Информатика

Ткаченко Кирилл Станиславович
Инженер 1-й категории
Федерального государственного автономного образовательного учреждения высшего образования «Севастопольского государственного университета».

Программная реализация решения головоломки «Господин S и господин P» на школьном алгоритмическом языке

Рассматривается программная реализация решения известной головоломки «Господин S и господин P» на школьном алгоритмическом языке. Эта реализация демонстрирует использование конструкций языка и обработку одномерных массивов.

Программирование головоломок – это часто непростая задача, которая может быть полезна не только школьникам, но и специалистам разного возраста и опыта. Для начинающих оно может предопределить выбор ими своей будущей деятельности. Поэтому существуют учебники, которые ориентированы на разработку и программирование решений таких задач [1].В свою очередь решение головоломок может «упираться» не только в непосредственное программирование, но и в логическую часть.

Известная головоломка «Господин S и господин P» в первую очередь сложна формулировкой задания для последующего программирования, причём само программирование её решения достаточно простое. Эта головоломка и процесс нахождения её решений уже рассмотрены и изучены [2, 3]. На определенных этапах получения решения требуется применение списка простых чисел, который можно получить, используя решето Эратосфена [4, 5]. Стоит отметить, что, в частности, имеется реализация нахождения решения головоломки на языке 1С [6], здесь она осуществлена на школьном алгоритмическом языке.

Условие головоломки следующее. Определённым способом были выбраны два целых положительных числа, оба из них, как и их сумма, не

меньше 2 и не больше 99. Исключительно господину S была сообщена сумма этих чисел, исключительно господину P – их произведение. Между господами S и P произошёл диалог. Господин S утверждает, что господин P не определил эти два числа. После этого господин P утверждает, что теперь знает их. А затем господин S понимает, что они ему также теперь известны.

По этому диалогу была сформулирована система условий [1–3]. Для определения целых положительных чисел А и Б, необходимо сгенерировать пары целых положительных чисел, для которых выполняются три условия:

I. Величина суммы Д+Е не может быть представлена суммой двух простых чисел;

II. Величина произведения В · Г может быть представлена суммой двух целых положительных чисел Д+Е, которая удовлетворяет условию I, и это представление единственно;

III. Величина суммы А+Б может быть представлена как произведение двух целых положительных чисел В · Г, удовлетворяющих условию II, и это представление единственно.

Программа будет генерировать пары целых положительных чисел и проверять для них условия в порядке III, II, I.

До алгоритмов программы определяются константа ВЕРХ_ГРАН, которая не меньше искомых целых положительных чисел и их суммы, и одномерный булев массив Решето, в котором отмечается значением «да», является ли число по индексу в массиве простым, и «нет», – в противном случае:

```
цел ВЕРХ_ГРАН = 99;
лог таб Решето[2 : ВЕРХ_ГРАН];
```

Программа начинает выполняться с алгоритма Потенциал, который производит инициализацию и поиск решения:

```
алг Потенциал
нач
  Инициализация
  Решение
кон
```

Алгоритм Инициализация выполняет построение решета Эратосфена, чтобы иметь возможность проверки, является ли число простым. Вначале программы определяются счётчики циклов А и Б:

```
алг Инициализация
нач
  цел А, Б;
```

Все целые положительные числа от 2 до ВЕРХ_ГРАН полагаются простыми:

```
  нц для А от 2 до ВЕРХ_ГРАН
    Решето[А] := да;
  кц
```

Для всех целых положительных чисел от 2 до ВЕРХ_ГРАН, если рассматриваемое число простое, то все кратные ему считаются составными:

```
  нц для А от 2 до ВЕРХ_ГРАН
    если Решето[А]
      то
        Б := А + А;
```

Пока мы не вышли за пределы доступных для головоломки чисел, кратные рассматриваемому числу отмечаются как составные:

```
        нц пока Б <= ВЕРХ_ГРАН
          Решето[Б] := нет;
          Б := Б + А;
        кц
    все
  кц
кон
```

Алгоритм Простое (цел А) воз-

вращает «да», если аргумент А принадлежит отрезку от А до ВЕРХ_ГРАН и при этом является простым, и «нет» – в противном случае:

```
алг лог Простое(цел А)
нач
  знач := (2 <= А) и
 (А <= ВЕРХ_ГРАН) и Решето[А];
кон
```

Алгоритм Условие I (цел Сум) возвращает «да», если аргумент Сум удовлетворяет условию I, и «нет» – в противном случае. Переменная А служит счётчиком цикла:

```
алг лог УсловиеI(цел Сум)
нач
  цел А;
```

Полагается, что результат «да»:

```
  знач := да;
```

Для всех целых положительных чисел, первое из которых не меньше второго и сумма которых равна Сум, если оба этих числа являются простыми, то условие I не выполняется и результат «нет»:

```
  нц для А от 2 до div(Сум, 2)
    если Простое(А) и
    Простое(Сум - А)
      то
        знач := нет;
    все
  кц
кон
```

Алгоритм Условие II (цел Произв) возвращает «да», если произведение Произв удовлетворяет условию II, и «нет» – иначе. Переменные А и Б – множимое и множитель, Количество – сколько раз удалось выполнить условие I для множимого и множителя:

```
алг лог Условие II (цел Произв)
нач
  цел А, Б;
  цел Количество;
```

Вначале условие I не было выполнено:

```
  Количество := 0;
```

Для всех целых положительных множимых А, допустимых для произведения Произв, находится множитель Б:

```
  нц для А от 2 до int(sqrt
 (Произв))
    если mod(Произв, А) = 0
      то
        Б := div(Произв, А);
```

Если множимое и множитель удовлетворяют условию I, то количество инкрементируется:

```
   если (2 <= Б) и (Б <= ВЕРХ_
ГРАН) и УсловиеI(А + Б)
          то
            Количество :=
            Количество + 1;
        все
    все
  кц
```

Если условие I было выполнено один раз, то условие II выполняется, иначе – нет:

```
  знач := (Количество = 1);
кон
```

Алгоритм Условие III (цел Сум) возвращает слагаемое, которое для суммы Сум удовлетворяет условию III, и 0, если такое слагаемое не было найдено. В переменной А находится искомое слагаемое. Найденное А – величина последнего найденного слагаемого, Количество – количество раз нахождения слагаемого:

```
алг цел УсловиеIII (цел Сум)
нач
  цел А;
  цел НайденноеА;
  цел Количество;
```

Изначально слагаемое не найдено:

```
  НайденноеА := 0;
  Количество := 0;
```

Если величина суммы удовлетворяет условию I, то для всех допустимых слагаемых:

```
если УсловиеI(Сум)
  то
    нц для A от 2 до div(Сум, 2)
```

Если рассматриваемое слагаемое и его дополнение до суммы являются множителями, удовлетворяющими условию II, то первое слагаемое отмечается и увеличивается количество найденных слагаемых:

```
        если УсловиеII(A *
        (Сум - A))
          то
            НайденноеA := A;
            Количество :=
            Количество + 1;
        все
      кц
  все
```

Когда такое слагаемое нашли один раз, то оно удовлетворяет условию III, иначе – нет:

```
  если Количество = 1
    то
      знач := НайденноеA;
    иначе
      знач := 0;
  все
кон
```

Алгоритм Решение находит решение головоломки. В переменной Сум находится расматриваемая сумма, A – первое слагаемое:

```
алг Решение
нач
  цел Сум, A;
```

Для всех допустимых сумм проверяется соблюдение условия III и удовлетворяющее ему слагаемое:

```
  нц для Сум от 2 до ВЕРХ_ГРАН
    A := УсловиеIII(Сум);
```

Когда условие IIIсоблюдается, то выводятся оба слагаемых для этой суммы:

```
    если A > 0
      то
        вывод A, ", ", Сум - A, нс;
    все
  кц
кон
```

Полный исходный текст программы находится в приложении А, результаты работы – в Б.

Полученная программа находит единственное допустимое по тексту головоломки и условиям I, II, III решение. Эта реализация может стать полезной демонстрацией использования конструкций школьного алгоритмического языка и обработки на нем одномерных массивов.

Источники информации

1. *Арсак Ж.* Программирование игр и головоломок. – М.: Наука, 1990. 224 с.
2. Sum and Product Puzzle // Wikipedia. URL: https://en.wikipedia.org/wiki/Sum_and_Product_Puzzle (дата обращения: 22.10.2018).
3. Sum and Product Puzzle // Rosetta Code. URL: http://rosettacode.org/wiki/Sum_and_Product_Puzzle (дата обращения: 21.10.2018).
4. *Перминов О.Н.* Программирование на языке Паскаль. – М.: Радио и связь, 1988. 224 с.
5. Sieve of Eratosthenes // Rosetta Code. URL: http://rosettacode.org/wiki/Sieve_of_Eratosthenes (дата обращения: 07.11.2018).
6. *Ткаченко К.С.* Программная реализация решения головоломки «Господин S и господин P» по Арсаку в 1С // Системный администратор. Вып. 12 (193)/2018. –М.: ООО «Издательский дом Положевец и партнёры», 2018. С. 44–46.

Приложение А. Полный исходный текст программы

```
цел ВЕРХ_ГРАН = 99;
лог таб Решето[2 : ВЕРХ_ГРАН];

алг Потенциал
нач
  Инициализация
  Решение
кон

алг Инициализация
нач
  цел А, Б;

  нц для А от 2 до ВЕРХ_ГРАН
    Решето[А] := да;
  кц

  нц для А от 2 до ВЕРХ_ГРАН
    если Решето[А]
      то
        Б := А + А;

        нц пока Б <= ВЕРХ_ГРАН
          Решето[Б] := нет;
          Б := Б + А;
        кц
    все
  кц
кон

алг лог Простое(цел А)
нач
  знач := (2 <= А) и (А <= ВЕРХ_ГРАН) и Решето[А];
кон

алг лог УсловиеI(цел Сум)
нач
  цел А;

  знач := да;

  нц для А от 2 до div(Сум, 2)
    если Простое(А) и Простое(Сум - А)
      то
        знач := нет;
```

```
    все
  кц
кон

алг лог УсловиеII(цел Произв)
нач
  цел А, Б;
  цел Количество;

  Количество := 0;

  нц для А от 2 до int(sqrt(Произв))
    если mod(Произв, А) = 0
      то
        Б := div(Произв, А);

        если (2 <= Б) и (Б <= ВЕРХ_ГРАН) и УсловиеI(А + Б)
          то
            Количество := Количество + 1;
        все
    все
  кц

  знач := (Количество = 1);
кон

алг цел УсловиеIII(цел Сум)
нач
  цел А;
  цел НайденноеА;
  цел Количество;

  НайденноеА := 0;
  Количество := 0;

  если УсловиеI(Сум)
    то
      нц для А от 2 до div(Сум, 2)
        если УсловиеII(А * (Сум - А))
          то
            НайденноеА := А;
            Количество := Количество + 1;
        все
      кц
  все
  если Количество = 1
```

```
    то
      знач := НайденноеА;
    иначе
      знач := 0;
  все
кон

алг Решение
нач
  цел Сум, А;

  нц для Сум от 2 до ВЕРХ_ГРАН
    А := УсловиеIII(Сум);

    если А > 0
      то
        вывод А, ", ", Сум - А, нс;
    все
  кц
кон
```

Приложение Б. Результаты работы программы

```
4, 13
```

Калейдоскоп Калейдоскоп Калейдоскоп

Не вероятно … но факт!

К концу прошлого века сложилось твёрдое общественное мнение, что современная наука очень затратна, нуждается в большом количестве участников исследований, следовательно, требует не домашних, а лабораторных условий. Однако относительно недавно произошёл случай, который опроверг это мнение.

Английский биохимик Питер Митчел не только выполнил дома сложные экспериментальные наблюдения, но и получил за них Нобелевскую премию! Когда он окончил университет, ему по состоянию здоровья пришлось жить в сельской местности. Он по своему усмотрению и на собственные деньги приобрёл измерительные приборы и всё, что ему понадобилось для исследований. А интересовала его , как передаётся энергия в клетки организма. Проведя множество опытов, измерений и вычислений, он решил сложнейшую научную проблему: установил механизм получения клеткой энергии, за что и был удостоен Нобелевской премии. Не исключено, что и в наши дни «домашняя наука» может принести удачу – раскрыть научную загадку!