УДК 004.42

# ИСПОЛЬЗОВАНИЕ ПОТОКОВ ВВОДА-ВЫВОДА ПРИ ИЗУЧЕНИИ ПРОГРАММИРОВАНИЯ НА ПРИМЕРЕ ЯЗЫКА ПАСКАЛЬ

**Г.Е. Злуникин**

*ФГБОУ ВПО «Борисоглебский государственный педагогический институт», г. Брисоглебск*

*Рецензент д-р пед. наук, профессор Е.А. Ракитина*

**Ключевые слова и фразы:** программирование; обучение информатике; файл.

**Аннотация:** Рассмотрен подход к изучению потоков ввода-вывода информации на начальной стадии знакомства с языками программирования. Материал может быть использован при построении как школьного, так и вузовского курсов программирования. Приведены примеры заданий.

В «Си»-подобных языках программирования часто используется понятие «поток ввода-вывода» информации, который, как правило, связывают с соответствующими устройствами ввода-вывода. При изучении языка программирования Паскаль эти вопросы, как правило, остаются малоизученными или вовсе не освещаются.

В статье делается попытка показать, как можно использовать на практике потоки ввода-вывода.

Рассмотрение потоков ввода-вывода начнем с простейших задач.

**Задача 1.** На вход программы подается два целых числа. Найти их сумму, разность и произведение.

Решение задачи, в первом приближении, тривиально и не требует комментариев.

```
{1} program zd1;
{2} var
{3}   a, b : integer;
{4}   sum, raz, mult : integer;
{5} begin
{6}    writeln('Введите два числа:');
{7}    readln(a, b);
{8} sum := a+b;
{9} raz := a-b;
{10} mult := a * b;
```

Злуникин Геннадий Евгеньевич – старший преподаватель кафедры прикладной математики, информатики, физики и методики их преподавания, начальник центра коммуникационно-информационных технологий образования, e-mail: ZL1@rambler.ru, ФГБОУ ВПО «Борисоглебский государственный педагогический институт», г. Борисоглебск.

```
{11} writeln(a,' + ', b, ' = ', sum);
{12} writeln(a,' - ', b, ' = ', raz);
{13} writeln(a,' * ', b, ' = ', mult);
{14} end.
```

Однако из условия задачи не понятно «откуда» берутся два числа, вводятся ли они с клавиатуры или берутся из файла. Если данные загружаются из файла, то, как правило, считается необходимым знать и уметь работать файловым типом данных и знать соответствующие подпрограммы. Однако этот «сложный» материал можно обойти, введя минимум новых понятий и используя потоки ввода-вывода.

Немного изменим программу, а затем прокомментируем ее.

```
{1} program zd2;
{2} var
{3}   a, b : integer;
{4}   sum, raz, mult : integer;
{5} begin
{5-1} Assign (input, 'in.dat'); Reset(input);
{5-2} Assign (output, 'out.dat'); Rewrite(output);
{6}    writeln('Введите два числа:');
{7}    readln(a, b);
{8} sum := a+b;
{9} raz := a-b;
{10} mult := a * b;
{11} writeln(a,' + ', b, ' = ', sum);
{12} writeln (a,' - ', b, ' = ', raz);
{13} writeln (a,' * ', b, ' = ', mult);
{14} end.
```

Итак, в строке {5-1} процедура `Assign` осуществляет логическое связывание файловой переменной `input` с файлом `'in.dat'`, который находится в текущем каталоге. Затем этот файл открывается процедурой `Reset`.

Переменная `input` является «стандартной» текстового типа, а также эта файловая переменная осуществляет взаимодействие со стандартным потоком ввода информации. В нашем случае мы перенаправили «стандартный» поток ввода (клавиатуру) на файл, то есть чтение (ввод) данных мы будем осуществлять не посредством ввода данных с клавиатуры, а посредством чтения данных из файла.

Переменная `output` также является «стандартной» (она объявлена как и `input` в модуле `system` и является переменной текстового типа), осуществляет взаимодействие с потоком вывода данных, который по умолчанию ассоциируется с монитором (дисплеем). В нашем случае поток вывода мы также связываем с файлом `'out.dat'`. Поток вывода надо создавать, поэтому мы применили процедуру `Rewrite`, которая формирует новый файл.

Обычно потоки ввода-вывода связываются с консолью ввода-вывода, для потока ввода стандартной консолью является клавиатура, для вывода – дисплей.

В любой момент программист может перенаправить поток, применив процедуру `Assign`. Чтобы вернуться к «стандартным» потокам в качестве файла используется специальная константа – `'Con'` (от слова *Console*).

Обратим внимание, что при запуске программы zd2 на экране монитора ничего не будет выведено, поскольку весь вывод осуществлен в файл.

Наиболее наглядными примерами использования потокового ввода-вывода, по нашему мнению, являются задания по теме «Структурированные типы данных» (начиная с массивов).

*Рассмотрим задачу*: дан двумерный массив (размерность определена) целых чисел (для простоты в интервале от –5 до +5), найти минимальный элемент и вывести его координаты.

Для проверки этой задачи выдаются тестовые задания – двумерные таблицы со значениями. Разобьем задачу на две части: 1) ввод массива; 2) решение (поиск).

Первая подзадача решается аналогично предыдущим задачам, ее листинг приведен ниже.

```
{1} Const n=3;        m=3;
{2}  Type mas=array[1..m,1..n] of integer;
{3}  Var
{4}      a:mas;
{5}     s,i,j : integer;
{6} begin
{7} { Assign(input,'in.txt'); Reset(input);}
{8}   Assign(output,'out.txt'); Rewrite(output);
{9}    Randomize;
{10}    Writeln(m:4,n:4);
{11}     For i:=1 to m do
{12}     For j:=1 to n do
{13}    a[i,j]:=Random(10)-5;
{14}    For i:=1 to m do
{15}    begin Writeln;
{16}    For j:=1 to n do
{17}    write(a[i,j] :3);
{18}     end;
{19} end.
```

Строка {7} отключена. Вводить данные из внешнего источника не надо, мы будем генерировать данные. В строке {8} мы определили, что выводить данные будем в файл 'out.txt'.

Последовательность решения второй подзадачи такова.

Предварительно переименовываем файл 'out.txt' в 'in.txt', получившийся массив из части 1 является входным массивом для части 2.

В первой строке файла (таблица) записана размерность массива[1], мы ее считываем, однако, предварительно объявив массив большего размера.

Затем, считываем весь массив в память, осуществляем поиск и выводим результаты.

Листинг (часть 2):

```
{1}Const nb=5;
{2}      mb=5;
{3}Type mas= array[1..mb,1..nb] of integer;
{4}Var
```

**Файл 'out.txt'**

| | | |
|---|---|---|
| 3 | 3 | |
| 3 | 1 | 4 |
| 3 | 4 | 5 |
| 4 | 1 | 0 |

[1] Размерность массива можно уменьшить, удалив при этом «лишние» элементы. При этом мы показываем универсальность алгоритма.

```
{5}    a:mas;
{6}   s,i,j : integer;      m,n  : integer;
{7}    min_i,min_j,min: integer;
{8} begin
{9}    Assign(input,'in.txt'); Reset(input);
{10}   Assign(output,'out.txt'); Rewrite(output);
{11}   Readln (m,n);
{12}   For i:=1 to m do
{13}   For j:=1 to n do
{14}   Read(a[i,j]);   min:= a[1,1]; min_i:=1; min_j:=1;
{15}   For i:= 1 to m do
{16}   For j:=1 to n do
{17}    if a[i,j]<min then
{18}      begin
{19}        min_i:=i;   min_j:=j;    min:=a[i,j];
{20}      end;
{21}  For i:=1 to m do
{22}  begin Writeln;
{23}  For j:=1 to n do
{24}  write(a[i,j] :3);
{25}   end;     Writeln;
{26}WriteLn('min=',min:4,'min_i=':8,min_i,'min_j=':8,min_j);
{27}end.
```

Как показывает опыт практической работы, изучение потоков вводится достаточно просто, оттачивает навыки по работе с файловым менеджером, более глубоко погружает обучающихся в интегрированную среду разработки, позволяет свободно владеть вводом-выводом данных как в файл, так и из файла.

При изучении в вузе курса «Теоретические основы информатики» студенты решают задачи по определению объема информации в некотором файле. При использовании потоков ввода-вывода студенты разрабатывают набор специализированных утилит, которые осуществляют требуемое действие.

---

**Using Input-Output Flows**
**in the Course of Programming in Pascal**

**G.E. Zlunikin**

*Borisoglebsk State Pedagogical Institute, Borisoglebsk;*

**Key words and phrases:** programming; text files; input-output flows; teaching computer science.

**Abstract:** The approach to the studying of input-output flows of the information at the initial stage of learning programming languages is considered; the material can be used in designing a construction of school and university programming courses. Sample tasks are given.

---