

Wyrażenia

Analogicznie do trzech typów stałych i zmiennych: numerycznych (w tym: całkowitych, rzeczywistych i zespolonych), logicznych i tekstowych – w Fortranie występują trzy grupy wyrażeń: arytmetyczne, logiczne i tekstowe.

Wyrażenia arytmetyczne

W wyrażeniu arytmetycznym mogą wystąpić następujące elementy (operandy):

- stałe arytmetyczne (całkowite, rzeczywiste oraz zespolone)
- zmienne
- elementy tablic
- wywołania funkcji

Można wykonywać następujące operacje arytmetyczne (operatory):

- potęgowanie **
- mnożenie i dzielenie * /
- dodawanie i odejmowanie + -

Operatory zostały podane w kolejności odpowiadającej ich priorytetom. Jeżeli w wyrażeniu występują operatory o takim samym priorytecie, to:

- dla potęgowania ** obliczenia wykonuje się w kolejności od prawej strony wyrażenia do lewej, czyli wartość wyrażenia $a^{**}b^{**}c$ oblicza się jako $a^{**}(b^{**}c)$. Jest to oczywiste, jeśli popatrzy się na tradycyjny zapis a^{b^c} .
- dla pozostałych operatorów obowiązuje kolejność od lewej do prawej. Np. wartość wyrażenia $a/b*c$ oblicza się jako $(a/b)*c$.

Nawiasy okrągłe () pozwalają zmienić tę kolejność, np. w wyrażeniu $a/(b*c)$ ‘naturalna’ kolejność została zaburzona.

Uwagi:

- Można wykorzystywać tylko nawiasy okrągłe. Liczba nawiasów otwierających musi być taka sama jak liczba nawiasów zamykających.
- Dwa operatory nie mogą wystąpić obok siebie. W związku z tym zapis $a*-b$ jest niepoprawny i należy zastąpić go przez $a*(-b)$.

Zabronione są następujące operacje:

- dzielenie przez zero,
- podnoszenie zera do ujemnej lub zerowej potęgi.

Typ wyniku

Każdy operand wyrażenia arytmetycznego (w szczególności stała lub zmienna) posiada typ. Również wartość wyrażenia arytmetycznego posiada typ, a jak wiadomo, z faktem przypisania do określonego typu wiążą się określone konsekwencje.

Sprawa jest prosta gdy oba operandy mają ten sam typ. W takim przypadku wynik również ma ten typ. Należy przy tym zwrócić szczególną uwagę przypadek dzielenia całkowitoliczbowego.

Dzielenie całkowitoliczbowe

Zgodnie z podaną przed chwilą zasadą, jeśli dwa operatory mają typ całkowity, to również wynik operacji, np. dzielenie, ma typ całkowity. Co będzie zatem wynikiem działania $1/3$? Skoro musi mieć on typ całkowity, to z całą pewnością nie może być równy 0.3333, bo jest to wartość rzeczywista. W arytmetyce całkowitoliczbowej $1/3 = 0$. Podobnie, $9/10=0$, $-9/10=0$,

$-11/10=-1$. W szczególności wartością wyrażenia $a^{** (1/k)}$, dla k całkowitego i różnego od 1, bez względu na wartość zmiennej a , jest 1 (dlaczego?).

W przypadku operatorów różnych typów, typ wyniku jest zgodny z typem tego operandu, który zajmuje wyższą pozycję na liście: complex, double precision, real, integer.

W związku z tym, o ile wynik wyrażenia $1/k$ (przy założeniu, że k jest zmienną całkowitą i różną od 0 oraz 1) był równy 0, to dla wyrażenia $1./k$ już tak nie jest. Wynika to z zastąpienia stałej całkowitej (1) stałą rzeczywistą (1.), co powoduje zmianę typu wyniku na rzeczywisty.

Na marginesie zwracamy uwagę, że zapis $a^{** (1./k)}$ odpowiada $\sqrt[k]{a}$.

Funkcje standardowe

Omawianie wyrażeń arytmetycznych jest dobrym momentem do wspomnienia o funkcjach standardowych, nazywanych też funkcjami wbudowanymi (ang. *intrinsic*). Fortran 90 posiada 113 funkcji wbudowanych, które można podzielić na kilka (nieformalnych) kategorii:

- Funkcje podstawowe, wśród których z kolei wyróżnia się funkcje matematyczne, numeryczne, znakowe oraz bitowe;
- Sprawdzające;
- Przekształcające.

Aby funkcja została wykonana należy ją wywołać.

Standardowe funkcje matematyczne:

zapis w Fortranie	zapis matematyczny
sin(x)	$\sin x$
cos(x)	$\cos x$
tan(x)	$\tan x$
asin(x)	$\arcsin x$
acos(x)	$\arccos x$
atan(x)	$\arctan x$
abs(x)	$ x $
sqrt(x)	\sqrt{x}
exp(x)	e^x
log(x)	$\ln x$
log10(x)	$\log_{10} x$

Obiekt x w powyższym zestawieniu pełni rolę *argumentu*, w języku programowania nazywanego *parametrem*. Od wartości parametru zależy wartość funkcji. W chwili *wywołania* funkcji wartość parametru musi być znana. Dla powyższych funkcji argumenty muszą mieć typ REAL, DOUBLE PRECISION lub COMPLEX

Wszystkie funkcje akceptujące argumenty typu REAL akceptują również argumenty typu DOUBLE PRECISION. Natomiast nie wolno ich wywoływać z parametrami o typie całkowitego. W szczególności wywołanie EXP(1) jest błędne (poprawnie: jest EXP(1.) lub EXP(1d0)); podobnie niepoprawne jest wywołanie SQRT(9) (poprawnie: SQRT(9.), SQRT(9.e0) lub SQRT(0.9d1)).

Uwagi:

- Parametry funkcji trygonometrycznych są podawane w radianach. (1 radian = $180/\pi$ stopni).
- Dla funkcji asin i acos argument musi należeć do przedziału [-1, 1].

- Próba wyznaczenia wartości funkcji dla parametru nie należącego do jej dziedziny zakończy się porażką, czyli błędem wykonania (ang. *run time error*). Np. błędem zakończy się próba wyznaczenia pierwiastka kwadratowego lub logarytmu z wartości ujemnej.

Na marginesie...

Funkcje trygonometryczne można wykorzystać do uzyskania wartości π , wiedząc że $\arctan 1 = \pi/4$ albo $\arcsin 0.5 = \pi/6$, patrz przykłady programów.

Inne funkcje standardowe

Wywołanie funkcji	Opis działania
AINT (a)	Obcięcie do wartości całkowitej; przy czym również wynik ma typ <i>rzeczywisty</i> . Wartością AINT (1.8) jest 1.0, zaś wartością AINT (-1.8) jest -1.0.
ANINT (a)	Wynikiem działania jest najbliższa argumentowi liczba całkowita. Zarówno parametr jak i wynik mają typ <i>rzeczywisty</i> . Wartością ANINT (1.8) jest 2.0, zaś wartością ANINT (-1.8) jest -2.0.
CEILING (a)	Najmniejsza liczba całkowita większa niż lub równa wartości argumentu. Argument ma typ <i>rzeczywisty</i> , zaś wynik ma typ <i>całkowity</i> . Wartością CEILING (1.8) jest 2, zaś wartością ANINT (-1.8) jest -1.
CMPLX (x, y)	Konwersja dwóch wartości rzeczywistych na wartość zespoloną, gdzie x jest częścią rzeczywistą a y jest częścią urojoną.
DBLE (x)	Konwersja argumentu (o typie całkowitym, rzeczywistym lub zespolonym) do typu DOUBLE PRECISION.
DIM (x, y)	Argumenty muszą mieć typ REAL lub INTEGER. Jeśli x>y to wtedy wartością DIM (x, y) jest wartość x-y. Jeśli x<y, wartością DIM (x, y) jest 0.
FLOOR (a)	Największa liczba całkowita mniejsza niż lub równa wartości argumentu. Argument ma typ <i>rzeczywisty</i> , zaś wynik ma typ <i>całkowity</i> . Wartością FLOOR (1.8) jest 1, zaś wartością FLOOR (-1.8) jest -2.
INT (a)	Obcięcie do wartości o typie INTEGER. Obcięcie następuje 'w stronę zera', wartością INT (0.9) jest 0, wartością INT (-0.9) jest 0.
MAX (a1, a2, a3, ...)	Wartość maksymalna z argumentów. Muszą wystąpić co najmniej dwa argumenty o typie INTEGER lub REAL.
MIN (a1, a2, a3, ...)	Wartość minimalna z argumentów. Muszą wystąpić co najmniej dwa argumenty o typie INTEGER lub REAL.
MOD (a, p)	Reszta z dzielenia a/p wyznaczona według wzoru a-INT (a/p) *p. Argumenty muszą mieć typ INTEGER lub REAL, p musi być różne od zera.
NINT (x)	Zaokrąglenie wartości rzeczywistej do najbliższej wartości całkowitej. Jeśli x>0, NINT (x) wyznacza się jako INT (x+0.5). Jeśli x<0 to NINT (x) jest równe INT (x-0.5).
REAL (x)	Konwersja na typ REAL.

Wywołanie funkcji	Opis działania
SIGN (a, b)	Wynik jest równy $ABS(a) * (b / ABS(b))$, co jest równoważne 'przeniesieniu' znaku drugiego argumentu na pierwszy argument. Argumenty mają typ rzeczywisty lub całkowity.

Wyrażenia tekstowe

W wyrażeniu tekstowym mogą wystąpić następujące elementy (operandy):

- stałe i zmienne tekstowe
- podłańcuchy
- wywołania funkcji tekstowych

Dostępna jest tylko jedna operacja, jaką można wykonywać na tekstach; jest nią *konkatenacja*. Operator konkatenacji oznacza się symbolem //. Konkatenacja polega na 'zlepianiu' ze sobą stałych lub zmiennych tekstowych występujących w wyrażeniu, np. wynikiem 'A'///'la'///' ma kota' jest 'Ala ma kota'. Nawiasy nie mają wpływu na działanie operatora konkatenacji.

Podłańcuchy (ang. *substring*)

Podłańcuch jest spójnym fragmentem zmiennej tekstowej. Definiuje się go jako nazwa(pierwszy_znak : ostatni_znak), gdzie wartość pierwszy_znak wskazuje na położenie pierwszego znaku podłańcucha w zmiennej nazwa. Brak tego elementu w definicji podłańcucha jest równoważne podaniu wartości 1. Wartość ostatni_znak wskazuje na ostatni znak w podłańcuchu, a jej brak oznacza, że końce zmiennej i podłańcucha pokrywają się.

Jeśli wartością zmiennej tekst jest 'Ala ma kota' to:

```
tekst(5:6)           ma
tekst( :3) (albo tekst(1:3))  Ala
tekst(8:) (albo tekst(8:11)) kota
tekst(:)             Ala ma kota
```

Standardowe funkcje tekstowe

Wywołanie funkcji Oznaczenia: i – stała/zmienna całkowita str – łańcuch znakowy ch – znak	Opis działania
ACHAR(i)	Wynikiem jest i-ty znak z tablicy znaków ASCII; wartość argumentu musi być z przedziału 0-127. Np. wartością ACHAR(113) jest 'p'.
ADJUSTL(str)	Argumentem funkcji jest łańcuch znakowy, zaś wynikiem jest ten sam łańcuch, z którego usunięto spacje wiodące, dopisując je na końcu. Np. wartością ADJUSTL(' ALA') jest 'ALA '.
ADJUSTR(str)	Argumentem funkcji jest łańcuch znakowy, zaś wynikiem jest ten sam łańcuch, z którego usunięto spacje końcowe, dopisując je na początku łańcucha jako spacje wiodące. Np. wartością ADJUSTR('KOT ') jest ' KOT'.

Wywołanie funkcji Oznaczenia: <i>i</i> – stała/zmienna całkowita <i>str</i> – łańcuch znakowy <i>ch</i> – znak	Opis działania
CHAR(<i>i</i>)	Wynikiem jest <i>i</i> -ty znak z tablicy porządku leksykograficznego kompilatora; wartość argumentu musi być z przedziału 0-127. Najczęściej wynik jest taki sam jak dla funkcji ACHAR.
IACHAR(<i>ch</i>)	Argumentem jest pojedynczy znak, zaś wynikiem jest położenie znaku w tablicy znaków ASCII; np. IACHAR('p') ma wartość 113.
ICHAR(<i>ch</i>)	Argumentem jest pojedynczy znak, zaś wynikiem jest położenie znaku w tablicy porządku leksykograficznego kompilatora. Najczęściej wynik jest taki sam jak dla funkcji IACHAR.
INDEX(<i>str</i> , <i>substr</i>)	Argumentami są dwa łańcuchy znakowe, <i>str</i> oraz <i>substr</i> , zaś wynikiem jest <i>pierwsze</i> wystąpienie łańcucha <i>substr</i> w łańcuchu <i>str</i> ; np. wartością INDEX('galanteria', 'ala') jest 2.
LEN(<i>str</i>)	Wynikiem jest długość łańcucha <i>str</i>
LEN_TRIM(<i>str</i>)	Wynikiem jest długość łańcucha <i>str</i> z <i>pominic iem</i> spacji końcowych.
LGE(<i>str1</i> , <i>str2</i>)	porównanie leksykalne dwóch łańcuchów zgodne z kolejnością znaków w tablicy znaków ASCII; wynikiem jest wartość logiczna. Odpowiada relacji 'większy lub równy'.
LGT(<i>str1</i> , <i>str2</i>)	porównanie leksykalne dwóch łańcuchów zgodne z kolejnością znaków w tablicy znaków ASCII; wynikiem jest wartość logiczna. Odpowiada relacji 'większy niż'.
LLE(<i>str1</i> , <i>str2</i>)	porównanie leksykalne dwóch łańcuchów zgodne z kolejnością znaków w tablicy znaków ASCII; wynikiem jest wartość logiczna. Odpowiada relacji 'mniejszy lub równy'.
LLT	porównanie leksykalne dwóch łańcuchów zgodne z kolejnością znaków w tablicy znaków ASCII; wynikiem jest wartość logiczna. Odpowiada relacji 'mniejszy niż'.
REPEAT(<i>str</i> , <i>i</i>)	konkatenuje łańcuch <i>str</i> <i>i</i> razy; np. wynikiem REPEAT('Aa', 3) jest 'AaAaAa'.
TRIM(<i>str</i>)	usuwa końcowe spacje.
VERIFY(<i>str</i> , <i>set</i>)	Argumentami są łańcuchy znakowe, zaś wynikiem jest położenie w łańcuchu pierwszego znaku, który nie znajduje się zmiennej ' <i>set</i> '. Na przykład wartością VERIFY('ALA', 'A') jest 2, natomiast wartością VERIFY('ALA', 'LA') jest 0.

Wyrażenia relacji

Wyrażenia relacji służą do porównywania wartości dwóch wyrażeń arytmetycznych lub tekstowych. Wynik wyrażenia ma wartość logiczną `.TRUE.` gdy relacja zachodzi oraz `.FALSE.` w przeciwnym przypadku. Nie można porównać ze sobą wartości wyrażenia arytmetycznego i tekstowego.

Elementy wyrażenia relacji (operandami) są:

- wyrażenia arytmetyczne
- wyrażenia znakowe

Operatory relacji (operandy):

Dopuszczalne są dwa równorzędne sposoby zapisywania operatorów relacji. Jeden z nich, literowy, jest zapisem pochodzącym ze starych wersji języka Fortran; zapis za pomocą symboli matematycznych pojawił się w Fortranie 90.

.GE.	>	większe niż
.GT.	>=	większe równe
.LE.	<=	mniejsze równe
.LT.	<	mniejsze niż
.NE.	/=	nie równe
.EQ.	==	równe

Na przykład wartością `5 .LT. 10` jest `.TRUE.`, zaś wartością `100 .NE. 100` jest `.FALSE.`

Uwagi:

- W jednym wyrażeniu relacji można używać dwóch zapisów.
- W operatorach ‘dwuznakowych’, np. `==` lub `<=` nie można umieszczać spacji.
- Kolejność wykonywania obliczeń podczas wyznaczania wartości operatora relacji jest taka, że najpierw wyznaczane są wartości porównywanych wyrażeń. Dlatego wartością wyrażenia relacji `3+1 .GT. 2` jest `.TRUE.`
- Sposób zapisu wartości rzeczywistych w pamięci komputera oraz związany z tym fakt, że wartości te są reprezentowane w sposób przybliżony sprawia, że relacja `.EQ.` lub `.NE.` w odniesieniu do wartości rzeczywistych nie ma ‘realnego’ znaczenia. Zamiast relacji `x.EQ.y` zaleca się stosować porównanie w postaci:
`abs(x-y) .LT. eps,`
gdzie `eps` jest zmienną o małej wartości.
- W przypadku porównywania ze sobą łańcuchów znakowych obowiązuje kolejność alfabetyczna zgodna z kolejnością znaków ASCII. Wartością wyrażenia `'OLA' .GT. 'ALA'` jest `.TRUE.` Gdy porównywane łańcuchy mają różną długość, krótszy z nich dla celów porównania rozszerzany do rozmiaru dłuższego poprzez dopisanie na końcu odpowiedniej liczby spacji.

Wyrażenia logiczne

Wartość wyrażenia logicznego ma typ logiczny. Wyrażenie jest prawdziwe (`.TRUE.`) lub fałszywe (`.FALSE.`).

W wyrażeniu logicznym mogą wystąpić następujące elementy (operandy):

- stałe i zmienne logiczne;
- elementy tablic logicznych;
- wywołania funkcji logicznych;
- wyrażenia logiczne.

Wyrażenia logiczne konstruuje się w oparciu następujące operatory logiczne:

.NOT.	negacja
.AND.	koniunkcja
.OR.	alternatywa
.EQV.	tożsamość
.NEQV.	nie tożsamość

Operatory zostały podane według ich priorytetów, czyli kolejności w jakiej wykonywane są działania podczas wyznaczania wartości wyrażeń logicznych.

- Kiedy dwa operatory mają ten sam priorytet, to najpierw wykonywana jest operacja znajdująca się z lewej strony;
- Operator `.NOT.` poprzedza argument; pozostałe operatory wymagają podania dwóch argumentów w kolejności: `argument1 operator argument2`.

Wynik działania operatorów logicznych podaje następująca tabela:

a	b	.NOT.a	a.AND.b	a.OR.b	a.EQV.b	a.NEQV.b
.TRUE.	.TRUE.	.FALSE.	.TRUE.	.TRUE.	.TRUE.	.FALSE.
.TRUE.	.FALSE.	.FALSE.	.FALSE.	.TRUE.	.FALSE.	.TRUE.
.FALSE.	.TRUE.	.TRUE.	.FALSE.	.TRUE.	.FALSE.	.TRUE.
.FALSE.	.FALSE.	.TRUE.	.FALSE.	.FALSE.	.TRUE.	.FALSE.

Jeżeli w wyrażeniu występują operatory arytmetyczne, znakowe, relacji oraz logiczne to obowiązuje następująca kolejność wykonywania działań:

- operacje arytmetyczne i znakowe (wyrażenia arytmetyczne i tekstowe)
- wartości wyrażeń relacji (wyrażenia relacji)
- operacje logiczne.
- Umieszczenie w wyrażeniu nawiasów okrągłych pozwala zmienić tę kolejność.