять суммы разрядов. А как выделить цифры в трёхзначном числе? Это легко сделать с помощью операций целочисленного деления. В Паскале они обозначаются DIV – собственно деление и MOD – получение остатка от деления. При выполнении этих операций с делителем 10 получаем: результат операции MOD – последняя цифра числа, результат операции DIV – число без последней цифры (у нас оно будет двузначным). Например, для числа 123 получим: 123 mod 10 = 3; 123 div 10=12. Если из получившегося двузначного числа с помощью той же операции MOD выделить последнюю цифру, получим среднюю цифру числа (количество десятков). Количество сотен можно получить из этого же числа, применив операцию DIV 10, а можно из первоначального, трёхзначного, с помощью операции DIV 100.

Запишем теперь всё вместе. Только нужно решить, как нам обозначить величины, с которыми будем работать, какие названия придумать переменным. Это довольно важный вопрос: переменных много, с ними надо будет делать разные действия, расставлять в нужном порядке, важно не запутаться.

Назовём первоначальное число буквой C – почему бы и нет. А вот названия переменных для цифр возьмём из условия задачи – там говорится о старшем, среднем и младшем разрядах. Получим имена: ST, SR, ML.

```
ST:= C div 100;
ML:= C mod 10;
SR:= C div 10 mod 10.
```

Осталось аккуратно, ничего не перепутав, сложить заданные числа. Результаты назовём N1 и N2 (потому что это новые числа). Дальше требуется записать эти числа в порядке невозрастания (то есть сначала большее число, затем меньшее) – воспользуемся условным оператором IF. Если первое число больше второго, печатаем сначала его, а затем второе, в противном случае (числа равны или второе больше) – печатаем сначала второе, а затем первое. Когда мы напечатаем числа в одной строчке, на экране они будут выглядеть как одно число (мы ведь не ставим разделитель, как это и велено в задаче).

Получается вот такая программа:

```
Program Task16;
Var C, ST, SR, ML, N1, N2: Integer;
Begin
   Readln(C);
   If  (C>99) AND (C<1000) Then
               Begin   ST:= C div 100;
               ML:= C mod 10;
               SR:= C div 10 mod 10;
               N1:=ST+SR;
               N2:=SR+ML;
              If N1>N2 Then Writeln(N1, N2)
                                      Else Writeln(N2, N1)
                                End
                                      Else Writeln('Ошибка')
End.
```

Нажимаем кнопочку «сохранить файл» и запускаем программу. Если что-то не так, исправляем ошибки (надо быть аккуратнее с точками с запятой, скобочками) и запускаем снова. Проверим программу для числа, приведённого в задании. Вводим 277, получаем, как и надо, 149. Достаточно ли одной проверки? Конечно, нет. Надо взять ещё несколько чисел, посчитать «вручную» результат для них, а потом сверить с тем, который получится у компьютера. Причём для более полной проверки надо взять такие числа, чтобы у одного была больше первая поразрядная сумма, а у другого – вторая. Это очень просто, например, 103 и 301. Для этих чисел и считать-то нечего, для обоих ответ – 31.

Задача решена? Но мы же не дочитали условие (кстати, очень частая ошибка на экзаменах): что там за числа внизу? Эти числа надо подать на вход программе? Но многие из них неправильные, четырёхзначные, наша программа не будет с ними работать. Читаем внимательнее; вот что написано: «Определите, сколько из приведённых ниже чисел могут получиться в результате работы автомата. 1616 169 163 1916 1619 316 916 116. В ответе запишите только количество чисел».

Эти числа не надо подавать программе на вход. Это те числа, которые могут (или не могут – это и требуется узнать) получиться на выходе.

Что же получается, зря программу писали? Нет, не зря, с её помощью можно проверять наши предположения. Задачка сложная, чтобы не ошибиться, надо не просто для каждого из предложенных чисел сказать «да» или «нет», но для неправильных чисел указать причину «неправильности», а для правильных – подобрать исходное число. Вот такие проверки – правда ли, что из нами придуманного числа получается заданное – и будет делать эта программа.

А для решения задачи можно написать другую программу. Она будет чуть побольше, в ней надо проверить довольно много условий.

Итак, рассмотрим эти условия.

1. Посмотрим диапазон допустимых чисел. Может получиться однозначное число? Нет, никак, мы ведь выписываем две суммы разрядов ненулевого числа. А может получиться число 10? Да, из 100 (вот и пригодилась программа, это предположение с её помощью можно проверить). Итак, минимальное допустимое число – 10. Какое максимальное? Максимальное значение суммы двух цифр – 18, значит, максимальное число 1818. Наша первая проверка – не выходят ли числа за границы допустимого диапазона.

Если число условию 1 удовлетворяет, проверяем его дальше. Удобно отдельно рассматривать двузначные, трёхзначные и четырёхзначные числа.

2. Работа с двузначными числами. Выделим из числа цифры. В правильном числе первая цифра должна быть больше или равна второй.

3. Трёхзначное число. Как его разбить на два числа? Вообще это можно сделать двумя способами, но в нашем случае, когда первое число должно быть больше второго, вариант только один: первое – двухзначное, а второе – однозначное, разбиваем с помощью обычных операций div и mod. Проверку на правильность следования чисел здесь делать не надо, мы этого добились правильным разбиением. Надо проверить, могут ли получиться такие суммы, ведь мы складываем цифры, а они от 0 до 9. Значит, максимальная сумма двух цифр не может превышать 18, если первое число больше 18 – ошибка. И ещё одно

условие, пожалуй, самое сложное. Вспомним нашу первую программу N1:=ST+SR; N2:=SR+ML.

Найдём разность этих чисел N1--N2=ST-ML. То есть разность этих чисел равна разности двух цифр. А такая разность не может быть больше 9.

4. Четырёхзначное число. Такое число надо разбить на два двузначных. Это делается операциями div и mod, только делителем будет не 10, а 100. Все возможные проверки мы уже обсудили выше: оба числа должны быть не больше 18 (проверять можно только первое, второе должно быть меньше его); первое должно быть меньше второго, и их разность не должна превышать 9.

В программе надо аккуратно перечислить все условия. Можно вначале проверить разрядность введённого числа, в зависимости от этого выделить его части, а потом работать с этими частями. Мы здесь поступили по-другому: по отдельности рассматриваем двузначное, трёхзначное и четырёхзначное числа, как это сделано при описании алгоритма. Такая программа объёмнее, но вычислений в ней столько же (ведь выполняется только одна «веточка»).

Пишем программу, аккуратно для каждого Then записываем Else, не забываем ставить операторные скобки Begin – End.

А что будет выдавать наша программа? Просто «да» или «нет»? Сделаем интереснее. Для правильных чисел, конечно же, пусть выдаёт «YES», а для неправильных пусть печатает «код ошибки»: в каждом месте, где обнаруживается, что число неправильное, будем выводить букву «N» и разные цифры – так мы сможем понять, почему число «забраковано».

```
Program Task16_2;
Var C,   N1, N2 : Integer;
Begin
   Error:=0;
   Readln(C);
   If C<10 Then
              Writeln('N1')
    Else
       If C>1818 Then
              Writeln('N2')
         Else If C<100 Then
                Begin N1:=C div 10; N2:=C mod 10;
                      If N1<N2 Then Writeln('N3')
                                Else Writeln('YES')
                End
            Else If C<1000 Then {трехзначное число}
              Begin N1:=C div 10; N2:=C mod 10;
                    If N1>18 Then Writeln('N4')
                             Else If N1-N2>9 Then Writeln('N5')
                                             Else Writeln('YES')
              End
                Else {четырехзначные числа}
                  Begin N1:=C div 100; N2:=C mod 100;
                        If N1<N2 Then Writeln('N6')
                          Else If N1>18 Then Writeln('N7')
                           Else If N1-N2>9 Then Writeln('N8')
                                           Else Writeln('YES')
                  End;
End.
```

Осталось проверить все числа. Если ответ «YES», подбираем исходное число, пропускаем его через первую программу, проверяем, получилось ли число из задания. Если играть с цифрами понравилось, можно дополнить вторую программу, чтобы она и числа сама подбирала (хотя бы один вариант). В случае ответа «N» смотрим код ошибки, ещё раз проверяем себя и программу.

Что мы получили? Мы не просто решили задачу. Мы стали обладателем программы-генератора задания для экзамена. Теперь можно брать любые числа, проверять, «хорошие» они или «плохие», и за несколько секунд составлять множество вариантов. Причём уже с ответами!!!

Конечно, для другой формулировки задачи 16 придётся составить другие программы, но это не займёт много времени, они все похожи. Только внимательно читайте условие и тщательно проверяйте!

Калейдоскоп Калейдоскоп Калейдоскоп

TIM: робот-инспектор Большого андронного коллайдера

Большой адронный коллайдер — это огромное подземное сооружение, обслуживать которое необходимо с особой тщательностью. Для этого ЦЕРН создал TIM'а, маленького монорельсового робота-инспектора, следящего за состоянием туннеля БАК. Большой адронный коллайдер ЦЕРН является крупнейшим в мире ускорителем элементарных частиц и может похвастаться тем, что длина его подземного туннеля составляет 27,3 км. Разумеется, такое внушительное сооружение требует постоянного ухода и проведения диагностики всех систем, не говоря уже о банальной уборке. Именно для этих целей ЦЕРН и создал TIM — робота-инспектора, чья задача состоит в проведении мониторинга обширной туннельной системы в режиме реального времени. TIM (Train Inspection Monorail), как нетрудно догадаться из его названия, осуществляет инспекцию во время того, как объезжает весь БАК по монорельсовой дороге, проложенной по потолку. Первоначально этот монорельсовый путь был установлен, когда туннель БАК был еще туннелем Большого электрон-позитивного коллайдера (LEP), который занимал это подземное пространство 11 лет (с 1989 по 2000 годы) и был демонтирован в 2001 году после завершения серии экспериментов. TIM проезжает по туннелю со скоростью 6 км/ч, используя для сканирования набор инструментов для проверки целостности покрытия, измерения температуры и процентного содержания кислорода в туннеле. Кроме того, он может отслеживать уровень излучения и, в случае необходимости, предоставить оператору визуальный и инфракрасный снимок территории. А если и этого недостаточно, то робот может возить с собой множество небольших вагончиков, оснащенных оборудованием для выполнения тех задач, которые самому роботу не под силу.

На самом деле, в настоящее время существует не один, а целых два маленьких TIM'а, готовых по команде из центра выехать на задание.