“Мир информатики”

Журнал для тех, у кого информатика – любимый школьный предмет

Выпуск № 2, июнь 2016 г.

01100010001010100100111110010001100100100010100100100101 1

Уважаемые коллеги!

Предлагаем вашему вниманию второй выпуск интернет-журнала «Мир информатики», предназначенного для учащихся.

Пожалуйста, знакомьте своих учеников с публикуемыми материалами и предлагайте им направить ответы в редакцию по адресу *mirinfo@infojournal.ru*.

Фамилии всех приславших правильные ответы и решения будут опубликованы (с указанием учебного заведения и учителя информатики), а победители конкурсов и авторы лучших ответов будут награждаться дипломами. Будут публиковаться также комментарии редакции к ответам и решениям.

Конечно, мы рассмотрим возможность публикации в журнале и ваших материалов, полезных и интересных учащимся.

С надеждой на сотрудничество,
редакция журнала «Информатика в школе»

Школа программирования

Когда человек хочет передвинуть гору,
он начинает с того, что убирает маленькие камни.
Восточная мудрость

Рассмотрим несколько вопросов, связанных с программированием. При описании будем использовать школьный алгоритмический язык (система программирования КуМир). Русский синтаксис этого языка делает его максимально понятным, и вы легко сможете разработать аналогичные программы на языке программирования, который изучаете.

Обмен значениями переменных величин

Рассмотрим задачу: «Даны значения двух переменных величин *a* и *b*. Произвести обмен их значений».

Очевидно, что решение, что называется, «в лоб»:

```
a := b
b := a
```

или

```
b := a
a := b
```

требуемого результата не даст (убедитесь в этом!). Как же быть? А так, как происходит обмен содержимого двух чашек, в одной из которых находится молоко, а в другой — чай. Нужна третья чашка![1] То есть в нашей задаче для решения требуется третья, вспомогательная, переменная. С её использованием обмен может быть проведен следующим образом:

```
c := a |Запоминаем значение величины a
a := b |Величине a присваиваем значение величины b
b := c |Величине b присваиваем "старое" значение величины a
```

или

```
c := b
b := a
a := c
```

Но, оказывается, обменять значения двух величин можно и без использования третьей величины. «Не может быть!» — скажете вы. Может, и сделать это можно даже несколькими способами. Вот один из них.

Итак, пусть значение переменной *a* равно 5, переменной *b* — 3. Смоделируем место в памяти, где хранятся значения переменных, в виде прямоугольников, рядом с которыми запишем имена величин:

Сложим значения переменных и присвоим результат переменной *a*:

```
a := a + b
```

Новая ситуация со значениями переменных будет такой:

Как теперь получить значение *b*, равное исходному значению переменной *a*? Ответ такой:

```
b := a - b
```

а новая ситуация со значениями переменных:

Для полного решения задачи осталось записать один оператор:

```
a := a - b
```

Результат:

Красиво, не правда ли? С молоком так не сделаешь! ☺. Остальные способы решения, а их по крайней мере ещё 7, найдите самостоятельно. А заодно решите следующую задачу: «Даны значения трёх переменных величин *a*, *b* и *c*. Составить программу, после выполнения которой величина *b* будет иметь заданное значение величины *a*, величина *c* — значение величины *b*, величина *a* — значение величины *c*. Дополнительные величины не применять». Интересно, сколько способов решения вы найдёте…

---

[1] Можно, конечно, использовать и две дополнительные чашки, но это уже, так сказать, «слишком».

Об условном операторе

Во всех современных языках программирования предусмотрены две разновидности так называемого «условного оператора»:

1) полный условный оператор;
2) неполный условный оператор.

Рассмотрим их особенности

1. Полный условный оператор

Этот оператор используется в случаях, когда в программе возможны два варианта действий, а выбор того или иного варианта зависит от некоторого условия.

Пример. Дано целое число. Определить, является ли оно чётным.

Решение

Начальная часть программы решения задачи:

```
алг Проверка_на_чётность
нач цел n
вывод нс, "Введите целое число "
ввод n
```

Далее возможны два варианта действий (два варианта оператора вывода информации на экран):

1)
```
вывод нс, "Это число чётное"
```
2)
```
вывод нс, "Это число нечётное"
```

Значит, необходимо использовать полный условный оператор.

В школьном алгоритмическом языке общий вид этого оператора следующий[2]:

```
если <условие>
  то
    <Действия 1-го варианта (1-я серия операторов)>
  иначе
    <Действия 2-го варианта (2-я серия операторов)>
все
```

В наиболее распространённом случае <условие> — это два арифметических выражения, между которыми записан знак сравнения (отношения): =, >, <, >=, <=, <>. Возможно также использование сложных условий (с логическими операциями конъюнкции, дизъюнкции и др.).

Схема работы полного условного оператора показана на *рис*. 1[3].

---

[2] С правилами оформления различных видов условного оператора на языке программирования, который вы изучаете, ознакомьтесь самостоятельно.

[3] Приведенную алгоритмическую конструкцию называют — «развилка» или «ветвление».

Рис. 1

Применительно к рассмотренному примеру соответствующий оператор оформляется так:

```
если mod(n, 2) = 0
  то
    вывод нс, "Это число чётное"
  иначе
    вывод нс, "Это число нечётное"
все
```

где *mod* — функция школьного алгоритмического языка, возвращающая остаток от деления своего первого аргумента на второй (в других языках программирования для этого используются не функция, а специальная операция).

Возникает вопрос: «А можем ли мы считать первым вариантом действий вывод на экран сообщения о том, что заданное число — нечетное?». Конечно, можем:

```
если <условие>
  то
    вывод нс, "Это число нечётное"
  иначе
    вывод нс, "Это число чётное"
все
```

Но при этом `<условие>` должно быть уточнено — оно должно быть истинным для случая выполнения в программе действий первого варианта (см. *рис.* 1), то есть таким:

```
mod(n, 2) = 1
```

Самостоятельно убедитесь в том, что для обоих способов оформления оператора результат, согласно схеме на *рис.* 1, будет одним и тем же как для чётных, так и для нечётных значений числа *n*.

Задания для самостоятельной работы

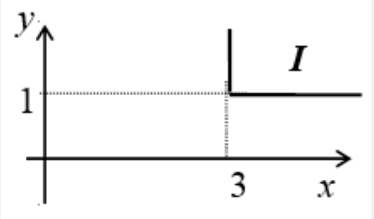
1. Даны два различных вещественных числа. Определить:

а) какое из них больше;

б) какое из них меньше.

2. Даны радиус круга и сторона квадрата. У какой фигуры площадь больше?

3. Определить, попадает ли точка с заданными координатами в область *I* (для простоты принять, что координаты точки не равны соответствующим границам этой области).



2. Неполный условный оператор

Эта разновидность условного оператора используется в случаях, когда какие-то действия в программе выполняются не всегда, а только при определённом условии.

Пример. Дано три целых числа, среди которых есть отрицательные. Вывести на экран отрицательные числа.

Начальная часть программы решения задачи:

```
алг Вывод_отрицательных_чисел
нач цел a, b, c
вывод нс, "Введите 3 целых числа (как минимум одно - отрицательное)"
ввод a, b, c
вывод нс, "Среди них отрицательные: "
```

Далее — в каком случае выводить значение первого числа *a*? Только если $a < 0$, то есть необходимо применить неполный условный оператор.

Общий вид этого оператора:

```
если <условие>
  то
    <Действия (серия операторов)>
все
```

Обратим внимание на то, что нет альтернативных вариантов действий, как в случае полного условного варианта, а есть единственно возможный, но не обязательный, вариант.

Применительно к рассмотренному примеру завершающая часть программы решения задачи оформляется так:

```
если a < 0
  то
    вывод a, " "
все
если b < 0
  то
    вывод b, " "
все
если c < 0
  то
    вывод c
все
```

Схема работы неполного условного оператора показана на *рис*. 2[4].

Рис. 2

[4] Приведенную алгоритмическую конструкцию называют — «обход».

Задания для самостоятельной работы

4. Даны три целых числа. Вывести на экран те из них, которые являются чётными.

5. Дано вещественное число. Вывести на экран его абсолютную величину (условно принимая, что соответствующей стандартной функции нет). Полный условный оператор не использовать.

6. Даны три вещественных числа. Вывести на экран те из них, которые принадлежат интервалу 1.6..3.8, включая его границы.

7. Даны четыре целых числа. Определить, сколько из них чётных.

Об операторах цикла

Как вы, очевидно, знаете, операторами цикла называют операторы, обеспечивающие повторение одних и тех же действий (поэтому их иногда называют также операторами повторений). В большинстве современных языков программирования существуют несколько разновидностей операторов цикла. Опишем некоторые особенности этих операторов.

1. Оператор цикла с параметром (со счётчиком)

Он применяется в тех случаях, когда в программе какие-то действия (операторы) повторяются и при этом некоторая величина меняется с постоянным шагом.

Пример. Для вывода «столбиком» всех целых чисел от 10 до 20 без использования оператора цикла в программе потребовалось бы записать:

```
вывод нс, 10
вывод нс, 11
…
вывод нс, 20
```

Видно, что есть повторяющиеся действия (вывод на экран числа) и что при этом выводимое число меняется с шагом, равным 1. Можно применить оператор цикла с параметром.

Общий вид этого оператора[5]:

```
нц для <параметр цикла> от <начальное значение> до <конечное значение>
  < тело оператора цикла>
кц
```

где <параметр цикла> — имя величины, являющейся параметром цикла (величины, меняющейся при повторении действий);

<начальное значение> — начальное значение этой величины;

<конечное значение> — то же, конечное; предполагается, что <конечное значение> больше <начальное значение> и что изменение от <конечное значение> до <начальное значение> происходит с шагом, равным 1;

<тело оператора цикла> — операторы, повторяющиеся при работе оператора цикла.

В рассмотренном примере параметром цикла является выводимое число, которое меняется от 10 до 20, а телом оператора цикла является один оператор — который выводит на экран значения числа, то есть оператор цикла оформляется следующим образом:

```
нц для i от 10 до 20  |i – имя величины - параметра цикла
  вывод нс, i
кц
```

---

[5] С правилами оформления этого и других операторов цикла на языке программирования, который вы изучаете, ознакомьтесь самостоятельно.

Схема, иллюстрирующая работу оператора цикла с параметром в общем случае, показана на *рис*. 3.

На схеме буквой *a* обозначено имя величины — параметра цикла, [н.з.] и [к.з.] — соответственно, начальное и конечное значения параметра.

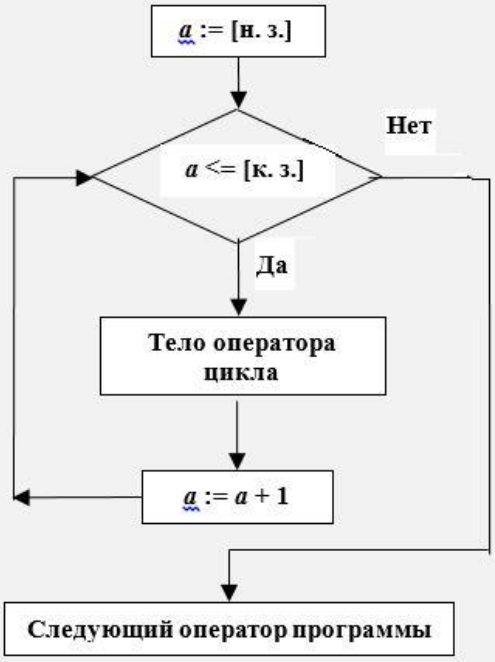


Рис. 3

Возникает вопрос: какое значение имеет величина — параметр цикла после окончания работы оператора? Ответы здесь разные для различных систем программирования. Например, в программе на Бейсике оно равно *a* + 1, в программах на Паскале и на школьном алгоритмическом языке — *a*[6].

В некоторых языках программирования предусмотрена возможность изменения значения величины — параметра цикла с шагом, отличным от 1. Например, в языке Паскаль шаг может быть равен –1, в языке Бейсик — любому числу, в том числе вещественному. В таких случаях при оформлении оператора указывается значение шага[7], а приведенная схема несколько меняется.

Задания для самостоятельной работы

8. Нарисуйте схему работы оператора цикла с параметром, значение которого меняется:

а) от большего начального значения до меньшего конечного с шагом, равным –1;

б) от меньшего начального значения до большего конечного с шагом, равным 2;

в) от большего начального значения до меньшего конечного с шагом, равным –3.

9. Ответьте, пожалуйста, на вопросы:

1) может ли тело оператора цикла с параметром не выполниться ни разу?

2) сколько раз будет выполняться тело оператора цикла с параметром при начальном (*a*) и конечном (*b*) значениях параметра цикла, равных:

— *a* = 1; *b* = 10;

---

[6] В первых версиях системы программирования КуМир, использующей школьный алгоритмический язык, значение величины — параметра цикла после окончания работы оператора вообще считалось неопределённым (не кажется ли вам, уважаемый читатель, что в этом есть определённая логика — параметр цикла нужен только для работы оператора, когда выполняются действия при различных его значениях).

[7] В языке программирования Паскаль при шаге, равном –1, в операторе вместо служебного слова **`To`** должно быть записано **`Downto`**.

— $a = 10$; $b = 20$;

— $a = N$; $b = M$ ($M \geq N$),

если во всех случаях шаг изменения величины-параметра равен 1?

3) сколько раз будет выполняться тело оператора цикла с параметром, если начальное значение параметра $a = N$, конечное значение $b = M$ ($M \geq N$), а шаг изменения равен s ($s \neq 1$)? (Конечно, в случае, если значение $s \neq 1$ допускается в языке программирования.)

Самостоятельно установите также, можно ли в языке программирования, который вы используете, изменять значение параметра в теле оператора цикла, а также определите значение параметра после окончания работы оператора. Это важные вопросы, ответы на которые помогут вам избежать ошибок при разработке программ. Заметим, что изменение параметра, а также его начального и конечного значений в теле оператора цикла может привести к непредсказуемым последствиям и считается плохим стилем программирования.

Задания на разработку программ

10. Напечатать «столбиком» квадраты всех целых чисел от 10 до b (значение b вводится с клавиатуры; $b \geq 10$);

11. Напечатать таблицу соответствия между массой в фунтах и массой в килограммах для значений 1, 2, ..., 10 фунтов (1 фунт = 453 г).

12. Напечатать таблицу умножения на 7:

```
1 x 7 = 7
2 x 7 = 14
...
9 x 7 = 63
```

Частным случаем оператора цикла с параметром является вариант, при котором величина — параметр цикла в теле оператора не используется. Например, если требуется, например, повторить какие-то действия 10 раз, то соответствующий фрагмент может быть записан (с учётом схемы на *рис*. 1 и анализа результатов ответов на вопросы в пункте 2 задания 2 для самостоятельной работы) как:

```
нц для i от 1 до 10
  …
кц
```

или

```
нц для i от 2 до 11
  …
кц
```

или другим подобным способом. Примером такой задачи является задача обработки последовательности чисел (определение суммы всех чисел последовательности, поиска максимального/минимального значения и его порядкового номера, нахождения количества чисел, обладающих некоторыми свойствами, и т.п.). В ней для ввода значений чисел последовательности может быть использован следующий фрагмент программы:

```
нц для i от 1 до n |n — общее количество чисел в последовательности
  вывод нс, "Задайте очередное число последовательности "
  ввод a    |Очередное число
  …         |Обработка заданного значения числа a
кц
```

Видно, что величина i в теле оператора не используется — при выполнении оператора она играет роль счётчика числа повторений (помните второе название обсуждаемого оператора?).

Задание для самостоятельной работы

13. Измените приведенный фрагмент таким образом, чтобы на экране при вводе очередных значений последовательно появлялись следующие сообщения (при $n = 20$):

```
Задайте 1-е число последовательности
Задайте 2-е число последовательности
…
Задайте 20-е число последовательности
```
— что, конечно, удобнее приведенного ранее варианта.

Задания на разработку программ

14. Напечатать ряд чисел 20 в виде:
20 20 20 20 20 20 20 20 20 20.

15. Составить программу вывода любого числа любое заданное количество раз в виде, аналогичном показанному в предыдущем задании.

16. Последовательность Фибоначчи образуется так: первый и второй члены последовательности равны 1, каждый следующий равен сумме двух предыдущих (1, 1, 2, 3, 5, 8, 13, ...). Составить программу для нахождения *k*-го члена последовательности Фибоначчи.

Оператор цикла с параметром может быть использован в программе для обеспечения некоторой паузы, если в его теле не приводить никаких операторов (применить так называемый «пустой цикл»). Например:

```
нц для i от 1 до 2000

кц
```

Продолжительность паузы будет зависеть от быстродействия компьютера и от конечного значения параметра цикла.

Задание для самостоятельной работы

17. Подберите на своем компьютере такое конечное значение параметра «пустого цикла», при котором в программе будет пауза длительностью примерно 5 сек.

2. Оператор цикла с условием

Этот оператор используется в тех случаях, когда число повторений действий в программе неизвестно. Приведем пример задачи.

Задача 1. Дано натуральное число. Разработать программу для определения суммы его цифр.

Для двузначного числа программа решения задачи имеет вид:

```
алг Сумма_цифр_двузначного_числа
нач цел число, перв, вт, сумма
  |Ввод числа
  вывод нс, "Задайте 2-значное число "
  ввод число
  |Определение его 1-й цифры
  перв := div(число, 10)
  |Определение его 2-й цифры
  вт := mod(число, 10)
  |Расчёт суммы цифр
  сумма := перв + вт
  |Вывод результата
  вывод нс, "Сумма его цифр равна ", сумма
кон
```

где *div* — функция школьного алгоритмического языка, возвращающая целочисленное частное от деления своего первого аргумента на второй (в других языках программирования для этого используется не функция, а специальная операция).

Один из возможных вариантов программы решения задачи для трёхзначного числа такой:

```
алг Сумма_цифр_трёхзначного_числа
нач цел число, перв, вт, тр, сумма
  |Ввод числа
  вывод нс, "Задайте 3-значное число "
  ввод число
  |Определение его 1-й цифры
  перв := div(число, 100)
  |Определение его 2-й цифры
  вт := div(mod(число, 100), 10)
  |или
  |вт := mod(div(число, 10), 10)
  |Определение его 3-й цифры
  тр := mod(число, 10)
  |Расчёт суммы цифр
  сумма := перв + вт + тр
  |Вывод результата
  вывод нс, "Сумма его цифр равна ", сумма
кон
```

Программу применительно к четырёхзначному числу разработайте самостоятельно.

А как решить задачу для числа неизвестной «значности»”?

Для ответа сначала предлагаем подумать над другим вопросом — какую цифру числа неизвестной «значности» (см. табличку ниже) определить можно?

1-я	*2-я*	*3-я*	*...*	*предпосл*	*посл*

цифры числа

Ответ — последнюю (последняя цифра любого целого числа равна остатку от деления этого числа на 10 — убедитесь в этом!):

```
посл := mod(число, 10)
```

А остальные цифры? Как, например, определить предпоследнюю цифру? Её можно найти только так — получить число без последней цифры исходного числа (разделив исходное нацело на 10) и для него определить последнюю цифру.

Аналогично можно получить и предпредпоследнюю и остальные цифры. После нахождения каждой цифры её надо сложить с суммой уже найденных до этого цифр.

Следовательно, для решения задачи в программе надо многократно выполнить следующие действия:

```
|Определить последнюю цифру числа
посл := mod(число, 100)
|Учесть её в найденной ранее сумме
сумма:= сумма + посл
|Получить новое число
число := div(число, 10)
```

Например, для числа 30 625 необходимо эти действия повторить 5 раз:

Шаг	**Последняя цифра**	**Сумма “обработанных” цифр**	**Новое значение числа**
1	5	5	3062
2	2	5 + 2 = 7	306
3	6	7 + 6 =13	30
4	0	13 + 0 = 13	3
5	3	13 + 3 = 16	0

то есть фрагмент программы подсчёта суммы цифр пятизначного числа можно оформить с использованием оператора цикла с параметром:

```
сумма := 0 |Начальное значение суммы цифр
           |В программе на школьном алгоритмическом языке
           |такая инициализация является обязательной
нц для i от 1 до 5
  посл := mod(число, 10)
  сумма := сумма + посл
  число := div(число, 10)
кц
```

Но в нашей «основной» задаче количество цифр числа неизвестно, то есть неизвестно количество повторений тела цикла. Именно в таких случаях и применяется оператор цикла с условием.

Возможны две разновидности этого оператора:
1) оператор цикла с предусловием;
2) оператор цикла с предусловием.

Рассмотрим их особенности.

2.1. Оператор цикла с предусловием

Оператор цикла с предусловием применяется тогда, когда в программе известно условие, при котором действия должны повторяться. Его общий вид:

```
нц пока <условие>
  <тело оператора цикла>
кц
```

где <условие> — условие, при котором выполняется <тело оператора цикла>.

Схема, иллюстрирующая работу этого оператора:

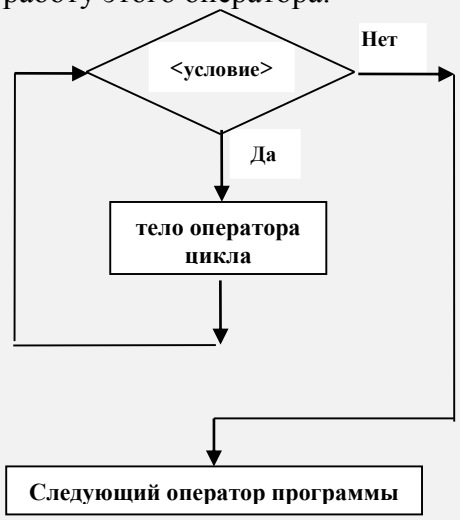


Рис. 4

Приведем полезное правило, с помощью которого можно установить условие, записываемое в таком операторе:

1) необходимо определить условие, при котором *нельзя* или *не нужно* выполнять повторяющиеся действия;

2) записать в операторе условие, противоположное найденному на предыдущем этапе.

Применительно к задаче определения суммы цифр любого натурального числа оператор цикла с предусловием оформляется следующим образом:

```
нц пока ?
  посл := mod(n, 10)
  …
кц
```

Чтобы определить условие, обозначенное символом «?», вспомним приведенное чуть выше правило. Мы должны прекратить вычисления, когда *число* = 0 (см. выше табличку для числа 30 625), значит, условие, записываемое после служебного слова **пока**, должно быть таким: *число* > 0 (отрицательным значение переменной величины *число* быть не может). Следовательно, можем оформить фрагмент так:

```
сумма := 0
нц пока число > 0
  посл := mod(число, 10)
  сумма := сумма + посл
  число := div(число, 10)
кц
```

Конечно, в данном случае условие, записываемое в программе после слова **пока**, можно было определить и не используя приведенное правило. В ряде случаев искомое условие не так очевидно. Вот пример задачи.

Задача 2. Даны два натуральных числа. Разработать программу для определения наибольшего общего делителя (НОД) заданных чисел.

Решение

Найти НОД двух натуральных чисел можно разными способами [1]. Лучшим из них является использование алгоритма Евклида, названного так в честь его автора — древнегреческого математика III века до нашей эры. Суть алгоритма можно изобразить схемой:

$$\text{НОД}(a, b) \begin{cases} = \text{НОД}(a - b, b) \text{ при } a > b \\ = \text{НОД}(a, b - a) \text{ при } b > a \\ = a \text{ (или } = b) \text{ при } a = b \end{cases}$$

Например, согласно схеме:

НОД(26, 18) = НОД(8, 18) = НОД(8, 10) = НОД(8, 2) = НОД(6, 2) = НОД(4, 2) = НОД (2, 2) = 2.

Анализ показывает, что для нахождения НОД нужно многократно вычитать из большего из двух чисел меньшее, причём количество повторений — неизвестно, то есть следует применить оператор цикла с условием, в частности — с предусловием:

```
алг Определение_НОД
нач цел a, b, НОД
  |Ввод чисел
  …
  нц пока ?
    если a > b
      то
        a := a - b
      иначе
        b := b - a
    все
  кц
  НОД := a
  |Вывод результата
  …
кон
```

Чтобы определить условие, обозначенное символом «?», также вспомним приведенное выше правило. Когда мы должны прекратить действия? Согласно алгоритму Евклида, — когда *a* = *b*. Значит, вместо вопросительного знака должно быть записано условие *a* <> *b*.

Задания для самостоятельной работы

17. Подумайте, чем в последней программе можно заменить многократные вычитания меньшего числа из большего, например, когда *a* = 348, *b* = 6.

18. Подумайте, пожалуйста, над вопросами:

1) может ли тело оператора цикла с предусловием не выполниться ни разу?

2) может ли тело оператора цикла с предусловием выполняться бесконечно (или до того момента, когда пользователь прервёт выполнение одновременным нажатием клавиш <**Ctrl**> и <**Break**>)?

Задания на разработку программ

19. Дано натуральное число. Определить количество цифр в нём.

20. Вывести на экран все натуральные числа, не превышающие заданное число *n*.

21. Дано число *n*. Из чисел 1, 4, 9, 16, 25, ... напечатать те, которые не превышают *n*.

Оператор цикла с предусловием также может быть использован в программе для обеспечения некоторой паузы в её работе. Составьте соответствующий вариант программы самостоятельно.

Обращаем внимание на одну особенность работы оператора цикла с предусловием. Как следует из *рис*. 4, условие проверяется только *перед* выполнением тела оператора, но не проверяется *в процессе* выполнения. Неучёт этого обстоятельства может привести к ошибкам. Очень наглядно это продемонстрировано в учебнике [2] на примере задачи с исполнителем Робот.

Задача такая. Робот, находящийся на поле из клеток и умеющий перемещаться на одну клетку влево, вправо, вверх или вниз, находится в клетке **А** (*рис*. 5) и выполняет фрагмент программы:

```
нц пока снизу свободно
  вниз
  вниз
кц
```

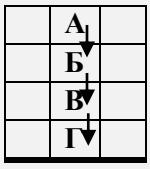


Рис. 5

Проследим работу оператора согласно схеме на *рис*. 4.

1. Исходное положение — Робот в клетке **А**. Проверяется условие `снизу свободно` — оно истинно, то есть выполнится тело оператора, и Робот, сместившись дважды вниз, окажется в клетке **В**.

2. Опять проверяется условие `снизу свободно` — оно также истинно, и начнется выполнение тела оператора — Робот сначала сместится в клетку **Г**, но вторая команда `вниз` выполниться не сможет (появится сообщение об ошибке).

Конечно, Робот — условный исполнитель. Приведем и реальную задачу. Пусть требуется вывести на экран числа в следующем порядке:

21 20
19 18
17 16
…
11

Казалось бы, можем записать в программе:

```
a := 21
нц пока a >= 11
  вывод нс, a, " ", a - 1
  a := a - 2
кц
```

Однако в результате выполнения этого фрагмента на экран будет выведено следующее (также согласно схеме на *рис*. 4):

```
21 20
19 18
17 16
…
11 10
```

что не соответствует условию.

Ещё более реальный пример приведен в [3]. В этой статье представлена программа, моделирующая игру двух участников, каждый из которых берёт некоторое количество спичек из одной из двух кучек. В программе должен быть фрагмент, схема выполнения которого является следующей:

Повторение пока игра не кончилась

Ход первого участника

Вывод новой позиции

Ход второго участника

Вывод новой позиции

конец повторений

Казалось бы, все правильно — участники по очереди делают ходы до тех пор, пока игра не закончилась (пока есть хотя бы одна спичка). Однако при таком оформлении программы, согласно схеме, если после хода первого участника игра закончится, блок `Ход второго участника` всё равно будет выполняться (на экран будут выводиться вопросы, связанные с ходом второго игрока), хотя условие окончания работы оператора соблюдается.

Аналогичные ситуации могут иметь место в других компьютерных программах, моделирующих игры двух и более участников, и в ряде других случаев. Помните об этом!

Задание для самостоятельной работы

22. Подумайте, как нужно изменить приведенную блок-схему для того, чтобы в описанной ситуации операторы блока `Ход второго участника` и следующего за ним блока `Вывод новой позиции` не выполнялись.

Примечание. В языке программирования QuickBasic имеется также оператор цикла с предусловием, в котором указывается условие окончания работы оператора (**DO UNTIL** <условие> … **LOOP**).

3. Оператор цикла с постусловием

Он также применяется тогда, когда число повторений действий в программе неизвестно. Общий вид этого оператора:

```
нц
  <тело оператора цикла>
кц при <условие>
```

где <условие> — условие, при котором оператор прекращает свою работу[8].

Например, применительно к задаче о сумме цифр (см. выше) такой оператор оформляется следующим образом:

```
сумма := 0
нц
  посл := mod(число, 10)
  сумма := сумма + посл
  число := div(число, 10)
кц_при число = 0
```

Схема, иллюстрирующая работу оператора цикла с постусловием, показана на *рис*. 6.

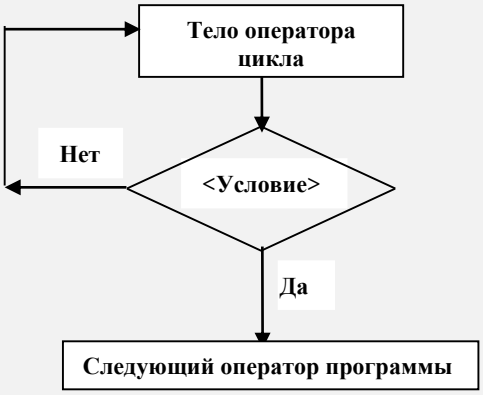


Рис. 6

Примечание. В языке программирования QuickBasic имеется также оператор цикла с постусловием, в котором указывается условие продолжения работы оператора (**DO** … **LOOP WHILE** <условие>).

Задание для самостоятельной работы

23. Подумайте, пожалуйста, над вопросами:

1) может ли тело оператора цикла с постусловием выполняться бесконечно (или до того момента, когда пользователь прервёт выполнение одновременным нажатием клавиш <**Ctrl**> и <**Break**>)?

2) может ли тело оператора цикла с постусловием не выполниться ни разу?

Из ответа на последний вопрос следует, что оператор цикла с постусловием можно использовать для проверки правильности вводимых в программу значений. Действия при этом обязательно выполняются один раз и в случае ввода ошибочного значения повторяются. Например, следующий вариант обеспечивает проверку правильности введённого значения коэффициента a в программе нахождения корней квадратного уравнения $ax^2 + bx + c = 0$ ($a \neq 0$):

---

[8] Сравните это условие с условием, указываемым в операторе цикла с предусловием.

```
нц
  вывод нс, "Задайте значение коэффициента a уравнения "
  ввод a
  если a = 0
    то
      вывод нс, "Коэффициент a не должен быть равен нулю!"
  все
кц при a <> 0
```

Задания для самостоятельной работы

24. Подготовьте фрагмент программы, в которой пользователь в ответ на вопрос: «Чёт (введите 2) или нечет (введите 1)?» должен ввести одно из двух значений — 1 или 2. В случае ввода другого числа на экран должно выводиться сообщение об ошибке, после чего действия должны повторяться до ввода правильного значения.

25. Подготовьте фрагмент программы, в которой пользователь должен ввести установленный пароль. В случае ввода неправильного пароля на экран должно выводиться сообщение об ошибке, после чего действия должны повторяться до ввода правильного значения. После этого на экран должно выводиться некоторое приветствие.

Ясно, что и рассматриваемая разновидность оператора цикла может быть применена в программе для обеспечения некоторой паузы в её работе. Здесь же заметим, что в большинстве современных языков программирования оператор цикла с постусловием используется для приостановки программы до нажатия любой клавиши. Например, в языке Паскаль:

```
Repeat
Until Keypressed;
```

где `Keypressed` — функция, возвращающая значение логического типа, указывающая, была ли нажата любая клавиша или нет, в языке Бейсик (вариант QuickBasic):

```
DO
LOOP UNTIL INKEY$ <> ""
```

где `INKEY$` — функция, возвращающая символ, считанный с клавиатуры; если символ не считан (нажатия клавиш нет), то возвращается «пустой» символ (""). Функция не дублирует ввод с клавиатуры выводом символа на экран (нажатый символ не выводится).

Преобразование одного вида оператора цикла в другой

Часто в программах необходимо заменить одну разновидность оператора цикла на другую. Обсудим, всегда ли это возможно.

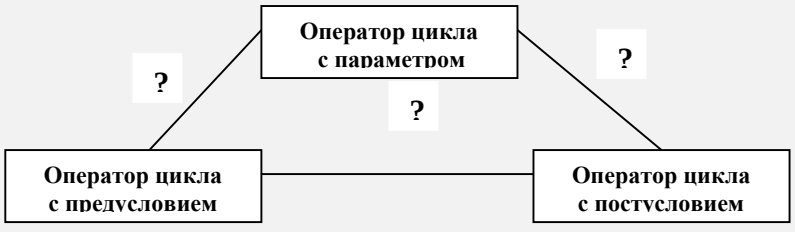


Рис. 7

Так как, зная условие продолжения работы оператора цикла с предусловием, можно определить условие окончания работы оператора цикла с постусловием и наоборот (каждое из этих условий противоположно другому — см., например, задачу о сумме цифр выше), то всегда одну из этих разновидностей оператора можно заменить на другую.

Более сложным является вопрос о возможности замены оператора цикла с параметром. Если проанализировать схему на *рис*. 3, то нетрудно увидеть, что для него:

1) известно значение величины *a* до начала выполнения тела оператора;

2) величины *a* увеличивается с шагом, равным 1;

3) условием окончания работы оператора является *a* `> [конечное значение].`

Это означает, что, имея оператор цикла с параметром, всегда можно заменить его на оператор цикла с постусловием (и, следовательно, — на оператор цикла с предусловием). Например, приведенный выше фрагмент:

```
нц для i от 10 до 20
   вывод нс, i
кц
```

можно оформить в виде оператора цикла с постусловием:

```
i := 10
нц
  вывод нс, i
  i := i + 1
кц при i > 20
```

или с предусловием:

```
i := 10
нц пока i <= 20
  вывод нс, i
  i := i + 1
кц
```

А наоборот? Вспомним задачу о сумме цифр некоторого натурального числа:

```
сумма := 0
нц
  посл := mod(число, 10)
  сумма := сумма + посл
  число := div(число, 10)
кц_при пока число = 0
```

Можем ли мы определить начальное и конечное значения величины *число* и шаг её изменения и тем самым применить оператор цикла с параметром? В данном случае — нет, не можем. Иногда это возможно (см., например, приведенные чуть выше операторы цикла с постусловием и с предусловием).

Итак, схему, иллюстрирующую возможность замены одной разновидности оператора цикла на другую (см. *рис*. 7), можно представить в виде:

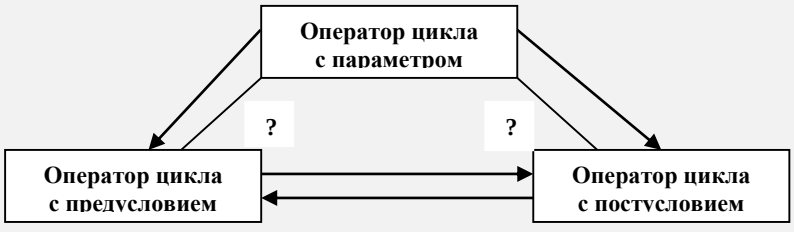


Рис. 8

Вывод: в некоторых случаях для реализации повторяющихся действий в программе может быть использована любая из трёх разновидностей оператора цикла, в некоторых — только две разновидности: операторы цикла с предусловием и с постусловием.

Рассмотрим ещё одну задачу, связанную с заменой одной разновидности оператора цикла на другую: «Дан массив целых чисел. Определить, имеется ли в массиве число 13».

Можно, конечно, подсчитать количество чисел в массиве, равных 13, и по этому значению получить ответ на вопрос в условии задачи:

```
k := 0
нц для i от 1 до n
  если m[i] = 13
    то
      k := k + 1
  все
кц
если k > 0
  то
    вывод нс, "Число 13 в массиве есть"
  иначе
    вывод нс, "Числа 13 в массиве нет"
все
```

Однако приведенный вариант является нерациональным — в нём происходит проверка всех *n* элементов массива, в то время как может оказаться, что число 13 записано уже, например, во втором элементе. Желательно прекратить проверки, как только встретится это число. Так как количество проверок до такого момента неизвестно, применим оператор цикла с предусловием. Условие, записываемое в нём, определим, используя правило, описанное при рассмотрении этого оператора. Будем рассуждать так. Проверки значений элементов массива надо прекратить, как только встретится число 13 или когда массив будет исчерпан. Соответствующее условие записывается в виде:

```
есть13 или i > n
```

где *есть*13 — величина логического типа, принимающая истинное значение, если число 13 встретится в массиве; *i* — индекс очередного проверяемого элемента массива.

Тогда, согласно правилу, условие в операторе цикла с предусловием должно быть противоположным:

```
не есть13 и i <= n
```

а весь фрагмент решения задачи оформляется следующим образом:

```
|Начальные значения величин i и есть13,
|обеспечивающие начало работы
|оператора цикла с предусловием
i := 1; есть13 := нет
нц пока не есть13 и i <= n
  если m[i] = 13
    то
      есть13 := да
    иначе |Переходим к рассмотрению следующего элемента
      i := i + 1
  все
кц
если есть13[9]
  то
    вывод нс, "Число 13 в массиве есть"
  иначе
    вывод нс, "Числа 13 в массиве нет"
все
```

Задание для самостоятельной работы

26. Используя схему на *рис*. 4, убедитесь, что применительно к массиву [21 6 13 7 77 12 0 120 12 15] работа оператора прекратится после проверки третьего элемента массива, а для массива [21 6 15 7 77 12 0 120 12 15], в котором число 13 отсутствует, результат также будет правильным.

---

[9] Обращаем внимание на условие, используемое в операторе.

Задания на разработку программ

27. Имеется фрагмент программы в виде оператора цикла с параметром, обеспечивающий вывод на экран «столбиком» всех целых чисел от 100 до 80. Оформить этот фрагмент в виде:

а) оператора цикла с предусловием;

б) оператора цикла с постусловием.

28. Известны оценки по информатике каждого из 20 учеников класса. В начале списка перечислены все пятёрки, затем все остальные оценки. Определить, сколько учеников имеют по информатике оценку «5»? Условный оператор и оператор цикла с предусловием не использовать. Рассмотреть два случая:

1) известно, что пятёрки имеют не все ученики класса;

2) допускается, что пятёрки могут иметь все ученики класса.

29. Начав тренировки, лыжник в первый день пробежал 10 км. Каждый следующий день он увеличивал пробег на 10% от пробега предыдущего дня. Определить, в какой день он пробежит больше 20 км.

Ответы и программы на языке программирования, который вы изучаете, пожалуйста, присылайте в редакцию по адресу *mirinfo@infojournal.ru*. Фамилии всех приславших ответы будут опубликованы, а авторы лучших работ будут награждены дипломами. Можно выполнять не все задания. Срок представления ответов не ограничен.

Литература

1. *Окулов С.М., Лялин А.В.* Алгоритмы поиска наибольшего общего делителя // «Информатика и образование», № 7/2011.

2. *Кушниренко А.Г., Лебедев А.Г., Зайдельман Я.Н.* Информатика 7–9: Учебник для общеобразовательных учебных заведений. М.: Дрофа, 2000.

Это полезно знать

Ученые обнаружили аномалии в структуре мозга у геймеров

В рамках наблюдений за работой мозга у 200 подростков, увлекающихся компьютерными играми, удалось раскрыть аномалии в структуре некоторых регионов мозга, которые отвечают за реакцию на раздражители. Таким людям свойственно импульсивное поведение, заставляющее в процессе игры забыть о сне, учёбе и еде.

По словам Джеффри Андерсона из университета Юты в Солт-Лейк-Сити, США, в основном такие изменения считаются благотворными. Однако бывают случаи, когда полезные изменения нельзя отделить от тех проблем, причинами которых они становятся.

За последние 20 лет в обществе появилась небольшая группа людей, страдающих чрезмерным увлечением компьютерными играми. Очень часто такие геймеры попадают в больницы или даже морги по причине истощения организма. Андерсон решил вместе со своими коллегами проверить, имеются ли какие-то нейрофизиологические причины для развития игромании. В эксперименте участвовали 200 детей и подростков, страдающих игроманией. В рамках исследования ученые отследили работу их мозга при помощи магнитно-резонансного томографа.

Мозги всех игроманов, сообщает агентство РИА, обладали одной общей чертой: два региона их мозга (задняя часть передней коры и височно-теменной узел) были сильно связаны между собой и активнее обменивались информацией во время игры, нежели у здоровых людей. Первый регион отвечает за сбор информации об окружающем мире из органов чувств, а второй — за функционирование рабочей памяти, планирование действий и поиск новой информации в данных из височно-теменного узла. Чрезмерно сильная связь между этими регионами делает человека более импульсивным. Следы подобной активности, по мнению ученого, можно использовать для определения предрасположенности ребенка к игромании и дальнейшей его защиты от последствий чрезмерного увлечения компьютерными играми.

Крис Касперски

Когда произносят фамилию «Касперский», большинство опытных (и не очень) пользователей компьютеров вспоминают известного российского программиста, автора распространённой антивирусной программы Евгения Касперского. Однако есть ещё один, менее известный в широких кругах пользователей, программист, связанный с указанной фамилией. Правда, «Касперски» (именно так!) — это его псевдоним. Об этом человеке мы и хотим рассказать[10].

Крис Касперски (настоящее имя — Николай Владимирович Лихачев) родился в 1976 году в селе Успенское Краснодарского края. В 7 лет собрал свой первый работающий радиоприёмник. Первый компьютер — «Правец 8D»[11] — ему привез из командировки отец. Компьютер подключался к магнитофону — аудиокассеты заменяли ему жёсткий диск, а в качестве монитора выступал телевизор. Инструкция к компьютеру была на болгарском языке, но у знакомых смогли найти словарь.

На “Правец 8D” будущий Крис Касперски написал свою первую программу — простенькую игру на тему рыбалки. Нолик и знак «>» (0>) символизировали рыбку, которая плавала по экрану взад — вперед, а в центре был рыбак в виде знака вопроса. Чтобы поймать рыбку, нужно было нажимать на клавишу «Пробел».

Следующий компьютер был «Электроника БК 0010», на котором Крис освоил язык программирования ассемблер. Потом были «ZX Spectrum» и «Агат».

В школе Николай учился хорошо, закончил на одни пятёрки. Без экзаменов поступил в Таганрогский радиотехнический университет на специальность «Проектирование микроконтроллеров». Но, проучившись всего три месяца, бросил учёбу и вернулся в родное село, поскольку в институте «программировать толком не давали».

«Год я сидел дома и ничего не делал, потом снова поступил, чтобы успокоить маму, и снова бросил. Но тогда мне купили IBM, жёсткий диск на 20 мегабайт и цветной монитор, целых шестнадцать цветов», — рассказывает Крис.

Во время учёбы в Таганроге он познакомился с предприимчивым студентом Шуриком, с которым они на паях открыли бизнес — оказывали услуги системного администрирования. Однако через некоторое время компаньон скрылся с деньгами, а Крису пришлось расплачиваться с рэкетирами. Отдав оставшиеся деньги и компьютер, он снова вернулся в село к родителям.

Все дальнейшие попытки организовать свой компьютерный бизнес в Армавире, Краснодаре и Ростове-на-Дону также не увенчались успехом.

С 1990 года Крис активно писал в так называемых «эхо-конференциях» Фидо[12]. В частности, написал целую серию книг и статей, содержащих «рабочие» техники работы с дизассемблером и отладчиком.

В 1999 году знакомый свел Криса с издательством технической литературы «СОЛОН-ПРЕСС», где тиражом в десять тысяч экземпляров вышла его первая книга «Техника и философия хакерских атак». На момент написания книги ему было всего 22 года.

---

[10] Статья подготовлена по материалам сайта *polit.ru*.

[11] «Правец 8D» — персональный компьютер, выпускавшийся в Болгарии.

[12] Фидо́ (полное название — Фидоне́т (англ. FidoNet, от созвучного греческого слова, означающего «коротко») — международная любительская компьютерная сеть, популярная в начале 1990-х годов (в бывшем СССР — до конца 1990-х).

Следующие десять лет Касперски вел затворническую жизнь сельского чудака, занимаясь написанием книг и статей для множества журналов, таких как «Системный администратор», «Байт», «Хакер» и др.

К 2008 году было издано уже шестнадцать книг Криса Касперски на русском и английском языках — по защите информации и оптимизации программ, компьютерным вирусам, дизассемблированию.

Примерно в это время в жизни Касперски неожиданно случился, как он сам пишет, «взрывной каскад путешествий по всему миру» по приглашениям на различные конференции. Всего за несколько лет он посетил огромное количество стран, в том числе и экзотических. Вершиной этого стало громкое выступление на тему удаленного взлома процессора Intel на конференции в Малайзии. Эти поездки дали Касперски возможность завести важные профессиональные знакомства и заключить несколько значимых контрактов по всему миру.

В июне 2008 года у Касперски появилась, наконец, постоянная работа. Американская фирма «Endeavor Security», занимающаяся безопасностью компьютеров и сетей, взяла его независимым консультантом. В 2009 году «Endeavor Security» была приобретена компанией McAfee. В McAfee Касперски проработал до сентября 2012. В настоящее время работает экспертом по безопасности в фирме «Check Point Software Technologies».

С 2008 года живёт в США в «Мекке» программистов — г. Рестон и является обладателем редкой американской визы O1 для людей с выдающимися способностями. По некоторым данным, уже стал долларовым миллионером.

Чем знаменит

Крис Касперски — один из самых известных в России специалистов в области компьютерной безопасности. Из-под его пера вышли такие книги, как «Техника и философия хакерских атак», «Образ мышления дизассемблера IDA», «Техника сетевых атак» и многие другие книги и статьи по программированию, взлому и отладке программ.

Иногда в книжных аннотациях можно встретить фразы типа: «Имя Криса Касперски известно едва ли не каждому пользователю персонального компьютера в России и за рубежом, благодаря популярному антивирусному программному обеспечению». Однако Крис не имел и не имеет никакого отношения к «Лаборатории Касперского» и не является родственником его основателя и создателя известного антивируса Евгения Касперского. В свое время во избежание путаницы с Евгением Касперским, он даже удалил последнюю букву из своего псевдонима.

Его книги, помимо собственно технических аспектов, не лишены и весьма серьезной философской составляющей. Стиль Касперски сложно с кем-то спутать; в своих работах он не просто учит хакерским премудростям и рассказывает, как нужно взламывать и как защищаться от атак, но и задается вопросом — зачем это делать.

В 2008 году Крис Касперски обнаружил уязвимость в процессорах Intel, которая позволяет совершить удаленный взлом системы при помощи скрипта на JavaScript или TCP/IP-пакета вне зависимости от операционной системы. О существовании «дыр» в процессорах Intel специалисты догадывались давно, но Касперски первым доказал, что уже написаны программы, которые помогут пробираться через них в компьютер. «Я всего лишь доказал, что это не миф», — говорит сам Касперски.

О чем надо знать

Помимо компьютеров, ещё одним огромным увлечением Криса Касперски является астрономия. Ещё в детстве отец построил для него обсерваторию рядом с домом. «Помимо возни с железом, я не расстаюсь с миром звёзд и моих телескопов», — говорит сам Крис. В одном из своих интервью он даже заявил однажды, что не хотел бы жить в городе как раз потому, что там совсем не видно звезд из-за сильной засветки.

Известные астрономические сайты публикуют его статьи, посвященные отечественным телескопам и наблюдению звёздного неба. Даже самая первая из опубликованных Крисом

статей была посвящена астрономии и вышла в журнале «Звездочёт» в 1993 году, когда он ещё учился в школе.

В 2003 году вышла в свет книга Криса Касперски «Энциклопедия примет погоды. Предсказание погоды по местным признакам».

Прямая речь

О себе: «Небрежно одетый мыщъх, не обращающий внимание ни на мир, ни на тело, в котором живёт, и обитающий исключительно в дебрях машинных кодов и зарослях технических спецификаций».

Об отношениях с Евгением Касперским: «Я не знаю почему, но как только в лаборатории разговор заходит обо мне, у них сразу портится настроение. Теперь они распространяют слухи, что я у них три недели работал, они взяли меня смеха ради, а потом выгнали. Сейчас мы с Касперским в параллельных плоскостях находимся, я не мешаю его бизнесу, а он не мешает моему. Я даже специально убрал “й” из псевдонима, чтобы нас не путали. Но он меня всё равно не любит».

О своих пристрастиях: «Скорее я хиппи. Не вандализм, а исследование. Хакером теперь называют даже школьников. А я люблю слово “кодокопатель”. Тот, кто роет вглубь».

О деньгах: «Миллион — это очень мало на самом деле. Это верхне-средний класс. С другой стороны, одни интересные люди предлагали мне два “лимона”. Я спросил у них, что можно сегодня купить на эти деньги. Интересные люди неправильно меня поняли и, не торгуясь, предложили сразу двадцать, но были посланы, потому что на текущие расходы я и сам заработаю, а большего мне не надо».

О вирусных угрозах: «Самый последний писк моды — в прицел атаки попали встраиваемые устройства. В первую очередь это, конечно, роутеры. Зловредный код в роутере очень сложно обнаружить. А тем временем хакеры наши способ проникнуть внутрь камер наблюдения, подключённых к Ethernet. На очереди “умная” бытовая техника (например, холодильники), а также атаки на бортовой компьютер автомобиля — это фантастика новой реальности».

Игорь Садреев о Крисе Касперски: «Он говорит с заметным южным акцентом, иногда проглатывает целые предложения, коверкает слова и смягчает окончания глаголов. Маленького роста, худой — “кожа да кости” в данном случае не преувеличение».

Сотрудник фирмы «Endeavor» Элис Чанг об известности Криса Касперски: «Известность Касперски за пределами России не преувеличена. Он очень известный хакер, причём именно в первоначальном значении этого слова: человек, который понимает самые основы того, как работает программа».

8 фактов о Крисе Касперски

1. Когда Крису было всего несколько недель (точнее, тогда — Николаю), врач по ошибке сделал мальчику инъекцию хлористого кальция, и малыш перенес инсульт. Частично омертвели ткани мозга, что привело к легкому аутизму.

2. Псевдоним, из-за которого Криса постоянно путают с создателем антивирусных программ, появился от большой любви к мультфильму про привидение Каспера и к музыканту и композитору Крису Кельми: «Я никогда не слушал его музыку, но от самого фонетического сочетания можно с ума сойти».

3. За исключением своего дня рождения, Крис вообще не помнит дат, а хронологию событий восстанавливает по моделям компьютеров, которые у него были в разное время.

4. Любит произведения Фрэнка Герберта и часто использует их для эпиграфов. Также читает Виктора Пелевина, Юлию Остапенко и Ф.Ю. Зигеля, о которых сам говорит так: «Именно они, как никто другой, повлияли на мое творчество, сформировали стиль и манеру изложения, зажгли искру, определили жизнь и заставили взяться за перо».

5. Другой свой псевдоним — «мыщъх» — был придуман Крисом на основе романа «Дюна», главный герой которого носил имя Муад'Диб, что означает пустынную мышь. Почти все остальные ники Криса — это то же слово «мышь» на разных языках мира. Иногда специально чуть искажённые, например: souriz — от *souris* (франц.).

6. Первое собеседование Криса в США закончилось неожиданно — он отказался выполнять предложенное задание, найдя в нём ошибки и «нечёткую спецификацию». В итоге это собеседование окончилось предложением работы с окладом $300 000 в год — первой официальной работой в его жизни.

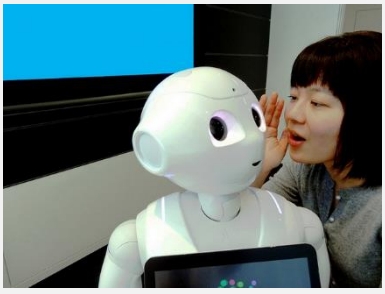
7. В первые годы жизни Касперски в Рестоне (США) к нему неожиданно нагрянуло с обыском ФБР, изъявшее все записи и носители. Никаких обвинений, однако, ему не предъявляли, а сама история закончилась внесудебным урегулированием с выплатой Крису «отступных в размере зарплаты сферического программиста из РФ за несколько лет». Однако свои записи и наработки Касперски пришлось довольно долго потом восстанавливать. С тех пор самые важные данные он хранит на винчестере в банковской ячейке.

8. В США Крис всерьез увлекся коллекционированием стрелкового оружия и спортивной стрельбой.

Цифровой мир

В Японии впервые в мире на учёбу в школу принят робот

13 апреля 2016 года впервые робот был принят на учёбу в школьное учреждение, где он будет учиться вместе с детьми. Это произошло в Японии, в средней школе японского города Васэда. Говорящий робот Pepper был создан специально для общения с людьми. Как утверждают разработчики, робот оснащён большим количеством камер и датчиков, которые позволяют ему распознавать практически весь спектр человеческих эмоций: радость, грусть, страх, волнение, раздражение. В зависимости от ситуации Pepper может смеяться в ответ на шутку или возьмётся утешать своего хозяина. Он также обладает способностью к самообучению.



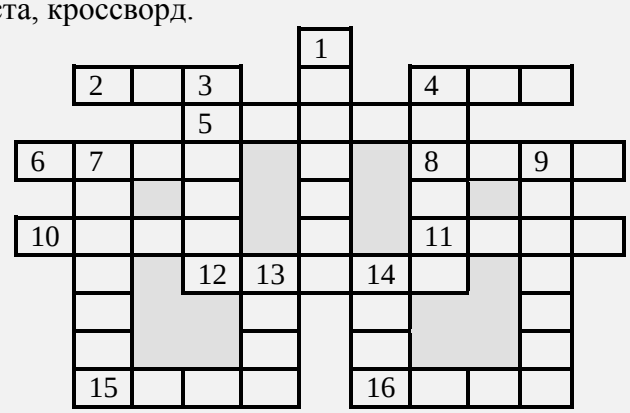
По случаю приёма робота в школе была устроена вступительная церемония, на которой учеников призвали максимально использовать уникальные шансы общения с андроидом, поскольку это им пригодится в будущей жизни. Сам Pepper заявил, что «никогда не думал, что будет принят в человеческую школу», и пообещал заниматься изо всех сил.

Отмечается, что устройство владеет не только японским, но и английским языком, поэтому его будут активно использовать на уроках иностранного.

Робот Pepper разработан японскими компаниями Aldebaran Robotics и Softbank Corp. Такие устройства предназначены пока что в основном для обслуживания клиентов в банках и в магазинах. В июне прошлого года первая партия из тысячи роботов Pepper по цене 214 тыс. иен (1,9 тыс. долл.) за штуку была раскуплена за одну минуту…

Кроссворд

Решите, пожалуйста, кроссворд.



По горизонтали

2. В программировании — одна из характеристик переменной величины.

4. Денежная единица, в которой получали зарплату болгарские программисты до введения в стране евро.

5. Величина, с помощью которой осуществляется счёт.

6. Компонент данных типа «запись».

8. Конечное число точек на плоскости, соединённых отрезками кривых линий.

10. Структура данных, в которых применен принцип «последним пришёл — первым вышел».

11. Семейство совместимых друг с другом компьютеров.

12. Буква греческого алфавита.

15. Название фигуры в настольной игре, для которой имеются компьютерные варианты.

16. Часть рабочей книги программы Microsoft Excel.

По вертикали

1. Язык программирования.

3. Элемент электронной таблицы.

4. Наука о законах и формах мышления.

7. Результат целочисленного деления.

9. Пользователь телефонной линии.

13. Константа логического типа (русский вариант).

14. Поименованная совокупность данных на носителе.

Ответы (можно осенью и не ко всем терминам) присылайте в редакцию.

Наше пожелание выпускникам: