1 Знакомство с FreePascal. Первая программа.

Программы на языке Pascal содержатся в обычных текстовых файлах. В операционной системе Windows такие файлы обычно редактируются программой, которая называется Блокнот (по-английски notepad). Файлы, которые, кроме собственно текста, содержат сложную информацию о форматировании, например, документы редактора Word, для программирования на Pascal (как и на любом другом языке программирования) не годятся.

После того, как программа написана и введена в компьютер в виде текстового файла (обычно файл с программой на языке Pascal имеет расширение .pas), для запуска программы на выполнение ее необходимо скомпилировать, т. е. преобразовать в другую программу, делающую то же самое, но написанную на машинном языке — языке, понятном процессору того компьютера, на котором эту программу придется выполнять (исполняемые файлы, содержащие программы на машинном языке, в операционной системе Windows имеют расширение .exe).

Наконец, после компиляции программы, полученный файл с расширением .exe можно запустить. Для этого можно воспользоваться, например, проводником Windows, открыв в нем папку с этим файлом и щелкнув на нем мышью два раза.

Для упрощения программирования на языке Pascal (как и на многих других языках программирования) созданы специальные среды, по-английски называемые IDE (integrated development environment, интегрированная среда разработки). В их состав обычно входит текстовый редактор, а также ряд вспомогательных средств для запуска компилятора, отладчика, профилировщика и других средств программирования.

Среди всех удобств использования IDE можно отметить возможность быстро найти в тексте программы то место, в котором компилятор выдал ошибку. Также полезной является возможность отладки программы, когда подлежащая выполнению строка подсвечивается прямо в текстовом редакторе, и в отдельном окне показаны значения переменных в данной точке программы.

Мы будем работать в IDE, поставляемой вместе с компилятором FreePascal. По виду и основным командам она напоминает TurboPascal.

Для ввода текста программы необходимо сначала завести новое окно редактора. Это делается при помощи пункта New меню File. В открывшемся окне редактора нужно ввести интересующую нас программу, после чего записать ее в файл (File/Save). Важным здесь является вопрос о том, где сохранять файл. Чтобы он не пропал при следующей перезагрузке компьютера, а также был доступен независимо от того, за каким компьютером Вы работаете, его нужно сохранять на диске W:. Это сетевой диск; его содержимое свое у каждого студента, и он подключается в момент входа в систему. Именно на нем студенты должны хранить все свои программы и прочие файлы, поскольку диск C: при каждой перезагрузке восстанавливается, и все записанные пользователями на него файлы пропадают.

Далее, после ввода и записи программы на диск, ее необходимо скомпилировать (Compile/Compile). Если в появившемся окне написано Compile successful, значит, компилятор не обнаружил ошибок в программе, и она успешно скомпилирована. В противном случае в окошке появляется надпись Compile failed (компиляция не удалась), и внизу экрана открывается окно с сообщениями компилятора об обнаруженных в программе ошибках. В этом случае нужно нажать клавишу Esc, чтобы закрыть окно компиляции, и затем нажать Enter на одном из сообщений об ошибках — курсор переместится в окно с текстом программы в место, где, по мнению компилятора, должна быть исправлена ошибка. При этом внизу окна появится текст сообщения об ошибке. После исправления этой ошибки можно вернуть окно с сообщениями компилятора, выбрав Compile/Compile messages, перейти к другой ошибке при помощи клавиш перемещения курсора вверх/вниз, и опять вернуться к исходному тексту программы, для исправления очередной ошибки.

Правда, имеется одно хорошее правило — исправлять всегда только первую ошибку. Дело в том, что компилятор в принципе не может знать, чего хотел программист, поэтому сообщения об ошибках могут отличаться от истинной причины возникновения той или иной ошибки, замеченной компилятором. В частности, одна ошибка может провоцировать несколько (иногда — очень много) сообщений об ошибках. Поэтому иногда исправление одной ошибки может убирать сотни сообщений об ошибках по всей программе; увы, внесение в программу дополнительных ошибок тоже может радикально влиять на количество сообщений, причем как в сторону его увеличения, так и в сторону уменьшения, что обычно удивляет новичков. Так что количество сообщений об ошибках не может быть критерием правильности программы в том смысле, что далеко не всегда программа с маленьким количеством сообщений правильнее программы с большим.

Кроме того, в программировании возможны ошибки разного рода. Здесь можно провести следующую аналогию с текстами на русском языке: бывают ошибки, например, выражающиеся в неправильном написании некоторого слова; если при этом получилось бессмысленное слово, ошибки такого рода компилятор может заметить и сообщить о них программисту. Но бывают ошибки смысловые — текст, их содержащий, написан на грамотном русском языке (или на языке программирования), но содержащееся в этом тексте утверждение неверно. Поскольку смысл программы компилятору неизвестен (и не может быть известен в принципе), ошибки такого рода компилятор обнаружить не может.

В итоге, программа успешно компилируется и запускается. Некоторые ошибки такого рода могут быть замечены в процессе выполнения программы — например, деление на 0. Однако часто такие ошибки так и остаются незамеченными, приводя к тому, что программа молча выдает неверный результат.

Для иллюстрации общей структуры программы на Pascale, приведем в качестве примера очень простую программу. Она выводит на экран сообщение «Hello, world!», ждет, пока пользователь нажмет клавишу Enter и затем завершается. Текст этой программы выглядит следующим образом:

```
program Hello;
```

```
begin
  writeln('Hello, world!');
  readln
end.
```

Первая строка файла содержит слово `program` и имя программы. Слово `program` — ключевое, т. е. особое в том смысле, что его значение фиксировано в языке Pascal, и его нельзя использовать в программах на Pascal'е для других целей, например для имен чего-то, созданного программистом. Оно используется для указания имени программы, которое следует за ним через пробел. Вообще говоря, указание имени программы не является строго обязательным, но служит признаком хорошего стиля программирования.

На остальных строках располагаются ключевые слова `begin` и `end`, за последним стоит символ «.», и между ними некоторый текст. Это так называемое тело программы. Между `begin` и `end` стоят так называемые операторы, составляющие программу, т. е. указания того, что должна делать программа.

В нашем случае этих операторов — два, и они разделяются символом «;». Первый из них называется оператором вывода, и в скобках у него указано то, что должно быть напечатано на экране в результате выполнения этого оператора. В данном случае это текст «Hello, world!». Вместо кавычек в Pascal используются апострофы, а вся конструкция называется «строковая константа». Такие конструкции указывают на то, что используемый между апострофами текст должен быть напечатан (или использован) как есть, без каких либо преобразований (например, замены имени переменной на ее значение).

Второй оператор — оператор ввода. В данном случае у него нет параметров (нет скобок после readln). В таком виде он означает просто ожидание до тех пор, пока пользователь не нажмет клавишу Enter. Этот оператор стоит здесь только для того, чтобы пользователь мог увидеть результат работы программы; в противном случае он промелькнет на экране очень быстро, и опять будет показано окно редактора с текстом программы. В этом случае увидеть результат, т. е. вывод программы, можно, выбрав Debug/User screen. Вернуться обратно к редактору можно, нажав Esc.

Задание 1. Поменяйте содержимое строковой константы, запустите программу и убедитесь, что теперь выводится новая строка. Что будет, если убрать апострофы? А если заменить их на двойные кавычки? А если вместо writeln написать write? Напишите программу, выводящую строку «Here was I». Напишите программу, выводящую две строки «First line» и «Second line».

2 Вычисление выражений

В программах на Паскале можно вычислять выражения. Синтаксис выражений, т. е. способ их записи, похож на принятый в математике. В выражениях можно использовать целые числа (как в математике), а также вещественные числа в обычном виде, как десятичные дроби, только вместо десятичной запятой используется точка. Например, запись 2.731 означает «2 целых 731 тысячная». Можно также использовать вещественные числа в так называемом экспоненциальном или научном формате, когда к числу сзади дописывается буква «e» и за ней целое число (вся эта запись не должна содержать пробелы), которое означает, что исходное число надо умножить на 10 в указанной степени. Например, число 1.23e2 означает просто 123, а 1e-6 означает одну миллионную.

В большинстве языков программирования, и Pascal не является исключением, нельзя использовать в выражениях слишком большие целые или вещественные числа. Более того, диапазон используемых чисел зависит от компьютера, операционной системы и компилятора. В пределах этого диапазона, т. е. если исходные числа и все промежуточные результаты попадают в этот диапазон, арифметические действия с целыми числами выполняются точно. Если же исходные числа или промежуточные результаты выходят за пределы этого диапазона, результат будет неверным, причем система не всегда сообщает об этом программисту. Чтобы она сообщала, для целых чисел, в начало программы надо поместить управляющий комментарий {$Q+}. Для вещественных чисел используется специальное значение, означающее «бесконечно большое число».

Даже если все исходные числа и промежуточные результаты в выражении с вещественными числами лежат в допустимых пределах, все равно вычисления с вещественными числами выполняются лишь приближенно. Это происходит по ряду причин. Во-первых, даже такое, казалось бы, нехитрое вещественное число, как одна десятая, не может быть представлено в компьютере точно, поскольку в памяти компьютера вещественные числа представляются конечными двоичными дробями, а одна десятая так представлена быть не может (чтобы получить одну десятую, нужно поделить не только на 2, но и на 5, а этого уже конечные двоичные дроби не умеют). Далее, число двоичных цифр в этих дробях ограничено, поэтому неизбежно необходимо округление результатов (например, при умножении точный результат должен содержать примерно в два раза больше цифр, чем каждый из операндов).

В выражениях также можно использовать обычные арифметические операции, такие как сложение, вычитание, умножение и деление. Для умножения правила несколько отличаются от математических: в качестве значка для умножения используется звездочка, и пропускать его нельзя. Кроме того, для целых чисел имеются две дополнительные операции: div (неполное частное) и mod (остаток от деления), которые дают целый результат. Также, разумеется, можно использовать скобки, но только круглые (смысл квадратных скобок будет ясен несколько позже).

Для вещественных чисел также можно использовать элементарные функции, названия некоторых из них отличаются от принятых в математике:

Функция	Запись	
	математика	язык Pascal
синус	$\sin x$	sin(x)
косинус	$\cos x$	cos(x)
тангенс*	$\mathrm{tg}\, x$	tan(x)
арксинус*	$\arcsin x$	arcsin(x)
арккосинус*	$\arccos x$	arccos(x)
арктангенс	$\mathrm{arctg}\, x$	arctan(x)
гиперболический синус*	$\mathrm{sh}\, x$	sinh(x)
гиперболический косинус*	$\mathrm{ch}\, x$	cosh(x)
гиперболический тангенс*	$\mathrm{th}\, x$	tanh(x)
гиперболический арксинус*	$\mathrm{arcsh}\, x$	arsinh(x)
гиперболический арккосинус*	$\mathrm{arcch}\, x$	arcosh(x)
гиперболический арктангенс*	$\mathrm{arcth}\, x$	artanh(x)
экспонента	e^x	exp(x)
натуральный логарифм	$\ln x$	ln(x)
квадрат	x^2	sqr(x)
степень*	x^y	power(x,y)
квадратный корень	$\sqrt{x}$	sqrt(x)
корень*	$\sqrt[y]{x}$	power(x,1/y)

Здесь знаком «*» помечены те функции, для использования которых необходимо подключить модуль math (т. е. в начале программы, после указания ее имени, нужно написать uses math;). Остальные функции этого не требуют.

Пример 1. Следующая программа вычисляет выражение

$$\frac{\sin 5 + 1.75^2}{3e^{\cos 7}}$$

и выводит его результат на экран.

```
program Calculate;
begin
  writeln((sin(5)+sqr(1.75))/(3*exp(cos(7)))); readln
end.
```

Иногда бывает удобно вывести подсказку, описывающую выводимое число. Это можно сделать, например, так:

```
program Calculate;
begin
  writeln('Result=');
  writeln((sin(5)+sqr(1.75))/(3*exp(cos(7)))); readln
end.
```

Чтобы и подсказка, и результат вывелись на одной строке, можно написать так:

```
program Calculate;
begin
  write('Result=');
  writeln((sin(5)+sqr(1.75))/(3*exp(cos(7)))); readln
end.
```

А можно и так:

```
program Calculate;
begin
  writeln('Result=', (sin(5)+sqr(1.75))/(3*exp(cos(7)))); readln
end.
```

Иногда также бывает удобно, особенно при выводе таблиц, задавать ширину поля (т. е. места на экране, куда будет выведено число). Если выводимое число занимает меньше места, оно будет дополнено пробелами спереди. Это делается так:

```
program Calculate;
begin
  writeln('Result=', (sin(5)+sqr(1.75))/(3*exp(cos(7))):15); readln
end.
```

В данном случае ширина поля равна 15 символам. Для вещественных чисел, кроме того, удобно задавать точность, с которой выводится результат. Для этого добавляется еще символ : и за ним — требуемое число знаков после десятичной точки. Например, если в предыдущем примере нам требуется 7 цифр после точки, нужно написать

```
program Calculate;
begin
  writeln('Result=', (sin(5)+sqr(1.75))/(3*exp(cos(7))):15:7); readln
end.
```

Задание 2. Вычислите в вещественных числах значение выражения

$$\frac{\sqrt[3]{\sin^2(1.28)+1}-26+\operatorname{arctg}(1.17+2.95)}{(2.01-\cos^2 3.86)^{5.84}+25.362}.$$

Выведите его значение на экран с подсказкой, шириной поля 20 и 10 цифрами после запятой.

3 Переменные

Иногда при вычислении громоздких выражений бывает удобно запомнить некоторые промежуточные результаты, чтобы впоследствии их использовать. Это также удобно, если возникает необходимость использовать значение одного и того же подвыражения многократно. Также иногда возникает необходимость получать от пользователя некоторые данные, их тоже нужно запоминать.

В языке Pascal для запоминания промежуточных результатов в выражениях существует механизм переменных. Переменная может хранить целое или вещественное число и позволяет выполнять два действия: записать в переменную число (это действие в языках программирования называется присваиванием), и извлечь хранящееся в переменной число для его дальнейшего использования в выражениях.

Прежде, чем мы сможем воспользоваться переменной, ее надо объявить, т. е. написать некоторый текст в программе, объясняющий компилятору, что нам требуется переменная. Текст объявления некоторой переменной в языках программирования называется декларацией.

Переменная имеет имя, тип и значение. Имя переменной — текст, используемый для указания того, что переменная используется в данном месте программы. Имя переменной должно состоять из букв, цифр и знаков подчеркивания и не начинаться с цифры. Пробелы в имени переменной не допускаются. Также, имя переменной не должно совпадать ни с одним из ключевых слов языка Pascal. В качестве примеров правильных имен переменных можно назвать `a`, `x12`, `pos_of_ptr`. В качестве примеров неправильных имен переменных можно назвать `x y`, `1xy2`, `begin`, `a+2`.

Тип переменной указывает на то, какого рода информация в ней хранится. Переменная, хранящая вещественное число, будет иметь тип `real`, а переменная, в которой хранится целое число — тип `integer`.

Наконец, значение переменной — то, что в ней записано; оно должно соответствовать типу переменной. Например, нельзя хранить вещественные числа в переменной типа `integer`.

В декларации переменной указываются ее тип и имя. Декларации переменных помещаются после ключевого слова `var`, между конструкцией `uses` и телом программы. Например, чтобы объявить переменную с именем `x` типа `real` для хранения вещественных чисел, нужно между конструкцией `uses` и телом программы написать `var x:real;`. Точка с запятой обязательна в конце любой декларации. Если нужно объявить несколько переменных одного типа, можно для каждой из них написать свою декларацию, но можно обойтись и одной декларацией, написав список нужных имен переменных через `,`. Так, чтобы объявить три переменные типа `real` с именами `x`, `y` и `z`, можно написать `var x, y, z:real;`. Если нужны переменные разных типов, можно написать только одно слово `var`, например так: `var x:real; i:integer;`.

Записать в переменную значение можно при помощи оператора присваивания; он выглядит следующим образом:

```
имя_переменной := выражение_подходящего_типа
```

При этом сама переменная вполне может участвовать в этом выражении; в этом случае в выражении будет использовано старое значение этой переменной (то, которое она имела до присваивания).

Чтобы извлечь значение переменной, достаточно просто указать ее имя в выражении. Однако, прежде чем использовать значение переменной, ее значение должно быть задано, например, при помощи присваивания.

Пример 2. Напишем программу, вычисляющую значение выражения

$$\frac{\sin 1.03+\sqrt{\sin 1.03-0.03}}{5-\sin^2 1.03}.$$

При вычислении этого выражения разумно сначала вычислить $\sin 1.03$ и запомнить это значение в переменной, а затем вычислить указанное выражение, пользуясь запомненным в переменной значением. Таким образом, текст программы будет выглядеть так:

```
program Calculate2;
var x:real;
begin
  x := sin(1.03);
  writeln('Result=', (x+sqrt(x-0.03))/(5-sqr(x))); readln
end.
```

Задание 3. Вычислить выражение

$$(\operatorname{tg} 3 - \sin 4)\frac{\sin 4 \operatorname{tg} 3 + 1}{2 + 3\sqrt{\sin 4 + \operatorname{tg} 3 + 7}}.$$

Пример 3. а) Если где-то в программе мы хотим записать в переменную `x` значение 1.72, мы должны там написать `x:=1.72`. б) Если мы хотим увеличить значение переменной `i` на единицу, мы должны поставить в программе `i:=i+1`. в) Если мы хотим увеличить значение переменной `y` в два раза, мы должны написать `y:=y*2`.

Задание 4. Объявите переменную `i` типа «целое число», присвойте ей начальное значение 10, затем увеличьте ее значение на 7, потом в 3 раза, после этого замените ее значение на остаток от него при делении на 15. Выведите на экран значение этой переменной в самом начале и после каждого преобразования. Убедитесь, что все вычисления выполнены правильно.

Задание 5. В программе заведите две целых переменных i и j. Присвойте им начальные значения 38 и 75. Затем присвойте первой из этих переменных их сумму, после этого второй — их разность и наконец первой — опять их разность. Выводите значения переменных на экран в самом начале и после каждого преобразования. Что делает эта программа?

Присваивание значения переменной — не единственный способ положить в переменную какое-либо значение. Можно считать это значение с клавиатуры при помощи конструкции `readln(имя)`.

Пример 4. Следующая программа просит пользователя ввести целое число и печатает его восьмую степень.

```
program EighthPower;
var x:integer;
begin
  write('x='); readln(x);
  writeln('x^8=', sqr(sqr(sqr(x)))); readln
end.
```

Задание 6. а) Введите с клавиатуры вещественные числа a, b и c, и напечатайте значение выражения $\frac{\cos(e^a - 2b) + 3}{\sqrt{5\sin(abc) + 100} - \operatorname{tg} c}$.

б) Введите длины сторон треугольника и выведите длины его высот.

в) Направления кодируются следующим образом: 0 — север, 1 — запад, 2 — юг и 3 — восток. Введите направление и выведите результат его поворота на 90° по часовой стрелке и против часовой стрелки.

г) Введите оценку по 99-балльной шкале (от 0 до 99) и выведите разряд, соответствующий этой оценке (оценкам от 90 до 99 соответствует разряд 'A', от 80 до 89 — 'B', от 70 до 79 — 'C' и т. д.).

д) Грузовик может увезти 32 коробки. Введите число коробок, которые необходимо увезти и выведите минимальное достаточное для этого число грузовиков.

е) Введите натуральное число и выведите число цифр в его десятичной записи (подсказка: целую часть вещественного числа x выдает функция $\operatorname{int}(x)$).

4 Условный оператор. Составной оператор.

Программам, получающим от внешнего мира какие-то данные, например, поступающие с клавиатуры, часто приходится анализировать полученную информацию и выполнять различные действия в зависимости от истинности или ложности некоторых условий. Конструкция, существующая в языке Pascal для реализации такого поведения, называется условным оператором.

Синтаксис, т. е. форма записи, условного оператора в языке Pascal следующий:

```
if условие then оператор1 else оператор2
```

Если условие истинно, выполняется оператор1 (он называется положительной альтернативой), если оно ложно — оператор2 (отрицательная альтернатива). При этом часть else и отрицательная альтернатива могут отсутствовать. В этом случае, если условие ложно, не выполняется ничего. В качестве операторов после then и else могут присутствовать любые операторы, например, присваивания или другой условный оператор. Если один условный оператор вкладывается в другой, например таким образом:

```
if x>1
    if y>2
    then writeln('случай 1')
    else writeln('случай 2')
```

возникает вопрос, к какому if относится else? Правило здесь такое: else всегда относится к последнему if, еще не имеющему else (при этом число пробелов перед else не имеет никакого значения). Если мы хотим, чтобы else относилось к внешнему if, внутренний должен быть заключен в составной оператор (см. позже).

Если мы хотим выполнить в случае, например, истинного условия несколько операторов, нужно опять использовать так называемый составной оператор (см. позже).

В языке Pascal условием может быть только выражение, имеющее логический результат (т. е. истина (true) или ложь (false); например, применение операции сравнения или логической операции).

Операций сравнения в языке Pascal шесть: = (равно), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), <> (не равно). Если операция состоит из двух символов, пробелы между ними не допускаются.

По поводу операций сравнения нужно сделать одно важное замечание. Как уже говорилось ранее, вычисления с вещественными числами выполняются лишь приближенно. Поэтому в большинстве случаев проверять два вещественных числа на точное равенство бессмысленно — результат практически всегда будет false, даже если математика утверждает обратное (проверьте для суммы трех 1/3, пяти 1/5 и семи 1/7, равно ли это 1.0). Поэтому на практике обычно задают определенный уровень точности, и два вещественных числа считаются равными, если их разность по модулю меньше этого числа. Это число зависит от того, с какой точностью выполняются вычисления, и более конкретных рекомендаций дать нельзя — все определяет практика, в основном метод проб и ошибок, поскольку точный учет ошибок округления — дело очень сложное.

Пример 5. Следующий фрагмент программы запишет в переменную x модуль того числа, которое хранилось там до выполнения этого фрагмента:

```
if x<0
    then x := -x
```

Задание 7. а) Введите два вещественных числа и выведите наименьшее из них.

б) Введите три вещественных числа и выведите наименьшее из них.

в) Введите целое число n и выведите $(-1)^n$.

Кроме того, имеется ряд логических операций (and, or, not), при помощи которых можно строить более сложные условия. Таблица, показывающая результат этих операций для всевозможных значений операндов, приводится ниже:

x	y	x and y	x or y	not y
false	false	false	false	true
false	true	false	true	false
true	false	false	true	true
true	true	true	true	false

Например, первая строка этой таблицы (не заголовок) означает, в частности, что false and false будет false (при x=false и y=false имеем x and y=false). Вообще, результат операции «и(and)» будет истина (true) только в том случае, если оба выражения, которые она соединяет, имеют значение истина. Результат логического «или(or)» будет истина (true), если хотя бы один операнд — истина, а логического «не(not)» — истина, если операнд имеет значение ложь (false).

К сожалению, в большинстве языков программирования (и Pascal — не исключение) неравенства нельзя соединять в цепочки, т. е. выражение вроде x<y<z является противозаконным. Вместо этого нужно писать (x<y) and (y<z), причем скобки в этой записи обязательны.

Пример 6. Следующий фрагмент программы вычисляет arcsin(x), если он определен, и выводит сообщение об ошибке, если нет:

```
if (-1<=x) and (x<=1)
    then writeln('arcsin(x)=', arcsin(x))
    else writeln('arcsin(x) is not defined')
```

Задание 8. а) Введите длины трех отрезков и выведите слово «да», если из них можно составить треугольник, и «нет», если нельзя.

б) Введите два ненулевых вещественных числа и выведите текст «одинаковые знаки», если они одного знака, и текст «разные знаки», если они разных знаков.

в) Введите коэффициенты квадратного уравнения и выведите его решение (текст «no solutions», если решений нет, «x=» и решение, если оно одно и «x1=», первый корень, «x2=», второй корень, если есть два решения).

В языке Pascal имеется еще одна разновидность операторов — так называемый составной оператор. Он уже упоминался ранее в связи с необходимостью поместить несколько операторов туда, где по правилам языка Pascal должен стоять один оператор. Составной оператор — это последовательность операторов, разделенных точкой с запятой, заключенная в специальные операторные скобки begin и end. Выполнение составного оператора состоит в последовательном выполнении входящих в него операторов в том порядке, в котором они встречаются в составном операторе.

Пример 7. Следующий фрагмент программы упорядочивает содержимое двух переменных a и b по возрастанию, чтобы a всегда было не больше b; используется дополнительная переменная t того же типа, что и a и b:

```
if a>b
  then
    begin
      t := a;
      a := b;
      b := t
    end
```

Задание 9. а) Напишите программу, решающую систему двух линейных уравнений с двумя неизвестными (необходимо рассмотреть все возможные случаи).

б) Введите декартовы координаты точки на плоскости и выведите ее полярные координаты (т. е. расстояние до начала координат и угол между положительной полуосью абсцисс и лучом из начала координат в заданную точку, отсчитываемый против часовой стрелки, от 0 до 2π).

в) Введите длины сторон двух треугольников и выведите текст «подобны», если они подобны, и текст «не подобны», если не подобны.

г) (для тех, кто знает комплексные числа) Напишите программу, вводящую вещественную и мнимую части комплексного числа и выводящую вещественную и мнимую части одного из квадратных корней из этого числа (без привлечения тригонометрической формы).

Иногда бывает удобно запоминать в переменных не только числа, но и истинность определенных условий. Для этого в языке Pascal имеется специальный тип переменных — логический. Он называется boolean; логические переменные объявляются также, как и любые другие; однако, они могут иметь только два разных значения — true (истина) или false (ложь). Им можно присваивать как сами эти значения, так и результаты логических выражений, например операций сравнения или логических операций. Их значения можно использовать в качестве аргументов логических операций, а также условий, например в условном операторе. Увы, логические значения нельзя вводить при помощи readln и выводить при помощи writeln.

Пример 8. Следующий пример показывает, как можно обойти предыдущее ограничение — ввод и вывод логических значений. Здесь используется соглашение о том, что 0 означает false, а 1 — true. Программа вводит логическое значение указанным способом, применяет к нему операцию логического отрицания, и затем выводит результат, также используя указанное соглашение.

```
program Inverse;
var i:integer;
    x:boolean;
begin
  readln(i);
  x := i=1;
  x := not x;
  writeln(ord(x));
  readln
end.
```

Здесь ord — функция, выдающая номер значения перечислимого типа, т. е. такого типа, переменная которого может иметь лишь небольшое количество разных значений. Перечислимые типы будут обсуждаться позже; здесь же скажем только, что boolean — один из них.

Задание 10. а) Упростите программу из примера 8.

б) Введите указанным выше способом два логических значения, получите из них при помощи только «и», «или» и «не» результат операции «исключающее или» (истина, если они различны), и выведите его. Также получите результат операции «импликация» (из первого следует второе; ложно только «из истины следует ложь») и выведите его.

в) (для тех, кто знает системы счисления с другими основаниями — не 10) Введите две двоичные цифры (т. е. цифры в двоичной системе счисления — 0 или 1). Превратите их в логические значения указанным выше способом, и получите из них при помощи только логических операций две двоичные цифры их суммы (по прежнему 0 соответствует false, а 1 — true), т. е. старший и младший разряды этой суммы. Затем то же самое для пяти двоичных цифр и трех разрядов результата.