А. И. ЕФИМОВ

УЧЕБНИК ПО ПРОГРАММИРОВАНИЮ НА ЯЗЫКЕ ОБЕРОН

ЧАСТЬ I

для самостоятельного обучения
и компьютерных курсов

ВВЕДЕНИЕ

Данное пособие представляет собой сборник задач с решениями и предназначено для использования на начальном этапе изучения программирования, а также для углублённого изучения проблем, возникающих при составлении алгоритмов и написании программ.

К условию каждой задачи прилагаются комментарии, советы по решению задачи, а также само решение на языке программирования Оберон с подробными пояснениями. Задачи расставлены по принципу «от простого к сложному». Решая задачи одну за другой, учащийся постепенно получает теоретические и практические навыки, необходимые для освоения профессии программиста.

Приведённые в пособии программы могут быть скомпилированы с помощью пакета программного обеспечения Фри-Оберон, доступного на сайте freeoberon.su. Исходные коды приведённых программ также доступны на указанном сайте.

ГЛАВА I.

ЛИНЕЙНЫЕ ПРОГРАММЫ.

§ 1. Консольный ввод и вывод. Тип INTEGER. Оператор IF THEN ELSE.

Задача № 1.
Вводятся два целых числа. Определить наименьшее и наибольшее из них. Пример работы программы:

```
A = 51
B = 17
Min: 17
Max: 51
```

Решение:

```
 1 |  MODULE MinMax2;
 2 |  IMPORT In, Out;
 3 |  VAR a, b: INTEGER;     (* исходные данные *)
 4 |      min, max: INTEGER; (* результаты *)
 5 |  BEGIN
 6 |    Out.String('Эта программа находит минимум и максимум двух чисел.');
 7 |    Out.Ln; Out.Ln;
 8 |    Out.String('A = '); In.Int(a);
 9 |    Out.String('B = '); In.Int(b);
10 |    IF a > b THEN
11 |      max := a;
12 |      min := b
13 |    ELSE
14 |      max := b;
15 |      min := a
16 |    END;
17 |    Out.String('Min: '); Out.Int(min, 0); Out.Ln;
18 |    Out.String('Max: '); Out.Int(max, 0); Out.Ln
19 |  END MinMax2.
```

Пояснения к решению:

Программа должна считывать вводимые пользователем числа с клавиатуры, для чего потребуется задействовать модуль In. Кроме того, программа выводит на экран некоторый текст-пояснение и ответ, для чего а значит необходимо задействовать модуль Out.

Для хранения *целых чисел* в памяти ЭВМ используется тип данных, называемый INTEGER. Объявляем четыре переменные типа INTEGER: *a*, *b*, *min* и *max*. В переменных *a* и *b* будут храниться считанные с клавиатуры числа, а в переменных *min* и *max* — результаты вычислений.

Рассмотрим оператор `Out.String('A = ')`. Он представляет собой *вызов процедуры* под названием String, находящейся в модуле Out.

Вообще, вызов некоторой процедуры означает временную *передачу управления* этой процедуре с последующим возвратом хода программы на то же место и продолжением работы. После Out.String в скобках перечисляются *параметры*, передаваемые этой процедуре. В данном случае передаётся единственный параметр — *строка* `'A = '`. Строки заключаются в одинарные или двойные кавычки и означают буквально то, что в них заключено. Компилятор не «вникает» в содержимое строки, поэтому программист волен заключать в кавычки любой текст. Результатом выполнения процедуры `Out.String('A = ')` будет вывод переданных четырёх *символов* (или *литер*) на экран, т. е. буквы A, пробела, знака равенства и ещё одного пробела.

В строке №7 находятся сразу два оператора: `Out.Ln; Out.Ln;` Данные операторы представляют собой два подряд идущих вызова процедуры Ln из модуля Out. Они расположены на одной строке, но с тем же успехом мы могли бы разместить их и на двух строках, эффект один — выполнение первой команды, а затем второй. Данные операторы написаны на одной строке для большей выразительности программы, а также с целью экономии места на экране монитора. Такое размещение операторов будет встречаться и дальше.

Смысл же оператора `Out.Ln` в том, чтобы перенести *курсор* на следующую строку. Положение курсора определяет, где будет выведен последующий текст. В момент запуска программы, курсор находится в левом верхнем углу экрана, а при выводе очередного символа сдвигается вправо. `Out.Ln` переносит курсор на следующую строку и сдвигает его влево до упора. Оператор `Out.Ln` также обеспечивает, чтобы весь выведенный ранее (на данной строке) текст стал виден, поэтому в конце практически любой программы также можно найти `Out.Ln`.

После того, как входные данные были прочитаны с клавиатуры, программа приступает непосредственно к решению поставленной задачи. Логика здесь простая: если первое из введённых значений (a) больше второго (b), то первое будем считать максимумом, а в торое — минимумом, иначе — наоборот.

Рассмотрим данный кусок кода подробно:

```
10 |    IF a > b THEN
11 |      max := a;
12 |      min := b
13 |    ELSE
14 |      max := b;
15 |      min := a
16 |    END;
```

Сначала проверяется условие `a > b`. Если это условие окажется истинным, то программа перейдёт к выполнению строк 11—12, а затем перейдёт на строку 17. Если же это условие окажется ложным, программа перейдёт на строку 14. Таким образом, будет выполнена только одна из двух ветвей оператора IF ELSE.

Строка `min := a` представляет собой операцию присваивания. Текущее значение переменной *a* будет скопировано в переменную *min*. Важно осознавать, что при этом старое значение переменной *min* будет навсегда утеряно. В данном случае нас это не волнует, т. к. *min* не хранила никакого конкретного значения.

Обратите внимание, что перед ключевыми словами ELSE и END точка с запятой не ставится. Назначение точки с запятой — разделять два подряд идущих оператора, тогда как ни ELSE, ни END не являются оператором. Ключевые слова THEN, ELSE и END (а также ELSIF) — это составные части оператора IF.

После выполнения оператором IF одной из своих ветвей, в переменной *max* будет содержаться либо такое же значение, как в переменной *a*, либо такое же, как в переменной *b*, в зависимости от того, какое из них было больше, а в переменной *min* будет содержаться меньшее значение. Рассмотрим частный случай, когда значения *a* и *b* были равны. Условие `a > b` окажется ложным, поэтому выполнится ветвь ELSE. В переменные *min* и *max* будет записано одно и то же число, что и будет правильным ответом.

Итак, правильный ответ вычислен, теперь необходимо вывести его на экран. Для этого снова используется процедура `Out.String`, а также `Out.Int` — для вывода числа. Процедура `Out.Int` принимает два целочисленных параметра: первый представляет собой число, которое необходимо вывести на экран, а второй — количество знаков, которое это число минимально должно занимать на экране. По необходимости, слева от числа будут выведено некоторое количество пробелов. Например, `Out.Int(17, 5)` выведет три пробела, а затем число 17. Общее количество знаков, таким образом, составит пять. Мы же передаём вторым параметром нуль, поэтому число будет выписано без дополнительных пробелов.

Задача № 2.
Вводятся три целых числа. Определить наименьшее и наибольшее из них. Пример работы программы:

```
A = 22
B = 26
C = 5
Min: 5
Max: 26
```

Решение:

```
 1 |  MODULE MinMax3;
 2 |  IMPORT In, Out;
 3 |  VAR a, b, c: INTEGER;  (* исходные данные *)
 4 |      min, max: INTEGER; (* результаты *)
 5 |  BEGIN
 6 |    Out.String('Эта программа находит минимум и максимум трёх чисел.');
 7 |    Out.Ln; Out.Ln;
 8 |    Out.String('A = '); In.Int(a);
 9 |    Out.String('B = '); In.Int(b);
10 |    Out.String('C = '); In.Int(c);
11 |    IF (a > b) & (a > c) THEN max := a
12 |    ELSIF b > c THEN max := b
13 |    ELSE max := c
14 |    END;
15 |    IF (a < b) & (a < c) THEN min := a
16 |    ELSIF b < c THEN min := b
17 |    ELSE min := c
18 |    END;
19 |    Out.String('Min: '); Out.Int(min, 0); Out.Ln;
20 |    Out.String('Max: '); Out.Int(max, 0); Out.Ln
21 |  END MinMax3.
```

Пояснения к решению:

Основное полезное действие в программе приходится на строки 11—14 (определение максимума) и 15—18 (определение минимума). Эти два куска кода представляют собой два отдельных оператора IF. Рассмотрим первый из них. Знак `&` (амперсанд) означает логическое И, т. е. оба условия (слева и справа от `&`) должны быть истинны, и только в этом случае всё условие в целом будет считаться истинным.

Если одновременно выполняются оба условия: `a > b` и `a > c`, то очевидно, что a и есть максимальное число, поэтому значение переменной a будет скопировано в переменную max.

Если же это условие ложно (и только в этом случае), будет проверено условие на строке 12: `b > c`. При этом, важно понимать, что если программа при исполнении составного оператора IF дошла до некоторого ELSIF, то это означает, что все предыдущие условия оказались ложными. Другими словами, при проверке условия `b > c`, мы уже знаем, что a —

не есть максимальное число, так как предыдущее условие (а именно `(a > b) & (a > c)`) оказалось ложным. Это упрощает нам задачу, ведь остаётся проверить только какое из чисел *b* и *c* является максимальным.

Наконец, если обе ветви оператора IF оказались «провальными», то это можно понимать следующим образом: «ни *a*, ни *b* не являются максимальными», а значит, безусловно, максимальное число — *c*. Это и учтено в строке: `ELSE max := c`.

Нечто очень похожее происходит и в следующем операторе IF. Чтобы превратить его в поиск минимума, оказалось достаточным заменить знак > на <, а также *max* на *min*.

Следует отметить, что во всей программе можно заменить < на <=, а также > на >=. Это не изменит ничего во внешнем проявлении программы, но потенциально сделает её чуточку быстрее, так как если какие-то два из трёх чисел, или даже все три, окажутся равными, то всё равно, какое из них считать за максимум (или минимум). При a = b = c все три числа являются и максимальными, и минимальными.

* * *

/ продолжение следует /