2. ОСНОВНИ ОПЕРАЦИИ

2.1. Операции

За всеки тип данни са позволени определени операции. Величините (константи и променливи), които участвуват в една операция, се наричат *операнди*. В следващата таблица са дадени допустимите операции в програмите на Паскал, допустимите типове на операндите в тях и резултата от всяка операцията.

Таблица 1

Операция		Знак на операция	Тип на операндите	Тип на резултата
Аритметични	Събиране	+	Реален или целочислен	Целочислен, когато и двата операнда са целочислени, иначе е реален
	Изваждане	-		
	Умножение	*		
	Деление	/		Реален
	Целочислено деление	div	Целочислен	Целочислен
	Остатък от целочислено деление	mod		
Логически	НЕ	not	Логически	Логически
	ИЛИ	or		
	И	and		
Сравнения	Равно	=	Всички скаларни типове	Логически
	Различно	<>		
	По-голямо	>		
	По-малко	<		
	По-голямо или равно	>=		
	По-малко или равно	<=		

При използване на операциите трябва да се съблюдава следното:

1. Всички операции, с изключение на операцията **not**, изискват два операнда, които трябва да са от един и същ тип. При операциите +, -, *, / и всички операции за сравняване се допуска единият операнд да е от реален тип, а другият да е от целочислен тип.

2. Операцията **div** дава целочислен резултат, който е равен на цялата част на частното на двата операнда.

3. Операцията **mod** дава целочислен резултат, който е равен на остатъка при целочисленото деление.

4. Сравняването на величини от логически и символен тип е възможно поради това, че съвкупността от допустимите стойности на всеки тип представлява наредено множество. Изразът за сравнение 'A'<'D' има стойност True защото 'A' предхожда 'D'.

5. Операцията **not** се записва във вида:

not A,

където A е операнд от логически тип. Тя дава като резултат стойност, която е противоположна на стойността на A.

6. Операцията **and** се записва във вида:

A **and** B,

където A и B са операнди от логически тип. Тя дава като резултат стойност True само когато и двата операнда са със стойност True. В останалите случаи резултатът е False.

7. Операцията **or** се записва във вида:

A **or** B,

където A и B са операнди от логически тип. Тя дава като резултат стойност False само когато и двата операнда са със стойност False. В останалите случаи резултатът е True.

2.2. Вградени функции

Някои по-сложни операции в Паскал са реализирани като вградени (стандартни) функции. В следващата таблицата са дадени тези функции, техните означения, типовете на операндите и на резултата.

Таблица 2

Функция	Обръщение	Тип на аргумента	Тип на резултата
$\lvert x \rvert$	abs (x)	Реален или	Съвпада с типа
x^2	sqr(x)	целочислен	на операнда
$\sqrt{x}$	sqrt(x)	Реален или целочислен	Реален
sin x	sin(x)		
cos x	cos(x)		
arctg x	arctan(x)		
e^x	exp(x)		
ln x	ln(x)		
Отделяне на цяла част	trunc(x)	Реален	Целочислен
Закръгляне	round(x)	Реален	Целочислен
Проверка за нечетност	odd(x)	Целочислен	Логически
Символ по зададен номер	chr(n)	Целочислен	Символен
Номер на зададена стойност	ord(x)	Дискретен	Целочислен
Предна стойност	pred(x)		Съвпада с типа
Следваща стойност	succ(x)		на операнда

При използване на вградените функции трябва да се съблюдават следните ограничения и правила:

1. Функцията sqrt(x) изисква неотрицателен аргумент.

2. Функциите sin(x) и cos(x) изискват аргумент в радиани.

3. Функцията ln(x) изисква положителен аргумент.

4. Функцията arctan(x) връща резултата в радиани.

5. Функцията trunc(x) отхвърля дробната част на аргумента и връща цялата му част като целочислена стойност.

6. Функцията round(x) закръгля стойността на аргумента до цяло число.

7. Функцията odd(x) връща като резултат стойност True, когато аргументът е нечетно число, и False, когато е четно число.

8. Функцията chr(n) връща символа, съответстващ на номер (код), посочен като аргумент (0 <= n <= 255).

9. Функцията ord(x) връща номера на зададена като аргумент символна константа или променлива.

10. Функцията pred(x) връща като резултат стойността, предшестваща стойността на аргумента в множеството от допустими стойности за съответния тип. Ако аргументът няма предшественик, резултатът е неопределен.

11. Функцията succ(x) връща като резултат стойността, следваща стойността на аргумента в множеството от допустими стойности за съответния тип. Ако аргументът няма наследник, резултатът е неопределен.

2.3. Изрази

Изразът представлява последователност от операции. В частен случай той може да се състои само от една константа, една променлива или една функция. Операциите се изпълняват не според реда на записването, а съгласно определен приоритет. В Паскал *приоритетът* на операциите е следния:

Операция	Приоритет
Обръщение към функция	5
not	4
*, /, div, mod, and	3
+, -, or	2
Сравнения: =, <>, <, <=, >=, >	1

Операциите с по-висок приоритет се изпълняват по-напред. Ако в един израз са записани последователно операции с еднакъв приоритет, те се изпълняват в реда, в който се срещат при обхождане на израза от ляво на дясно. За промяна на приоритета на операциите се използват скоби. Използват се ***само малки скоби,*** независимо от нивото на вложеност на скобите. Всяка операция от израза има резултат от определен тип (табл.2.1 и табл.2.2).

Типът на резултата от последната изпълнена операция определя типа на израза.

Примери:

1. 15 **div** 3

Изразът е от целочислен тип и резултатът 5 е от цял тип.

2. 15 **div** 3 = 2

Най-напред ще се изпълни операцията div, и тя ще даде резултат 5, след това - операцията сравнение 5 = 2, а тя ще даде резултат false. Тъй като последната изпълнена операция е логическа, изразът е от логически тип.

3. 15 >10

Изразът е от логически тип, а резултатът е true.

4. (12 > 10) **and** (15 - 5 < 20)

Изразът е от логически тип, а резултат е true.

Най-напред ще се извършат действията в скобите. От първите скоби ще се получи true. Във вторите скоби първо ще се изпълни операцията изваждане като операция с по-висок приоритет и след това - операцията сравнение. Резултатът от сравнението е true. Последна ще се изпълни операцията **and** и резултатът от нея е също true, тъй като и двата й операнда имат стойности true.

5. 10 **mod** 3 * 4

Операциите **mod** и умножение са с един и същ приоритет и ще се изпълнят в реда на записването. Първо ще се изпълни операцията mod и резултатът 1 ще бъде от целочислен тип. След това ще се изпълни операцията умножение и резултатът 4 ще бъде също от целочислен тип. Следователно изразът е от целочислен тип.

6. (6 + 2) / (5 - 2.5)

Ще се получи резултат от реален тип: 3.2

Най-напред ще се изпълни събирането в скобите, след това изваждането във вторите скоби и накрая делението, което дава резултат от реален тип 3.2. Изразът е от реален тип.

7. trunc(3.2 * 4) **div** 3

Най-напред ще се изпълни умножението в скобите, след това - функцията trunc и накрая - целочисленото деление. Ще се получи резултат 4 от цял тип.

8. chr(5 * 13)

Най-напред ще се изпълни умножението в скобите, след това - функцията chr. Ще се получи като резултат символа A. Изразът е от символен тип.

9. 11 **mod** 4 **div** 2 <>0

Най-напред ще се намери остатъкът от делението 11 на 4, който е 3. След това - цялата част от делението 3 на 2, която е 1. Накрая ще се изпълни операцията сравнение за неравенство и тъй като 1 не е равно на 0, ще получи като краен резултат true. Изразът е от логически тип.

10. (3>7) **or** ('A'< 'L') **and** ('9'>='8')

Най-напред ще се изпълнят изразите в скобите - стойността на израза в първите скоби е False, стойността на израза в вторите скоби е true, защото в ASCII номерът на символа 'A' е по-малък от номера на символа 'L', и стойността на израза в третите скоби е true, защото в ASCII номерът на символа '9' е по-голям от номера на символа '8'. След това ще се изпълни операцията **and**, тъй като тя е с по-висок приоритет от операцията **or**. Резултатът е true, защото и двата операнда на тази операция са със стойност true. Накрая ще се изпълни операцията **or**. Резултатът от нея също ще бъде, защото единият от операндите й е със стойност true. Изразът е от логически тип.

2.4. Оператор за присвояване

Общ вид на оператора:

Променлива := израз;

Променливата и изразът трябва да са от един и същ тип. Единствено изключение е, че на променлива от реален тип може да се присвои стойност на израз от цял тип. Присвояването е разрешена операция за всички типове данни с изключение на типа файл.

Действието на оператора се състои в изчисляване на израза и присвояване на получената стойност на променливата. По този начин в мястото от паметта на компютъра, заделено за променливата, се записва изчислената стойност.

Примери:

Нека са описани променливите I, J и M от цял тип, K - от реален тип, C - от символен тип, L, L1 и L2 - от логически тип и операторите:

```
I    := 10;
C    := 'X';
J    := I + 5;
```

```
K   := J / 8.0 + 3;
M   := J div 5 - 2;
I   := I + 2;
L   := true;
L   := L and (J >M);
L1  := (I <0) or (I = -10);
L2  := (C >= 'A') and (C <= 'Z');
```

След изпълнение на тези оператори за присвояване променливите ще имат следните стойности:

C = 'X', J= 15, K = 4.875, M = 1, I = 12, L = true, L1 = false, L2 = true.

2.5. Въвеждане и извеждане на данни

За въвеждане и извеждане на данни се използват множество различни входни и изходни устройства - клавиатури и екрани, печатащи устройства, устройства за работа с магнитни ленти, магнитни дискове и др. Използването им дава възможност данни, получени и записани от една програма, да бъдат използвани от друга, и по този начин освобождава потребителя от необходимостта да въвежда данните от клавиатурата при работа с всяка програма.

Въвежданите и извежданите от различни входно/изходни устройства данни, могат да се разглеждат като ***входни и изходни потоци***. Тук накратко ще бъде разгледано въвеждането на данни от клавиатурата и извеждането на данни на екрана, където входният и изходният потоци се състоят от редове.

2.5.1. Въвеждане на данни от клавиатурата

Въвеждането на данни от клавиатурата представлява задаване стойности на променливи посредством клавиатурата. Операторите за въвеждане на данни от клавиатурата са:

Read (Входен списък);

Readln (Входен списък);

Readln,

където Read (прочети), Readln (прочети и премини на следващ ред) са запазени думи, а *Входен списък* представлява списък от променливи, разделени със запетаи. Променливите могат да бъдат от целочислен, реален и символен тип и от тип символен низ.

Например операторите

Read(Xn,Yn) и Readln(Xk,Yk)

са предназначени за въвеждане на стойностите на Xn, Yn, Xk и Yk.

За да се изпълнят тези два оператора трябва от клавиатурата да прибавим към входния поток от данни стойностите на Xn, Yn, Xk и Yk. Четирите стойности могат да бъдат разположени на един ред или на няколко реда. След стойността на Yk редът трябва да завършва, защото операторът Readln след прочитане на тази стойност ще приключи с четенето от този ред, следващ оператор вече ще чете от нов ред.

Редовете от входния поток се въвеждат един по един. Не е необходимо стойностите на всички променливи от даден входен списък да са включени в един и същи ред. С натискането на клавиша Enter в реда се извежда знака за край на ред, който е невидим, редът завършва и компютърът веднага прочита съдържащите се в него стойности и ги присвоява на съответните променливи от

входния списък. Ако те са недостатъчни за всички променливи от списъка, компютърът очаква нови стойности в нови редове.

Редовете могат да съдържат до 127 символа. Във всеки ред може да има цели числа, реални числа, символи и символни низове. Те трябва да са подредени в същата последователност, в която са подредени съответстващите им променливи във входния списък. Между променливите от входния списък и стойностите им във входния поток трябва да има и съответствие по тип. Допуска се само едно изключение - на променлива от реален тип във входния списък да съответства целочислена стойност във входния поток.

Броят на символите на стойност от тип символен низ трябва да е равен на декларираната дължина на съответната променлива. Ако е по-малък, стойността трябва да се допълни със съответен брой празни позиции.

След стойност на числова променлива в реда трябва да има поне една празна позиция за отделянето ѝ от стойността на следващата променлива от същия входен списък. Символите и символните низове не трябва да се разделят с празни позиции. Когато пред символ има празна позиция, тя се приема за стойност на променливата от символен тип от входния списък, а самият символ ще бъде прочетен като стойност на следващата променлива от входния списък и от тук нататък се нарушава съответствието между променливите във входния списък и стойностите в реда и всичко се обърква. Същото става и когато в реда се очаква символ, а се появи знак за край на ред. Ето защо трябва да се избягва във входен списък нечислова променлива да следва числова. Ако обаче това не може да се избегне, трябва във входния списък да включим след числовата променлива някаква измислена променлива от символен тип, която да прочете като своя стойност празната позиция между числовата и нечисловата стойности и по този начин да се запази съответствието между променливи и стойности.

Операторите Read(Xn,Yn) и Readln(Xk,Yk) се изпълняват по различен начин. Първият оператор прочита като стойности на Xn и Yn поредните две числа от реда със стойности и запазва възможността следващият оператор да прочете следващи числа от същия ред. Вторият оператор прочита следващите две числа от реда със стойности като стойности на Xk и Yk и приключва с четенето от този ред. Ако след стойностите на Xk и Yk има други стойности, те ще останат непрочетени. Следващ оператор за четене може да чете само от началото на нов ред.

Няколко примера:

1. Нека в оператора

```
Readln(XCoor,YCoor,R,N);
```

XCoor , YCoor и R са реални променливи, а N е целочислена променлива. Във входния поток първите три числа могат да бъдат реални или целочислени, докато четвъртото число задължително трябва да е целочислено. Не е съществено на колко реда ще бъдат разположени тези четири числа, но е важно реалните числа да са преди цялото, тъй като в списъка на оператора Read най-напред са записани имената на реалните променливи, а след това - на цялата. Един възможен начин за оформяне на входния поток в този случай е следният:

```
-25.3  50.7  20  3
```

Изпълнението на оператора за въвеждане започва с търсене във входния поток на стойност за първата променлива от списъка. Търсенето започва от мястото във входния поток, където е приключил предният оператор за въвеждане (четене). Четенето чрез оператора Read може да приключи по средата на реда и тогава следващият оператор (Read или Readln) ще чете от продължението на същия ред.

2. Нека N, XP1, XP2, YP1 и YP2 са числови променливи, които се въвеждат чрез операторите:

```
Readln(N); Read(XP1, YP1, XP2, YP2);
```

Ако входният поток е 5 25.7 32 30.4 -29, то първият оператор ще въведе числото 5 като стойност на N и ще подготви четенето със следващия оператор да започне от следващ ред, т.е. ще прескочи остатъка от реда и за XP1, YP1, XP2 и YP2 няма да се прочетат числата 25.7, 32, 30.4 и -29.

Очевидно числото 5 трябва да е на отделен ред, т.е. входният поток може да бъде оформен коректно например по следния начин:

```
5
25.7 32 30.4 -29.
```

3. Нека да напишем оператор за въвеждане от входен поток

```
125.32  42 QZ-35
```

за реалните променливи Vreal1 и Vreal2 съответно стойностите 125.32 и 42.0, за символните променливи Vchar1 и Vchar2 съответно стойностите Q и Z и за цялата променлива Vint - стойността -35.

Особеното на този входен поток е, че между числото 42 и символа Q има празна позиция. Входният оператор трябва да има вида:

```
Readln(Vreal1, Vreal2, C, Vchar1, Vchar2, Vint);
```

където C е символна променлива и служи само за прочитане на празната позиция след числото 42 или

```
Readln(Vreal1, Vreal2, Vchar1, Vchar1, Vchar2, Vint);
```

При такава ситуация често стойностите се групират в отделни редове, в случая например така:

```
125.32  42
QZ-35
```

и се въвеждат със следните два оператора:

```
Readln(Vreal1, Vreal2);
Readln(Vchar1, Vchar2, Vint);
```

2.5.2. Извеждане на данни на екрана

Данни се извеждат на екрана посредством операторите:

Write*(Изходен списък);*

Writeln*(Изходен списък);*

Writeln;

Изходният списък може да включва изрази (константа, променлива и функция са частни случаи на израз), разделени със запетаи. Изразите могат да бъдат от цял, реален, символен и логически тип и от тип символен низ.

Изпълнението на оператора се заключава в изчисляване стойностите на изразите от списъка и извеждане на изчислените стойности на екрана, подредени в редове. Максималната дължина на един екранен ред е 80 символа. Чрез оператора Writeln може да се извеждат и по къси редове. С изпълнението на такъв оператор завършва формираният до този момент ред и следващият оператор за извеждане започва да извежда на нов ред. Оператор Writeln без изходен списък не извежда нищо на екрана, а само прекратява извеждания изходен ред. Следващ оператор за извеждане извежда стойностите на изразите от изходния си списък на нов ред. Ако този следващ оператор е отново без

списък, той прекратява реда празен и минава на следващ, т.е. извежда на екрана един празен ред.

За всеки израз от изходния списък се отделя поле, чиято ширина зависи от неговия тип. Ширината на полето (в Турбо Паскал) за стойности на изрази от целочислен, символен и логически тип и тип символен низ е такава, каквато е необходима за самата стойност. Стойностите на изразите от реален тип се извеждат във формат с плаваща точка, т.е. във вида

x.xxxxxxxxxxxExx,

като броят на цифрите в мантисата съответства на конкретния подтип.

Няколко примера:

1. Write(A,B);

Извежда стойностите на A и B една след друга без никакъв разделител между тях.

2. Write('A*B е равно на ',A*B);

Извежда текстовата константа такава каквато е (разбира се без апострофите в двата края) и плътно след нея извежда стойността на израза A*B в съответствие с неговия тип.

3. Write(M=N);

Изразът е от логически тип. Ще се намери стойността му (true или false в зависимост от това дали M е равно на N или не) и ще се изведе.

4. Writeln('Стойности на X и Y:'); Write('X=',X,', Y=',Y);

Първият оператор ще изведе константата от тип символен низ 'Стойности на X и Y:' и ще прекрати реда. т.е. на екрана ще се появи следният текст:

Стойности на X и Y:

Ако X и Y са целочислени със стойности съответно 25 и -100, вторият оператор ще изведе следния следващ ред:

X=25, Y=-100

Съществува възможност програмистът да форматира изхода, като предоставя за стойностите на изразите в изходния списък полета, различни, обикновено по- големи, от необходимите за самите стойности. Така може да се изведе колона от различно големи числа подравнени по последната цифра. Размер на полето за стойността на израз се задава както следва:

Израз : брой позиции на полето,

където броят позиции се указва с израз от цял тип.

Например

Writeln(A:8, B:15, C*D:12);

Когато полето е по-голямо от необходимото, стойностите на изразите се разполагат в дясната част на същото, а когато е по-малко, то автоматично се увеличава до необходимото за целите, логическите символните и символните низове, а при реалните се съкращава броят на цифрите в мантисата. Обикновено се задават полета, по-големи от необходимите, за да остават празни позиции пред извежданите стойности, които служат като разделители.

Реалните числа, изведени във формат с плаваща точка, са непрегледни. Затова е предвидена възможност за заявяване извеждането им и с фиксирана точка. Това става по следния начин:

Израз : брой позиции на полето : брой позиции за дробната част,

където броят позиции се указва с израз от цял тип.

Примери:

1. Writeln(A:8:3,B:5);

Ако променливата A е реална, а B - цяла и техните стойности са 22,5 и 34, ще се изведе ред със следното съдържание:

22.500 34,

т.е. редът ще съдържа: две празни позиции (Защо?), числото 22.500, три празни позиции (Защо?), числото 34.

2. Writeln('A*B е равно на ',A*B:10:3);

Ако променливите и техните стойности са същите както в предния пример, ще се изведе следният ред:

A*B е равно на 765.000

3. Write('Въведете стойностите на A,B и C: '); Readln(A,B,C);

Първият оператор ще изведе константата от тип символният низ, курсорът ще остане в края на изведения текст и от тук нататък може да започне редът със стойности от входния поток. След задаване на стойностите, както е показано по-долу, и натискане на клавиша Enter ще се прочетат стойностите и курсорът ще премине на нов ред. Следващ оператор за въвеждане или извеждане започва от нов ред.

На екрана това изглежда така:

Въведете стойностите на A, B и C: 50 25.55 -3.75

В този случай входният и изходният поток се смесват и Readln и Writeln предизвикват преминаване на нов ред, без да се държи сметка дали следващото действие е въвеждане или извеждане.

2.6. Програма с линейна структура

Разгледаните до тук оператори са достатъчни за съставяне на програми с линейна структура. Такива програми обикновено реализират въвеждане на входни данни от клавиатурата, пресмятане на определени изрази и присвояване резултатите на определени променливи и извеждане на получените резултати на екрана. Ще разгледаме няколко примера.

Пример 2.1. Програма, която по зададени радиус и височина на цилиндър пресмята обема и лицата на околната и пълната повърхнини на цилиндъра.

С променливите R и H са означени радиусът и височината на цилиндъра, с S1 и S2 - съответно лицата на околната и пълната повърхнина на цилиндъра и с V - обема му. Всички променливи са от реален тип, тъй като съответните им величини могат да бъдат произволни реални числа. Стойността на π е известна в Турбо Паскал като стандартна константа от реален тип с име Pi.

```
1  Program PovObCyl;
2   Var  R, H, V, S1, S2,B : real;
3   Begin
4   Write('Задайте радиуса: '); Readln(R);
5   Write('Задайте височината: '); Readln(H);
6   S1 := 2 * Pi * R * H;
7   B := Pi*sqr(R);
8   S2 := S1 + 2*B;
9   V := B * H;
10 Writeln; Writeln;
11 Writeln(' ':10,'ВХОДНИ ДАННИ:');
12 Writeln(' ':12,'- радиус.....',R:7:2);
```

```
13 Writeln(' ':12,'- височина....',H:7:2);
14 Writeln;
15 Writeln(' ':10,'РЕЗУЛТАТИ:');
16 Writeln(' ':12,'- околна повърхнина...',S1:8:2);
17 Writeln(' ':12,'- пълна повърхнина.....',S2:8:2);
18 Writeln(' ':12,'- обем.....',R:8:2);
19 Readln
20 End.
```

В програмите на Паскал не е допустимо номериране на редовете. Тук то е направено за удобство при коментара на програмата, който следва.

Ред 1 е заглавието на програмата. То съдържа името PovObCyl. По-нататък ще пропускаме заглавието, тъй като за ТУРБО ПАСКАЛ то не е задължително.

Ред 2 представя описателната част на програмата и дефинира използваните в програмата променливи - те са две входни (R и H), три изходни (S1, S2 и V) и една помощна (B).

Редовете с номера 3 ..19 представляват изпълнимата част на програмата.

Редове 4 и 5 съдържат по два оператора - първият извежда подсказващ текст, който съобщава на потребителя какво да направи, за да се изпълни вторият оператор, т.е. за да се въведе съответната стойност. Първият оператор е write, а не writeln, за да остане курсорът в края на подсказващото съобщение и въведената стойност да се появи непосредствено след него.

Редове 6 .. 9 съдържат операторите, които извършват необходимите пресмятания.

Редове 10 .. 18 извеждат данните и резултатите в прегледен вид.

Ред 19 съдържа оператора Readln. Той ще принуди компютъра да чака да се въведе завършен ред, т.е. да се натисне клавиша Enter. Това ще даде възможност да се видят резултатите преди да е приключила програмата и те да са изчезнали от екрана.

При изпълнение на програмата с R=15 и H=10, върху екрана на монитора ще се изведе следната информация:

```
Задайте радиуса: 15
Задайте височината: 10
 ВХОДНИ ДАННИ:
  - радиус.....  15.00
  - височина....10.00
 РЕЗУЛТАТИ:
  - околна повърхнина...     942.48
  - пълна повърхнина.....    2356.19
  - обем.....                7068.58
```

Пример 2.2. Програма, която извежда на екрана TRUE или FALSE в зависимост от това дали точка със зададени координати лежи или не лежи в окръжност със зададени координати на центъра и радиус. С променливите Xp и Yp са означени координатите на точката, а с Xc, Yc и R - параметрите на окръжността.

```
Var
  Xp, Yp, Xc, Yc, R : real;
Begin
  Write('Задайте координатите на точката:');  Readln(Xp,Yp);
  Write('Задайте координатите на центъра на окръжността:');
  Readln(Xc,Yc);
  Write('Задайте радиуса на окръжността:'); Readln(R);
  Writeln;
```

```
  Writeln(sqrt(sqr(Xp-Xc)+sqr(Yp-Yc)) <= R);
  Readln
End.
```

В тази програма е по-особен предпоследният оператор. Той е оператор за извеждане на данни с изходен списък от тип израз. Списъкът се състои само от един единствен израз, при това от логически тип. Тук най-напред се пресмята стойността на израза, която може да бъде True или False, и след това тя се извежда. Друг вариант на тази програма може да съдържа една променлива от логически тип, на която предварително се присвоява стойността на логическия израз и след това тя се извежда. При изпълнение на тази програма върху екрана на монитора ще се появи следното:

Задайте координатите на точката: 20 30
Задайте координатите на центъра на окръжността: 10 10
Задайте радиуса на окръжността: 20
FALSE

Пример 2.3. Програма, която при зададени стойност на покупката и сума, дадена от купувача, извежда следната информация:

Стойност на покупката: xxx.xx
Сума, дадена от купувача: xxx.xx
Ресто xxx.xx

Използвани са две променливи от реален тип - Sum1 и Sum2 за означаване на входните данни.

```
Var
  Sum1, Sum2 : real;
Begin
  Write('Стойност на покупката'); Readln(Sum1);
  Write('Сума, дадена от купувача'); Readln(Sum2);
  Writeln('Стойност на покупката',Sum1:6:2);
  Writeln('Сума, дадена от купувача',Sum2:6:2);
  Writeln('Ресто  ',Sum2-Sum1:6:2);
End.
```