МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тульский государственный университет»

Интернет-институт

ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ

по дисциплине

«Интеллектуальные системы в промышленности»

Семестр 6

Вариант 3

Выполнил: студент гр. ИБ262521-ф

Артемов Александр Евгеньевич

Проверил: канд. техн. наук, доц.

Французова Юлия Вячеславовна

Тула 2024

Лабораторная работа № 1.

Название работы: Основные понятия Пролога.

Цели работы: Узнать и усвоить базовые понятия языка и основы написания программ на языке Пролог.

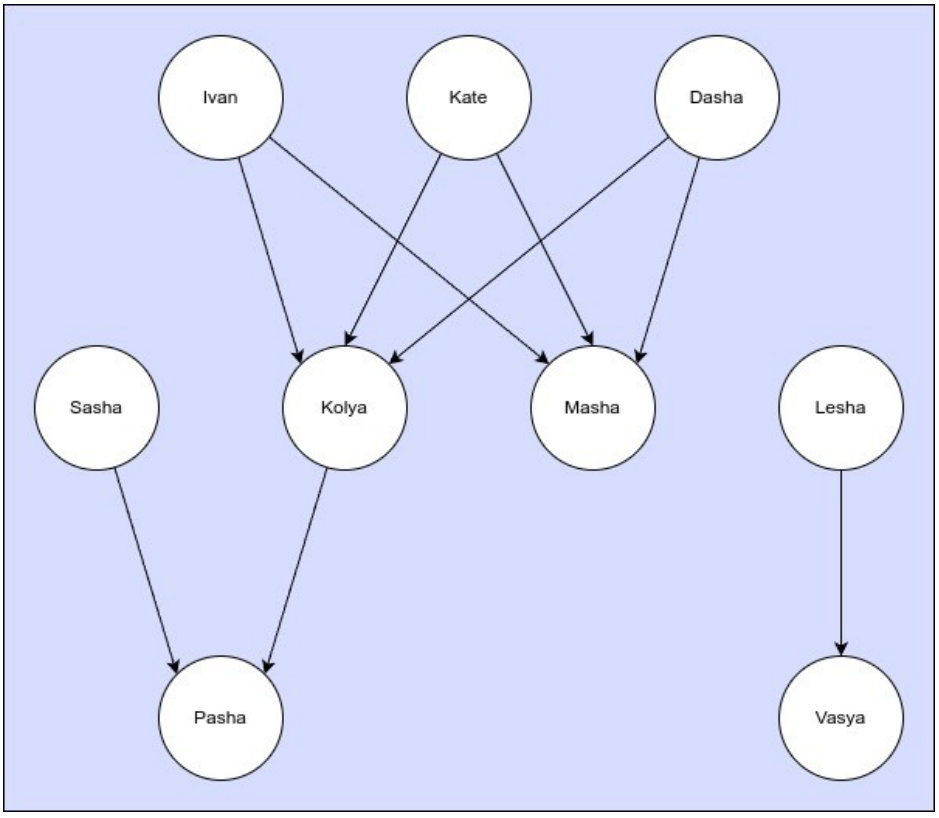
Задание:
Создайте предикат, проверяющий, являются ли два человека:

1. сестрами;
2. братьями;
3. дедушкой и внуком (внучкой);
4. дядей и племянником (племянницей);
5. супругами;
6. родственниками.

Выполнение лабораторной работы.

Изучены теоретические сведения лабораторной работы о базовых понятиях языка и основах написания программ на языке Пролог.

Для выполнения лабораторной работы определим предметную область для всех заданий как некоторое генеалогическое дерево, представленное на рисунке ниже:



Стрелками определяется отношение «является родителем». В данном случае Паша является родителем Саши и Коли, а Вася — родителем Леши. В свою очередь, Коля и Маша являются родителями Ивана, Кати и Даши, от куда видно, что Иван, Катя и Даша — братья и сестры, а Коля и Маша — предположительно, супруги.

Упреждая выполнения заданий, отметим, что Паша является дедом для Ивана, Кати и Даши, а Саша — их дядей. Так же видно, что Вася и Леша не имеют родственников, кроме друг друга.

Исходный код программы сохраним в файл lab1.pl, а запускать его будем при помощи интерпретатора SWI-Prolog 9.2.7.

1. Создадим предикат, проверяющий, являются ли два человека сестрами. Для этого добавим в файл строки, определяющие отношение «быть сестрами» согласно нашего «древа жизни», где сестрами являются только Катя и Даша:

```
issister(kate, dasha).
issister(dasha, kate).
```

Так как отношение «быть сестрами» должно работать в обе стороны, добавляем два предиката, то есть Катя — сестра Даши и Даша — сестра Кати. При вводе вопроса в Пролог-интерпретатор являются ли Катя и Даша сестрами получаем положительный ответ. Аналогично при вводе вопроса являются ли Даша и Катя сестрами. А вот при вводе вопроса являются ли Даша и Иван сестрами — ответ отрицательный.

2. Создадим предикат, проверяющий, являются ли два человека братьями. Согласно «древа жизни», братьями являются только Саша и Коля. Для этого добавим в файл строки:

```
isbrother(kolya, sasha).
isbrother(sasha, kolya).
```

Так как отношение «быть братьями» должно работать в обе стороны, добавляем два предиката, то есть Коля — брат Саши и Саша — брат Коли. При вводе вопроса в Пролог-интерпретатор являются ли Коля и Саша братьями получаем положительный ответ. Аналогично при вводе вопроса являются ли Саша и Коля братьями. А вот при вводе вопроса являются ли Саша и Маша братьями — ответ отрицательный.

3. Создадим предикат, проверяющий, являются ли два человека дедушкой и внуком (внучкой). Так как отношение родства «быть дедушкой» человека означает «быть родителем родителя» человека, необходимо ввести отношение «быть родителем». Это отношение является однонаправленным, так как, например, согласно нашего древа Коля является родителем Ивана, но Иван не является родителем Коли, он является потомком. Добавим в код предикаты отношения «быть родителем»; в нашем примере Коля и Маша являются родителями Ивана, Кати и Даши, Паша является родителем Саши и Коли, а Вася — Леши.

```
isparent(kolya, ivan).
isparent(kolya, kate).
isparent(kolya, dasha).
isparent(masha, ivan).
isparent(masha, kate).
isparent(masha, dasha).
isparent(vasya, lesha).
isparent(pasha, sasha).
isparent(pasha, kolya).
```

Согласно нашего древа отношение «являться дедушкой и внуком (внучкой)» просматривается у Паши (дедушка) с Иваном (внук), Катей и Дашей (внучки). Создадим правило определяющее данное отношение через отношение «быть родителем»:

```
isgrandpa(X, Y) :- isparent(X, Z), isparent(Z, Y).
```

При вводе вопроса в Пролог-интерпретатор является ли Паша дедом Ивану получаем положительный ответ. Аналогично при вводе вопроса является ли Паша дедом Кате. А вот при вводе вопроса является ли Паша дедом Маше — ответ отрицательный.

4. Создадим предикат, проверяющий, являются ли два человека дядей и племянником (племянницей). Так как отношение родства «быть дядей» человека означает «быть братом родителя» человека, то правило данного отношения определяется через предикаты «быть братом» и «быть родителем». Согласно нашего древа отношение «дядей и племянником (племянницей)» просматривается у Саши (дяди) с Иваном (племянник), Катей и Дашей (племянницы). Создадим правило определяющее данное отношение:

```
isuncle(X, Y) :- isbrother(X, Z), isparent(Z, Y).
```

При вводе вопроса в Пролог-интерпретатор является ли Саша дядей Ивану получаем положительный ответ. Аналогично при вводе вопроса является ли Саша дядей Кате. А вот при вводе вопроса является ли Саша дядей Маше — ответ отрицательный:

5. Создадим предикат, проверяющий, являются ли два человека супругами. Так как отношение «быть супругами» должно работать в обе стороны, добавляем два предиката, то есть Коля — супруг Маши и Маша — супруга Коли:

```
ismarried(kolya, masha).
ismarried(masha, kolya).
```

При вводе вопроса в Пролог-интерпретатор являются ли Коля и Маша супругами получаем положительный ответ. Аналогично при вводе вопроса являются ли Маша и Коля супругами. А вот при вводе вопроса являются ли Маша и Вася супругами — ответ отрицательный:

6. Создадим предикат, проверяющий, являются ли два человека родственниками. В предыдущих примерах реализованы большинство отношений родства, кроме отношений родства между братом и сестрой, а так же между братом супруга и супругой. Тип родства между братом и сестрой просматривается между Иваном и Катей, Иваном и Дашей: отношение должно являться двунаправленным. Добавим соответствующие предикаты:

```
ischildren(ivan,kate).
ischildren(ivan,dasha).
ischildren(kate,ivan).
ischildren(dasha,ivan).
```

Наименование «ischildren» выбрано с семантикой «являются детьми одних родителей». Отношение между Катей и Дашей не добавляется, так как будет использовано отношение «являться сестрами».

Отношение родства между братом супруга и супругой в нашем примере: Саша — деверь для Маши (брат мужа для жены), а Маша — сноха для Саши (замужняя женщина по отношению к родным ее мужа). Добавим в код эти правила на основе существующих:

```
isdever(X,Y) :- isbrother(X, Z), ismarried(Z, Y).
issnoxa(X,Y) :- ismarried(X, Z), isbrother(Z, Y).
```

По нашему «древу жизни» видим, что Паша, Саша, Коля, Маша, Иван, Катя и Даша являются родственниками между собой, но не являются родственниками ни Леше, ни Васе. Так же стоит отметить, что в данном случае не проработано отношение родства от потомка к родителю, соответственно, и к прародителю. В дополнение расширим отношение «являться снохой» - супруга к родителям мужа:

```
isdescendant(ivan,kolya).
isdescendant(kate,kolya).
isdescendant(dasha,kolya).
isdescendant(ivan,masha).
isdescendant(kate,masha).
isdescendant(dasha,masha).
isdescendant(sasha,pasha).
isdescendant(kolya,pasha).
isdescendant(lesha,vasya).
issnoxa(X,Y) :- ismarried(X, Z), isdescendant(Z, Y).
```

Добавим все заданные отношения родства в правило «isrelative» - «являться родственниками»:

```
isrelative(X, Y) :- issister(X,Y).
isrelative(X, Y) :- isbrother(X,Y).
isrelative(X, Y) :- ischildren(X,Y).
isrelative(X, Y) :- isparent(X,Y).
isrelative(X, Y) :- ismarried(X,Y).
isrelative(X, Y) :- isgrandpa(X,Y).
isrelative(X, Y) :- isuncle(X,Y).
isrelative(X, Y) :- isdever(X,Y).
isrelative(X, Y) :- issnoxa(X,Y).
isrelative(X, Y) :- isdescendant(X,Y).
```

```
isrelative(X, Y) :- isdescendant(X,Z), isdescendant(Z,Y).
isrelative(X, Y) :- isdescendant(X,Z), isbrother(Z,Y).
```

Два последних отношения родства определяют отношения «являться внуком (внучкой)» и «являться племянником (племянницей)» в направлении обратном уже реализованному, то есть от потомка.

При вводе вопроса в Пролог-интерпретатор являются ли Катя и Паша родственниками получаем положительный ответ. Аналогично при вводе вопроса являются ли Маша и Паша родственниками. Аналогично при вводе вопроса являются ли Маша и Саша родственниками. А вот при вводе вопроса являются ли Маша и Леша родственниками — ответ отрицательный:

Просмотрим всех родственников Маши и Леши: как видно, между собой они родственниками не являются:

Полный листинг кода программы в файле lab1.pl:

```
issister(kate, dasha).
issister(dasha, kate).
isbrother(kolya, sasha).
isbrother(sasha, kolya).
isparent(kolya, ivan).
isparent(kolya, kate).
isparent(kolya, dasha).
isparent(masha, ivan).
isparent(masha, kate).
isparent(masha, dasha).
```

```
isparent(vasya, lesha).
isparent(pasha, sasha).
isparent(pasha, kolya).
isgrandpa(X, Y) :- isparent(X, Z), isparent(Z, Y).
isuncle(X, Y) :- isbrother(X, Z), isparent(Z, Y).
ismarried(kolya, masha).
ismarried(masha, kolya).
ischildren(ivan,kate).
ischildren(ivan,dasha).
ischildren(kate,ivan).
ischildren(dasha,ivan).
isdever(X,Y) :- isbrother(X, Z), ismarried(Z, Y).
issnoxa(X,Y) :- ismarried(X, Z), isbrother(Z, Y).
isdescendant(ivan,kolya).
isdescendant(kate,kolya).
isdescendant(dasha,kolya).
isdescendant(ivan,masha).
isdescendant(kate,masha).
isdescendant(dasha,masha).
isdescendant(sasha,pasha).
isdescendant(kolya,pasha).
isdescendant(lesha,vasya).
issnoxa(X,Y) :- ismarried(X, Z), isdescendant(Z, Y).

isrelative(X, Y) :- issister(X,Y).
isrelative(X, Y) :- isbrother(X,Y).
isrelative(X, Y) :- ischildren(X,Y).
isrelative(X, Y) :- isparent(X,Y).
isrelative(X, Y) :- ismarried(X,Y).
isrelative(X, Y) :- isgrandpa(X,Y).
isrelative(X, Y) :- isuncle(X,Y).
isrelative(X, Y) :- isdever(X,Y).
isrelative(X, Y) :- issnoxa(X,Y).
isrelative(X, Y) :- isdescendant(X,Y).
isrelative(X, Y) :- isdescendant(X,Z), isdescendant(Z,Y).
isrelative(X, Y) :- isdescendant(X,Z), isbrother(Z,Y).
```

Лабораторная работа № 2.

Название работы: Основы Турбо Пролога. Структура программы на Турбо Прологе. Директивы компилятора.

Цели работы: Изучить специфику среды Турбо Пролог, отличия языка Пролог от других языков программирования, способы организации управления программой при программировании.

Задание:

1. Создайте программу, решающую квадратное уравнение.
2. Создайте предикат, имеющий пять аргументов, и проверяющий, попадает ли точка, чьи координаты заданы первыми двумя параметрами, в круг, центр которого определяют третий и четвертый параметр, а радиус – пятый.
3. Создайте предикат, находящий абсолютное значение числа.
4. Создайте предикат, вычисляющий длину гипотенузы прямоугольного треугольника по длинам катетов.
5. Создайте предикат, вычисляющий периметр треугольника по двум сторонам и углу между ними.
6. Создайте предикат, вычисляющий площадь треугольника по двум сторонам и углу между ними.
7. Создайте предикат, вычисляющий площадь вписанного правильного треугольника в окружность радиусом R.
8. Создайте предикат, вычисляющий площадь описанного правильного треугольника вокруг окружности радиусом R.
9. Создайте предикат, вычисляющий площадь окружности, описанной вокруг правильного треугольника со стороной a.
10. Создайте предикат, вычисляющий площадь окружности, вписанной в правильный треугольник со стороной a.
11. Создайте предикат, вычисляющий площадь четырехугольника по четырем точкам с проверкой на выпуклость.
12. Создайте предикат, вычисляющий площадь треугольника по трем точкам.
13. Создайте предикат, проверяющий является ли угол между двумя векторами острым, прямым или тупым.
14. Создайте предикат, вычисляющий длину вектора по координатам начала и конца.
15. Создайте предикат, проверяющий направлен ли вектор по оси X или против.
16. Создайте предикат, проверяющий направлен ли вектор по оси Y или против.
17. Создайте предикат, определяющий проекцию вектора по заданной оси.

18. Создайте предикат, проверяющий направлен ли вектор по оси Z или против.

Выполнение лабораторной работы.

Изучены теоретические положения лабораторной работы по основам Турбо Пролога, структуры программы на Турбо Прологе, директивам компилятора.

Для решения заданий лабораторной работы используем среду разработки Visual Prolog.

1. Программа, решающая квадратное уравнение.

Квадратное уравнение имеет общий вид $ax^2+bx+c=0$, где a, b, c — вещественные числа, вводимые пользователем.

Решениями уравнения являются $x_1=\frac{-b+\sqrt{D}}{2a}$ и $x_2=\frac{-b-\sqrt{D}}{2a}$, если $D=b^2-4ac>0$; $x=\frac{-b}{2a}$, если $D=0$. При $D<0$ уравнение не имеет решения.

Создадим новый консольный проект и запишем в файл main.pro следующий код:

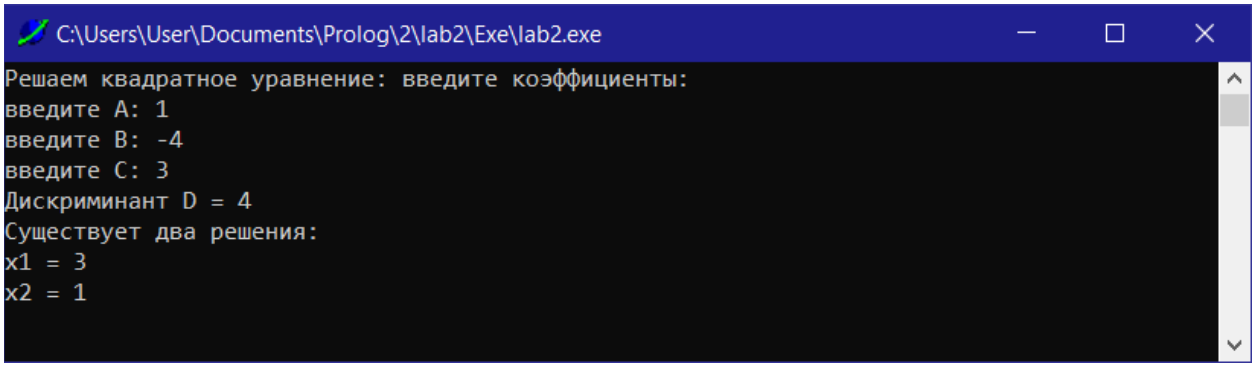
```
implement main
      open core, console

class facts
    a : real32 := 0.
    b : real32 := 0.
    c : real32 := 0.
    d : real32 := 0.

clauses
    run() :-
        stdio::write("Решаем квадратное уравнение, введите коэффициенты: "),
        nl,
        stdio::write("введите A: "),
        a := toTerm(stdio::readLine()),
        stdio::write("введите B: "),
        b := toTerm(stdio::readLine()),
        stdio::write("введите C: "),
        c := toTerm(stdio::readLine()),

        d := b * b - 4 * a * c,
        stdio::write("Дискриминант D = "),
        stdio::write(d),
        nl,

        if d > 0 then
            stdio::write("Существует два решения:"),
            nl,
            stdio::write("x1 = "),
            stdio::write((-b + math::sqrt(d)) / (2 * a)),
            nl,
            stdio::write("x2 = "),
            stdio::write((-b - math::sqrt(d)) / (2 * a)),
            nl
        elseif d = 0 then
```

```
            stdio::write("Существует одно решение:"),
            nl,
            stdio::write("x = "),
            stdio::write(-b / (2 * a)),
            nl
        else
            stdio::write("Решений нет (("),
            nl
        end if,

        _ = stdio::readLine().

end implement main

goal
    console::runUtf8(main::run).
```

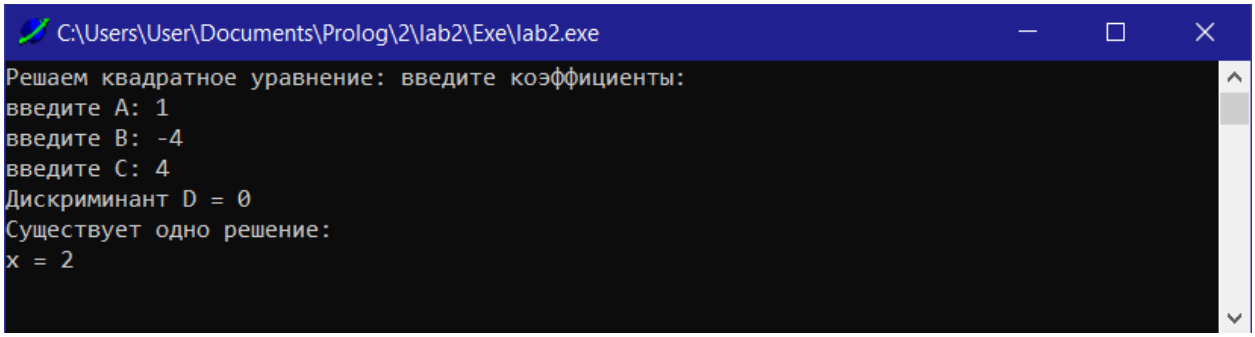
В данном коде в разделе clauses создается предложение run(), которое выполняется в разделе goal. Код предложения run() выводит приглашения для ввода коэффициентов квадратного уравнения и записывает введенные значения в соответствующие переменные. Далее вычисляется дискриминант и, в зависимости от его значения, вычисляются решения уравнения.

Проверим решение уравнения на примере коэффициентов:

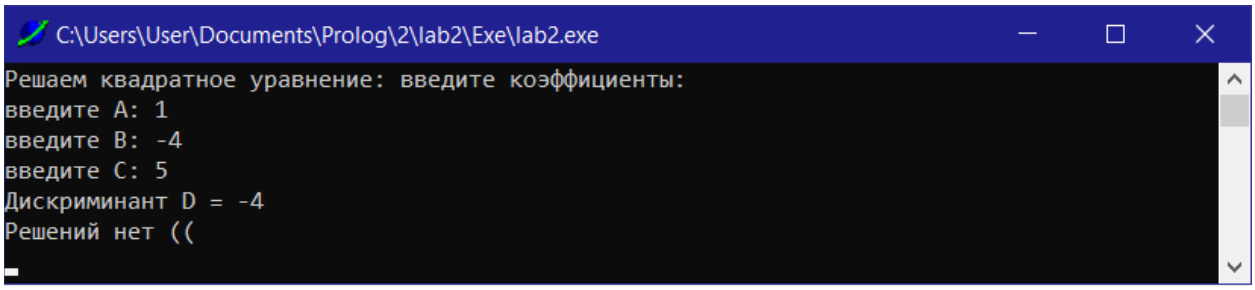
a = 1, b = -4, c = 3



a = 1, b = -4, c = 4



a = 1, b = -4, c = 5



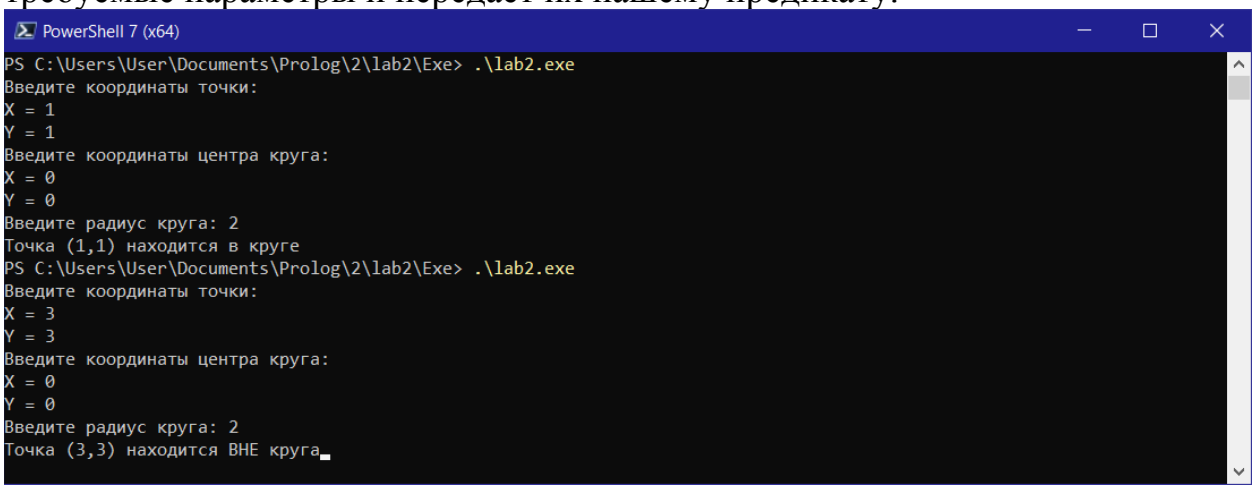
2. Предикат, имеющий пять аргументов, и проверяющий, попадает ли точка, чьи координаты заданы первыми двумя параметрами, в круг, центр которого определяют третий и четвертый параметр, а радиус – пятый.

Для решения задачи воспользуемся соотношением $(x_T - x_C)^2 + (y_T - y_C)^2 \leq R^2$, где (x_T, y_T) - координаты проверяемой точки, (x_C, y_C)- координаты центра круга, а R — радиус.

Наш предикат при помощи операции ветвления определяет истинность соотношения при переданных аргументах и выводит соответствующее сообщение.

```
myPredicate(X, Y, XC, YC, RC) :-
      stdio::write("Точка ("),
      stdio::write(X),
      stdio::write(","),
      stdio::write(Y),
      if (X - XC) ^ 2 + (Y - YC) ^ 2 <= RC ^ 2 then
            stdio::write(") находится в круге")
      else
            stdio::write(") находится ВНЕ круга")
      end if.
```

Предикат run() тестовой программы предлагает пользователю ввести требуемые параметры и передает их нашему предикату.



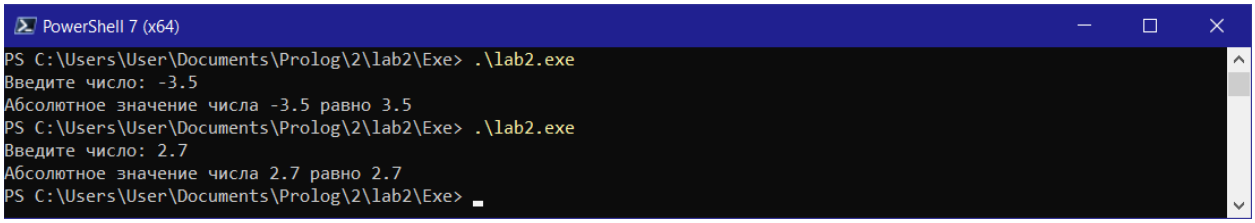
При тестировании предиката проверяем точки с координатами (1,1) и (3,3), а так же круг с центром в начале координат и радиусом 2. Соответственно, точка (1,1) находится внутри круга, а точка (3,3) — нет.

3. Предикат, находящий абсолютное значение числа.

Код предиката:

```
myPredicate() :-
	stdio::write("Введите число: "),
	X = stdio::read(),
	hasDomain(real, X),
	stdio::write("Абсолютное значение числа "),
	stdio::write(X),
	stdio::write(" равно "),
	if X >= 0 then
		stdio::write(X)
	else
		stdio::write(-X)
	end if.
```

При тестировании предиката проверяем числа -3,5 и 2,7.

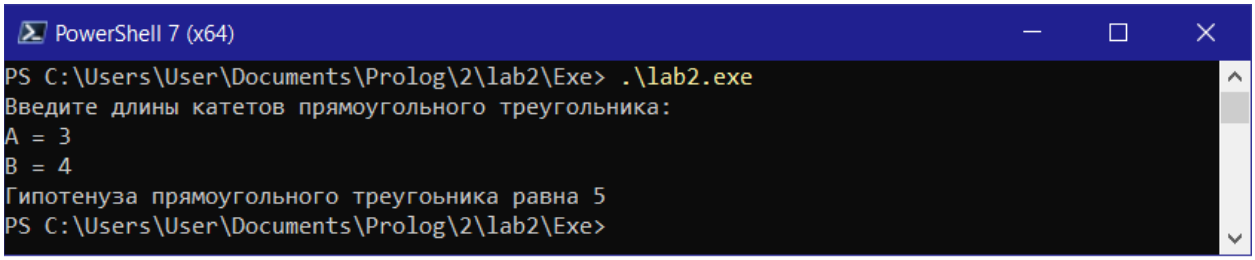


4. Предикат, вычисляющий длину гипотенузы прямоугольного треугольника по длинам катетов.

Согласно теоремы Пифагора $C^2=A^2+B^2$, откуда получаем длину гипотенузы $C=\sqrt{A^2+B^2}$.

```
myPredicate() :-
	stdio::write("Введите длины катетов прямоугольного треугольника: "),
	nl,
	stdio::write("A = "),
	A = stdio::read(),
	hasDomain(real, A),
	stdio::write("B = "),
	B = stdio::read(),
	hasDomain(real, B),
	stdio::write("Гипотенуза прямоугольного треугольника равна "),
	stdio::write(math::sqrt(A ^ 2 + B ^ 2)).
```

При тестировании предиката проверяем катеты с длинами 3 и 4, соответственно, длина гипотенузы равна $\sqrt{3^2+4^2}=\sqrt{25}=5$.



5. Предикат, вычисляющий периметр треугольника по двум сторонам и углу между ними.

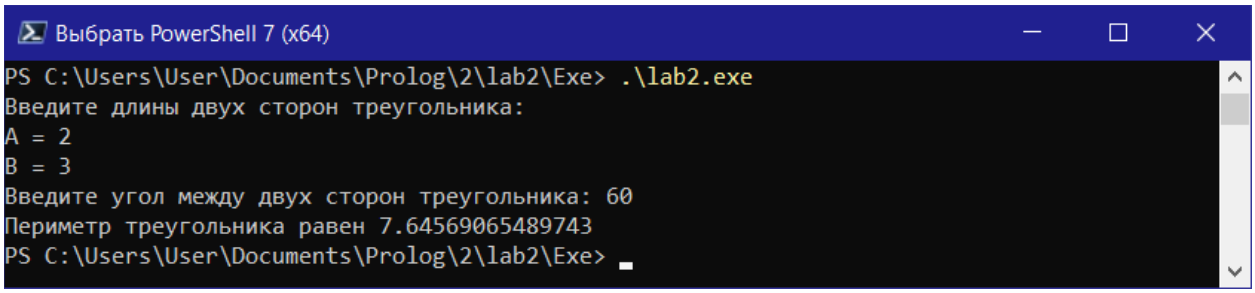
Согласно теореме косинусов находим 3 сторону: $C=\sqrt{A^2+B^2-2AB\cos\alpha}$.

```
myPredicate() :-
	stdio::write("Введите длины двух сторон треугольника: "),
	nl,
	stdio::write("A = "),
```

```
	A = stdio::read(),
	hasDomain(real, A),
	stdio::write("B = "),
	B = stdio::read(),
	hasDomain(real, B),
	stdio::write("Введите угол между двух сторон треугольника: "),
	C = stdio::read(),
	hasDomain(real, C),
	stdio::write("Периметр треугольника равен "),
	stdio::write(A + B + math::sqrt(A ^ 2 + B ^ 2 - 2 * A * B * math::cos(C * 3.1415 / 180))).
```

Стоит обратить внимание, что функция cos() в прологе принимает аргументы в радианах, поэтому необходим перевод в градусы.

При тестировании предиката проверяем стороны треугольника с длинами 2 и 3 и углом между ними 60º, соответственно, периметр треугольника равен $2+3+\sqrt{2^2+3^2-\cos 60}\approx 5+2.646=7.646$.

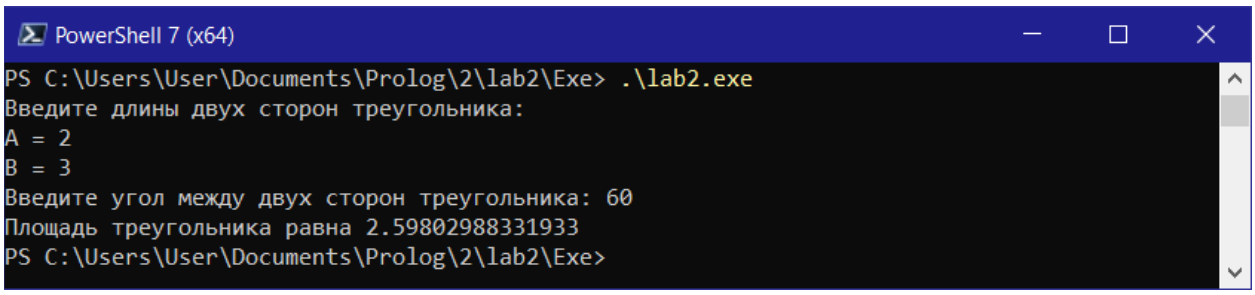


6. Предикат, вычисляющий площадь треугольника по двум сторонам и углу между ними.

Согласно теореме косинусов находим 3 сторону, а площадь по формуле Герона $S=\sqrt{p(p-A)(p-B)(p-C)}$, где A, B, C — длины сторон, а p — полупериметр.

```
myPredicate() :-
	stdio::write("Введите длины двух сторон треугольника: "),
	nl,
	stdio::write("A = "),
	A = stdio::read(),
	hasDomain(real, A),
	stdio::write("B = "),
	B = stdio::read(),
	hasDomain(real, B),
	stdio::write("Введите угол между двух сторон треугольника: "),
	Alpha = stdio::read(),
	hasDomain(real, Alpha),
	C = math::sqrt(A ^ 2 + B ^ 2 - 2 * A * B * math::cos(Alpha * 3.1415 / 180)),
	P = (A + B + C) / 2,
	stdio::write("Площадь треугольника равна "),
	stdio::write(math::sqrt(P * (P - A) * (P - B) * (P - C))).
```

При тестировании предиката проверяем стороны треугольника с длинами 2 и 3 и углом между ними 60º, соответственно, длина 3 стороны треугольника равна $\sqrt{2^2+3^2-\cos 60}\approx 2{,}646$, полупериметр $\frac{2+3+2{,}646}{2}=3{,}823$, а площадь $S=\sqrt{3{,}823(3{,}823-2)(3{,}823-3)(3{,}823-2{,}646)}=2{,}598$.

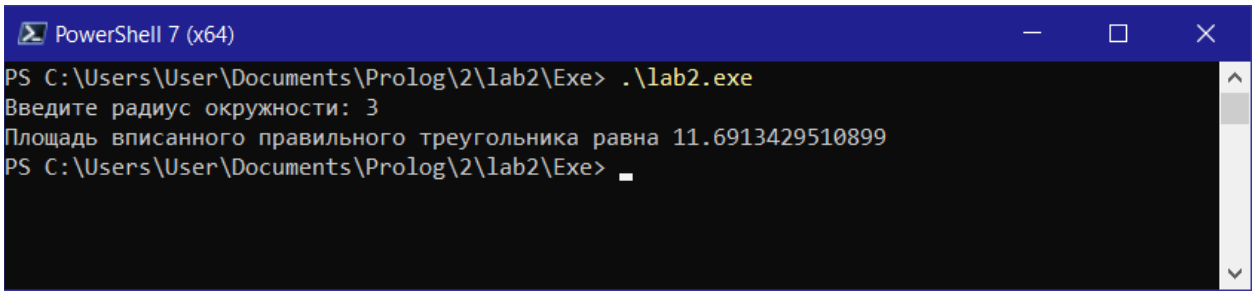


7. Предикат, вычисляющий площадь вписанного правильного треугольника в окружность радиусом R.

Площадь вписанного правильного треугольника в окружность вычисляется по формуле $S=\frac{3\sqrt{3}R^2}{4}$, где R — радиус окружности.

```
myPredicate() :-
	stdio::write("Введите радиус окружности: "),
	R = stdio::read(),
	hasDomain(real, R),
	stdio::write("Площадь вписанного правильного треугольника равна "),
	stdio::write(3 * math::sqrt(3) * R ^ 2 / 4).
```

При тестировании предиката проверяем радиус окружности длиной 3. Получаем площадь $S=\frac{3\sqrt{3}\cdot 3^2}{4}\approx 11{,}69134$.

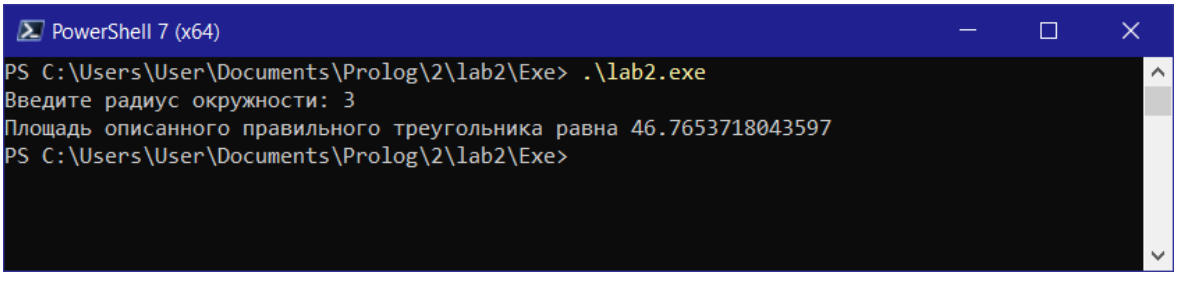


8. Предикат, вычисляющий площадь описанного правильного треугольника вокруг окружности радиусом R.

Площадь описанного правильного треугольника вокруг окружности вычисляется по формуле $S=3\sqrt{3}R^2$.

```
myPredicate() :-
	stdio::write("Введите радиус окружности: "),
	R = stdio::read(),
	hasDomain(real, R),
	stdio::write("Площадь описанного правильного треугольника равна "),
	stdio::write(3 * math::sqrt(3) * R ^ 2).
```

При тестировании предиката проверяем радиус окружности длиной 3. Получаем площадь $S=3\sqrt{3}\cdot 3^2\approx 46{,}76537$.

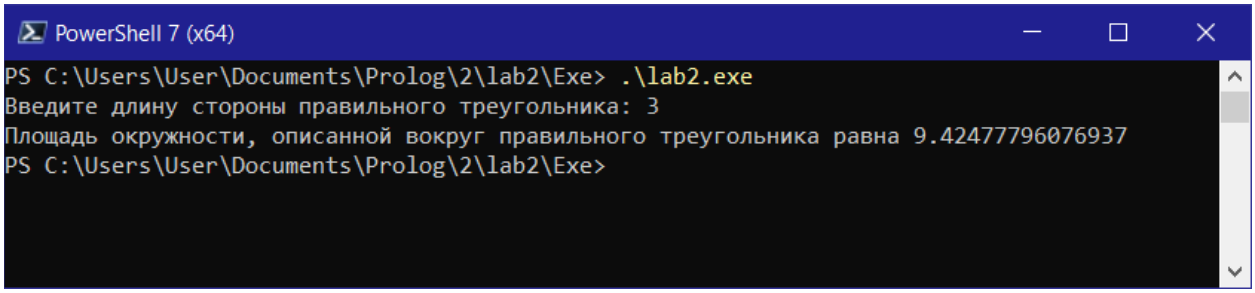


9. Предикат, вычисляющий площадь окружности, описанной вокруг правильного треугольника со стороной a.

Для правильного треугольника со стороной a радиус описанной окружности равен $R=\frac{a}{\sqrt{3}}$, а площадь окружности $S=\pi R^2$, откуда $S=\frac{\pi \cdot a^2}{3}$.

```
myPredicate() :-
	stdio::write("Введите длину стороны правильного треугольника: "),
	A = stdio::read(),
	hasDomain(real, A),
	stdio::write("Площадь окружности, описанной вокруг правильного
треугольника равна "),
	stdio::write(A ^ 2 * math::pi / 3).
```

При тестировании предиката вводим длину стороны треугольника равную 3. Получаем площадь $S=\frac{\pi \cdot a^2}{3} \approx 9{,}42478$.

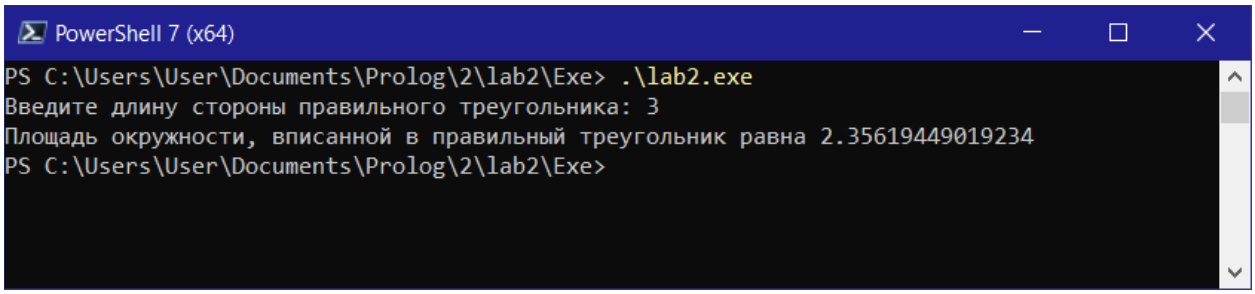


10. Предикат, вычисляющий площадь окружности, вписанной в правильный треугольник со стороной a.

Для правильного треугольника со стороной a радиус вписанной окружности равен $R=\frac{a}{2\sqrt{3}}$, а площадь окружности $S=\pi R^2$, откуда $S=\frac{\pi \cdot a^2}{12}$.

```
myPredicate() :-
	stdio::write("Введите длину стороны правильного треугольника: "),
	A = stdio::read(),
	hasDomain(real, A),
	stdio::write("Площадь окружности, вписанной в правильный треугольник
равна "),
	stdio::write(A ^ 2 * math::pi / 12).
```

При тестировании предиката вводим длину стороны треугольника равную 3. Получаем площадь $S=\frac{\pi \cdot a^2}{12} \approx 2{,}35619$.

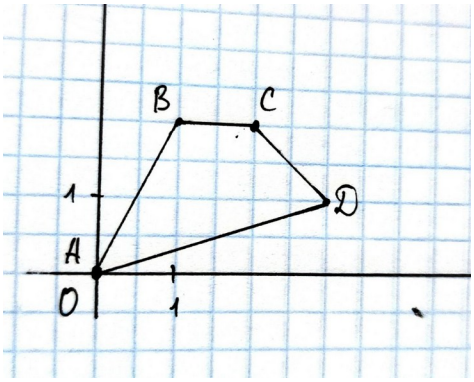


11. Предикат, вычисляющий площадь четырехугольника по четырем точкам с проверкой на выпуклость.

Для проверки четырехугольника на выпуклость воспользуемся свойством, которое говорит, что сумма углов выпуклого четырехугольника равна 360°. Для этого вычислим сумму косинусов всех углов четырехугольника и сравним с 0. Площадь вычислим как сумму площадей двух треугольников, получаемых разбиением четырехугольника по диагонали. Для вычисления косинуса угла и площади треугольника воспользуемся предикатами cosVect и sqrTre из 13-го и 12-го заданий соответственно. Для упрощения положим, что вершины четырехугольника вводятся в порядке по часовой стрелке, т. е. 1-ая и 3-ья вершины являются противолежащими и через них проведена диагональ. Для вычисления площади четырехугольника определим предикат sqrArea(real, real, real, real, real, real, real, real) → real, принимающий значения координат вершин четырехугольника, выводящий сообщение является ли четырехугольник выпуклым и возвращающий вычисленное значение площади. Данный предикат объявляется в секции class predicates, а определяется в секции clauses, код которой приведен ниже:

```
sqrArea(XA, YA, XB, YB, XC, YC, XD, YD) = S :-
    CosA = cosVect(XA, YA, XB, YB, XA, YA, XD, YD), hasDomain(real, CosA),
    CosB = cosVect(XB, YB, XA, YA, XB, YB, XC, YC), hasDomain(real, CosB),
    CosC = cosVect(XC, YC, XB, YB, XC, YC, XD, YD), hasDomain(real, CosC),
    CosD = cosVect(XD, YD, XA, YA, XD, YD, XC, YC), hasDomain(real, CosD),
    if CosA + CosB + CosC + CosD = 0 then
        write("Треугольник  ABCD  - выпуклый!!"), nl
    else
        write("Треугольник  ABCD  НЕ выпуклый (("), nl
    end if,
    S = sqrTre(XA, YA, XB, YB, XC, YC) + sqrTre(XA, YA, XC, YC, XD, YD).
run() :-
    stdio::write("Введите координаты точки A: "), nl,
    write("X = "), XA = read(), hasDomain(real, XA),
    write("Y = "), YA = read(), hasDomain(real, YA),

    write("Введите координаты точки B: "), nl,
    write("X = "), XB = read(), hasDomain(real, XB),
    write("Y = "), YB = read(), hasDomain(real, YB),

    stdio::write("Введите координаты точки C: "), nl,
    write("X = "), XC = read(), hasDomain(real, XC),
    write("Y = "), YC = read(), hasDomain(real, YC),

    stdio::write("Введите координаты точки D: "), nl,
    write("X = "), XD = read(), hasDomain(real, XD),
    write("Y = "), YD = read(), hasDomain(real, YD),

    write(sqrArea(XA, YA, XB, YB, XC, YC, XD, YD)),
    _ = readLine().
```

Тестируем нахождение площади четырехугольника с координатами вершин A(0,0), B(1,2), C(2,2) и D(3,1). Такой четырехугольник является выпуклым, а его площадь равна 3. Схематически четырехугольник имеет вид:




```
PowerShell 7 (x64)
PS C:\Users\User\Documents\Prolog\2\lab2\Exe> .\lab2.exe
Введите координаты точки A:
X = 0
Y = 0
Введите координаты точки B:
X = 1
Y = 2
Введите координаты точки C:
X = 2
Y = 2
Введите координаты точки D:
X = 3
Y = 1
Треугольник  ABCD  - выпуклый!!
3
PS C:\Users\User\Documents\Prolog\2\lab2\Exe>
```

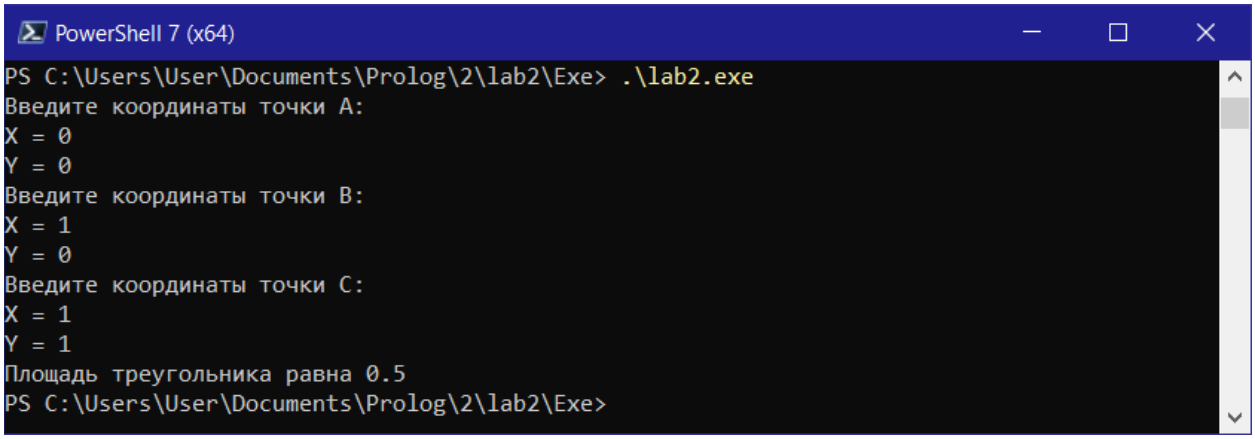
12. Предикат, вычисляющий площадь треугольника по трем точкам.

Площадь треугольника ABC вычисляется по формуле $S_{\Delta ABC}=\frac{1}{2}\cdot AB\cdot AC\cdot \sin A$, где $\sin A=\sqrt{(1-\cos^2 A)}$. Для вычисления длины сторон треугольника используем предикат lenVect из 14-го задания, а для вычисления косинуса угла между двумя векторами — предикат cosVect из 13-го задания. Для вычисления площади треугольника определим предикат sqrTre(real, real, real, real, real, real) → real, принимающий значения координат вершин треугольника и возвращающий вычисленное значение площади. Данный предикат объявляется в секции class predicates, а определяется в секции clauses, код которой приведен ниже:

```
sqrTre(XA, YA, XB, YB, XC, YC) = S :-
    AB = lenVect(XA, YA, XB, YB),
    AC = lenVect(XA, YA, XC, YC),
    CosA = cosVect(XA, YA, XB, YB, XA, YA, XC, YC),
    SinA = math::sqrt(1 - CosA ^ 2),
    S = 1 / 2 * AB * AC * SinA.
run() :-
    stdio::write("Введите координаты точки A: "), nl,
    write("X = "), XA = read(), hasDomain(real, XA),
    write("Y = "), YA = read(), hasDomain(real, YA),

    write("Введите координаты точки B: "), nl,
    write("X = "), XB = read(), hasDomain(real, XB),
    write("Y = "), YB = read(), hasDomain(real, YB),

    stdio::write("Введите координаты точки C: "), nl,
    write("X = "), XC = read(), hasDomain(real, XC),
    write("Y = "), YC = read(), hasDomain(real, YC),
```

```
        write("Площадь треугольника равна "),
        write(sqrTre(XA, YA, XB, YB, XC, YC)),
        _ = readLine().
```

Тестируем нахождение площади треугольника с координатами вершин (0,0), (1,0) и (1,4). Площадь такого треугольника равна 0,5.



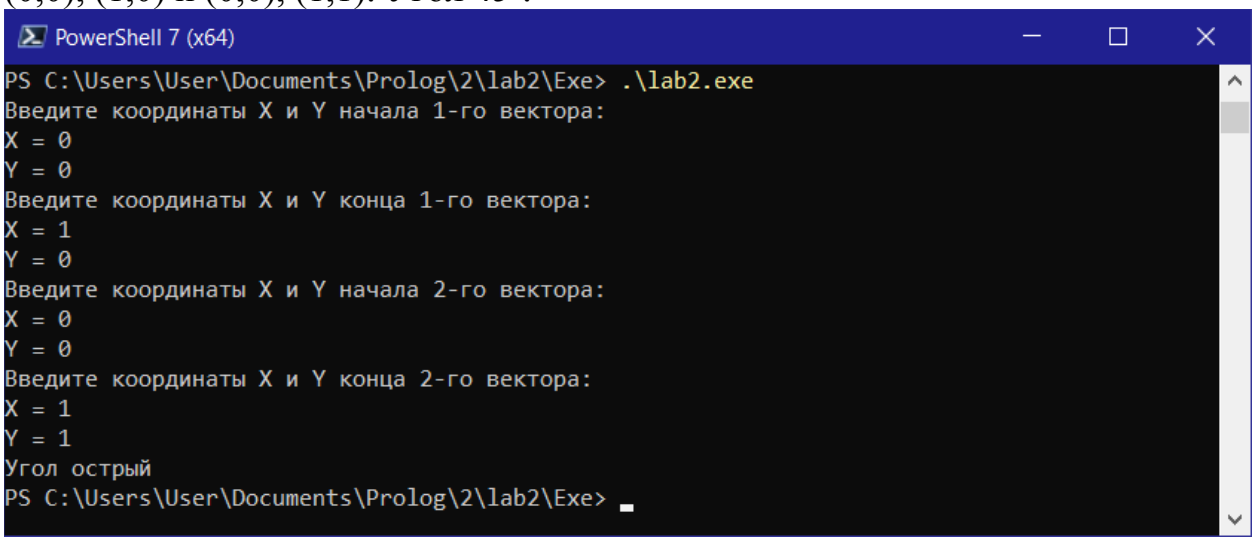
13. Предикат, проверяющий является ли угол между двумя векторами острым, прямым или тупым.

Для определения типа угла будем пользоваться фактом, что косинус 90º равен 0, острого угла — положителен, а тупого — отрицателен. Косинус угла между двумя векторами $\vec{a}=(a_1,a_2)$ и $\vec{b}=(b_1,b_2)$ вычисляется по формуле $\cos(\vec{a},\vec{b})=\frac{a_1b_1+a_2b_2}{|\vec{a}|\cdot|\vec{b}|}$, где (a_1,a_2) и (b_1,b_2) - координаты векторов $\vec{a}$ и $\vec{b}$ соответственно. Координаты вектора равны $\{x_2-x_1;y_2-y_1\}$, где (x_1,y_1) — координаты точки начала вектора, а (x_2,y_2) — конца вектора. Для вычисления длины вектора используем предикат lenVect из 14-го задания. Для вычисления косинуса угла между двумя векторами определим предикат cosVect(real, real, real, real, real, real, real, real) → real, принимающий значения координат точек и возвращающий вычисленное значение. Данный предикат объявляется в секции class predicates, а определяется в секции clauses, код которой приведен ниже:

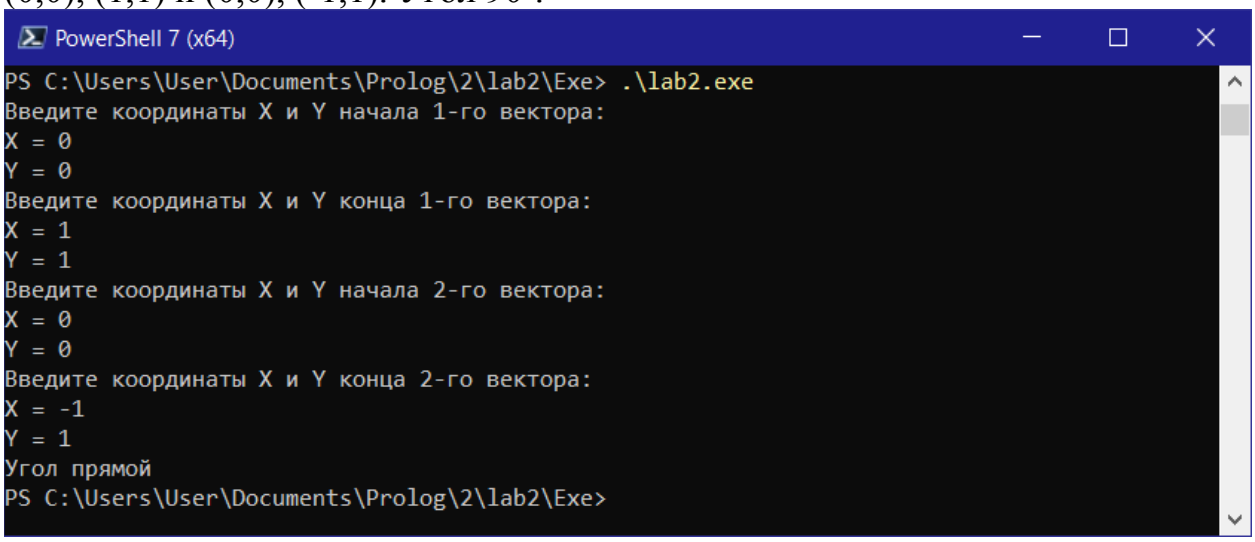
```
    cosVect(XA1, YA1, XA2, YA2, XB1, YB1, XB2, YB2) = C :-
        A1 = XA2 - XA1, A2 = YA2 - YA1,
        B1 = XB2 - XB1, B2 = YB2 - YB1,
        C = (A1 * B1 + A2 * B2) / (lenVect(XA1, YA1, XA2, YA2) * lenVect(XB1,
YB1, XB2, YB2)).
    run() :-
        stdio::write("Введите координаты X и Y начала 1-го вектора: "), nl,
        write("X = "), XA1 = read(), hasDomain(real, XA1),
        write("Y = "), YA1 = read(), hasDomain(real, YA1),
        write("Введите координаты X и Y конца 1-го вектора: "), nl,
        write("X = "), XA2 = read(), hasDomain(real, XA2),
        write("Y = "), YA2 = read(), hasDomain(real, YA2),

        stdio::write("Введите координаты X и Y начала 2-го вектора: "), nl,
        write("X = "), XB1 = read(), hasDomain(real, XB1),
        write("Y = "), YB1 = read(), hasDomain(real, YB1),
        write("Введите координаты X и Y конца 2-го вектора: "), nl,
        write("X = "), XB2 = read(), hasDomain(real, XB2),
```

```
write("Y = "), YB2 = read(), hasDomain(real, YB2),
C = cosVect(XA1, YA1, XA2, YA2, XB1, YB1, XB2, YB2),
hasDomain(real, C),
if C = 0 then write("Угол прямой")
elseif C > 0 then write("Угол острый")
else write("Угол тупой")
end if,
_ = readLine().
```

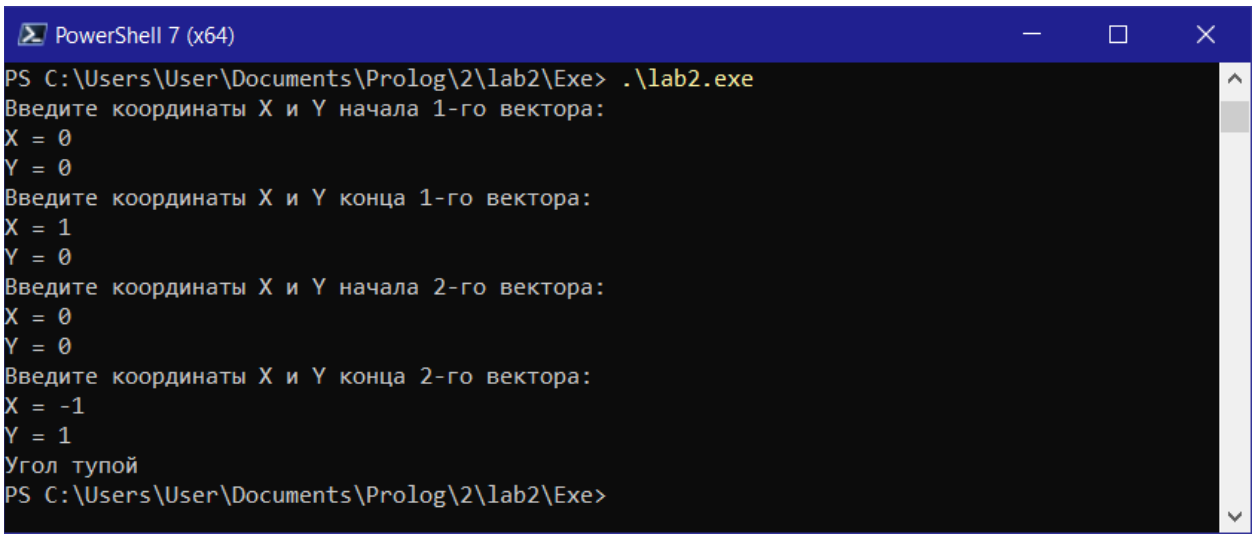
Тестируем определение острого угла между векторами с координатами (0,0), (1,0) и (0,0), (1,1). Угол 45º.



Тестируем определение прямого угла между векторами с координатами (0,0), (1,1) и (0,0), (-1,1). Угол 90º.



Тестируем определение тупого угла между векторами с координатами (0,0), (1,0) и (0,0), (-1,1). Угол 135º.

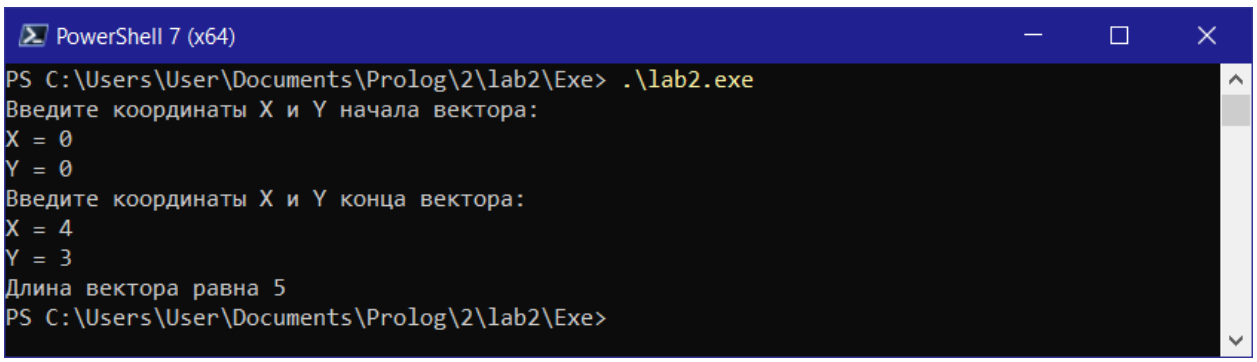


14. Предикат, вычисляющий длину вектора по координатам начала и конца.

Длина вектора по координатам начала и конца вычисляется по формуле $|\overrightarrow{AB}|=\sqrt{(x_B-x_A)^2+(y_B-y_A)^2}$. Для вычисления длины вектора определим предикат lenVect(real, real, real, real) → real, принимающий значения координат точек и возвращающий вычисленное значение. Данный предикат объявляется в секции class predicates, а определяется в секции clauses, код которой приведен ниже:

```
lenVect(X1, Y1, X2, Y2) = math::sqrt((X2 - X1) ^ 2 + (Y2 - Y1) ^ 2).
run() :-
    stdio::write("Введите координаты X и Y начала вектора: "), nl,
    write("X = "), X1 = read(), hasDomain(real, X1),
    write("Y = "), Y1 = read(), hasDomain(real, Y1),
    write("Введите координаты X и Y конца вектора: "), nl,
    write("X = "), X2 = read(), hasDomain(real, X2),
    write("Y = "), Y2 = read(), hasDomain(real, Y2),
    write("Длина вектора равна "),
    write(lenVect(X1, Y1, X2, Y2)),
    _ = readLine().
```

При тестировании предиката вводим координаты (0,0) начала вектора, координаты (4,3) конца вектора. Получаем длину вектора $|\overrightarrow{AB}|=\sqrt{(4-0)^2+(3-0)^2}=5$.

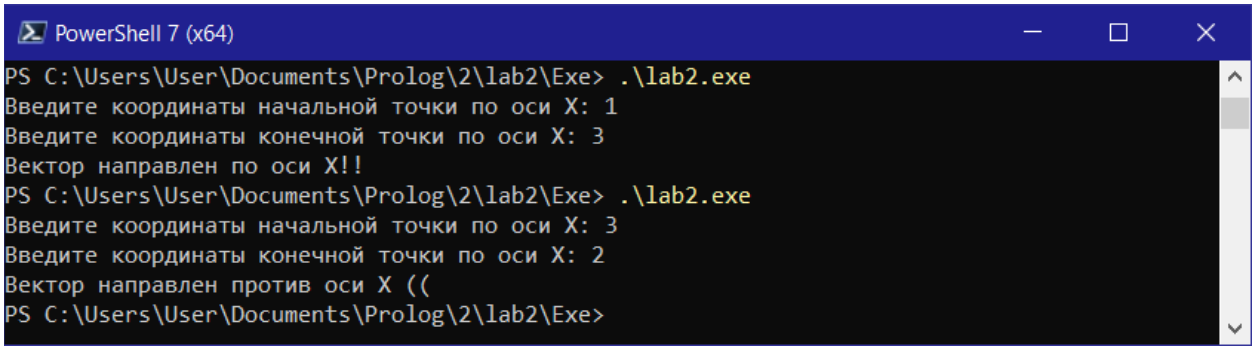


15. Предикат, проверяющий направлен ли вектор по оси X или против.

Пусть вектор задается координатами начальной и конечной точки в трехмерной системе координат. Тогда в случае направленности вектора по оси X его проекция на ось X будет положительной, в обратном случае отрицательной. Определим это по разности координат по оси X конечной и начальной точки вектора. Для вычисления координаты по оси X вектора определим предикат proX(real, real) → real, принимающий значения координат точек по оси X начальной и конечной точки и возвращающий вычисленное значение. Данный предикат объявляется в секции class predicates, а определяется в секции clauses, код которой приведен ниже:

```
proX(X1, X2) = X2 - X1.
run() :-
    stdio::write("Введите координаты начальной точки по оси X: "),
    X1 = read(),
    hasDomain(real, X1),
    stdio::write("Введите координаты конечной точки по оси X: "),
    X2 = read(),
    hasDomain(real, X2),
    if proX(X1, X2) > 0 then
        write("Вектор направлен по оси X!!"),
        nl
    else
        write("Вектор направлен против оси X (("),
        nl
    end if,
    _ = readLine().
```

Тестируем определение направления вектора со значениями координат 1 и 3, а так же 3 и 2 по оси X.



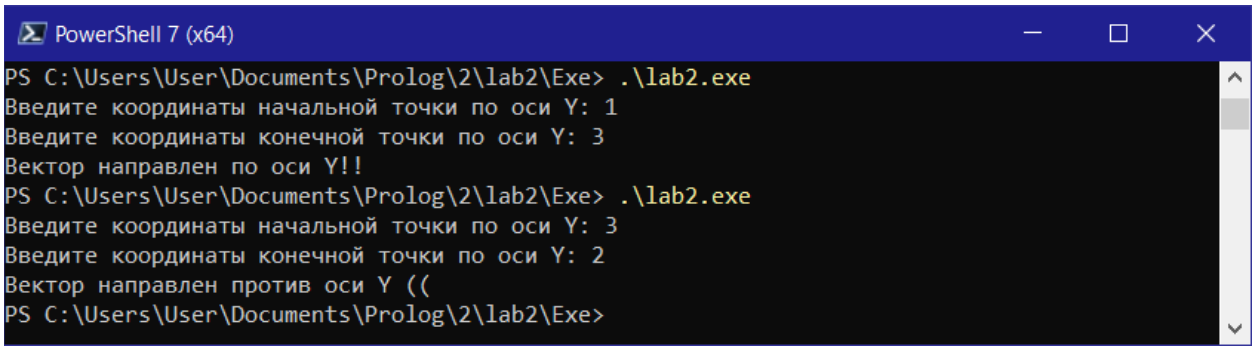
16. Предикат, проверяющий направлен ли вектор по оси Y или против.

Пусть вектор задается координатами начальной и конечной точки в трехмерной системе координат. Тогда в случае направленности вектора по оси Y его проекция на ось Y будет положительной, в обратном случае отрицательной. Определим это по разности координат по оси Y конечной и начальной точки вектора. Для вычисления координаты по оси Y вектора определим предикат proY(real, real) → real, принимающий значения координат точек по оси Y начальной и конечной точки и возвращающий

вычисленное значение. Данный предикат объявляется в секции class predicates, а определяется в секции clauses, код которой приведен ниже:

```
proX(X1, X2) = X2 - X1.
run() :-
    stdio::write("Введите координаты начальной точки по оси Y: "),
    Y1 = read(),
    hasDomain(real, Y1),
    stdio::write("Введите координаты конечной точки по оси Y: "),
    Y2 = read(),
    hasDomain(real, Y2),
    if proY(Y1, Y2) > 0 then
        write("Вектор направлен по оси Y!!"),
        nl
    else
        write("Вектор направлен против оси Y (("),
        nl
    end if,
    _ = readLine().
```

Тестируем определение направления вектора со значениями координат 1 и 3, а так же 3 и 2 по оси Y.



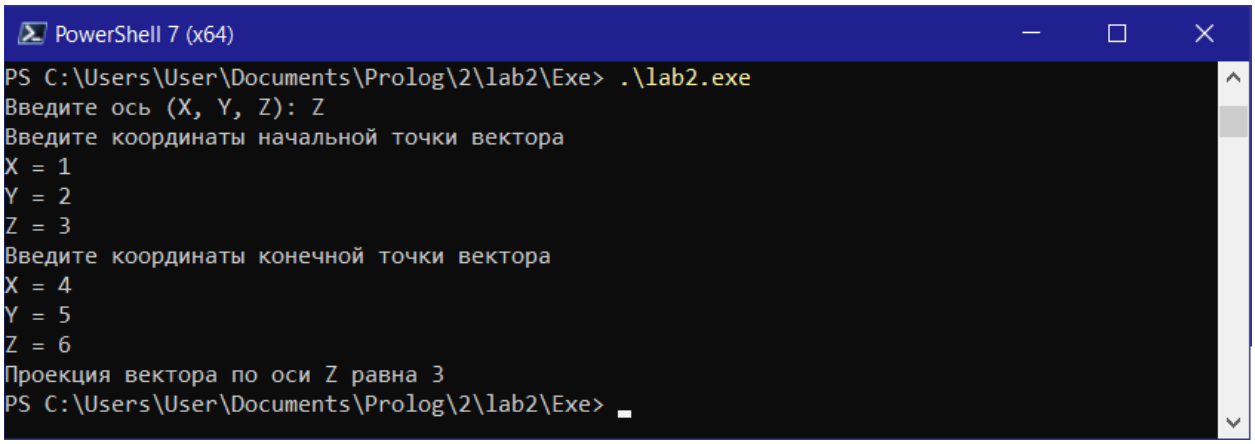
17. Предикат, определяющий проекцию вектора по заданной оси.

Пусть вектор задается координатами начальной и конечной точки в трехмерной системе координат. Для вычисления проекции вектора на заданную ось используем предикаты proX, proY и proZ из 15-го, 16-го и 18-го заданий. Для вычисления проекции вектора определим предикат proVect(real, real, real, real, real, real, real, real) → real, принимающий значения координат точек и возвращающий вычисленное значение. Данный предикат объявляется в секции class predicates, а определяется в секции clauses, код которой приведен ниже:

```
proVect(X1, Y1, Z1, X2, Y2, Z2, Axis) = P :-
    if Axis = "X" then
        write("Проекция вектора по оси ", Axis, " равна "),
        P = proX(X1, X2)
    elseif Axis = "Y" then
        write("Проекция вектора по оси ", Axis, " равна "),
        P = proY(Y1, Y2)
    elseif Axis = "Z" then
        write("Проекция вектора по оси ", Axis, " равна "),
        P = proZ(Z1, Z2)
    end if.

run() :-
    stdio::write("Введите ось (X, Y, Z): "),
```

```
    AX = readLine(),

    if AX = "X" or AX = "Y" or AX = "Z" then
        stdio::write("Введите координаты начальной точки вектора"), nl,
        write("X = "), X1 = read(), hasDomain(real, X1),
        write("Y = "), Y1 = read(), hasDomain(real, Y1),
        write("Z = "), Z1 = read(), hasDomain(real, Z1),

        stdio::write("Введите координаты конечной точки вектора"), nl,
        write("X = "), X2 = read(), hasDomain(real, X2),
        write("Y = "), Y2 = read(), hasDomain(real, Y2),
        write("Z = "), Z2 = read(), hasDomain(real, Z2),

        PR = proVect(X1, Y1, Z1, X2, Y2, Z2, AX),
        write(PR)
    else
        write("Ось задана неверно!!")
    end if,
    _ = readLine().
```

Тестируем вычисление проекции вектора с началом в точке (1, 2, 3) и окончанием в (4, 5, 6) по оси Z.



18. Предикат, проверяющий направлен ли вектор по оси Z или против.

Пусть вектор задается координатами начальной и конечной точки в трехмерной системе координат. Тогда в случае направленности вектора по оси Z его проекция на ось Z будет положительной, в обратном случае отрицательной. Определим это по разности координат по оси Z конечной и начальной точки вектора. Для вычисления координаты по оси Z вектора определим предикат proZ(real, real) → real, принимающий значения координат точек по оси Z начальной и конечной точки и возвращающий вычисленное значение. Данный предикат объявляется в секции class predicates, а определяется в секции clauses, код которой приведен ниже:

```
proZ(Z1, Z2) = Z2 - Z1.
run() :-
    stdio::write("Введите координаты начальной точки по оси Z: "),
    Z1 = read(),
    hasDomain(real, Z1),
    stdio::write("Введите координаты конечной точки по оси Z: "),
    Z2 = read(),
    hasDomain(real, Z2),
```

```
if proZ(Z1, Z2) > 0 then
    write("Вектор направлен по оси Z!!"),
    nl
else
    write("Вектор направлен против оси Z (("),
    nl
end if,
_ = readLine().
```

Тестируем определение направления вектора со значениями координат 1 и 3, а так же 3 и 2 по оси Z.