УДК [004.421+004.438](076.5)

МЕТОДИКА БЫСТРОГО ОБУЧЕНИЯ ПРОГРАММИРОВАНИЮ НА ОСНОВЕ ИЗУЧЕНИЯ КЛАССОВ ЗАДАЧ (16–17)

***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, файлы, модули.

DOI: 10.21777/2312-5500-2016-3-3-14

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.

Ю.А. Аляев

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.

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 двум (16 и 17) из девятнадцати выделенных классов задач (1–15 классы задач рассмотрены в [3–7]).

16. Файлы

Задача 1. Создать файл, описывающий группу студентов. Для каждого студента нужно указать фамилию, дату рождения, пять оценок за экзамены прошедшей сессии.

Решение. Опишем структуру записи файла. Для работы файл нужно открыть в режиме перезаписи (rewrite). Затем организовать цикл считывания информации о студентах с клавиатуры и занесения ее в файл. Об окончании работы будет свидетельствовать считывание «пустой» фамилии.

```
typedat=record        {формат даты}
den:1..31;   {день}
mes:1..12;   {месяц}
god:integer; {год}
end;
stud=record       {формат анкеты студента}
fio:string[10];  {фамилия}
dr:dat;          {дата рождения}
```

```
        note:array [1..5] ofbyte; {оценки за прошедшую сессию}
        end;
        vargrup:fileofstud; {файловая переменная}
          st:stud;              {анкета студента}
          s:string;             {имя файла}
          i:integer;            {номер оценки}
beginwrite('Введите имя файла');
        readln(s);
        assign(grup,s);  {создание файла}
        rewrite(grup);
write('фамилия?');readln(st.fio);
        whilest.fio<>'' do
            begin write('день рождения: день месяц год');
        readln(st.dr.den,st.dr.mes,st.dr.god);
        write('пять оценок за экзамены прошедшей сессии');
        for i:=1 to 5 do read(st.note[i]);readln;
        write(grup,st);
        write('фамилия?');readln(st.fio);
        end;
        close(grup);
        end.
```

Задача 2. Добавить информацию в файл, описывающий студентов, задав исходные данные с помощью датчика случайных чисел.

Решение. Вначале программа запрашивает имя файла, с которым будет работать. Затем проверяется существование файла с заданным именем. Если указанного файла не существует, программа останавливается. Если файл существует, дописываем новые записи в конец. Новые записи формируем с помощью датчика случайных чисел. Для получения фамилии используем две строковые константы, содержащие прописные и строчные буквы русского алфавита. Разыграв случайные номера от 1 до 32, выбираем соответствующую букву константы. Вначале находим первую букву фамилии – прописную. Затем определяем, сколько еще букв должно быть получено и определяем их. Наконец формируем окончание русской фамилии. Здесь используются наиболее распространенные окончания русских фамилий: -ов, -ова, -ин, -ина. Дата рождения и оценки также формируются случайным образом.

```
        const alfb='БВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ';
     alfm='абвгдежзийклмнопрстуфхцчшщъыьэюя';
type dat=record  {формат даты}
           den:1..31;   {день}
        mes:1..12;   {месяц}
        god:integer; {год}
        end;
        stud=record  {формат анкеты студента}
        fio:string[10]; {фамилия}
                  dr:dat;       {дата рождения}
                  note:array [1..5] of byte; {оценки за прошедшую сессию}
        end;
        var grup:file of stud;
        st:stud;
        s:string;
        i,n,m,j:integer;
        begin write('Введите имя файла');
        readln(s);
```

```
assign(grup,s);
{проверим, есть ли файл с заданным именем}
{$I-}
reset(grup);
    {$I+}
if ioresult<>0
thenbeginwrite('Указанный файл не существует');
halt;
end;
    write('Сколько записей добавить?');
readln(n);
    {переместились в конец файла, дописываем новые записи}
seek(grup,filesize(grup));
for i:=1 to n do
begin st.fio:=copy(alfb,1+random(32),1);
{первая буква фамилии должна быть прописной}
  m:=2+random(5);
        {сколько еще букв нужно добавить?}
for j:=1 to m do st.fio:=st.fio+copy(alfm,1+random(32),1);
        {добавляем строчные буквы}
case m mod 4 of
  0:st.fio:=st.fio+'ов';
  1:st.fio:=st.fio+'ова';
  2:st.fio:=st.fio+'ин';
        3:st.fio:=st.fio+'ина';
end;
        {сформировали окончание фамилии}
        {формируем дату рождения и оценки}
  st.dr.den:=1+random(31);
  st.dr.mes:=1+random(12);
  st.dr.god:=60+random(31);
for j:=1 to 5 do st.note[j]:=2+random(4);
write(grup,st);
end;
close(grup);
end.
```

Задача 3. Вывести содержимое файла с информацией о студентах в несколько столбиков на экран. Если экран заполнится, а файл не будет исчерпан, то выдать сообщение о необходимости дальнейшего просмотра.

Решение. Вычислим длину записи файла в байтах. Поскольку дата содержит поля, для которых базовым является тип integer (2 байта), то ее длина равна 3*2=6 байтам. Данные о фамилии занимают 11 байт (тип string). Пять оценок занимают 5 байтов. Таким образом, в сумме длина записи равна 22 байтам. При выводе на экран будем отделять одну запись от другой вертикальной чертой. Всего в одной строке экрана можно разместить три записи. Переменные x и y указывают позицию на экране, в которую будет выводиться текущая запись. Они изменяются так, чтобы вначале заполнялся первый столбец, затем второй и, наконец, третий. Если экран заполнится, а записи в файле не будут исчерпаны, то будет выдано сообщение «Дальше?» и компьютер будет ждать, когда нажмут клавишу для продолжения просмотра.

```
uses crt;
const dl='   ';  {константа для выравнивания фамилий}
type dat=record          {формат даты}
```

```
den:1..31;   {день}
mes:1..12;   {месяц}
god:integer; {год}
end;
stud=record            {формат анкеты студента}
fio:string[10];  {фамилия}
dr:dat;          {дата рождения}
note:array [1..5] ofbyte; {оценки за прошедшую сессию}
end;
var grup:file of stud;
st:stud;
s:string;
i:integer;
x,y:integer;
ch:char;
begin textbackground(7);
textcolor(1);
clrscr;
write('Введите имя файла');
readln(s);
assign(grup,s);
{проверим, есть ли файл с заданным именем}
{$I-}
reset(grup);
     {$I+}
if ioresult<>0
thenbeginwrite('Указанный файл не существует');
halt;
end;
clrscr;
x:=1;y:=1;
while not eof(grup) do
begin read(grup,st); {читаем запись из файла}
  {переход в позицию x,y, форматирование и вывод на экран}
gotoxy(x,y);
          st.fio:=st.fio+copy(dl,1,10-length(st.fio));
write(st.fio:10,' ',st.dr.den:2,'.',st.dr.mes:2,'.',st.dr.god:2,' ');
for i:=1 to 5 do write(st.note[i]);write('|');
y:=y+1;
if y>24
then begin x:=x+26;
if x>80-23
then begin gotoxy(x,y+1);
textcolor(12);
gotoxy(10,25);
write('дальше? ');
textcolor(1);
ch:=readkey;
clrscr;
x:=1;y:=1;
end
else y:=1;
```

```
        end;
        end;
        ch:=readkey;
        close(grup);
        end.
```

Задача 4. В файле студентов заменить все оценки «два» на оценки «три», создав страховую копию.

Решение. Страховая копия – это файл, содержимое которого совпадает с содержимым исходного файла. Имя копии совпадает с именем исходного файла, но имеет расширение bak. Переименуем исходный файл. Затем будем читать исходный файл, проверять, есть ли в прочтенной записи оценки «два», и заменять их оценкой «три». В любом случае запись переписывается в новый файл, имя которого совпадает с именем исходного.

Подобные алгоритмы работы с двумя файлами используются столь широко, что для них придумано специальное имя: «алгоритмы перезаписи». Обратите внимание на приведенный пример и запомните его структуру.

```
        typedat=record  {формат даты}
        den:1..31;   {день}
        mes:1..12;   {месяц}
        god:integer; {год}
        end;
        stud=record  {формат анкеты студента}
        fio:string[10]; {фамилия}
        dr:dat;        {дата рождения}
        note:array [1..5] ofbyte; {оценки за прошедшую сессию}
        end;
        var grup,grup1:file of stud;
        st:stud;
        s,s1:string;
        i:integer;
        begin
        write('Введите имя файла');
readln(s);
assign(grup,s);
    {проверим, есть ли файл с заданным именем}
    {$I-}
reset(grup);
    {$I+}
if ioresult<>0
then begin write('Указанный файл не существует');
        halt;
        end;
            {получение имени файла, где будет храниться страховая копия}
    s1:=copy(s,1,pos('.',s))+'bak';
close(grup);
rename(grup,s1);
assign(grup,s1);
reset(grup);
assign(grup1,s);
        rewrite(grup1);
        while not eof(grup) do
        begin read(grup,st);
```

```
for i:=1 to 5 do if st.note[i]=2
then st.note[i]:=3;
write(grup1,st);
end;
close(grup);close(grup1);
end.
```

Задача 5. Упорядочить файл с данными о студентах в алфавитном порядке по фамилиям.

Решение. Используем самый простой метод сортировки и возможность прямого доступа к файлу. Для каждой записи просмотрим все следующие за ней. Если следующая запись содержит фамилию, стоящую в алфавитном порядке раньше просматриваемой, то меняем записи местами. Просмотр продолжается. Для изменения записи необходимо после прочтения вернуться к ней вновь.

```
type dat=record  {формат даты}
den:1..31;   {день}
mes:1..12;   {месяц}
god:integer; {год}
end;
stud=record  {формат анкеты студента}
fio:string[10]; {фамилия}
dr:dat;        {дата рождения}
note:array [1..5] ofbyte; {оценки за прошедшую сессию}
end;
var grup:file of stud;
st,st1,r:stud;
s:string;
i,j:integer;
begin
write('Введите имя файла');
readln(s);
assign(grup,s);
{проверим, есть ли файл с заданным именем}
{$I-}
reset(grup);
     {$I+}
if ioresult<>0
thenbeginwrite('Указанный файл не существует');
halt;
end;
for i:=0 to filesize(grup)-2 do {перебираем все записи, кроме последней}
begin seek(grup,i);read(grup,st);
for j:=i+1 to filesize(grup)-1 do {сравниваем со всеми следующими записями}
begin seek(grup,j);read(grup,st1);
if st.fio>st1.fio
then begin seek(grup,i);write(grup,st1);
seek(grup,j);write(grup,st);
r:=st1; st1:=st; st:=r
end;
end;
end;
close(grup);
end.
```

Задача 6. Удалить из файла студентов m записей, начиная с записи с номером n (нумерация записей начинается с нуля).

Решение. Переписываем записи с номерами n+m+i, где i изменяется от 0 до m или до конца файла, на записи с номерами n+i.

```
type dat=record  {формат даты}
den:1..31;  {день}
mes:1..12;  {месяц}
god:integer; {год}
end;
stud=record  {формат анкеты студента}
fio:string[10]; {фамилия}
dr:dat;        {дата рождения}
note:array [1..5] ofbyte; {оценки за прошедшую сессию}
end;
var grup:file of stud;
st:stud;
s:string;
i,j:longint;
n,m,q:longint;
begin
write('Введите имя файла');
readln(s);
assign(grup,s);
  {проверим, есть ли файл с заданным именем}
      {$I-}
reset(grup);
      {$I+}
if ioresult<>0
thenbeginwrite('Указанный файл не существует');
halt;
end;
    q:=filesize(grup);
writeln('Всего записей в файле:',q);
    write('Введите номер первой удаляемой записи');
    readln(n);
    write('Введите количество удаляемых записей');
readln(m);
if n+m>=q
then begin seek(grup,n);truncate(grup) end
else begin i:=0;
while (n+m+i<q) and (i<=m-1) do
begin seek(grup,n+m+i);
read(grup,st);
seek(grup,n+i);
write(grup,st);
i:=i+1;
end;
seek(grup,n+m);
truncate(grup);
end;
close(grup);
end.
```

Задача 7. Переписать файл студентов в текстовый файл.

Решение. Структура программы совпадает со структурой программы перезаписи.

```
const dl='    '; {константа для выравнивания фамилий}
type dat=record  {формат даты}
   den:1..31;   {день}
  mes:1..12;   {месяц}
  god:integer; {год}
  end;
  stud = record  {формат анкеты студента}
  fio:string[10]; {фамилия}
        dr:dat;        {дата рождения}
        note:array [1..5] of byte; {оценки за прошедшую сессию}
  end;
  var grup:file of stud;
    grup1:text;
  st:stud;
  s:string;
  i:integer;
  begin
  write('Введите имя файла');
readln(s);
assign(grup,s);
  {проверим, есть ли файл с заданным именем}
      {$I-}
  reset(grup);
      {$I+}
  if ioresult<>0
  thenbeginwrite('Указанный файл не существует');
  halt;
  end;
  assign(grup1,'spisok.txt');
  rewrite(grup1);
  while not eof(grup) do
  begin read(grup,st);
       st.fio:=st.fio+copy(dl,1,10-length(st.fio));
  write(grup1,st.fio:10,' ',st.dr.den:2,'.',
  st.dr.mes:2,'.',st.dr.god:2,' ');
  for i:=1 to 5 do write(grup1,st.note[i]);
  writeln(grup1);
  end;
  close(grup);close(grup1);
  end.
```

Задача 8. В текстовом файле заданы фамилии студентов и новый набор оценок. Фамилии и оценки разделяются одним пробелом. Внести соответствующие корректировки в типизированный файл, отсортированный по фамилиям.

Решение. Просматриваем текстовый файл и отыскиваем нужную фамилию с помощью дихотомического поиска.

```
type dat=record  {формат даты}
   den:1..31;   {день}
   mes:1..12;   {месяц}
   god:integer; {год}
```

```
        end;
    stud = record  {формат анкеты студента}
        fio:string[10]; {фамилия}
                dr:dat;         {дата рождения}
                note:array [1..5] of byte; {оценки за прошедшую сессию}
        end;
        var grup:file of stud;
           grup1:text;
        st:stud;
        s,s1:string;
        i,j,k,n,m:integer;
        f:boolean;
        begin
        write('Введите имя файла с данными о студентах');
readln(s);
assign(grup,s);
     {проверим, есть ли файл с заданным именем}
     {$I-}
reset(grup);
     {$I+}
        if ioresult<>0
        thenbeginwrite('Указаный файл не существует');
        halt;
                end;
            write('Введите имя файла с исправлениями');
        readln(s);
        assign(grup1,s);
        {проверим, есть ли файл с заданным именем}
        {$I-}
        reset(grup1);
            {$I+}
        if ioresult<>0
        thenbeginwrite('Указаный файл не существует');
        halt;
        end;
        while not eof(grup1) do
        begin readln(grup1,s);
                s1:=copy(s,1,pos(' ',s)-1);
        delete(s,1,pos(' ',s));
        i:=0;j:=filesize(grup)-1;f:=false;
        while (i<=j) and not f do
        begin k:=(i+j) div 2;
        seek(grup,k);
        read(grup,st);
        if st.fio=s1
        then begin f:=true;
        for n:=1 to 5 do
        begin val(copy(s,1,1),st.note[n],m);
        delete(s,1,2)
        end;
        seek(grup,k);
        write(grup,st);
```

```
end
else if st.fio<s1
then i:=k+1
else j:=k-1;
end;
end;
close(grup);close(grup1);
end.
```

17. Организация работы с модулями

Модули поддерживают принципы модульного программирования, согласно которым взаимодействие логически независимых фрагментов программы должно быть сведено к минимуму. Модуль содержит набор процедур, функций и объектов, которые могут использоваться другими пользователями. Структура модуля такова:

```
unit<имя модуля>;
Interface
  <интерфейсная часть содержит объявления констант, переменных, типов, заголовки процедур, функций>
Implementation
  <реализация процедур и функций>
[begin <инициализация модуля>]
end.
```

Задача 1. Разработать модуль для работы с комплексными числами.

Решение. Используем известные правила выполнения операций с комплексными числами.

```
{Модуль должен размещаться в файле CmplVal.pas}
unit CmplVal;
interface
  type complex=record Re,Im:real; end;

  procedure InitC(r,i:real; var c:complex);
  {получение комплексного числа из двух вещественных}

  procedure AddC(c1,c2:complex; var c:complex);
  {сложение комплексных чисел}

  procedure SubC(c1,c2:complex; var c:complex);
  {вычитание комплексных чисел}

  procedure MultC(c1,c2:complex; var c:complex);
  {умножение комплексных чисел}

  procedure DivC(c1,c2:complex; var c:complex);
  {деление комплексных чисел}

  procedure WriteC(c1:complex);
  {печать комплексного числа}

  Implementation
  procedure InitC(r,i:real; var c:complex);
  begin with c do begin Re:=r; Im:=i; end; end;

  procedure AddC(c1,c2:complex; var c:complex);
  begin with c do
          begin Re:=c1.Re+c2.Re;
                Im:= c1.Im+c2.Im;
          end;
```

```
  end;

  procedure SubC(c1,c2:complex; var c:complex);
  begin with c do
        begin Re:=c1.Re-c2.Re;
              Im:=c1.Im-c2.Im;
        end;
  end;

  procedure MultC(c1,c2:complex; var c:complex);
  begin with c do
        begin Re:=c1.Re*c2.Re+c1.Im*c2.Im;
              Im:=c1.Im*c2.Re+c1.Re*c2.Im;
        end;
  end;

  procedure DivC(c1,c2:complex; var c:complex);
  var r:real;
  begin with c2 do r:=Re*Re+Im*Im;
        with c do
        begin Re:=(c1.Re*c2.Re+c1.Im*c2.Im)/r;
        Im:=(c1.Im*c2.Re+c1.Re*c2.Im)/r;
        end;
  end;

  procedure WriteC(c:complex);
  begin with c do
        begin write(Re);
        if Im then exit;
        if Im>0 then write('+');
        write(Im);
        write('i');
        end;
        end;

beginwrite('Подключен модуль работы с комплексными числами');
end.
```

Используем разработанный модуль.

```
usesCmplVal;
var p1,p2,p3:complex;
begin initc(1,2,p1);
  initc(3,4,p2);
  multc(p1,p2,p3);
  writec(p3);
  divc(p3,p2,p1);
  writec(p1);
end.
```

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

16. Файлы.

17. Организация работы с модулями.

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

18. Указатели. Динамическая память. Структуры данных.
19. Отладка Паскаль-программ.

Литература

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

2. *Аляев Ю.А., Гладков В.П., Козлов О.А.* Практикум по алгоритмизации и программированию на языке Паскаль. – 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. 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. 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. 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. http://www.muiv.ru/vestnik/pdf/pp/ot_2015_4_026-043.pdf.

7. *Аляев Ю.А.* Методика быстрого обучения программированию на основе изучения классов задач (11–15) // Образовательные ресурсы и технологии. 2016. № 1 (13). С. 8–20. http://www.muiv.ru/vestnik/pdf/pp/ot_2016_1_008-020.pdf.

Methods of the quick education to programming on base of the study of the classes of the problems (16–17)

***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
http://pvivv.ru

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, files, modules.

УДК 378.14

МОДЕЛИРОВАНИЕ И НАГЛЯДНОСТЬ В ОБУЧЕНИИ ИНОСТРАННОМУ ЯЗЫКУ СТУДЕНТОВ НЕЯЗЫКОВЫХ ВУЗОВ

***Александр Викторович Аниол**, канд. техн. наук, доцент,*
e-mail: aaniol@muiv.ru,
Московский университет имени С. Ю. Витте,
http://www.muiv.ru

В статье рассмотрена практика использования знаковых моделей на занятиях по иностранному языку в неязыковом вузе. Разработан ряд моделей, иллюстрирующих некоторые особенности английской грамматики и сложности перевода. Показана эффективность их использования при обучении студентов с разным уровнем подготовки. Показана важность такого свойства моделей, как наглядность, для использования их в учебном процессе.

Ключевые слова: модель, наглядность, межкультурная коммуникация, грамматика, перевод, английский язык.

DOI: 10.21777/2312-5500-2016-3-18-24