**Решение простых задач на языке Паскаль**

**9 класс**

**Формулировка.** Вывести на экран сообщение «Hello World!».

Некоторые учебные курсы по программированию рассматривают эту задачу как самую первую при изучении конкретного языка или основ программирования.

**Решение.** Эта задача включает в себя лишь демонстрацию использования оператора вывода **write** (или **writeln**), который будет единственным в теле нашей маленькой программы. С помощью него мы будем осуществлять вывод на экран константы 'Hello World!' типа **string** (или, как допускается говорить, строковой константы). В данном случае будем использовать оператор **writeln**.

Напомню, что при использовании оператора **write** курсор останется в той же строке, в которой осуществлялся вывод, и будет находиться на одну позицию правее восклицательного знака во фразе «Hello World!», а при использовании оператора **writeln** – на первой позиции слева в следующей строке.

**Код:**

program HelloWorld;

begin

writeln('Hello World!')

end.

**Формулировка.** Вывести на экран три введенных с клавиатуры числа в порядке, обратном их вводу.

Другими словами, мы ввели с клавиатуры три числа (сначала первое, потом второе и третье), и после этого единственное, что нам нужно сделать – это вывести третье, затем второе и первое.

**Решение.** Так как с клавиатуры вводится три числа, необходимо завести три переменные. Обозначим их как **a**, **b** и **c**. Ввиду того, что нам ничего не сказано о том, в каком отрезке могут располагаться введенные числа, мы возьмем тип **integer**, так как он охватывает и положительные, и отрицательные числа в некотором диапазоне (от -2147483648 до 2147483647). Затем нам нужно использовать оператор вывода **write** (**writeln**), в списке аргументов которого (напомним, что список аргументов **write**(**writeln**) может содержать не только переменные, но и константы и арифметические выражения) эти переменные будут находиться в обратном порядке. В данном случае будем использовать оператор **writeln**, который после вывода результата переведет курсор на следующую строку:

writeln(c, b, a);

Однако если мы оставим его в таком виде, то увидим, что при выводе между переменными не будет никакого пробела, и они будут слеплены и визуально смотреться как одно число. Это связано с тем, что при вводе мы использовали пробелы для разделения чисел, а сами пробелы никаким образом не влияют на содержимое переменных, которые будут последовательно выведены оператором **writeln** без каких-либо дополнений. Чтобы избежать этого, нам нужно добавить в список аргументов **writeln** две текстовые константы-пробелы. Проще говоря, пробельная константа – это символ пробела, заключенный в одиночные апострофы (апостроф – символ «'»). Первая константа будет разделять переменные **a** и **b**, вторая – **b** и **c**. В результате наш оператор вывода будет таким:

writeln(c, ' ', b, ' ', a);

Теперь он работает так: выводит переменную **c**, затем одиночный символ пробела, затем переменную **b**, потом еще один символ пробела и, наконец, переменную **a**.

**Код:**

program WriteThree;

var

a, b, c: integer;

begin

readln(a, b, c);

writeln(c, ' ', b, ' ', a)

end.

**Формулировка.** Дано натуральное число меньше 256. Сформировать число, представляющее собой его квадрат.

**Решение.** Задача является аналогом **задачи 2**. Единственное отличие состоит в том, что условие данной задачи требует предварительное сохранение результата в некоторой переменной. Так как нам ничего не сообщается о необходимости сохранить исходное число, то для получения квадрата мы можем использовать ту же самую переменную, в которую считывали число с клавиатуры. Обозначим эту переменную как **a**.

В условии задачи дается ограничитель величины вводимого числа – фраза «меньше 256». Это означает, что оно может быть охвачено типом **byte**. Но что произойдет, если в переменную **a**будет введено число 255, и затем мы попытаемся присвоить ей его квадрат 65025? Естественно, это вызовет переполнение типа данных, так как используемой для переменной **a** ячейки памяти не хватит для того, чтобы вместить число 65025. Значит, для ее описания мы должны использовать более емкий числовой тип. При этом типом минимальной размерности, охватывающим данный отрезок (от 1 (это 12) до 65025), является тип **word**. Его мы и будем использовать при описании **a**.

Далее нужно сформировать в переменной **a** квадрат. Для этого присвоим ей ее прежнее значение, умноженное само на себя:

a := a \* a;

Теперь остается вывести результат на экран. Для этого будем использовать оператор **writeln**.

**Код:**

program SqrOfNum;

var

a: word;

begin

readln(a);

a := a \* a;

writeln(a)

end.

**Формулировка.** Сформировать число, представляющее собой реверсную (обратную в порядке следования разрядов) запись заданного трехзначного числа. Например, для числа 341 таким будет 143.

Давайте разберемся с условием. В нашем случае с клавиатуры вводится некоторое трехзначное число (трехзначными называются числа, в записи которых три разряда (то есть три цифры), например: 115, 263, 749 и т. д.). Нам необходимо получить в некоторой переменной число, которое будет представлять собой реверсную запись введенного числа. Другими словами, нам нужно перевернуть введенное число «задом наперед», представить результат в некоторой переменной и вывести его на экран.

**Решение.**Определимся с выбором переменных и их количеством. Ясно, что одна переменная нужна для записи введенного числа с клавиатуры, мы обозначим ее как **n**. Так как нам нужно переставить разряды числа **n** в некотором порядке, следует для каждого из них также предусмотреть отдельные переменные. Обозначим их как **a**(для разряда единиц), **b**(для разряда десятков) и **c**(для разряда сотен).

Теперь можно начать запись самого алгоритма. Будем разбирать его поэтапно:

1)      Вводим число **n**;

2)      Работаем с разрядами числа **n**. Как известно, последний разряд любого числа в десятичной системе счисления – это остаток от деления этого числа на 10. В терминах языка **Pascal** это означает, что для получения разряда единиц нам необходимо присвоить переменной **a** остаток от деления числа **n** на 10. Этому шагу соответствует следующий оператор:

a := n mod 10;

Получив разряд единиц, мы должны отбросить его, чтобы иметь возможность продолжить работу с разрядом десятков. Для этого разделим число **n** на 10. В терминах **Pascal**, опять же, это означает: присвоить переменной **n** результат от деления без остатка числа **n** на 10. Это мы сделаем с помощью оператора

n := n div 10;

3)      Очевидно, что после выполнения **п. 2**в переменной **n** будет храниться двухзначное число, состоящее из разряда сотен и разряда десятков исходного. Теперь, выполнив те же самые действия еще раз, мы получим разряд десятков исходного числа, но его уже нужно присваивать переменной **b**.

4)      В результате в переменной **n** будет храниться однозначное число – разряд сотен исходного числа. Мы можем без дополнительных действий присвоить его переменной **c**.

5)      Все полученные в переменных числа – однозначные. Теперь переменная **n** нам больше не нужна, и в ней нужно сформировать число-результат, в котором **a** будет находиться в разряде сотен, **b** – десятков, **c** – единиц. Легко понять, что для этого нам следует умножить **a** на 100, прибавить к полученному числу **b**, умноженное на 10 и **c**без изменения, и весь этот результат присвоить переменной **c**. Это можно записать так:

n := 100 \* a + 10 \* b + c;

6)      Далее остается только вывести полученное число на экран.

**Код:**

program ReverseNum;

var

n, a, b, c: word;

begin

readln(n);

a := n mod 10;

n := n div 10;

b := n mod 10;

n := n div 10;

c := n;

n := 100 \* a + 10 \* b + c;

writeln(n)

end.

Проверим работу программы на произвольном варианте введенных данных. Для этого выполним ее «ручную прокрутку», проделав с введенным числом те же действия, которые должен выполнить алгоритм.

Пусть пользователем введено число 514. Покажем в таблице, какие значения будут принимать переменные **после**выполнения соответствующих строк. При этом прочерк означает, что значение переменных на данном шаге не определено, а красным цветом выделены переменные, которые изменяются:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ строки** | **n** | **a** | **b** | **c** |
| 7 | 514 | — | — | — |
| 8 | 514 | 4 | — | — |
| 9 | 51 | 4 | — | — |
| 10 | 51 | 4 | 1 | — |
| 11 | 5 | 4 | 1 | — |
| 12 | 5 | 4 | 1 | 5 |
| 13 | 415 | 4 | 1 | 5 |

Нетрудно понять, что написанная программа будет выводить правильный ответ для любого заданного трехзначного числа, так как в соответствии с алгоритмом заполнение данной таблицы возможно лишь единственным образом. Это значит, что мы можем представить число в виде абстрактного трехзначного числа ***xyz***, (в нем каждая буква должна быть заменена на любое число от 0 до 9, конечно, за исключением тех случаев, когда оно перестает быть трехзначным), и работая с разрядами этого числа, показать, что в результате работы ответом будет число ***zyx***.

**Формулировка.** Дано натуральное число меньше 16. Посчитать количество его единичных битов. Например, если дано число 9, запись которого в двоичной системе счисления равна 10012(подстрочная цифра 2 справа от числа означает, что оно записано в двоичной системе счисления), то количество его единичных битов равно 2.

**Решение.**Нам необходима переменная для ввода с клавиатуры. Обозначим ее как **n**. Так как мы должны накапливать количество найденных битов, то возникает потребность в еще одной переменной. Обозначим ее как **count** («*count*» в переводе с англ. означает «считать», «подсчет» и т. д.). Переменные возьмем типа **byte** (они могут принимать значения от 0 до 255), и пусть в данном случае такой объем избыточен, но это не принципиально важно.

Как же сосчитать количество битов во введенном числе? Ведь число же вводится в десятичной системе счисления, и его нужно переводить в двоичную?

На самом деле все гораздо проще. Здесь нам поможет одно интересное правило:

*Остаток от деления любого числа****x****в системе счисления с основанием****p****на само число****p****дает нам разряд единиц числа****x****(его крайний разряд справа).*

То есть, в десятичной системе счисления мы получаем разряд единиц некоторого числа, взяв остаток от деления этого числа на 10. Возьмем, например, число 3468. Если остаток от деления на 10 равен 8, то есть разряду единиц этого числа.

Такие же правила господствуют и в арифметике в других системах счисления, и в том числе в двоичной системе. Предлагаю поэкспериментировать: запишите на бумаге десятичное число, затем, используя любой калькулятор с функцией перевода из одной системы счисления в другую, переведите это число в двоичную систему счисления и также запишите результат. Затем разделите исходное число на 2 и снова переведите в двоичную систему. Как оно изменилось в результате? Вполне очевидно, что у него пропал крайний разряд справа, или, как мы уже говорили ранее, разряд единиц.

Но как это использовать для решения задачи? Воспользуемся тем, что в двоичной записи числа нет цифр, кроме 0 и 1. Легко убедиться в том, что сложив все разряды двоичного числа, мы получаем как раз таки количество его единичных битов. Это значит, что вместо проверки значений разрядов двоичного представления числа мы можем прибавлять к счетчику сами эти разряды – если в разряде был 0, значение счетчика не изменится, а если 1, то повысится на единицу.

Теперь, резюмируя вышеприведенный итог, можно поэтапно сформировать сам алгоритм:

1)      Вводим число **n**;

2)      Обнуляем счетчик разрядов **count**. Это делается потому, что значения всех переменных при запуске программы считаются неопределенными, и хотя в большинстве компиляторов **Pascal** они обнуляются при запуске, все же считается признаком «хорошего тона» в программировании обнулить значение переменной, которая будет изменяться в процессе работы без предварительного присваивания ей какого-либо значения.

3)      Прибавляем к **count** разряд единиц в двоичной записи числа **n**, то есть остаток от деления **n** на 2:

count := count + n mod 2;

Строго говоря, мы могли бы не прибавлять предыдущее значение переменной **count** к остатку от деления, так как оно все равно было нулевым. Но мы поступили так для того, чтобы сделать код более однородным, далее это будет видно. Учтя разряд единиц в двоичной записи **n**, мы должны отбросить его, чтобы исследовать число далее. Для этого разделим **n** на 2. На языке **Pascal** это будет выглядеть так:

n := n div 2;

4)      Теперь нам нужно еще два раза повторить **п. 3**, после чего останется единственный двоичный разряд числа **n**, который можно просто прибавить к счетчику без каких-либо дополнений:

count := count + n;

5)      В результате в переменной **count** будет храниться количество единичных разрядов в двоичной записи исходного числа. Осталось лишь вывести ее на экран.

**Код:**

program BinaryUnits;

var

n, count: byte;

begin

readln(n);

count := 0;

count := count + n mod 2;

n := n div 2;

count := count + n mod 2;

n := n div 2;

count := count + n mod 2;

n := n div 2;

count := count + n;

writeln(count)

end.

Программа работает правильно на всех вариантах правильных исходных данных, в чем несложно убедиться с помощью простой проверки.