Лекция 2

Типы данных

Типы данных

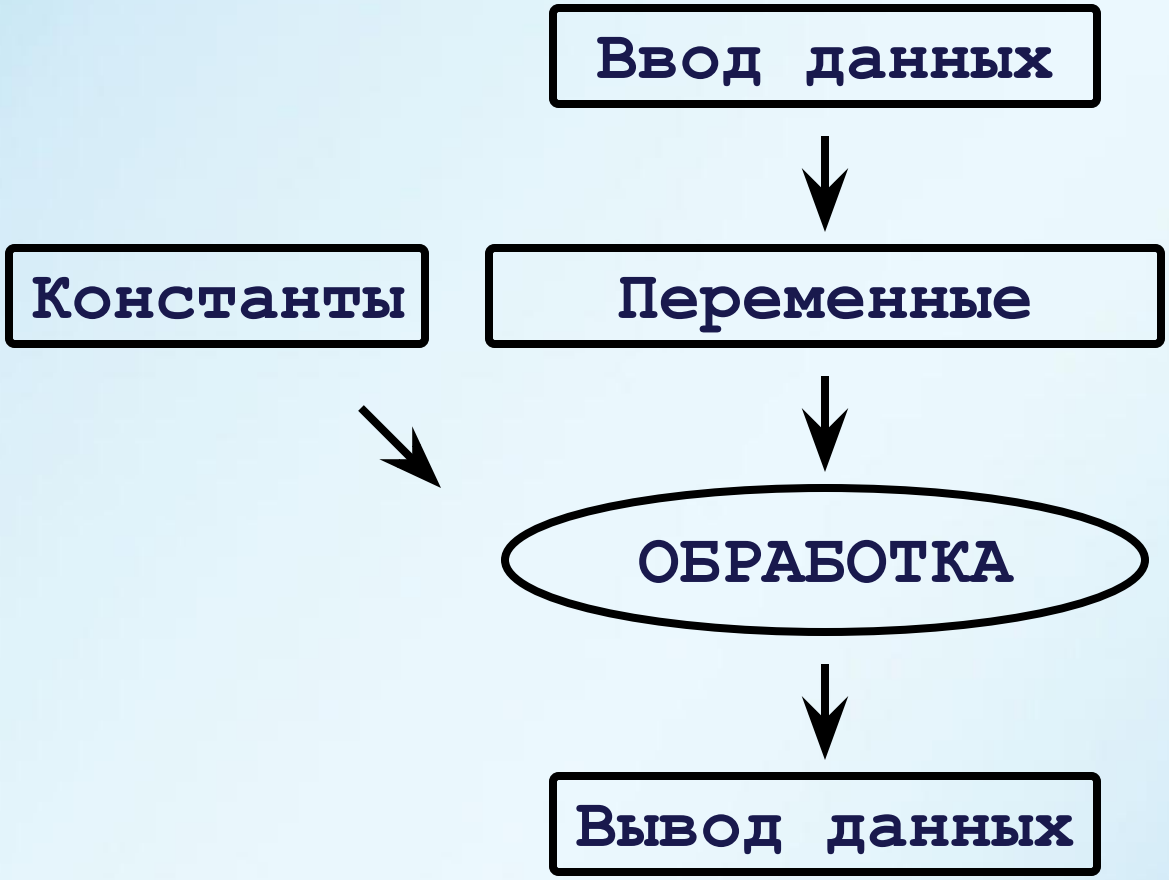
Стандартные

`integer`	целый
`real`	вещественный
`complex`	комплексный
`logical`	логический
`character`	символьный

Производные

`type(имя типа)`

Переменные и константы



Значения переменных могут изменяться, константы содержат всегда оно и тоже значение.

Задание имён

* Латинские буквы **A..Z, a..z**
  (маленькие и большие не различаются).

* Цифры и знак подчеркивания со 2-й позиции.

* Длина имени не более 63 символа.

R_0 **Vortex** **a1** **DistPol** Правильно	**2pressure** **func(x)** **sd:q** **w.x** Неверно

Используйте осмысленные имена!

Целочисленный тип

тип	длина (байт)	диапазон
`integer(1)`	1	-128 .. 127
`integer(2)`	2	-32768 .. 32767
`integer(4)`	4	-2^{31} .. $2^{31}-1$
`integer(8)`	8	-2^{63} .. $2^{63}-1$

Номера, счетчики, переменные циклов, границы и индексы массивов.

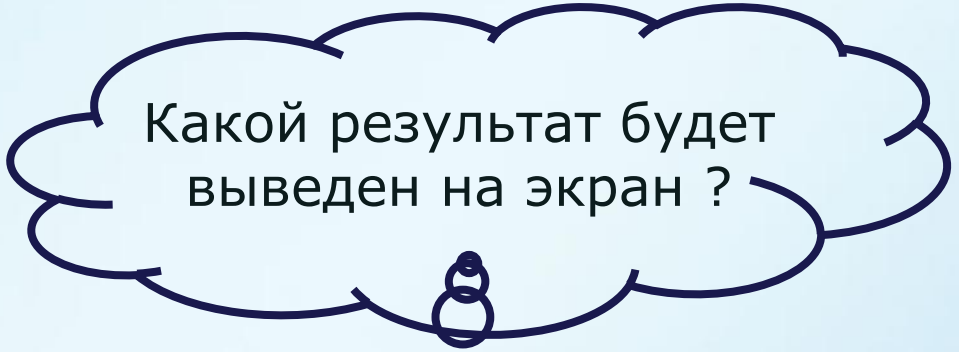
Целочисленный тип

```
! ---------------------- переменные
integer(1) a5
integer(4) nomer
integer index
integer*2 b2
! ---------------------- константы
integer, parameter :: T0 = 500
integer(2), parameter :: fact = 10100
integer(8), parameter :: QW = 3**20

index = 1000      ! переменным присвоили значения
a5 = B'1000101    ! двоичное представление
b2 = O'347'       ! восьмеричное
c7 = Z'AAB'       ! шестнадцатеричное
```

Инициализация переменных

```
program unknown
  integer koef
  write(*,*) koef
end
```



Инициализация переменной –
объявление + присваивание значения.

```
integer :: a = 10 ! инициализация переменной
integer :: s = 2**8+3**7+4**6 ! арифметические операции
```

Всегда ли есть соответствие ?

```
integer :: a = 10
```

```
integer a
a = 10
```

Вещественный тип

тип	длина (байт)	точность (знаков)	диапазон
real(4)	4	7	$1.2*10^{-38}$ .. $3.4*10^{+38}$
real(8) или double precision	8	15	$2.3*10^{-308}$ .. $1.7*10^{+308}$
real(16)	16	33	$3.4*10^{-4932}$ .. $1.1*10^{+4932}$

Переменные используемые для математических вычислений.

Вещественный тип

```
! ---------------------- переменные
real(4) :: p = 3.14159 ! 3.14159_4
real(4) :: s = 0.00001 ! .00001  или  1E-5

real(4)  :: A = 6.79E+15
real(4)  :: B = -9.0E-10

real(8)  :: q = 123456789D+5
double precision :: f = +2.7843D0

real(16) :: p1 = 123456789Q4000
real(16) :: p2 = -1.23Q-400

! ---------------------- константа
real, parameter :: pressure = 1e+10
```

Комплексный тип

тип	длина (байт)	точность (знаков)
complex(4)	8	7
complex(8)	16	15
complex(16)	32	33

```
! ---------------------- переменные
complex(4) c1
complex :: i1 = (0.0, 1.0) ! мнимая единица

! ---------------------- константа
complex, parameter :: z = (2.0, 3.0) ! 2+3i
```

Переменные для обработки комплексных данных (корни уравнений, преобразования Фурье).

Арифметические операции

операция	название	порядок	выполнение
**	степень	1	←
*	умножение	2	→
/	деление	2	→
-,+	знак числа	3	←
+	сложение	4	→
-	вычитание	4	→

Целочисленная арифметика

Деление целого числа на целое – результат целое.

```
S = 1/3 + 1/3 + 1/3  ! S = 0.0
P = 16**(1/4)        ! P = 1.0
```

Запись целого числа в вещественной форме.

```
S = 1.0/3.0 + 1./3. + 1./3 ! S = 1.0
P = 16**(1.0/4)            ! P = 2.0
```

Деление целого числа на нуль – ошибка выполнения.

```
m = 2/3
k = n/m  ! деление на нуль
```

Переполнение значения.

```
integer(1) :: bt = 127
bt = bt+1 ! bt = -128
```

Вещественная арифметика

Действительные числа представлены
с определенной точностью.

$$a + (b+c) \neq (a+b) + c$$

$$(a+b)^2 \neq a^2+2ab+b^2$$

```
a = 1.0/3; b = 4.0/7
  write(*,*) (a+b)**2              ! 0.8185941
  write(*,*) a**2+2*a*b+b**2       ! 0.8185942
```

Не рекомендуется сравнивать на равенство
вещественные числа!

```
real a,b
...
if (a==b) then
  ...
```

```
real, parameter :: eps=1.E-5
real a,b
...
if (abs(a-b)<eps) then
  ...
```

Вещественная арифметика

Переход от типа с меньшей точности к большей может привести к погрешности.

```
program lost_precision
  real(4) :: a = 1.03
  real(8) b

  b = a

  write(*,*) "a = ", a    ! 1.030000
  write(*,*) "b = ", b    ! 1.02999997138977

end
```

Вещественная арифметика

Не следует использовать в одном выражении значения, различие между которыми превышает число значащих цифр.

```
program arifm

  real(4) :: a = 1.E+10, b = -1.E+10, c = 5.0

  write(*,*) a+b+c  ! 5.000000
  write(*,*) a+c+b  ! 0.0000000E+00

end
```

Вещественная арифметика

Деление вещественного числа на нуль – бесконечность.

```
real(4) a,b
a = 1.0/0.0     ! результат Infinity
b = -1.0/0.0    ! -Infinity
```

Результат **Nan** "нет числа" – недопустимый результат.

```
real(4) a,b
a = (-2.0)**0.34 ! Nan
b = asin(2.0)    ! Nan
```

Логическая функция **IsNan(x)** проверки на значение **Nan**.

Смешанная арифметика

Автоматическое приведение типов по схемам

"целый → вещественный → комплексный"

"от меньшей разрядности → к большей"

Исключение – операция возведение в степень.

Запись `2**5` равносильна `2*2*2*2*2`

`2**5.0` равносильна `exp(5.0*ln(2.0))`

Преобразование типов может приводить к появлению погрешности !

Снижение погрешности

① Не вычитайте близкие числа.

② Не делите большие по модулю числа на малые.

③ Сложение (вычитание) длинной последовательности чисел начинайте с меньших чисел.

④ Уменьшайте число операций.

⑤ Используйте алгоритмы, для которых известны оценки ошибок.

⑥ Не сравнивайте на равенство вещественные числа.

Арифметические выражения

Математика	Fortran
$2a+3(b+c)$	`2*a+3*(b+c)`
$\frac{a+b}{c+d}$	`(a+b)/(c+d)`
$\frac{a}{b\cdot c\cdot d}$	`a/(b*c*d)` или `a/b/c/d`
$\sqrt[7]{a^5}$	`a**(5.0/7.0)`
a^2+b^5	`a*a + b**5`

Используйте дополнительные переменные для повышения читаемости кода программы.

Математические процедуры

Описание	Имя
$\sqrt{x}$	sqrt(x)
$\lvert x \rvert$	abs(x)
e^x	exp(x)
$\ln x$	log(x)
$\lg x$	log10(x)
$\mathrm{sh}x$	sinh(x)
$\mathrm{ch}x$	cosh(x)
$\mathrm{th}x$	tanh(x)
$\mathrm{Im}(z)$	aimag(z)
$\mathrm{Re}(z)$	real(z)

Описание	Имя
$\lvert z \rvert = \sqrt{x^2 + y^2}$	abs(z)
сопряжение	conjg(z)
остаток	mod(a,b)
abs(a) умножен-ное на знак b	sign(a,b)
округление ↓	floor(x)
округление ↑	ceiling(x)
максимум	max(x1,…)
минимум	min(x1,…)
max(x-y,0)	dim(x,y)

Математические процедуры *

Описание	Имя
радианы	
$\sin x$	`sin(x)`
$\cos x$	`cos(x)`
$\mathrm{tg}x$	`tan(x)`
$\mathrm{ctg}x$	`cotan(x)`
$\arcsin x$	`asin(x)`
$\arccos x$	`acos(x)`
$\mathrm{arctg}x$	`atan(x)`
$\mathrm{arctg}(y/x)$	`atan2(y,x)`

Описание	Имя
градусы	
$\sin x$	`sind(x)`
$\cos x$	`cosd(x)`
$\mathrm{tg}x$	`tand(x)`
$\arcsin x$	`asind(x)`
$\arccos x$	`acosd(x)`
$\mathrm{arctg}x$	`atand(x)`
$\mathrm{arctg}(y/x)$	`atan2d(y,x)`

Преобразование числовых типов

Приведение к целому типу

int(a,kind), kind = 1,2,4,8

Приведение к вещественному типу

real(a,kind), kind = 4,8,16

Приведение к комплексному типу

cmplx(a,kind), kind = 4,8,16

a – целого, вещественного или комплексного типов.

```
integer :: a = 10
real(16) s
  s = real(a,16) ! привели к типу real(16)
```

Операция присваивания

```
k = k+1   ! увеличение значения на 1
k = k-1   ! уменьшение значения на 1
k = 2*k   ! увеличение в 2 раза
k = k/2   ! уменьшение в 2 раза
k = -k    ! смена знака

s = s+k   ! увеличение s на k
s = s-k   ! уменьшение s на k
s = s*k   ! увеличение s в k раз
s = s+k   ! уменьшение s в k раз

tmp = a ! поменяли местами значения переменных a и b
a = b
b = tmp
```

Логический тип

тип	длина (байт)	значения
logical(1)	1	.TRUE.
logical(2)	2	
logical(4)	4	.FALSE.
logical(8)	8	

```
! ---------------------- переменные
logical(4)  :: st = .FALSE.
logical     :: res = .TRUE.
```

Переменные-флаги, проверки наступления событий, конструкции **if**.

Операции отношения

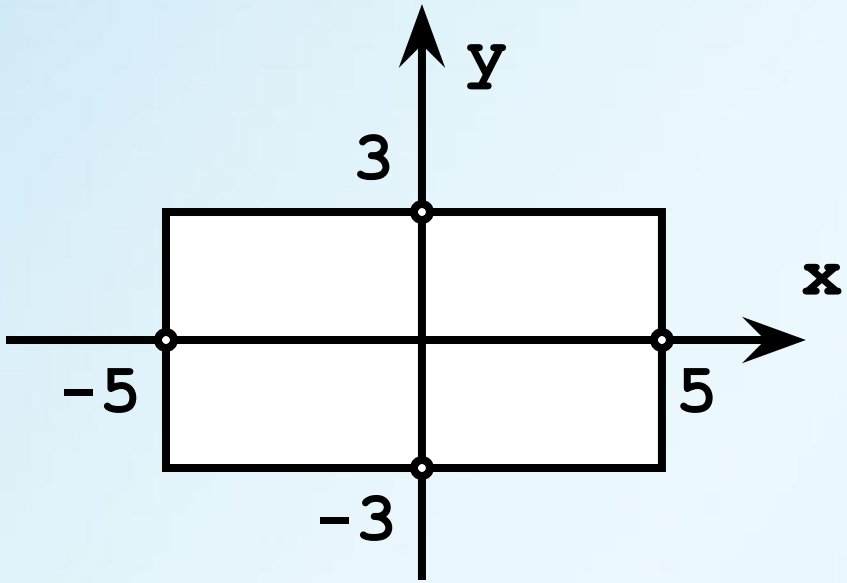
Операция	Имя
> или .GT.	больше
< или .LT.	меньше
== или .EQ.	равно
/= или .NE.	не равно
>= или .GE.	больше либо равно
<= или .LE.	меньше либо равно

```
logical position
 position = 3<5    ! .TRUE.
 position = 3==0   ! .FALSE.
```

Операция AND

Логическое умножение, конъюнкция.

```
1 .AND. 1 = 1      0 .AND. 1 = 0
1 .AND. 0 = 0      0 .AND. 0 = 0
```



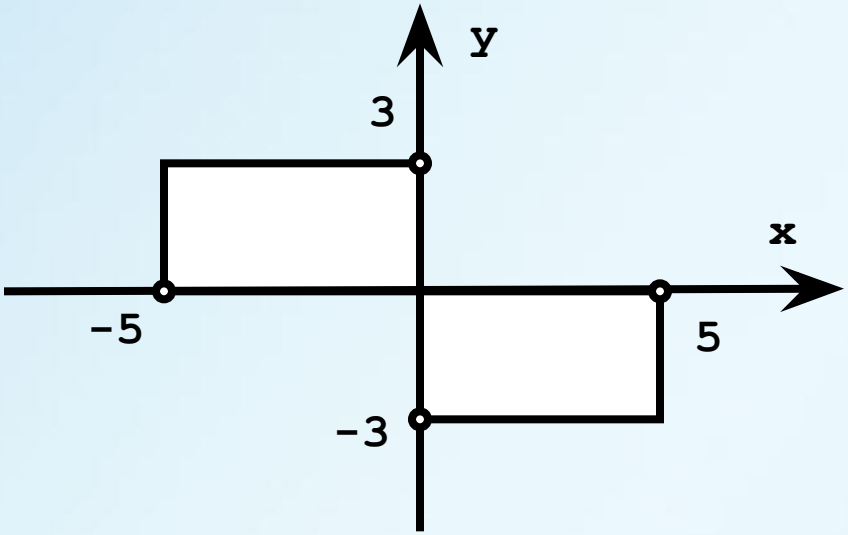
$$\begin{cases} -5 < x < 5 \\ -3 < y < 3 \end{cases}$$

```
(x>-5).AND.(x<5).AND.(y>-3).AND.(y<3)
```

Операция OR

Логическое сложение, дизъюнкция.

```
1 .OR. 1 = 1        0 .OR. 1 = 1
1 .OR. 0 = 1        0 .OR. 0 = 0
```



$$\begin{cases} -5 < x < 0 \\ 0 < y < 3 \end{cases}$$

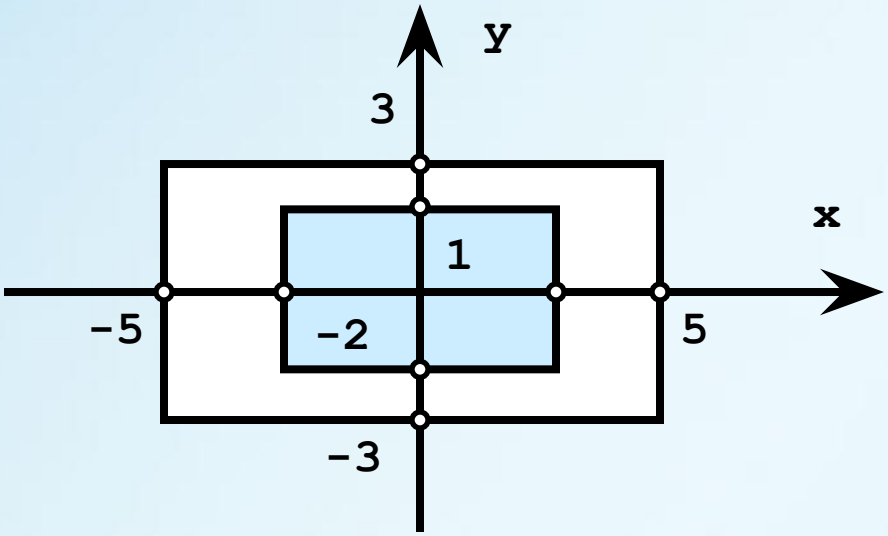
$$\begin{cases} 0 < x < 5 \\ -3 < y < 0 \end{cases}$$

```
(x>-5).AND.(x<0).AND.(y>0 ).AND.(y<3).OR. &
   (x>0 ).AND.(x<5).AND.(y>-3).AND.(y<0)
```

Операция XOR

Логическое исключающее "или", строгая дизъюнкция.

```
1 .XOR. 1 = 0      0 .XOR. 1 = 1
1 .XOR. 0 = 1      0 .XOR. 0 = 0
```



$$\begin{cases} -5 < x < 5 \\ -3 < y < 3 \end{cases}$$

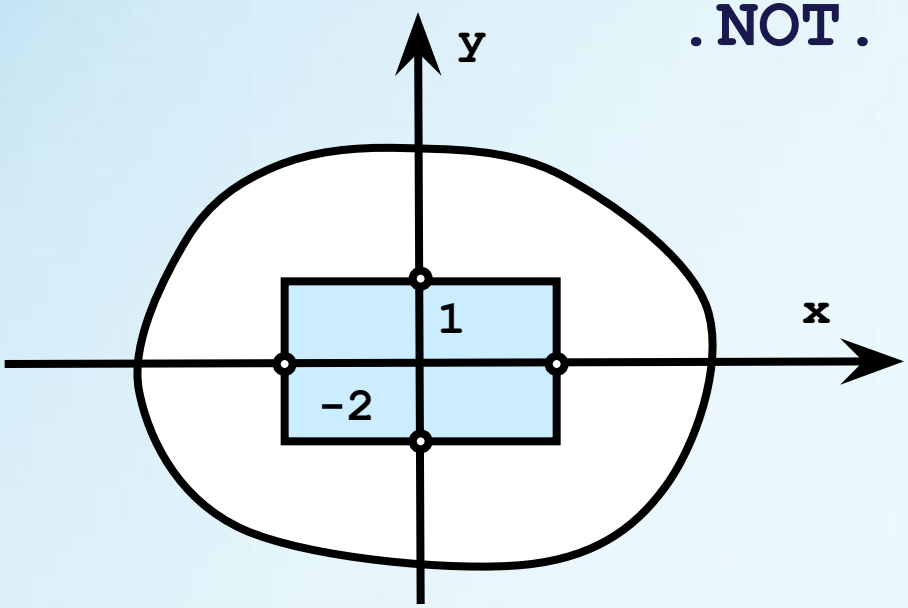
$$\begin{cases} -2 < x < 2 \\ -1 < y < 1 \end{cases}$$

```
(x>-5).AND.(x<5).AND.(y>-3).AND.(y<3).XOR. &
   (x>-2).AND.(x<2).AND.(y>-1).AND.(y<1)
```

Операция NOT

Логическое отрицание, инверсия.

```
.NOT. 1 = 0
.NOT. 0 = 1
```

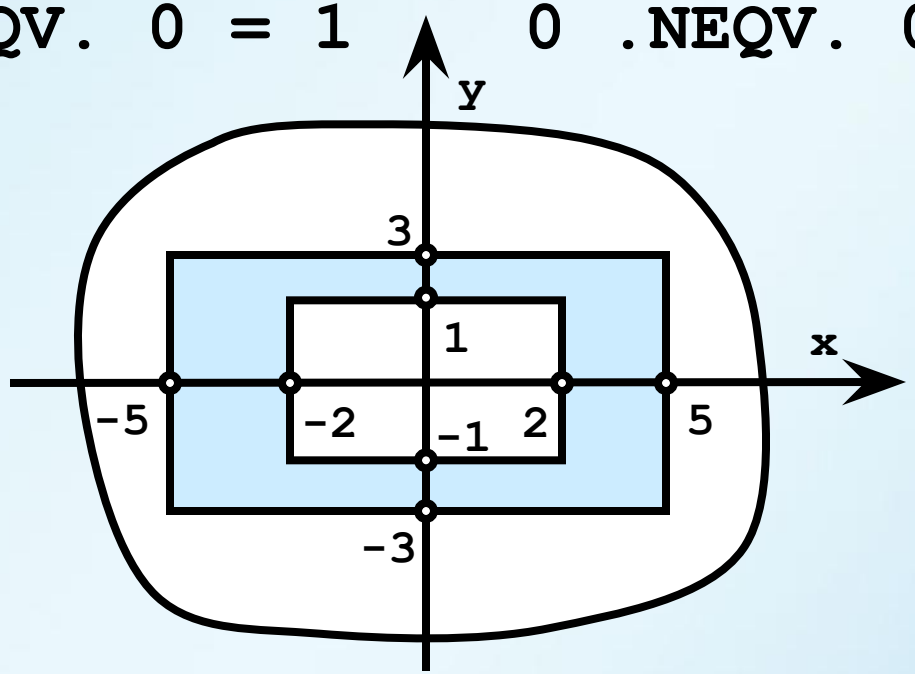


$$\begin{cases} -2 < x < 2 \\ -1 < y < 1 \end{cases}$$

```
.NOT.((x>-2).AND.(x<2).AND.(y>-1).AND.(y<1))
```

Операции эквивалентности

```
  1  .EQV.  1 = 1        0  .EQV.  1 = 0
  1  .EQV.  0 = 0        0  .EQV.  0 = 1
1  .NEQV.  1 = 0      0  .NEQV.  1 = 1
1  .NEQV.  0 = 1      0  .NEQV.  0 = 0
```




```
(x>-5).AND.(x<5).AND.(y>-3).AND.(y<3).EQV.  &
   (x>-2).AND.(x<2).AND.(y>-1).AND.(y<1)
```

Символьный тип

Объявления переменной для хранения 1 символа.

```
character key
character(1) ch
character(LEN = 1) symbol
```

Объявления строки для хранения 100 символов.

```
character(100) str
character word*100
character(LEN = 100) path
```

Имена файлов, обработка клавиш,
внутренние файлы, любая текстовая информация.

Символьный тип

Объявления символьных констант.

```
character, parameter :: key = 'A'
character(1), parameter :: ch = "Q"
character(100), parameter :: str = "C:\"
character(LEN = 11), parameter :: path = "D:\data.txt"
```

Присваивание символьных значений.

```
str = ' It''s very good! '  ! '' 1 апостроф
adr = '"TEXT"'  ! "TEXT"
```

C - строки

Символьная константа заканчивающаяся символом C.

```
character(100) cstr

cstr="Fortran & C++"C    ! cstr – C-строка
```

Управляющие символы в C-строках:

**** – слеш;

\a – звуковой сигнал;

\b – на 1 символ назад;

\n – новая строка;

\r – возврат каретки;

\t – горизонтальная табуляция;

и другие.

Операции со строками

// - конкатенация (сцепление, соединение) строк.

```
character a*5, b*2, c*20
...
a = 'AAAAA'
b = '..'
c = a//b//a  ! AAAAA..AAAAA
```

Обращение к подстроке, нумерация с единицы.

```
character (100) str, substr
str = '1234567890'
substr = str(1:3)    ! 123
```

Процедуры обработки строк *

Процедура	Описание
`len(str)`	длина строки
`len_trim(str)`	длина строки без хвостовых пробелов
`index(str,sub)`	номер первого вхождения строки **substr** в строку **str**
`iachar(ch)`	**ASCII**-код символа
`achar(code)`	возврат символа с кодом code
`getcharqq()` **	возврат нажатого символа
`peekcharqq(x)` **	определение нажатия клавиши

Ввод/вывод

Дескрипторы данных

Дескриптор	Тип	Представление
nIw[.m]	Целый	Целое число
nFw.d	Вещественный	**F**-форма
nEw.d	Вещественный	**E**-форма
nLw	Логический	**T** и **F**, **.T** и **.F**, **.TRUE.** и **.FALSE.**
nAw	Символьный	Строка символов

n – число повторений;

w – количество выводимых символов;

m – число ведущих нулей;

d – число цифр после десятичной точки.

Примеры вывода данных

```
integer :: a = 10, b = 20, c = 30
real    :: s = 1.237, p = 1.87342E+10
complex :: k = (0.0,1.0)
logical :: st = .TRUE.
character :: key = 'A'

write(*,"(3i4)") a,b,c     ! ^^10^^20^^30 ! упр
write(*,"(f10.5)") s       ! ^^^1.23700
write(*,"(E10.2)") p       ! ^^0.19E+11
write(*,"(2f5.1)") k       ! ^^0.0^^1.0
write(*,"(L2)") st         ! ^T
write(*,"(A4)") key        ! ^^^A

write(*,"(I4)") 1000000      ! **** ошибка
write(*,"(F5.4)") 123.456    ! ***** ошибка
```

Примеры ввода данных

```
program prog
  integer X, Y
  character(100) str

  write(*,"(A,\)") "Enter coordinates x,y "
  read(*,"(2I4)") X, Y
  write(*,"(A,I4)") "Summa = ", X+Y
                    ! 4^^^5 результат 9
                    ! 4^^^^^^^^^^^5 результат 4

  write(*,"(A,\)")   "Path..."
  read(*,"(A)")      str
  write(*,"(A)")     str(1:3)
                    ! Path...C:\DOCUM\1.txt
                    ! результат C:\
end
```

Примеры ввода данных

```
program prog
  real(4) c1, c2, c3
  complex(4) z1

  write(*,"(A,\)") "Values c1, c2, c3"
  read(*,"(3E10.2)") c1, c2, c3
  write(*,"(E10.2)") (c1+c2+c3)/2.0
      !1.E+0^^^^^2.E+0^^^^^9.E+0
      !результат ^^0.60E+01

  write(*,"(A,\)") "Complex Z1 = "
  read(*,"(2f5.2)") Z1 ! мнимая и действительная части
  write(*,"(A,f5.2)") "Im(z) = ", aimag(Z1)
  write(*,"(A,f5.2)") "Re(z) = ", real(Z1)
      !2.0^^3.0
      !результат Im(z) =^^3.0   Re(z) =^^2.0
end
```

Ввод/вывод

Дескрипторы управления:

nX – вывод **n** пробелов;

SP – вывод знака **"+"** в числовых данных;

SS – не выводить знак **"+"**;

S – восстановление действия дескриптора **SS**;

Tn – абсолютная табуляция;

TRn – относительная правая табуляция;

TLn – относительная левая табуляция;

BN – игнорировать пробелы;

BZ – интерпретировать пробелы как нули;

/ – переход на следующую строку;

**** – не переходить на следующую строку.

Ввод/вывод бесконечностей

-Inf или **-Infinity**

– отрицательная бесконечность;

Inf, +Inf, Infinity или **+Infinity**

– положительная бесконечность.

```
program infinity

   real(4) p
   write(*,"(A,\)") "p = " ! Inf
   read(*,*) p
   write(*,*) -p           ! -Infinity

end
```

Обработка ошибок

```
write(*,*,ERR = целочисленная метка) ...
read (*,*,ERR = целочисленная метка)  ...
```

```
program check_error
integer k

read(*,*,ERR = 100) k  ! если введен недопустимый символ

write(*,*) k*1000
stop

100 stop "ERROR"
end
```

Умолчания о типах данных

По умолчанию все объекты программы, имена которых начинаются с букв **i,j,k,l,m,n** или **I,J,K,L,M,N** являются типа **integer**.

Все остальные объекты имеют тип **real**.

Оператор **implicit** изменяет правила умолчания.

```
implicit integer (A-B), logical (C-D)
```

implicit none – все имена должны быть объявлены явно.

Перечисления enum

Множество целых констант.

Используется для взаимодействия с языком C.

```
enum, bind(C)
  enumerator plus
  enumerator :: minus = 4, div = 9
  enumerator equal
end enum
! или
integer, parameter :: plus = 0, &
                      minus = 4, div = 9, equal = 10
```

Ссылки и адресаты

Ссылка – переменная, связанная
с другой переменной, называемой адресатом.

При обращении к ссылке будет происходить
обращение к адресату и наоборот.

```
integer, pointer :: p  ! ссылка
integer, target :: a   ! адресат
```

Ссылки позволяют создавать динамические структуры данных - списки, стеки, деревья, очереди.

Ссылки и адресаты

Операция => прикрепление ссылки к адресату.

```
program  prog
integer, pointer :: p
integer, target :: a
  a = 100
  p => a        ! прикрепили ссылку к адресату
  write(*,*) p

  p = 100     ! a = 100
  a = 500     ! p = 500
end
```

Все изменения, происходящие с адресатом, дублируются в ссылке.

Ссылки и адресаты

Массивные указатели

```
real, pointer :: a_ptr(:)
real, target :: a_trg(5) = [1,2,3,4,5]
a_ptr => a_trg
print*, a_ptr
end
```

Результат

1.000000 2.000000 3.000000 4.000000 5.000000

Ссылки и адресаты

Функция **associated(pt, addr)** возвращает **.TRUE.** если ссылка **pt** прикреплена к адресату **addr**.

```
program prog
integer, pointer :: p1, p2, p3
integer, target :: a,b
  a = 100;  b = 2; p1 => a;  p2 => a
  write(*,*) associated(p1,p2)   ! TRUE
  write(*,*) associated(p1)
  write(*,*) associated(p2,a)
  p1 => b
  write(*,*) associated(p3)      ! FALSE
  write(*,*) associated(p1,p2)
  write(*,*) associated(p1,a)
end
```

Ссылки и адресаты

Оператор **nullify** - открепление ссылки от адресата.

```
program prog
integer, pointer :: p1, p2
integer, target :: a

  a = 1000;  p1 => a;  p2 => a
  ! если к адресату прикреплены две ссылки,
  ! то отсоединим последнюю

  if (associated(p1,p2)) nullify(p2)

  write(*,*) associated(p1), associated(p2) ! T, F

end
```

Целочисленные указатели

Целочисленный указатель – переменная целого типа, содержащая адрес некоторой переменной, называемой адресной переменной.

```
real a              ! базируемая переменная
pointer (p,a)       ! p – целочисленный указатель
                    ! на переменную типа real

character ch        ! ch – базируемая переменная
pointer (pc,ch)     ! pc – целочисленный указатель
                    ! на тип character
```

Целочисленный указатель и базируемая переменная используются совместно.

Целочисленный указатель часто используется для обращения к функциям языка С.

Целочисленные указатели

Функция **LOC** вычисляет адрес переменной.

```
program arrow
integer a     ! базируемая переменная
pointer(p,a)  ! указатель на целый тип

integer :: b = 100

  p = loc(b)        ! вычислили адрес переменной b
  a = 500           ! базируемой переменной поместим в b
                    ! значение 500

  write(*,*)"address = ",p, &   ! 5038080
            "  value = ",a, &   ! 500
            "  b     = ",b      ! 500
end
```

Конструкция Associate

Организует промежуточные расчетные блоки.

```
program prog
  ...
  associate (R => sqrt(x*x+y*y+z*z))
    res = (R<10).AND.(R>3) ! R только в правой части
  end associate

associate (ARRAY => AB % D (I, :) % X)
  ARRAY (3) = ARRAY (1) + ARRAY (2)
end associate

end
```

Задание

Треугольник задан тремя точками на плоскости *
(x1;y1), (x2;y2), (x3,y3).

Вычислить:

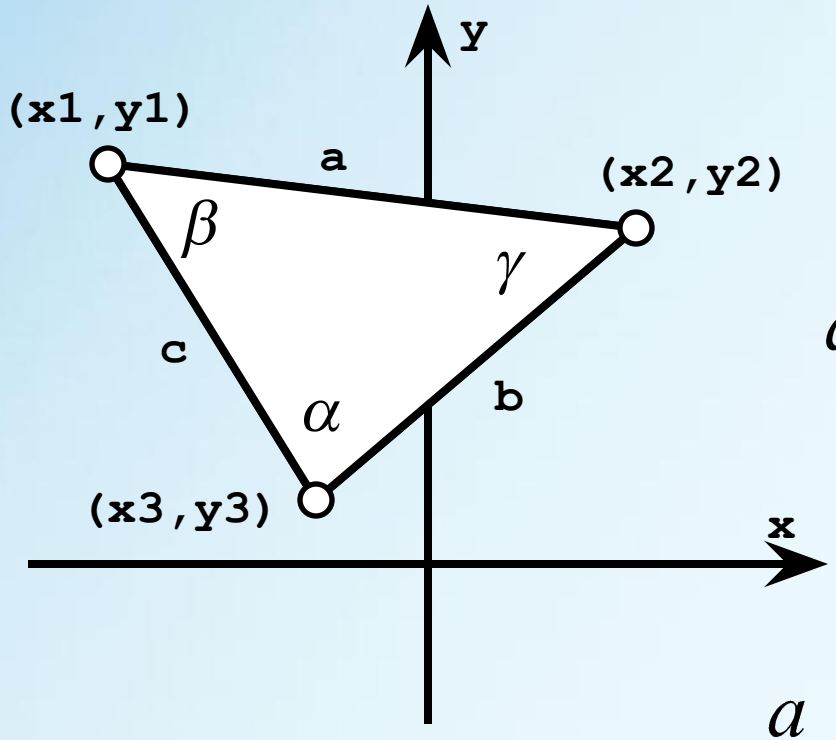
1) площадь;
2) периметр.

Проверить:

3) сумму углов треугольника;
4) неравенство треугольника;
5) теорему синусов.

* Случай, когда точки лежат на одной прямой не рассматриваем.

* З а д а н и е *



$$S = \pm\frac{1}{2}\cdot\begin{vmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{vmatrix}$$

$$a < b + c; \quad b < c + a; \quad c < a + b$$

$$a = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$\frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma}$$

$$c^2 = a^2 + b^2 - 2\cdot a\cdot b\cdot\cos\gamma$$

Задание

```
program prog
implicit none

  real(4), parameter :: eps = 1.0E-5 ! погрешность
  real(4) x1,y1,x2,y2,x3,y3          ! координаты
  real(4) a,b,c                      ! стороны
  real(4) alfa, betta, gamma         ! углы
  real(4) S, P                       ! площадь, периметр
  logical NRV   ! проверка неравенства треугольника
  logical TSIN  ! проверка теоремы синусов
  logical SUM   ! проверка суммы углов

  ! ввод координат
  write(*,"(A,\)") "1 point....";  read(*,*) x1, y1
  write(*,"(A,\)") "2 point....";  read(*,*) x2, y2
  write(*,"(A,\)") "3 point....";  read(*,*) x3, y3

  ! вычисление площади
  S=0.5*abs((x2-x1)*(y3-y1)-(x3-x1)*(y2-y1))

  ! нахождение сторон
  a=sqrt((x1-x2)**2+(y1-y2)**2)
  b=sqrt((x3-x2)**2+(y3-y2)**2)
  c=sqrt((x1-x3)**2+(y1-y3)**2)

  ! вычисление периметра
  P=a+b+c

  ! проверка неравенства треугольника
  NRV=(a<b+c).AND.(b<a+c).AND.(c<a+b)

  ! нахождение углов
  alfa =acos((a*a-c*c-b*b)/(-2*c*b))
  betta=acos((b*b-a*a-c*c)/(-2*a*c))
  gamma=acos((c*c-a*a-b*b)/(-2*a*b))

  ! проверка суммы углов треугольника
  SUM=abs(alfa+betta+gamma-2*acos(0.0))<eps
```

```
  ! проверка суммы углов треугольника
  SUM=abs(alfa+betta+gamma-2*acos(0.0))<eps

  ! проверка теоремы синусов
  TSIN=( (abs(a/sin(alfa )-b/sin(betta))<eps).AND. &
         (abs(b/sin(betta)-c/sin(gamma))<eps)     )

  ! вывод результатов
  write(*,"(A,F10.5)") "Square    = ", S
  write(*,"(A,F10.5)") "Perimeter = ", P

  write(*,"(A,L)") "Sum of corners      = ", SUM
  write(*,"(A,L)") "Triangle inequality = ", NRV
  write(*,"(A,L)") "Theorem of sine     = ", TSIN

end
```

Результат работы программы.

```
1 point....2 3
2 point....4 5
3 point....3 7
Square    =    3.00000
Perimeter =    9.18760
Sum of corners      =  T
Triangle inequality =  T
Theorem of sine     =  T
Для продолжения нажмите любую клавишу . . .
```