Златопольский Дмитрий Михайлович
Кандидат технических наук, доцент кафедры информатики и прикладной математики Московского городского педагогического университета.

Считаем «счастливые» билеты, или Оптимизация программ

В статье на примере программы решения задачи подсчёта количества так называемых «счастливых билетов» показываются возможности оптимизации программ. При этом предлагаются оригинальные методы, приводящие к уменьшению времени работы программы.

Как известно, шестизначный номер автобусного или подобного билета называют «счастливым», если сумма трех его первых цифр равна сумме трех последних. Например, счастливым считается билет с номером 627294. Интересно, а сколько существует таких билетов? Понятно, что для ответа целесообразно разработать компьютерную программу.

Самый очевидный алгоритм решения данной задачи аналогичен алгоритму действия человека, определяющего, является ли тот или иной номер билета «счастливым»:

- определяется каждая цифра номера;
- рассчитывается сумма трех первых цифр;
- рассчитывается сумма трех последних цифр;
- рассчитанные суммы сравниваются между собой; если они равны, то данный номер – «счастливый» и количество найденных «счастливых» номеров увеличивается на 1.

Разработаем соответствующую программу с использованием школьного алгоритмического языка. Его русский синтаксис делает программу максимально понятной и легко переносимой на любой другой язык программирования.

В программе применим следующие величины:

- *номер* – проверяемый номер;
- *пер*, *вт*, *тр*, *чет*, *пят*, *шес* – соответственно первая, вторая, ..., шестая цифры номера;
- *кол* – искомое количество.

Прежде чем представлять программу, заметим, что для выделения цифр следует использовать функции

mod и *div*, возвращающие, соответственно, остаток от деления одного целого числа на другое и их целочисленное частное (в других языках программирования используются не функции, а специальные операции).

Примем также, что номера могут иметь начальные нули, при этом номера 000001..000999 можно не рассматривать, а номер билета 000000 будем считать невозможным.

Итак, программа:

```
алг Число_счастливых_номеров_Вариант_1
нач цел номер, кол, пер, вт, тр, чет, пят, шес
  кол := 0
  |Рассматриваем все номера
  нц для номер от 1000 до 999999
    |Выделяем каждую цифру номера
    шес := mod(номер, 10)
    пят := mod(div(номер, 10), 10)
    чет := mod(div(номер, 100), 10)
    тр := mod(div(номер, 1000), 10)
    вт := mod(div(номер, 10000), 10)
    пер := div(номер, 100000)
    |Сравниваем суммы троек цифр
    если пер + вт + тр = чет + пят + шес
      то |Встретилось очередное «счастливое» число
        кол := кол + 1
    все
  кц
  |Выводим ответ
  вывод нс, кол
кон
```

Обратим внимание на то, что, хотя часть проверяемых номеров – менее чем 6-значные, для них также рассчитываются 6 цифр («недостающие» цифры будут равны нулю).

Подумаем над тем, как можно сократить размер программы. Анализ показывает, что в ней для каждого номера дважды определяется сумма трёх цифр. Это наводит на мысль, что можно создать вспомогательную функцию, рассчитывающую сумму цифр трёхзначного числа. Эта функция имеет вид:

```
алг цел СуммаЦифр(арг цел число)
нач цел пер, вт, тр
  пер := div(число, 100)
  вт := mod(div(число, 10), 10)
  тр := mod(число, 10)
  знач := пер + вт + тр |Значение функции
кон
```

С её использованием основная часть программы оформляется достаточно компактно:

```
алг Число_счастливых_номеров_Вариант_2
нач цел номер, полов1, полов2, кол
  кол := 0
  нц для номер от 1000 до 999999
    полов1 := div(n, 1000)
    полов2 := mod(n, 1000)
    если СуммаЦифр(полов1) = СуммаЦифр(полов2)
      то
        кол := кол + 1
    все
  кц
  вывод нс, кол
кон
```

где *полов*1 и *полов*2 – соответственно первая и вторая половины шестизначного (в общем случае) числа.

Однако, как показывают расчёты, время выполнения программы при этом увеличивается примерно на 20%. Это связано с многократными обращениями (вызовами) к вспомогательной функции.

Идею разбиения номера на две части можно применить следующим образом. Можно рассмотреть все возможные значения первой половины шестизначного номера (от 1 до 999) и для каждого значения, найдя сумму его цифр, сравнить её с суммой цифр всех возможных значений второй половины. Соответствующий вариант программы имеет вид:

```
алг Число_счастливых_номеров_Вариант_3
нач цел кол, полов1, полов2, пер, вт, тр,
    сумма123, сумма456 |суммы троек цифр
  кол := 0
  нц для полов1 от 1 до 999
    пер := div(полов1, 100)
    вт := mod(div(полов1, 10), 10)
    тр := mod(полов1, 10)
    сумма123 := пер + вт + тр
    нц для полов2 от 1 до 999
      пер := div(полов2, 100)
      вт := mod(div(полов2, 10), 10)
      тр := mod(полов2, 10)
      сумма456 := пер + вт + тр
      если сумма123 = сумма456
        то
          кол := кол + 1
      все
    кц
  кц
  вывод нс, кол
кон
```

Расчёты показывают, что время выполнения программы по сравнению с первым вариантом уменьшается примерно на 30%!

Разовьём идею разбиения числа на части и разработаем вариант программы, в котором рассматриваются отдельные цифры номера и все их возможные сочетания, то есть используем в программе 6 операторов цикла с параметром:

```
алг Число_счастливых_номеров_Вариант_4
нач цел кол, пер, вт, тр, чет, пят, шес
  кол := 0
  нц для пер от 0 до 9
```

```
    нц для вт от 0 до 9
      нц для тр от 0 до 9[1]
        нц для чет от 0 до 9
          нц для пят от 0 до 9
            нц для шес от 0 до 9
              если пер + вт + тр = чет + пят + шес
                то
                  кол := кол + 1
              все
            кц
          кц
        кц
      кц
    кц
    вывод нс, кол
кон
```

Оказывается, в этом случае время выполнения программы по сравнению с первым вариантом уменьшается более чем на 55%!

Казалось бы, все резервы ускорения программы исчерпаны. Однако это не так. Направления дальнейшей оптимизации автору предложила Елена Александровна Мирончик, учитель информатики лицея № 111 г. Новокузнецка Кемеровской обл., которую с полным правом можно считать соавтором данной статьи.

Одно из направлений основано на следующих рассуждениях. Если мы знаем первые пять цифр проверяемого номера, например, 16712, 16774 или 10145, то можем определить шестую из условия «счастливости» номера. Для приведённых примеров соответствующие значения равны:

$1 + 6 + 7 - 1 - 2 = 11$

$1 + 6 + 7 - 7 - 4 = 3$

$1 + 0 + 1 - 4 - 5 = -7$

Ясно, что подходящими будут не все рассчитанные значения шестой цифры, а только равные 0...9. Это означает, что в последнем варианте программы вместо шестого («самого внутреннего») оператора цикла можно записать два оператора:

[1] При указанных границах параметров в число «счастливых» номеров попадёт и номер 000000, который мы условились считать несуществующим.

```
алг Число_счастливых_номеров_Вариант_5
нач цел кол, пер, вт, тр, чет, пят, шес
  кол := 0
  …
  нц для пят от 0 до 9
    |Определяем требуемую шестую цифру
    шес := пер + вт + тр - (чет + пят)
    |Проверяем, подходит ли она
    если шес >= 0 и шес <= 9
      то     |Подходит
        кол := кол + 1
    все
  кц
…
```

Исключение одного оператора цикла уменьшает, как показывают расчёты, продолжительность выполнения программы ещё на 2,0 – 2,5%.

Ещё одно направление оптимизации заключается в ограничении возможных значений четвёртой и пятой цифр. Например, если номер начинается с 121, то четвёртая цифра может быть 0, 1, 2, 3 или 4. Это значит, что можно записать следующее условие:

```
если сумма123 < 10
  то
    макс_чет := сумма1
  иначе
    макс_чет := 9
все
```

где *сумма*123 – сумма трёх первых цифр номера, *макс_чет* – максимально возможное значение четвёртой цифры (его можно использовать в операторе цикла с параметром *чет*).

Аналогично для пятой цифры максимально возможное значение равно:

```
если сумма123 - чет < 10
  то
    макс_пят := сумма123 - чет
```

```
    иначе
      макс_пят := 9
  все
```

(В дальнейшем для лучшего понимания и краткости разность *сумма*123 – *чет* будем обозначать *сумма* 56.)

Можно также ограничить *минимально* возможные значения четвёртой и пятой цифр. Например, если номер начинается с 998, то четвёртая цифра может быть 9 или 8. Анализ таблицы:

Сумма трех первых цифр	Возможные значения четвёртой цифры
27	9
26	8 9
...	
19	1 2 3 4 5 6 7 8 9
18	0 1 2 3 4 5 6 7 8 9
17	0 1 2 3 4 5 6 7 8 9

показывает, что минимально возможное значение четвёртой цифры может быть определено следующим образом:

```
если сумма123 > 18
  то
    мин_чет := сумма123 - 18
  иначе
    мин_чет := 0
все
```

Аналогично для пятой цифры можно установить:

```
если сумма56 > 9   |сумм123 - чет > 9
  то
    мин_пят := сумма56 - 9
  иначе
    мин_пят := 0
все
```

А теперь – внимание! Сейчас обсудим важное и неочевидное утверждение. Получив значение величины *сумма*56, мы знаем сумму, которую должны составлять пятая и шестая цифры, чтобы номер оказался «счастливым». При этом количество возможных значений пятой цифры также из-

вестно (оно равно *макс_пят* – *мин_пят* + 1). Именно столько будет и «счастливых» номеров при известных значениях величин *мин_пят* и *макс_пят*.

Это значит, что можно отказаться от перебора всех возможных значений пятой цифры, расчёта шестой и проверки всех цифр номера на его «счастливость», а просто после расчёта двух последних величин добавить значение *макс_пят* – *мин_пят* + 1 к предыдущему значению искомой величины *кол*.

Всё сказанное выше позволяет сформировать вариант программы, решающей обсуждаемую задачу за минимальное время:

```
алг Число_счастливых_номеров_Вариант_6
нач цел кол, пер, вт, тр, чет, сумма123, сумма56,
        мин_чет, мин_пят, макс_чет, макс_пят
  кол := 0
  нц для пер от 0 до 9
    нц для вт от 0 до 9
      нц для тр от 0 до 9
        сумма123 := пер + вт + тр
        |Уточняем границы значений 4-й цифры
         если сумма123 < 10
           то
             макс_чет := сумма123
           иначе
             макс_чет := 9
         все
         если сумма123 > 18
           то
             мин_чет := сумма123 - 18
           иначе
             мин_чет := 0
         все
         нц для чет от мин_чет до макс_чет
           |Уточняем границы значений 5-й цифры
           сумма56 := сумма123 - чет
```

```
                если сумма56 < 10
                  то
                    макс_пят := сумма56
                  иначе
                    макс_пят := 9
                все
                если сумма56 > 9
                  то
                    мин_пят := сумма56 - 9
                  иначе
                    мин_пят := 0
                все
               |Учитываем все новые
               «счастливые» номера (см. выше)
                кол := кол + макс_пят - мин_пят +  1
            нц
          нц
        нц
      нц
      вывод нс, кол
кон
```

Итак, в результате оптимизации программы мы получили вариант, решающий задачу подсчёта количества «счастливых» билетов за время, примерно в 4 раза меньшее, чем первый вариант.

В заключение заметим следующее. Конечно, может возникнуть вопрос: «А какая разница, сколько времени будет выполняться программа – 10 сек или 3 сек, главное, чтобы результат был?» Во-первых, существуют задачи, в которых время выполнения программы является важным или даже определяющим фактором. Во-вторых, на олимпиадах по программированию часто ограничивают время решения задачи. В любом случае, как писал великий немецкий математик Карл Гаусс: «Достоинство науки требует, чтобы всемерно совершенствовались средства, ведущие к достижению цели»...

Задания для самостоятельной работы

1. Оказывается (и интуитивно понятно), что на числовой оси на отрезке от 0 до 999999 «счастливые» шестизначные номера расположены симметрично:

000000 и 999999
001001 и 998998
001010 и 998989
001100 и 998899
...
499976 и 500023
499985 и 500014
499994 и 500005

Это означает, что в приведённом в статье варианте 1 программы можно рассматривать только номера от 0 до 499999, то есть в 2 раза меньше, но при этом найденное значение *кол* должно быть удвоено. Разработайте соответствующий вариант программы.

2. Разработав любой из приведённых вариантов программы на языке программирования, которым вы владеете, определите количество «счастливых» шестизначных номеров.

3. Сравните время выполнения различных вариантов программ в системах программирования, которые вы применяете (автор все расчеты проводил в системе программирования PascalABC.net). Для этого используйте процедуры и функции, возвращающие текущее время, предусмотренные в системе программирования. Так как на продолжительность работы программы, кроме её особенностей, влияют и другие факторы, для каждого варианта проведите серию расчётов и используйте минимальное значение в серии.

4. Разработав соответствующие программы, определите количество «счастливых» четырёх- и восьмизначных номеров.

Мудрые мысли Мудрые мысли Мудрые мысли

Когда бы я захотел читать, ещё не зная букв, это было бы бессмыслицей. Точно так же, если бы я захотел судить о явлениях природы, не имея никакого представления о началах вещей, это было бы такой же бессмыслицей.

М.В. Ломоносов

Изучайте азы науки, прежде чем пытаться взойти на её вершины. Никогда не беритесь за последующее, не окончив предыдущего, никогда не пытайтесь прикрыть недостатки ваших знаний хотя бы и самыми смелыми догадками и гипотезами.

И.П. Павлов

Знание, пассивно усвоенное памятью, без умения применять его на практике – это ещё совершенно мёртвый балласт в наших плаваниях по житейскому морю.

С.Г. Струмилин

Умственные способности растут, как мускулы, при тренировке.

В.А. Обручев