Литература

1. *Дорошенко Е.Г., Пак Н.И., Рукосуева Н.В., Хегай Л.Б.* О технологии разработки ментальных учебников // Вестник Томского государственного педагогического университета (Tomsk State Pedagogical University Bulletin). 2013. Вып. 12 (140). С. 145–151.
2. *Новак Д., Канас А.* Теория построения и практика применения карт понятий. URL: http://cmap.ihmc.us/Publications/ResearchPapers/TheoryCmaps/TheoryUnderl
3. *Шенк Ф.Б.* Ментальные карты: конструирование географического пространства в Европе / пер. с нем. А. Жоровой // Политическая наука. Политический дискурс: История и современные исследования. 2001. Вып. 4. С. 4–17.
4. *Шаталов В.Ф.* Эксперимент продолжается. М.: Педагогика, 1989. 334 с.
5. *Калитина В.В.* Электронная энциклопедия как средство повышения уровня запоминания учебного материала // Вестник КГПУ. 2013. № 1 (23). С. 111–114.
6. *Мюллер Х.* Составление ментальных карт: метод генерации и структурирования идей / пер. с нем. В.В. Мартыновой, М.М. Демина. М.: Омега-Л, 2007. 126 с.
7. *Пак Н.И.* Гипермозг как основа становления ментальной дидактики. Интернет – свободный, безопасный, образовательный // Межрегион. науч.-практ. конф. (18–19 октября, 2013 г., г. Омск): сб. матер. / под общ. ред. М.П. Лапчика. Омск: Полиграфический центр КАН, 2013. 278 с.
8. *Пак Н.И.* Информационное моделирование: учебное пособие. // КГПУ им. В.П. Астафьева. Красноярск, 2010. 152 с.
9. *Колесник В.* Ментальные карты. URL: http://kolesnik.ru/2005/mindmapping
10. *Петрова И.А., Ракова Е.П.* Использование структурированных графических схем в изучении информатики // Успехи современного естествознания. 2013. № 10. С. 35–36.
11. *Найссер У.* Познание и реальность. М.: Прогресс, 1981. 252 с.
12. *Бруннер Е.Ю.* Применение технологии mind map в учебном процессе // Развитие международного сотрудничества в области образования в контексте Болонского процесса: материалы международной науч.-практ. конф. г. Ялта (5–6 марта 2008 г.). Ялта: РИО КГУ, 2008. Вып. 19. Ч. 1. С. 50–53.
13. *Бабич А.В.* Эффективная обработка информации (Mind mapping). URL: http://www.intuit.ru/studies/courses/647/503/lecture/11414?page=8

Use in educational process mental map

***Damir Rashitovich Khakimov**, Master SibGTU, Siberian State Technological University,*

In this paper, the technique used in the training process of mental maps, the technology development which is based on the information model of thinking. Using this technique significantly affects the intensification of training and intensifying training activities due to higher than traditional teaching methods, the degree of visualization of the material presented.

Keywords: mental map, image, information model of thinking, structuring the learning process

УДК 004.421+004.438

МЕТОДИКА БЫСТРОГО ОБУЧЕНИЯ ПРОГРАММИРОВАНИЮ НА ОСНОВЕ ИЗУЧЕНИЯ КЛАССОВ ЗАДАЧ (11–15)

***Юрий Александрович Аляев**, доц., доц. кафедры программного обеспечения вычислительной техники и автоматизированных систем,*
e-mail: alyr1@yandex.ru,
Пермский военный институт внутренних войск МВД России,
http://pvivv.ru

Предлагается методика быстрого обучения программированию на основе изучения классов задач, разработанная и применяющаяся на практике в процессе обучения программированию студентов вузов.

Ключевые слова: алгоритм, программа, язык программирования Паскаль, массив, процедуры и функции, рекурсия, множество, запись

Введение

Разделы курса «Информатика» – алгоритмизация и программирование – остаются наиболее важными для формирования алгоритмического мышления. Поскольку в школах данные разделы преподаются в недостаточном объеме, в вузе возникает необходимость начинать обучение с нуля и достичь хорошего уровня программирования при ограниченном количестве часов преподавания.

Этого удается добиться за счет применения рациональных методов обучения, прежде всего, последовательно проводя идеи обучения на основе выделения элементарных операций деятельности по построению алгоритмов и программ; выявления структуры алгоритма и форм ее записи на алгоритмическом языке; одинаковой формы алгоритма для решения задач с одинаковой структурой исходных данных [1–2]. Благодаря этим идеям, задачи по программированию удается разбить на ряд классов и типизировать методы решения задач каждого класса.

Ю.А. Аляев

Предлагаемая методика быстрого обучения программированию на основе изучения классов задач, появилась и применяется на протяжении многих лет в процессе обучения программированию студентов пермских вузов благодаря В.П. Гладкову [2]. В статье рассматриваются методики решения по пяти (11–15) из девятнадцати выделенных классов задач (1–10 классы задач рассмотрены в [3–6]).

11. Данные типа string

Задача 1. Подсчитать, сколько раз в заданной строке встречается указанная буква.

Решение 1. Пусть исходная строка хранится в переменной s, а искомая буква в переменной a. Для решения задачи будем просматривать строку *s* посимвольно и каждый символ сравнивать с заданной буквой.

```
k:=0; {количество указанных букв в строке}
for i:=1 to length(s) do
    if copy(s,i,1)=a then k:=k+1;.
```

Решение 2. Будем искать положение указанной буквы в строке до тех пор, пока ее удастся найти. Затем отбрасываем ту часть строки, где была найдена указанная буква, и повторяем поиск.

```
k:=0; j:=pos(a,s); {позиция первого вхождения a в строку s}
while j<>0 do
begin k:=k+1;
   s:=copy(s,j+1,length(s)-j);
   j:=pos(a,s);
end.
```

Задача 2. Проверить, входят ли в строку s две буквы a.

Решение. Для двух букв a, стоящих подряд: if pos('aa',s)>0 then write('входят') else write('НЕ входят'). Здесь требуется проверить наличие двух букв a, стоящих в любом месте строки. Эта задача является поисковой. Если строка закончится и две буквы a не будут найдены, то ответ на вопрос задачи отрицательный. Если при поиске будут найдены две буквы a, то ответ на вопрос задачи положительный. Для подсчета найденных букв a использовать счетчик.

```
k:=0; {счетчик букв a}
f:=false; {пока не нашли две буквы a}
i:=1; {номер исследуемого символа строки}
while (i<=length(s)) and not f do
   if copy(s,i,1)='a'
   then begin k:=k+1;
```

```
                    if k=2 then f:=true;
                end
    else i:=i+1;
if f
then write('в строке есть две буквы а')
else write('в строке нет двух букв а');.
```

Другое решение этой задачи можно получить, основываясь на втором решении задачи 10.1.1.

```
j:=pos('а',s);
if j>0
then    begin s:=copy(s,j+1,length(s)-j);
          j:=pos('а',s);
          if j>0
          then write('в строке есть две буквы а')
          else write('в строке нет двух букв а');
  end
else write('в строке нет двух букв а');.
```

Задача 3. В строке s заменить символы а на символы я.

Решение 1. Просматриваем строку посимвольно, удаляем найденный символ а, вставляем на его место я.

```
for i:=1 to length(s) do
if copy(s,i,1)='а'
then    begin delete(s,i,1)
          insert('я',s,i);
  end;.
```

Решение 2. Просматриваем исходную строку посимвольно и переписываем в выходную строку символы, отличные от а. Вместо символа а переписываем символ я.

```
s1:=''; { выходная строка }
for i:=1 to length(s) do
   if copy(s,i,1)='а'
   then s1:=s1+'я'
   else s1:=s1+copy(s,i,1);.
```

Решение 3. Оно основывается на решении 2 задачи 10.1.1.

```
j:=pos('а',s);
while j<>0 do
begin s:=copy(s,1,j-1)+'я'+copy(s,j+1,length(s)-j);
   j:=pos('а',s);
end;.
```

Задача 4. Будем считать словом любую последовательность букв и цифр. Строка состоит из слов, разделенных одним или несколькими пробелами. Удалить лишние пробелы, оставив между словами по одному пробелу.

Решение. Лишними пробелами называются второй, третий и т.д., следующие за первым пробелом. Следовательно, чтобы найти лишний пробел, нужно искать два пробела, стоящие рядом, и удалять второй пробел в каждой найденной паре.

```
j:=pos(' ',s);
while j<>0 do
begin delete(s,j,1);
   j:=pos(' ',s);
end;.
```

Задача 5. Строка символов – это любая последовательность символов, заключенная в апострофы. Задана строка символов, состоящая из слов и строк,

разделенных одним или несколькими пробелами. Удалить из строки все незначащие пробелы. Незначащими пробелами называются пробелы, не стоящие в апострофах.

Решение. Запишем формально определение незначащего пробела. Текущий пробел незначащий, если предыдущий символ является пробелом и этот пробел не стоит в апострофах. Введем логическую переменную p, которая принимает значение false, если предыдущий символ не является пробелом, и значение true, если предыдущий символ – пробел. Введем логическую переменную q, которая принимает значение false, если пробел находится не в апострофах, и значение true, если пробел в апострофах. Тогда формальное определение незначащего пробела запишется так:

(copy(s,i,1)=’ ’) and p and not q.

Составляем программу:

```
p:=false;
q:=false;
s1:=’’;
for i:=1 to length(s) do
begin if not((copy(s,i,1)=’ ’) and p and not q)
   then s1:=s1+copy(s,i,1);
   if copy(s,i,1)=’ ’
   then p:=true else p:=false;
   if copy(s,i,1)=’’’’ then q:=not q;
end;.
```

Задача 6. Подсчитать количество гласных букв русского алфавита в строке.

Решение. Гласная буква – это такая буква, которая принадлежит множеству гласных букв. Для решения задачи просматриваем строку посимвольно и проверяем каждый символ на принадлежность гласным буквам:

```
k:=0; {количество гласных букв}
for i:=1 to length(s) do
if pos(copy(s,i,1),’аоуэыяёюеи’)>0
then k:=k+1;.
```

Задача 7. Задано предложение, состоящее из слов, разделенных одним или несколькими пробелами. Определить самое длинное слово предложения.

Решение. Для того чтобы выделить окончание слова, нужно анализировать два символа: первый символ должен быть отличен от пробела, а второй должен быть пробелом. Для одинаковой обработки всех символов добавим к концу предложения дополнительный символ – пробел. Как только обнаружится конец слова, вычислим его длину и проверим на максимум:

```
smax:=’’; {слово максимальной длины}
readln(s); {исходное предложение}
s:=s+’ ’; {исходное предложение с дополнительным пробелом}
ss:=’’; {текущее слово предложения}
for i:=1 to length(s)-1 do {просмотр предложения по два символа}
  if (copy(s,i,1)<>’ ’) and (copy(s,i+1,1)=’ ’)
  {если текущий символ не пробел, а следующий – пробел}
  then    begin ss:=ss+copy(s,i,1); {дописали последний символ}
                  if length(smax)<length(ss)
                  then smax:=ss; {если длина нового слова больше, чем длина smax,
то запоминаем его}
                  ss:=’’; {готовим место для следующего слова}
          end
  else if copy(s,i,1)<>’ ’ then ss:=ss+copy(s,i,1);
  {если текущий символ не пробел, то запоминаем его в слове}.
```

Задача 8. Задано предложение, состоящее из слов, разделенных одним или несколькими пробелами. Расположить слова предложения в алфавитном порядке.

Решение. Перепишем слова предложения по одному в элементы одномерного массива. Отсортируем массив по возрастанию и перепишем слова из массива в строку.

```
const nn=100; {максимальное количество слов в предложении}
type mas=array[1..nn]of string; {тип массива строк}
var       a:mas; {массив слов предложения}
  i,j, {индексы массивов, i – номер обрабатываемого символа}
  k:integer; {количество слов в предложении, индекс массива слов}
  s,  {исходное предложение и результат}
  r:string;{текущее слово предложения, переменная для обмена слов}
begin
  write('Введите строку ');
  readln(s);
  s:=s+' '; {добавили пробел в конце для однотипной обработки всех слов}
  k:=0; {количество слов в предложении}
  r:=''; {текущее слово}
  for i:=1 to length(s)-1 do {просмотр предложения по два символа}
          if (copy(s,i,1)<>' ') and (copy(s,i+1,1)=' ')
                  {если текущий символ не пробел, а следующий пробел}
          then    begin   r:=r+copy(s,i,1); {дописать символ к слову}
                          k:=k+1; {увеличить количество слов}
                          a[k]:=r; {записать слово в массив}
                          r:=''; {подготовить место для следующего слова}
                  end
          else if copy(s,i,1)<>' ' then r:=r+copy(s,i,1);
  {если текущий символ не пробел, то записать его в слово}
  {три следующих оператора сортируют массив}
  for i:=1 to k-1 do
  for j:=i+1 to k do
          if a[i]>a[j] then begin
                  r:=a[i];a[i]:=a[j];a[j]:=r; end;
  {сцепление слов из массива в новое предложение}
  s:='';
  for i:=1 to k do s:=s+a[i]+' ';
  write(s);
end.
```

12. Процедуры и функции

Задача 1. Написать программы для вычисления числа сочетаний из n по m, оформив вычисления факториала процедурой без параметров, процедурой с параметрами, функцией. Сравнить решения.

Решение 1. Воспользуемся известной формулой:

$$C_n^m = \frac{n!}{m!(n-m)!}.$$

```
{В решении используется процедура без параметров}
var n,m:longint; {исходные данные}
    c:longint;   {число сочетаний}
    fn,fm,fnm:longint; {переменные для хранения n!, m!, (n-m)!}
    p:longint; {глобальная переменная для хранения факториала}
    q:longint;  {глобальная переменная для числа, факториал которого
отыскивается}
```

```
procedure fact1;
var i:longint;
begin p:=1;
        for i:=1 to q do p:=p*i;
end;
begin {главная программа}
  write('Введите n и m ');
  readln(n,m);
  q:=n; fact1; fn:=p; {исходное данное занесли в q, вычислили факториал, результат сохранили в p}
  q:=m fact1; fm:=p;
  q:=n-m; fact1; fnm:=p;
  c:=fn div (fm*fnm);
  write('Число сочетаний из ',n,' по ',m,' равно ',c);
end.
```

Решение 2.

```
{В решении используется процедура с параметрами}
var n,m:longint; {исходные данные}
    c:longint;   {число сочетаний}
    fn,fm,fnm:longint; {переменные для хранения n!, m!, (n-m)!}
{p – параметр для хранения факториала, q : longint; {параметр для числа, факториал которого отыскивается}
procedure fact2(q:longint;var p:longint);
var i:longint;
begin p:=1;
        for i:=1 to q do p:=p*i;
end;
begin {главная программа}
  write('Введите n и m ');
  readln(n,m);
  fact2(n,fn);
  fact2(m,fm);
  fact2(n-m,fnm);
  c:=fn div (fm*fnm);
  write('Число сочетаний из ',n,' по ',m,' равно ',c);
end.
```

Решение 3.

```
{В решении используется функция}
var n,m:longint; {исходные данные}
    c:longint;   {число сочетаний}
function fact3(q:longint):longint;
var i:longint;
begin p:=1;
        for i:=1 to q do p:=p*i;
         fact3:=p;
end;
begin {главная программа}
  write('Введите n и m ');
  readln(n,m);
  write('Число сочетаний из ',n,' по ',m,' равно ',
  fact3(n) div (fact3(m)*fact3(n-m)));
end.
```

Задача 2. Даны два числа a и b. Разработать алгоритм и написать программу для определения наибольшего общего делителя (НОД) трех величин: a+b, |a–b|, a·b. Расчет НОД двух чисел оформить в виде пользовательской процедуры.

Математическая модель данной задачи имеет вид:

Ввод: a,b.

Расчет: x=a+b; y=|a–b|; z=a·b.

Для расчета наибольшего общего делителя используем следующее выражение: НОД(x,y,z)=НОД(НОД(x,y),z). Для расчета НОД двух чисел используем алгоритм Евклида.

Вывод: NOD.

Фрагмент алгоритма для расчета наибольшего общего делителя k двух чисел n и m имеет вид:

```
. . . . . .
Цикл-ПОКА (m≠n);
Если m>n То;
   m:=m–n;
Иначе
   n:=n–m;
Конец-Если;
Конец-Цикла;
k:=m;
. . . . . .
```

В Паскаль-программе пользовательская процедура размещается в разделе описания процедур и функций.

```
program z1;
var a,b,c:integer;
procedure evklid(m,n:integer; var k:integer);
begin
            while m<>n do
              if m>n then m:=m–n
                    else n:=n–m;
            k:=m;
             end;
begin
  read(a,b);
          writeln('a=',a,' b=',b);
          evklid(a+b,abs(a–b),c);
          evklid(c,a*b,c);
          writeln('нод=',c);
end.
```

В данной программе обращение к пользовательской процедуре осуществляется дважды: первый раз – для расчета НОД суммы и модуля разности чисел a и b, второй раз – для расчета НОД числа, полученного от первого обращения к процедуре и произведения чисел a и b.

Пользовательская процедура имеет три формальных параметра: параметры-значения – m и n, а также параметр-переменную – k. Формальному параметру k соответствует фактический параметр c, с помощью которого выводится на печать искомый результат.

Задача 3. Решить задачу 11.2.2, используя для нахождения НОД пользовательскую функцию. Программа на языке Паскаль для решения этой задачи имеет следующий вид:

```
program z2;
```

```
var a,b,c:integer;
function evklid(m,n:integer):integer;
begin
while m<>n do
  if m>n then m:=m–n
                else n:=n–m;
evklid:=m;
end;
begin
  read(a,b);
          writeln('a=',a,' b=',b);
          c:=evklid(evklid(a+b,abs(a–b)),a*b);
            writeln('нод=',c);
end.
```

Для вызова пользовательской функции применяется оператор присваивания, в котором в качестве операнда используется обращение к функции, содержащее имя функции и список фактических параметров. В соответствии с правилами приоритета первым выполняется обращение evklid(a+b,abs(a–b)). Для возвращения результата решения пользовательская функция должна содержать оператор присваивания, у которого в левой части должно стоять имя функции, а в правой – возвращаемый в основную программу результат.

13. Рекурсия

Задача 1. Известно рекурсивное определение факториала:

$$n = \begin{cases} 1, \text{если } n = 0 \text{ или } n = 1, \\ (n-1)!, \text{если } n > 1. \end{cases}$$

Здесь n – неотрицательно. Записать эту функцию на языке Паскаль.

Решение. В первой строке определения явно указано, как вычислить факториал, если аргумент равен нулю или единице. В любом другом случае для вычисления n! необходимо вычислить предыдущее значение (n-1)! и умножить его на n. Уменьшающееся значение гарантирует, что, в конце концов, возникнет необходимость найти 1! или 0!, которые вычисляются непосредственно.

```
program task5;
var n:integer; { исходное значение }

function fact(i:integer):integer;
begin if (i=1) or (i=0)
          then fact:=1
          else fact:=fact(i-1)*i;
end;

begin write('Введите нужное значение n ');
   readln(n);
   writeln('Факториал ',n,' равен ',fact(n));
end.
```

Вспомним, что на время выполнения вспомогательного алгоритма основной алгоритм приостанавливается. При вызове новой копии рекурсивного алгоритма вновь выделяется место для всех переменных, объявляемых в нем, причем переменные других копий будут недоступны. При удалении копии рекурсивного алгоритма из памяти удаляются и все его переменные. Активизируется предыдущая копия рекурсивного алгоритма, становятся доступными ее переменные. Пусть необходимо вычислить 4!. Основной алгоритм: вводится n=4, вызов fact(4). Основной алгоритм

приостанавливается, вызывается и работает fact(4): 4<>1 и 4<>0, поэтому fact:=fact(3)*4. Работа функции приостанавливается, вызывается и работает fact(3): 3<>1 и 3<>0, поэтому fact:=fact(2)*3. В данный момент в памяти компьютера две копии функции *fact*. Вызывается и работает fact(2): 2<>1 и 2<>0, поэтому fact:=fact(1)*2. В памяти компьютера уже три копии функции *fact* и вызывается четвертая. Вызывается и работает fact(1): 1=1, поэтому fact(1)=1. Работа этой функции завершена, продолжает работу fact(2). fact(2):=fact(1)*2=1*2=2. Работа этой функции также завершена, и продолжает работу функция fact(3). fact(3):=fact(2)*3=2*3=6. Завершается работа и этой функции, и продолжает работу функция fact(4). fact(4):=fact(3)*4=6*4=24. Сейчас управление передается в основную программу и печатается ответ: «Факториал 4 равен 24».

14. Тип данных множество

Задача 1. Задать множество целых чисел от заданного числа до числа в три раза большего, чем заданное.

Решение. Используем описание множеств на языке Паскаль и операторы для работы с множествами.

Если количество элементов *n* в множестве известно заранее, то задача решается так:

```
const n=50;
type setnum=set of byte;
const mn:setnum=[n..3*n];
{В тексте программы остается использовать созданное множество}.
```

Если начальное значение задается пользователем, то задача решается так:

```
type setnum=set of byte;
var mn:setnum;
   n,i:byte;
begin write('задайте первый элемент множества ');
  readln(n);
  if 3*n<256 then for i:=n to 3*n do mn:=mn+[i]
  else write('заданное количество элементов не поместится в множестве ');
end.
```

Задача 2. Вывести элементы множества, содержащего прописные и строчные буквы латинского алфавита, на экран.

Решение. В цикле проверим вхождение всех элементов базового типа и выводим те, которые входят в множество.

```
var zn:set of 'A'..'z';
   i:char;
begin for i:='A' to 'Z' do
        if i in zn then write(i,' ');
     for i:='a' to 'z' do
        if i in zn then write(i,' ');
end.
```

Задача 3. Написать программу, которая в заданном слове, состоящем из строчных букв, определяет составляющие его буквы, глухие и звонкие согласные, затем все согласные и все гласные буквы.

Решение.

```
type setchar = set of char;
const GL:setchar=['п','ф','к','т','ш','с','х','ц','ч','щ'];
     {множество глухих согласных}
ZV:setchar=['л','м','н','р','й','б','в','г','д','ж','з'];
     {множество звонких согласных}
```

```
  ALF:string='абвгдеёжзийклмнопрстуфхцчшщъыьэюя';
    {строка – русский алфавит}
var s:string; {заданное русское слово}
  i:integer; {номер обрабатываемого символа в слове}
  mgl:setchar; {множество глухих согласных}
  mzv:setchar; {множество звонких согласных}
  buk:setchar; {множество всех букв слова}
  sog:setchar; {множество согласных слова}
  gla:setchar; {множество гласных слова}
procedure print(s:string;mn:setchar);
{процедура вывода элементов множества с предшествующим комментарием}
var i:integer;
begin write(s,' ');
    for i:=1 to length(ALF) do
     if ALF[i] in mn then write(ALF[i],' ');
    writeln;
end;
begin write('Введите русское слово ');
    readln(s);
    mgl:=[]; mzv:=[]; buk:=[];
    {вначале множества букв пусты}
    for i:=1 to length(s) do
    begin buk:=buk+[s[i]]; {объединили букву с множеством букв}
        if s[i] in GL
        then mgl:=mgl+[s[i]]
        {объединили глухую согласную с множеством глухих согласных}
        else if s[i] in ZV
            then mzv:=mzv+[s[i]];
        {объединили звонкую согласную с множеством звонких согласных}
    end;
    {находим множество согласных букв}
    sog := mgl + mzv;
    {находим множество гласных букв}
    gla:=buk-(sog+['ь','ъ']);
    print('слово состоит из букв; ',buk);
    print('гласные буквы: ',gla);
    print('согласные буквы: ',sog);
    print('глухие согласные: ',mgl);
    print('звонкие согласные; ',mzv);
end.
```

Задача 4. Заданы два слова. Определить буквы, которые не являются общими для обоих слов.

Решение. Образуем множества, содержащие буквы первого и второго слова. Затем найдем разности первого и второго, второго и первого множеств. Их объединение даст ответ.

```
type setchar=set of char;
const ALF:string='абвгдеёжзийклмнопрстуфхцчшщъыьэюя';
    {строка – русский алфавит}
var s1,s2:string; {заданные слова}
  i:integer; {номер обрабатываемого символа в слове}
  ms1:setchar; {множество букв первого слова}
  ms2:setchar; {множество букв второго слова}
```

```
   g1,g2:setchar;
{множества букв, которые входят в первое слово, но не входят во второе,
 и входят во второе, но не входят в первое}
procedure print(mn:setchar);
{процедура вывода элементов множества}
var i:integer;
begin for i:=1 to length(ALF) do
      if ALF[i] in mn then write(ALF[i],' ');
     writeln;
end;
begin write('Введите два русских слова, разделив их нажатием клавиши Enter ');
     readln(s1);readln(s2);
     ms1:=[]; ms2:=[];
     {множество букв первого слова}
     for i:=1 to length(s1) do ms1:=ms1+[s1[i]];
     {множество букв второго слова}
     for i:=1 to length(s2) do ms2:=ms2+[s2[i]];
     g1:=ms1-ms2;
     g2:=ms2-ms1;
     print(g1+g2);
end.
```

Задача 5. Написать программу для нахождения простых чисел с помощью «решета Эратосфена».

Решение.

1. Поместим все числа между 2 и n (n<=255) в решето.
2. Выберем из решета наименьшее из чисел.
3. Поместим это число среди простых.
4. Переберем и вынем из решета все числа, кратные данному.
5. Если решето не пустое, то повторим шаги 2 – 5.

```
const n=255; {количество элементов в множестве}
var interval, {решето}
     prost:set of 2..n; {множество простых чисел}
     next:integer;      {наименьшее число в решете}
     c:integer;         {новое простое число}
     j:integer;         {переменная для удаления из решета чисел, кратных текущему
простому числу}
begin interval:=[2..n];
  prost:=[];
  next:=2;
  repeat {поиск очередного простого числа}
     while not (next in interval) do next:=succ(next);
     prost:=prost+[next];
     c:=2*next–1;
     j:=next;
     while j<=n do
     begin interval:=interval–[j]; j:=j=c; end;
  until interval=[];
end.
```

Задача 6. Натуральные числа вводятся с клавиатуры до тех пор, пока не будет введено число нуль (признак окончания ввода). Написать программу для определения цифры, которая встречается во всех введенных числах.

Решение. Для каждого введенного числа образуем множество его цифр и найдем его пересечение с множествами цифр других чисел. Для первого числа не существует множества цифр предыдущих чисел, поэтому в ответе следует записать все множество цифр первого числа. Эта ситуация контролируется в программе с помощью переменной – признака p.

```
var s,s1:set of byte; {множества всех введенных чисел и цифр очередного числа}
   a, {очередное число}
   p:integer; {равно 1, если число первое}
begin s:=[];p:=1;
      read(a);
      while a<> 0 do {пока есть числа}
       begin s1:=[];
            {определяем множество цифр очередного числа}
            while a>0 do
             begin s1:=s1+[a mod 10];
                   a:=a div 10;
             end;
             {пересечение множеств}
             if p=1 then begin s:=s1;p:=0; end
             else s:=s*s1;
              read(a);
       end;
       for a:=0 to 9 do if a in s then write(a,' ');
end.
```

15. Тип данных запись

Задача 1. Создайте массив автовладельцев. Для каждого автовладельца известен номер, марка автомобиля, фамилия и адрес. Нужно подсчитать количество владельцев автомобиля определенной марки и вывести все сведения о них.

Решение. Сведения об автовладельцах представим массивом записей. Исходные данные вводятся с клавиатуры. Работа с массивом записей аналогична работе с одномерным массивом.

```
const nn=100;{максимальное количество автовладельцев}
type mash=record nomer:integer;       {номер владельца}
                 marka:string[20];      {марка автомобиля}
                 fio:string[40]; {фамилия и инициалы владельца}
                 adres:string[60];      {адрес владельца}
            end;
  mas=array[1..nn] of mash; {тип массива владельцев автомобилей}
var v:mas; {массив автовладельцев}
   n:integer; {количество автовладельцев}
   i:integer; {индекс}
   s:string; {заданная марка автомобиля}
   k:integer; {количество владельцев указанной марки}
begin write('Введите n ');readln(n);
   writeln('Введите ',n,' автовладельцев');
   for i:=1 to n do
   begin v[i].nomer:=i;
      writeln('автовладелец номер ',v[i].nomer);
      write('Фамилия? ');readln(v[i].fio);
      write('марка ? ');readln(v[i].marka);
      write('адрес? ');readln(v[i].adres);
```

```
    end;
    write('Какая марка вас интересует? ');
    readln(s);
    k:=0;
    for i:=1 to n do
      if v[i].marka=s
      then begin with v[i] do writeln(nomer,' ',marka,' ',fio,' ',adres);
                k:=k+1;
           end;
    write('Количество владельцев марки ',s,'=',k);
end.
```

Таким образом, представлены методики решения по пяти из девятнадцати выделенных классов задач:

11. Данные типа string.
12. Процедуры и функции.
13. Рекурсия.
14. Тип данных множество.
15. Тип данных запись.

В следующей статье мы продолжим знакомство с методикой быстрого обучения программированию на основе изучения классов задач. Будут рассмотрены методики по следующим двум из девятнадцати выделенных классов задач:

16. Файлы.
17. Организация работы с модулями.

Литература

1. *Аляев Ю.А.* Алгоритмизация и языки программирования Pascal, C++, Visual Basic / Ю.А. Аляев, О.А. Козлов. М.: Финансы и статистика, 2002, 2004, 2007. 320 с.

2. *Аляев Ю.А.* Практикум по алгоритмизации и программированию на языке Паскаль / Ю.А. Аляев, В.П. Гладков, О.А. Козлов. М.: Финансы и статистика, 2004, 2007. 528 с.

3. *Аляев Ю.А.* Методика быстрого обучения программированию на основе изучения классов задач (1–3) // Образовательные ресурсы и технологии. 2015'1(9). С. 3–14. URL: http://www.muiv.ru/vestnik/pdf/pp/ot_2015_1_3-14.pdf

4. *Аляев Ю.А.* Методика быстрого обучения программированию на основе изучения классов задач (4–5) // Образовательные ресурсы и технологии. 2015'2(10). С. 3–16. URL: http://www.muiv.ru/vestnik/pdf/pp/ot_2015_2_3-16.pdf

5. *Аляев Ю.А.* Методика быстрого обучения программированию на основе изучения классов задач (6–7) // Образовательные ресурсы и технологии. 2015'3(11). С. 3–20. URL: http://www.muiv.ru/vestnik/pdf/pp/ot_2015_003_020.pdf

6. *Аляев Ю.А.* Методика быстрого обучения программированию на основе изучения классов задач (8–10) // Образовательные ресурсы и технологии. 2015'4(12). С. 26–43. URL: http://www.muiv.ru/vestnik/pdf/pp/ot_2015_4_026-043.pdf

Methods of the quick education to programming on base of the study of the classes of the problems (11–15)

***Yuri Alexandrovich Alyaev**, assistant professor, assistant professor of the pulpit of software of the computing machinery and automated systems, Perm military institute of internal troops of the MIA of Russia,*

Is offered methods of the quick education to programming on base of the study of the classes of the problems, designed and using in practice in process of the education to programming student high school.

***The Keywords**: algorithm, program, programming language Pascal, array, procedures and functions, recursion, set, record*