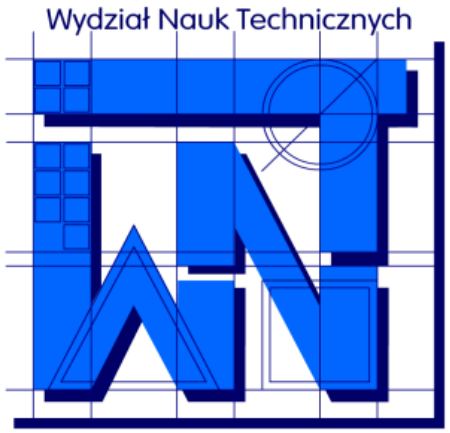


UNIVERSITY OF WARMIA AND MAZURY IN OLSZTYN
The Faculty of Technical Sciences
POLAND, 10-957 Olsztyn, M. Oczapowskiego 11
tel.: (48)(89) 5-23-32-40, fax: (48)(89) 5-23-32-55
URL: http://www.uwm.edu.pl/edu/sobieski/ (in Polish)

Języki Programowania

Elementy języków programowania

Wojciech Sobieski

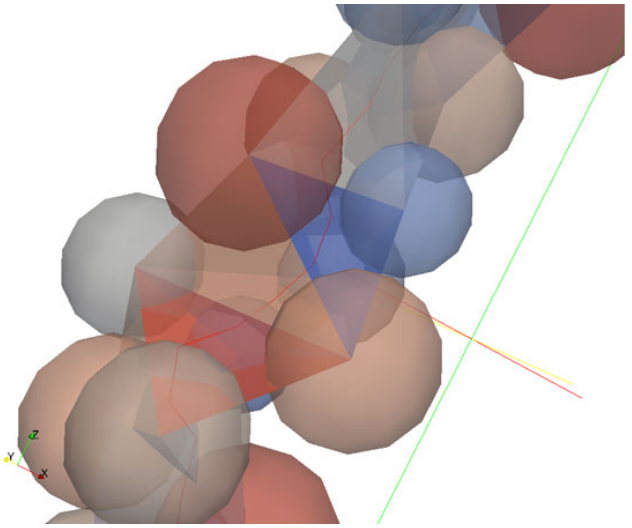
Olsztyn, 2001-2021

Definicja

Elementy języków programowania – cechy i funkcjonalności charakterystyczne dla różnych (tu: arytmetycznych) języków programowania.

Funkcjonalność – zbiór atrybutów urządzenia, oprogramowania lub systemu, określających zdolność do dostarczenia funkcji zaspokajających wyznaczone i zakładane potrzeby, podczas używania w określonych warunkach.

niezależnie od tego, w jakim języku napisany jest program, wymagamy od niego na przykład, aby dało się zapisać na dysku wyniki jego pracy, chociażby po to, żeby później dokonać ich wizualizacji



UWAGA: Omawiane na wykładzie zagadnienia mają charakter ogólny, poza przykładami, które to dotyczą głównie języka Fortran.

Znaki i symbole

Ściśle określony zbiór znaków i symboli.

Zakres znaków obejmuje wielkie i małe litery (rozróżniane bądź nie), cyfry oraz znaki specjalne i symbole wieloznakowe. Mimo, że wiele języków programowania posiada podobny zestaw znaków dopuszczalnych, niektóre z nich mogą mieć zupełnie inne znaczenie. Przykładem może być znak "//" - w FORTRANIE służący do łączenia łańcuchów tekstowych, zaś w PASCALU do oznaczenia wiersza komentarza.

Fortran	**Pascal**
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z	
a b c d e f g h i j k l m n o p q r s t u v w x y z	
1 2 3 4 5 6 7 8 9 0	
+ - / * . , = ( ) : _	+ - * / = ^ < > ( ) [ ] { } . , : ; ' # $ @ spacja _
.LT. .LE. .EQ. .NE. .GE. .GT. .AND. .OR. .NOT. .EQV. .NEQV.	:= <> <= >= .. (. .) (* *)

zestawienie znaków dopuszczalnych języka Fortran 77 oraz Free Pascal

Słowa kluczowe

Skończona (zazwyczaj) liczba słów kluczowych.

Słowa kluczowe są to określone zbiory znaków posiadające konkretne znaczenie dla translatora (najczęściej są to wyrazy z języka angielskiego, np. WRITE, READ, PROGRAM, IF, PROCEDURE, FUNCTION, itd.).

```
program hello
print *, 'Witaj swiecie :)'
read *
end
```

korzystnie, jeśli edytor kodu koloruje słowa kluczowe

```
PROGRAM hello
PRINT *, 'Witaj swiecie :)'
READ *
END
```

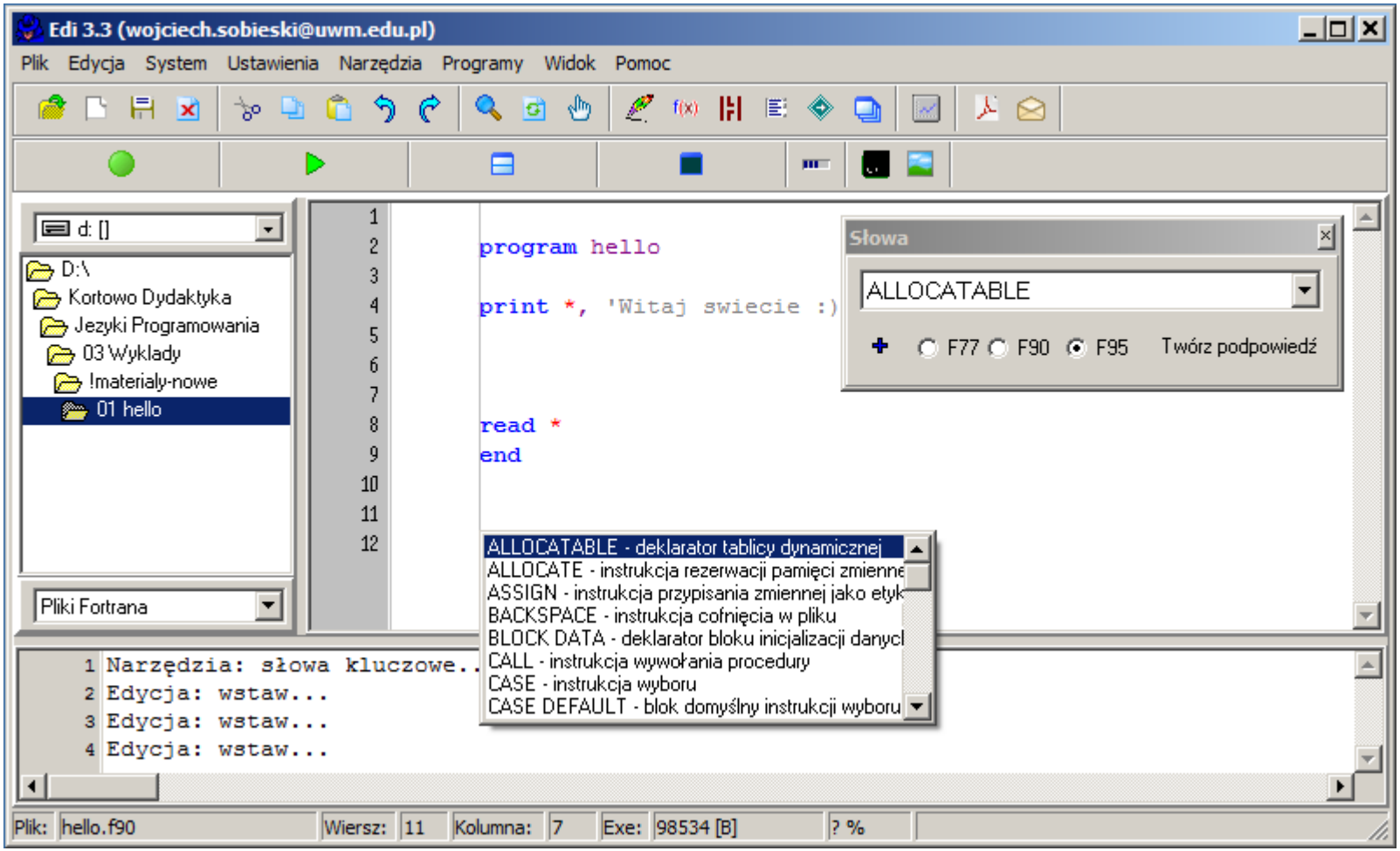
jeżeli edytor nie koloruje słów kluczowych, warto pisać je wielką literą

Słowa kluczowe

	Fortran			**Pascal**	
assign	endfile	parameter	and	if	repeat
backspace	entry	pause	array	implementation	set
block data	equivalence	print	asm	in	shl
call	err	program	begin	inherited	shr
character	external	read	case	inline	string
close	file	real	const	interface	then
common	format	return	constructor	label	to
complex	function	rewind	destructor	mod	type
continue	goto	save	div	nil	unit
data	if	single	do	not	until
dimension	implicit	status	downto	object	uses
do	integer	stop	else	of	var
double precision	inquire	subroutine	end	or	while
else	intrinsic	then	file	packed	with
elseif	iostat	unit	for	procedure	xor
end	logical	write	function	program	
endif	open		goto	record	

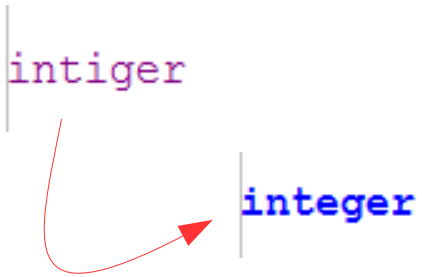
zestawienie słów kluczowych języka Fortran 77 oraz Free Pascal

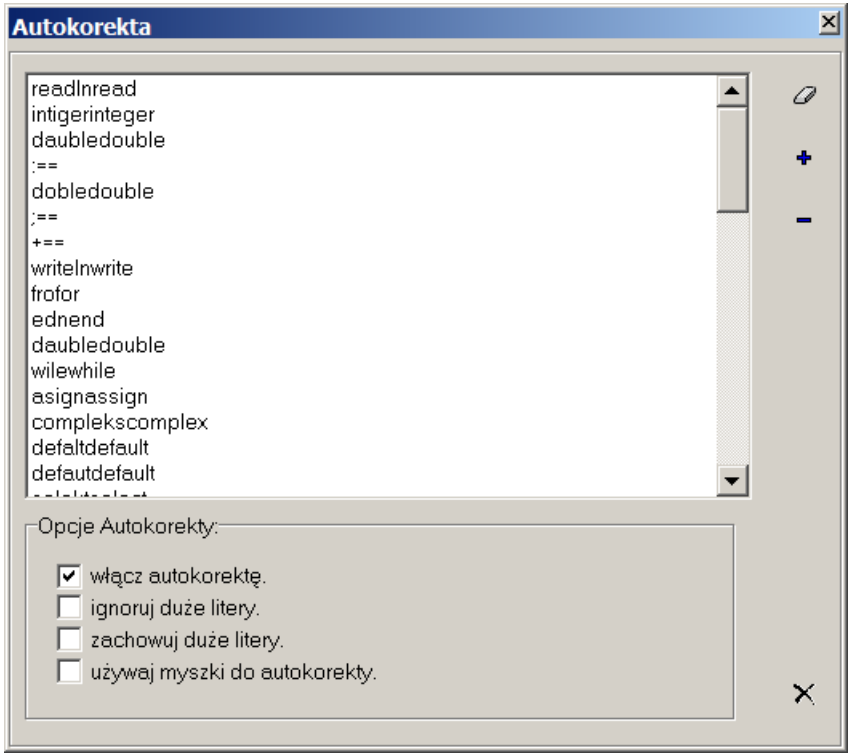
Słowa kluczowe



w pakiecie Edi dostępne jest narzędzie wspomagające używanie słów kluczowych

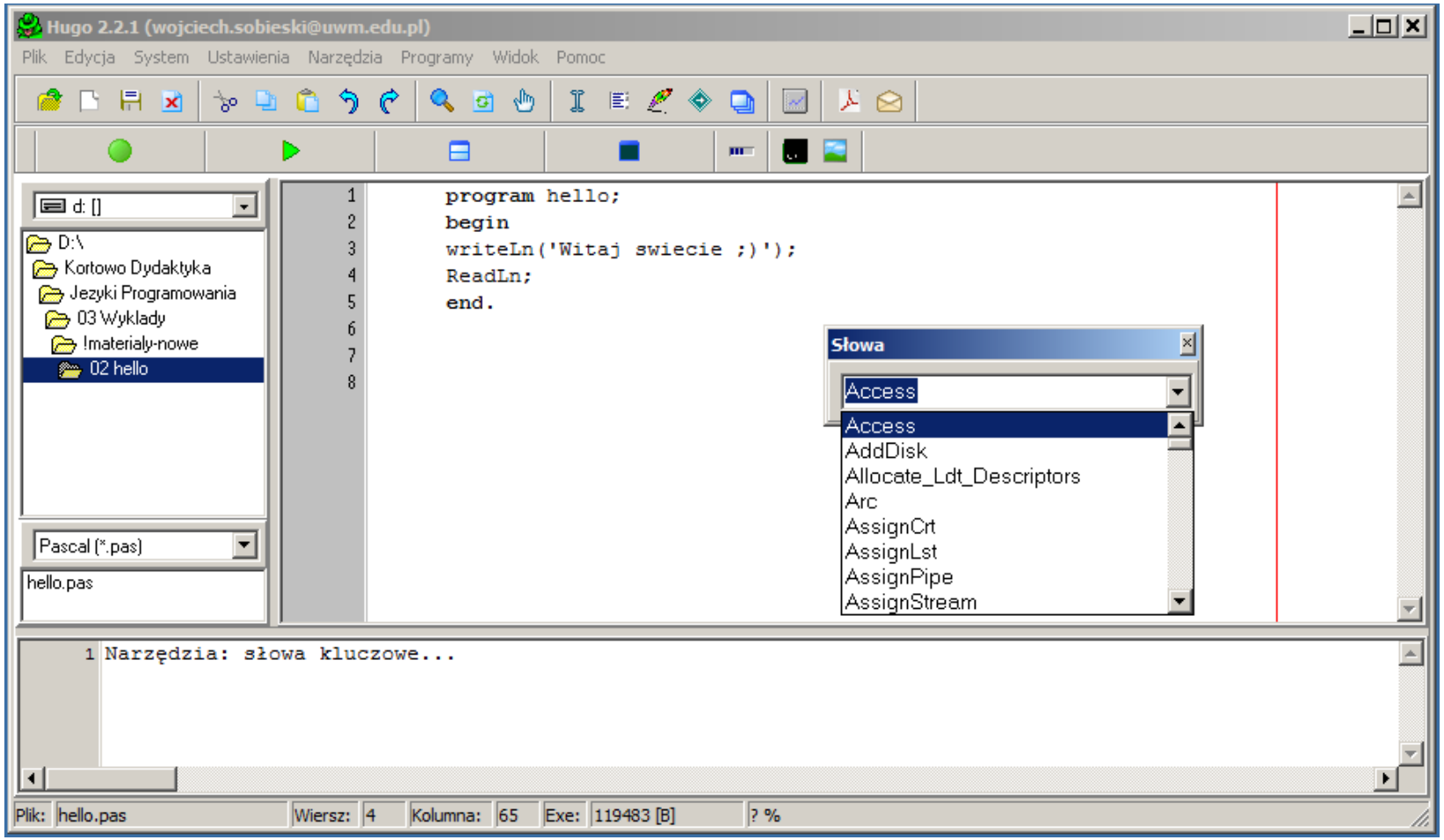
Słowa kluczowe





w pakiecie Edi dostępne jest również narzędzie autokorekty wspomagające używanie słów kluczowych

Słowa kluczowe



lista słów kluczowych w programie Hugo (odpowiednik programu Edi, ale dla języka Pascal)

Struktura kodu

Ściśle określona struktura i zasady pisania kodu źródłowego.

Kod źródłowy składa się z przynajmniej jednego modułu (programu), który musi mieć sztywno określoną budowę (szczegóły zależą od języka):

- wyraźnie określony początek i koniec
- obszary deklaracji obiektów i typów występujących zmiennych
- obszary zawierające bloki instrukcji, funkcji lub procedur

```
program hello                          - początek
print *, 'Witaj swiecie :)'            - blok instrukcji
read *                                 - blok instrukcji
end                                    - koniec
```

zestawienie zasad pisania kodu źródłowego w języku Fortran oraz Free Pascal

Zasady edycji kodu

Fortran	Free Pascal
1. Używanie jedynie znaków dopuszczalnych.	1. Używanie jedynie znaków dopuszczalnych.
2. Brak rozróżnienia znaków pisanych małą lub wielką literą.	2. Brak rozróżnienia znaków pisanych małą lub wielką literą.
3. Znaków narodowych nie należy używać (nawet w komentarzach).	3. Znaków narodowych nie należy używać (nawet w komentarzach).
4. Nazwy (identyfikatory) stałych, zmiennych, podprogramów, funkcji, itp. powinny zaczynać się od litery i powinny być możliwie krótkie (słowa kluczowe mają budowę identyfikatorów, ale mogą być dłuższe).	4. Nazwy (identyfikatory) stałych, zmiennych, podprogramów, funkcji, itp. powinny zaczynać się od litery – długość dowolna.
5. Treść instrukcji wpisuje się między 7 a 132 kolumną. Nie ma obowiązku zaczynania od 7 kolumny. Znaki za 132 kolumną są ignorowane.	5. Treść instrukcji wpisuje się w dowolnej kolumnie.
6. Odstępy umieszczane w instrukcjach są ignorowane – nie ma różnicy pomiędzy np. GOTO i GO TO.	6. Odstępy umieszczane w instrukcjach są ignorowane – nie ma różnicy pomiędzy np. x:=4 a x := 4.

Zasady edycji kodu

Fortran	Free Pascal
7. W celu zwiększenia czytelności programu należy stosować wcięcia – szczególnie podczas stosowania instrukcji zagnieżdżonych.	7. W celu zwiększenia czytelności programu należy stosować wcięcia – szczególnie podczas stosowania instrukcji zagnieżdżonych.
8. W jednym wierszu powinna znajdować się tylko jedna instrukcja (ale można pisać wiele).	8. W jednym wierszu może znajdować się kilka instrukcji, np. if x=1 then y:=1 else y:=2;.
9. Instrukcje przypisania realizowane są za pomocą znaku „=”.	9. Instrukcje przypisania realizowane są za pomocą znaku „:=”.
10. Na końcu wiersza nie stawia się żadnego znaku.	10. Na końcu wiersza zazwyczaj jest średnik.
11. Poszczególne bloki programu zaczynają się i kończą w dowolny sposób.	11. Poszczególne bloki programu zaczynają się i kończą w dowolny sposób, ale muszą być umieszczone między słowami BEGIN i END.
12. Główny blok programu rozpoczyna się słowem PROGRAM z nazwą, a kończy słowem END. Po END nie stawia się kropki	12. Główny blok programu rozpoczyna się słowem PROGRAM z nazwą, a kończy słowem END z kropką na końcu.
13. Warunki arytmetyczne i logiczne określone są symbolami (dawniej skrótami słownymi).	13. Warunki arytmetyczne i logiczne określone są symbolami.

Zasady edycji kodu

Fortran	Free Pascal
14. Jedna instrukcja może zajmować maksymalnie 40 wierszy, przy czym pierwszy wiersz nazywa się wierszem początkowym, a pozostałe wierszami kontynuacji. Każdy wiersz kontynuacji należy zaznaczyć poprzez umieszczenie w szóstej kolumnie znaku różnego od zera i spacji.	14. Jedna instrukcja może zajmować dowolną liczbę wierszy.

15. W programie należy umieszczać komentarze – zaczynają się one znakiem „*”, „C” lub „!”. Komentarze można dodawać na początku linii lub w dowolnym jej miejscu. Wiersze z samymi spacjami traktowane są jak wiersze komentarzy	15. W programie należy umieszczać komentarze. Bloki komentarza ujmuje się w nawiasy klamrowe:
	{
	komentarz...
	}

16. W kolumnach 1-5 można wpisywać tzw. etykiety – niepowtarzalny ciąg cyfr – identyfikujące określone miejsce w programie.	16. Brak etykiet (istnieje możliwość zaznaczania miejsc w kodzie poleceniem LABEL).

Zasady edycji kodu

Fortran	Free Pascal
17. Numery etykiet powinny być nadawane w odstępach (standardowo co 10) – daje to możliwość dopisania nowych etykiet pomiędzy już istniejące.	17. -
18. W programie najpierw umieszcza się deklaracje a dopiero po nich instrukcje.	18. W programie najpierw umieszcza się deklaracje a dopiero po nich instrukcje.
19. Deklaracja zmiennych: **typ** :: *nazwa*	19. Deklaracja zmiennych: *nazwa* : **typ**
20. Zmienne są lokalne, tzn. obowiązują w obszarze jednego modułu. Wymiana wartości odbywa się poprzez odpowiednie instrukcje.	20. Zmienne mogą być lokalna oraz globalne (zależy to od miejsca deklaracji).
21. Podczas pisania programów należy dążyć do zminimalizowania liczby zmiennych.	21. Podczas pisania programów należy dążyć do zminimalizowania liczby zmiennych.
22. Należy dbać o przejrzystość i czytelność kodu – dlatego też zaleca się stosowanie podprogramów i funkcji.	22. Należy dbać o przejrzystość i czytelność kodu – dlatego też zaleca się stosowanie podprogramów i funkcji.

Identyfikatory

Jednoznaczność identyfikatorów w obrębie jednego programu[1].

Identyfikatory są to nazwy programów, modułów, bloków, podprogramów, procedur, funkcji i zmiennych.

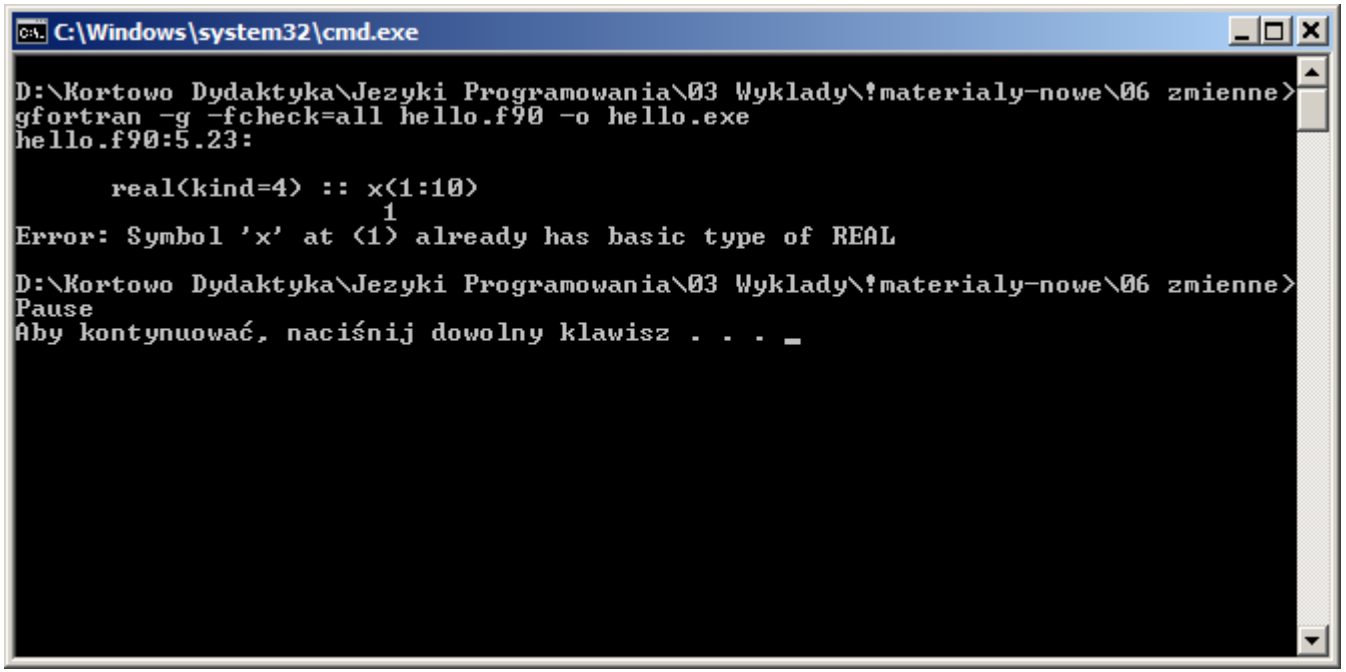
Fortran		**Pascal**	
Przykłady:		Przykłady:	
program Silnia	- poprawnie	program Kalkulator;	- poprawnie
program Write	- błędnie	program Public;	- błędnie
integer n, krok	- poprawnie	n, krok : byte;	- poprawnie
real program, sin	- błędnie	program, sin : real;	- błędnie
procedure read	- błędnie	procedure open;	- błędnie

przykłady poprawnych i błędnych identyfikatorów w języku Fortran oraz Free Pascal

[1]Identyfikatory obiektów lokalnych mogą posiadać takie same nazwy.

Identyfikatory

```
real(kind=4)  :: x
real(kind=4)  :: x(1:10)
```



przykład błędu wynikającego ze zdublowania identyfikatora

Operacje wejścia/wyjścia

Konieczność komunikacji programu z otoczeniem (operacje I/O).

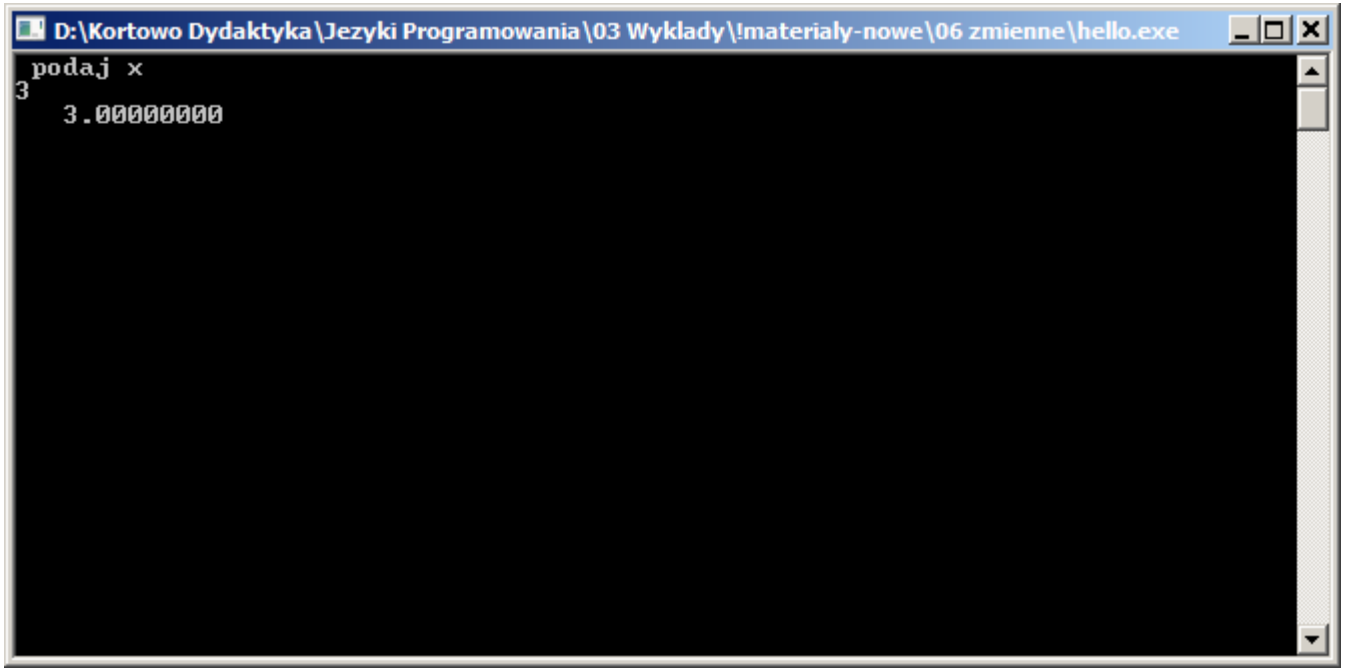
Standardowo do komunikacji z użytkownikiem służy klawiatura (domyślne wejście) oraz monitor (domyślne wyjście).

Pobieranie danych z klawiatury (lub pliku) wymaga wcześniejszego zadeklarowania typu zmiennej (chyba, że istnieje możliwość niejawnej deklaracji typu) – wprowadzany typ będzie wówczas jednoznaczny i zgodny z oczekiwaniami.

Podczas wyprowadzania danych na monitor (lub do pliku) należy ustalić ich format (wygląd). Podczas zapisu (odczytu) danych do pliku należy dodatkowo określić rodzaj pliku (tekstowy czy binarny).

Operacje wejścia/wyjścia

```
print *, 'podaj x '
read *, x
print *, x
```



przykład operacji I/O na urządzeniach standardowych (domyślnych)

Typy zmiennych

Korzystanie z różnych typów zmiennych.

Rozróżnia się typ **całkowity**, **rzeczywisty**, **zespolony**, **tekstowy**, **logiczny** i **inne**. Zależnie od języka, poszczególne typy mogą mieć wiele odmian.

Niektóre języki dopuszczają typ **variant** (może spowalniać program), który interpretowany jest w zależności od przyjętej wartości.

Oprócz typów prostych, wiele języków umożliwia tworzenie własnych typów lub struktur danych o bardziej złożonej budowie. Odpowiednie dobranie typów zmiennych pozwala zaoszczędzić pamięć operacyjną oraz skrócić czas wykonywania obliczeń.

W niektórych językach (ale nie w Fortranie) istnieje konieczność częstej konwersji typów, czyli zamiany zawartości jednego typu na inny.

Typy zmiennych

Fortran				
Nazwa typu	Zakres min.	Zakres max.	Bajty	Dokładność
INTEGER	- 2.147483648	+ 2.147483647	4	-
REAL	$-3.4028235 \cdot 10^{38}$	$+3.4028235 \cdot 10^{38}$	4	6-7
DOUBLE PRECISION	$-1.7976931348623 16 \cdot 10^{308}$	$+1.7976931348623 16 \cdot 10^{308}$	8	15-16
COMPEX	Wartość typu COMPEX jest uporządkowaną parą liczb typu REAL.			
DOUBLE COMPEX	Typ DOUBLE COMPEX jest uporządkowaną parą liczb typu DOUBLE PRECISION.			
LOGICAL	TRUE lub FALSE		4	-
CHARACTER	0 znaków	256 znaków	1 na znak	-

typy zmiennych oraz ich zakresy w języku Fortran 77

```
print *, maxexponent(x)
print *, minexponent(x)
print *, precision(x)
```

```
 128
-125
   6
```

instrukcje umożliwiające sprawdzenie zakresu i dokładności zmiennych w języku Fortran

Typy zmiennych

Pascal				
Nazwa typu	Zakres min.	Zakres max.	Bajty	Dokładność
INTEGER	- 32768	+ 32767	2	-
SHORTINT	- 128	+ 128	1	-
LONGINT	- 2147483648	+ 2147483647	4	-
BYTE	0	255	1	-
REAL	$-2.9 \cdot 10^{39}$	$+1.7 \cdot 10^{38}$	6	11-12
SINGLE	$-1.5 \cdot 10^{45}$	$+3.4 \cdot 10^{38}$	4	7-8
DOUBLE	$-5.0 \cdot 10^{324}$	$+1.7 \cdot 10^{308}$	8	15-16
EXTENDED	$-1.9 \cdot 10^{4951}$	$+1.1 \cdot 10^{4932}$	10	19-20
COMP	0	$+9.2 \cdot 10^{18}$	8	19-20
BOOLEAN	TRUE lub FALSE		4	-
STRING	0 znaków	256 znaków	1 na znak	-

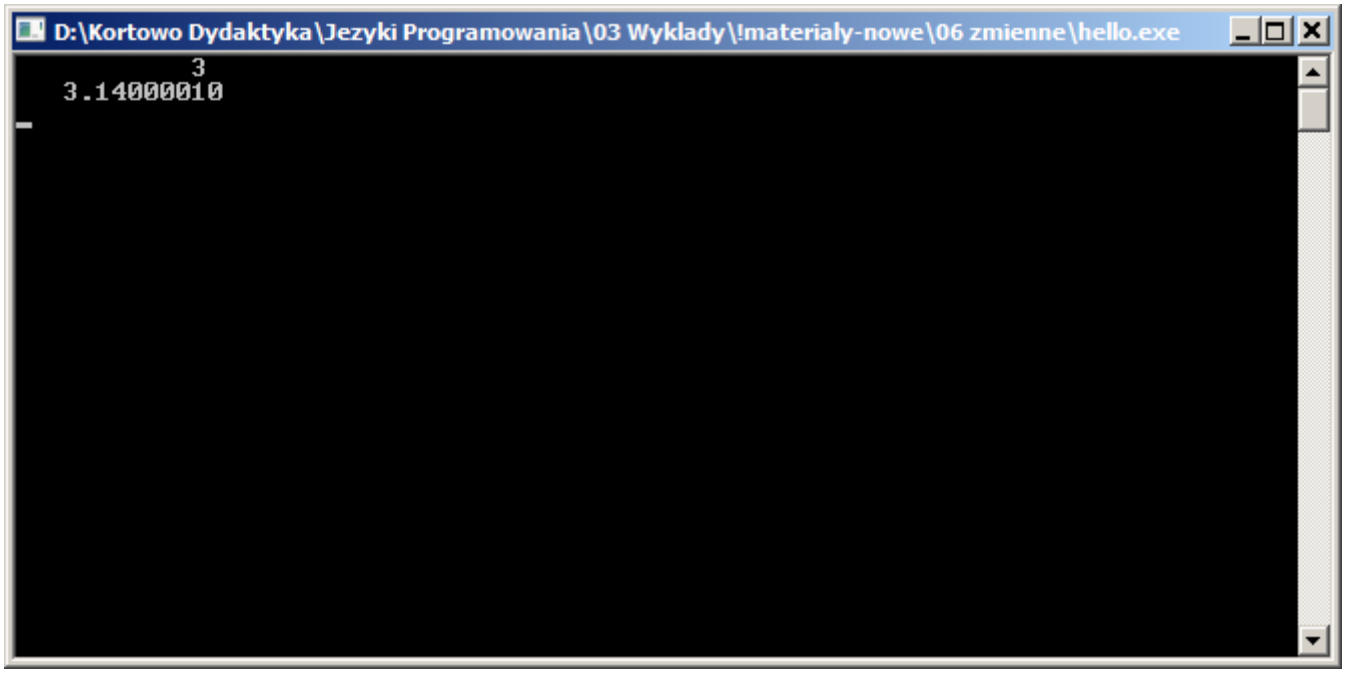
typy zmiennych oraz ich zakresy w języku Free Pascal

Typy zmiennych

Niejawna deklaracja typu – określanie typu zmiennej na podstawie pierwszej litery jej nazwy (np. Fortran) lub też na podstawie wartości (np. Basic – typ variant).

```
i = 3.14
x = 3.14

print *, i
print *, x
```



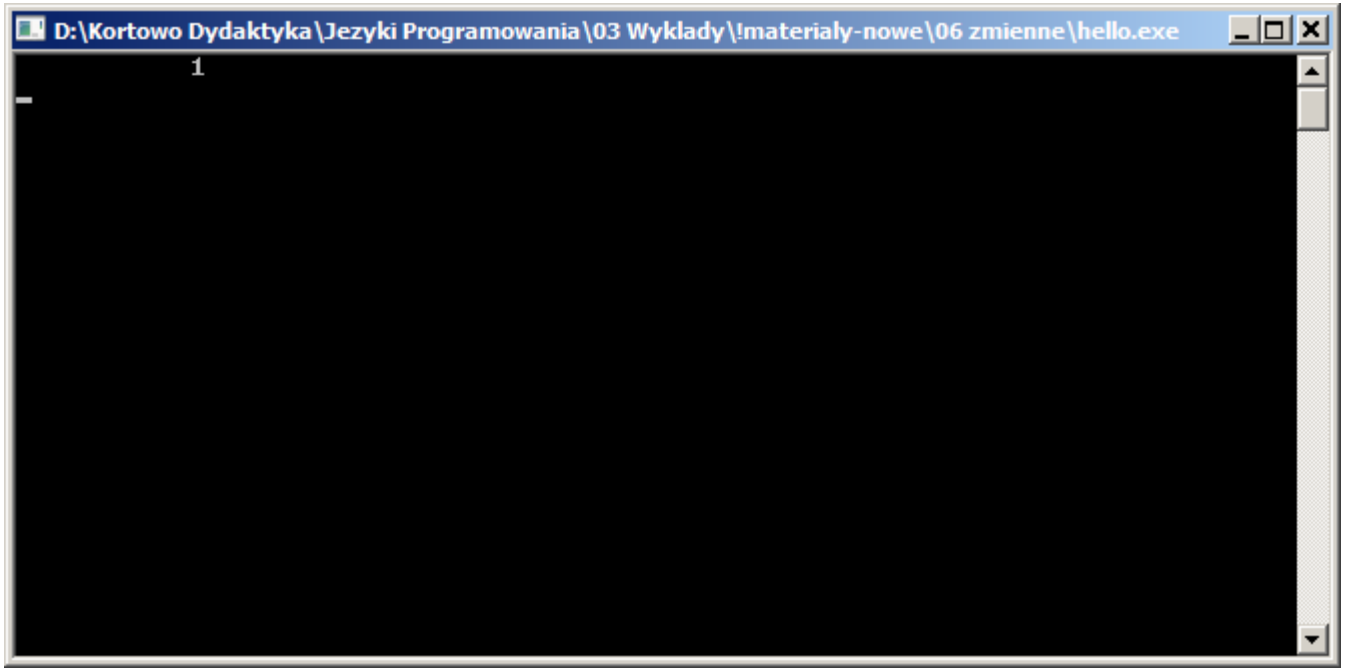
w Fortranie zmienne zaczynające się od liter **i**, **j**, **k**, **l**, **m**, **n**, mają typ całkowity, reszta ma typ rzeczywisty

Typy zmiennych

Niezgodność typów może prowadzić do poważnych błędów, które trudno wykryć!

```
x = 1.
i = 3.14/2

print *, i
```



przykład błędu wynikającego z braku kontroli nad typami zmiennych

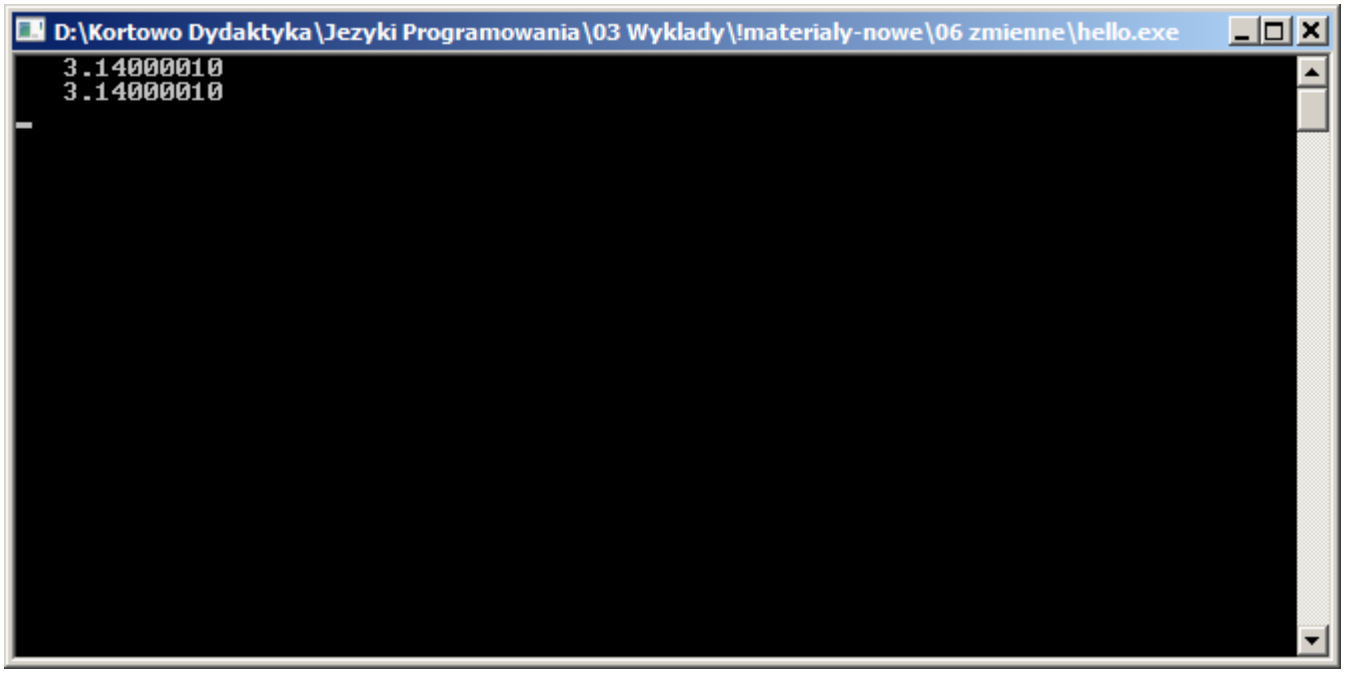
Typy zmiennych

Jawna deklaracja typu – określanie typu zmiennej na podstawie deklaracji.

```
real(kind=4) :: i
real(kind=4) :: x

i = 3.14
x = 3.14

print *, i
print *, x
```



teraz pierwsza litera nie ma znaczenia

Typy zmiennych

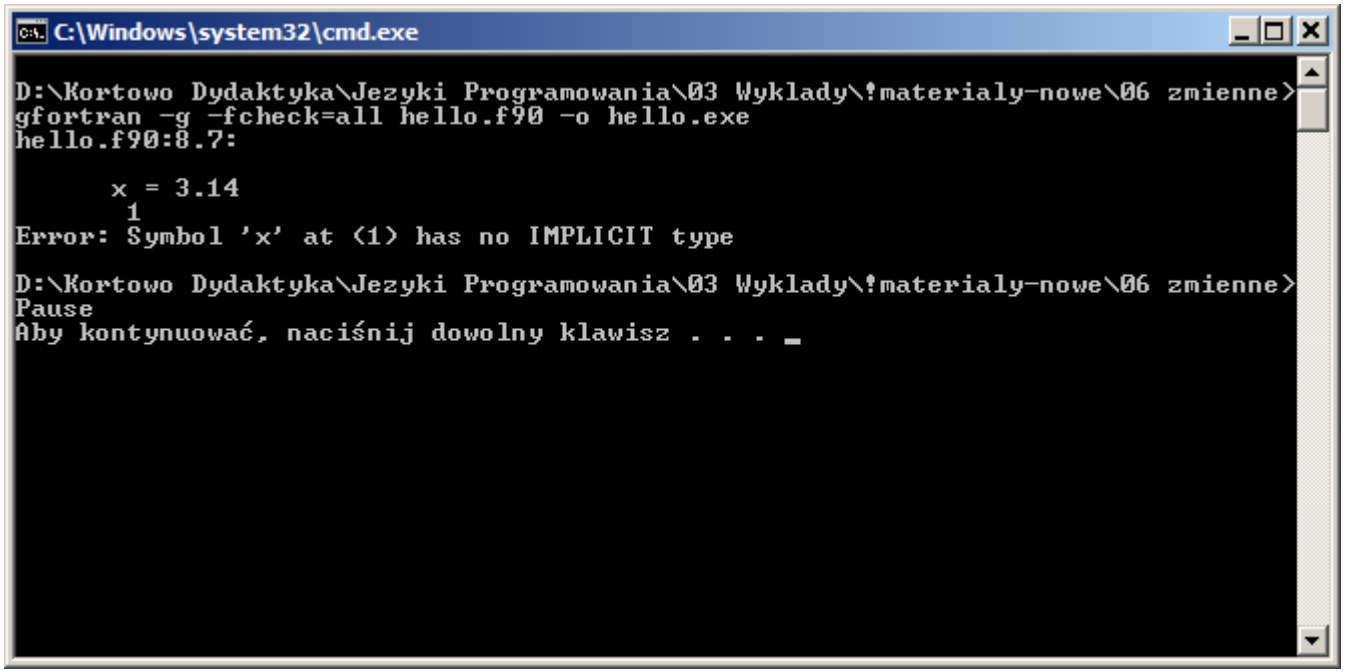
W języku Fortran można wymusić konieczność deklaracji typów (zalecane).

```
implicit none

real(kind=4) :: i

i = 3.14
x = 3.14

print *, i
print *, x
```



przykład błędu wynikającego z braku deklaracji typu użytej zmiennej

Formatowanie

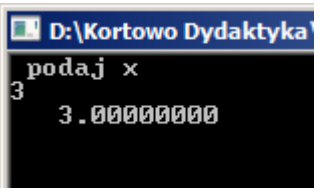
Potrzeba sterowania formatem (wyglądem) zmiennych.

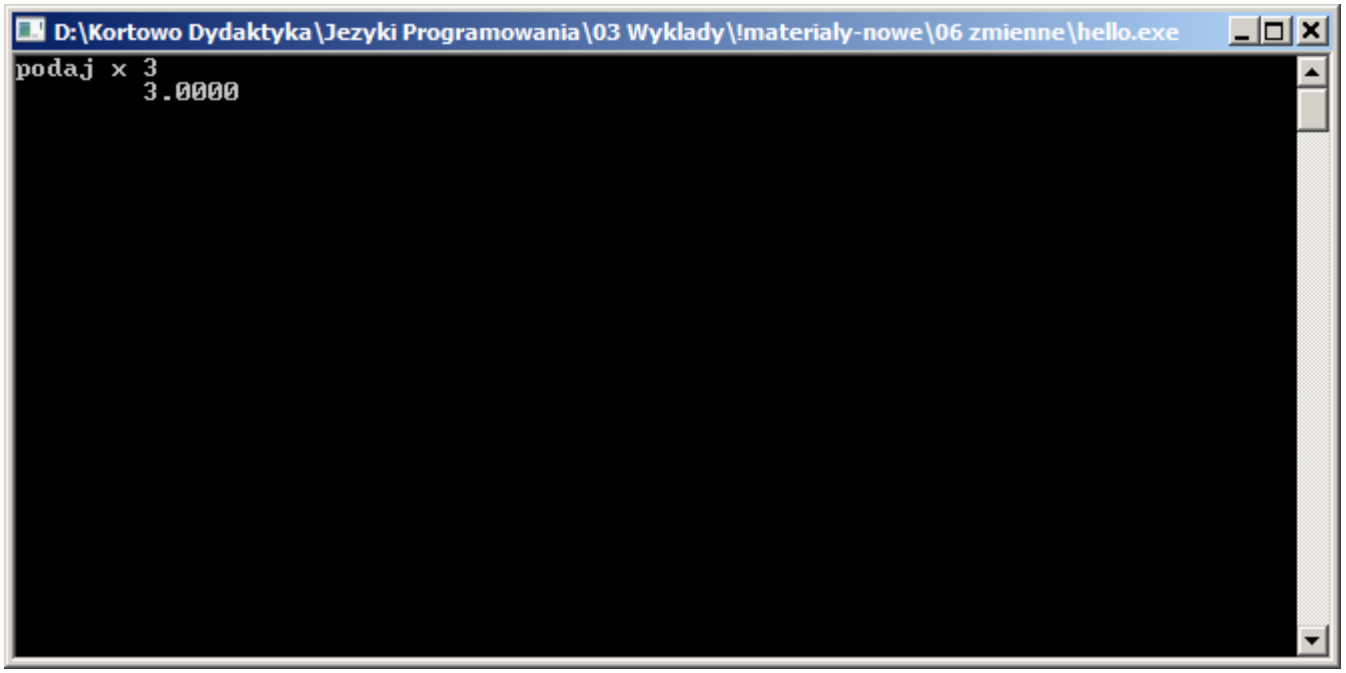
Formatowanie jest dokonywane podczas wyprowadzania wartości danych na ekran monitora lub podczas zapisu do plików.

Formatowanie zmiennych nie zawsze jest konieczne (każdy język programowania posiada domyślne sposoby zapisu poszczególnych typów zmiennych) ale może być bardzo przydatne – można zażądać, aby wszystkie liczby miały tą samą ilość znaków i były zapisane np. w postaci wykładniczej o określonej liczbie cyfr po przecinku.

Formatowanie

```
print '(A$)', 'podaj x '
read *, x
print '(F14.4)', x
```



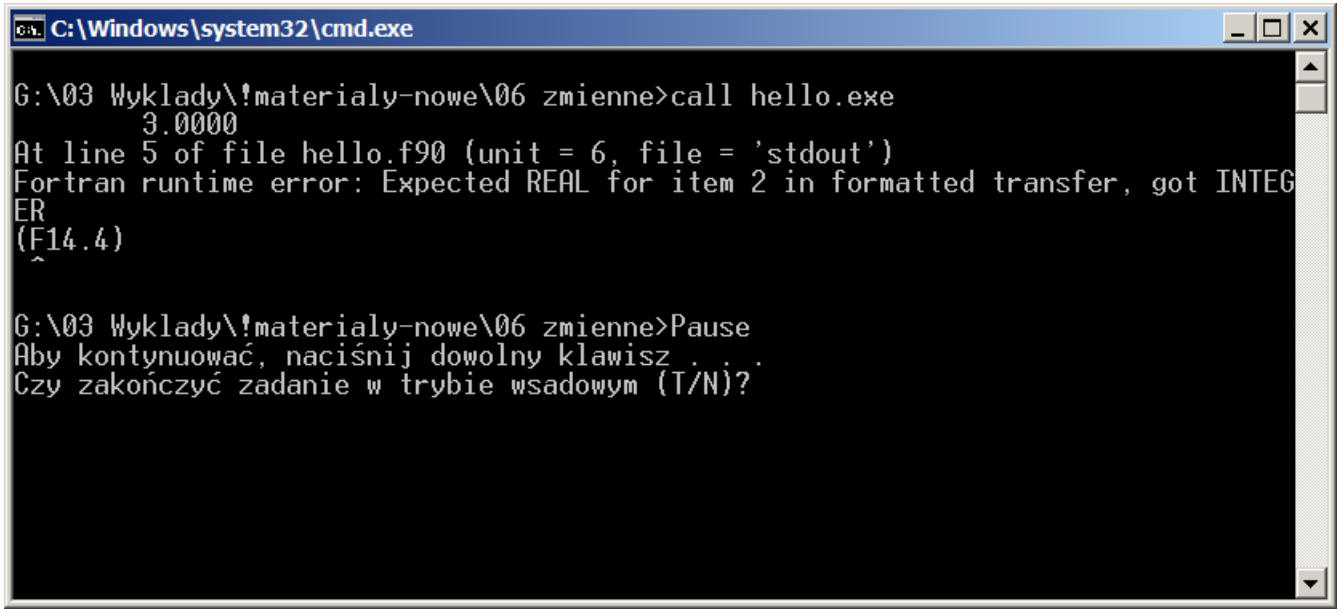


przykład formatowania operacji I/O na urządzeniach standardowych (domyślnych)

Formatowanie

```
print '(F14.4)', 3.
print '(F14.4)', 1
```

kropka jest ważna!



w języku Fortran liczby bez znaku separatora (kropki) traktowane są jaki całkowite

Formatowanie

```
print '(F14.4)', 3.
print '(F14.4)', 1
```

14 – informacja, że należy zarezerwować 14 pól do wyświetlenia zmiennej

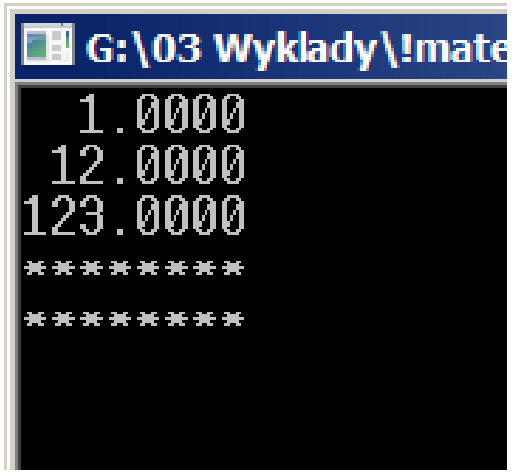
znak separatora zabiera jedno pole

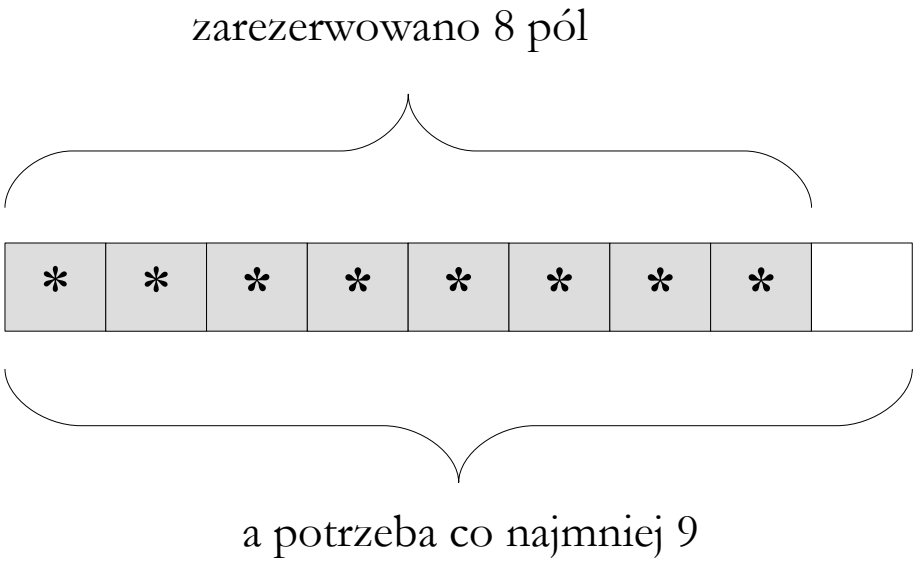
F – informacja, że chodzi o liczby rzeczywiste pojedynczej precyzji

4 – informacja, że na część po przecinku rezerwuje się 4 pola (liczba jest zaokrąglana, ale tylko na urządzeniu wyjściowym – nie w pamięci)

Formatowanie

```
print '(F8.4)', 1.
print '(F8.4)', 12.
print '(F8.4)', 123.
print '(F8.4)', 1234.
print '(F8.4)', 12345.
```





4 znaki mają być po przecinku
1 znak jest na separator
4 znaki przed przecinkiem

Zakresy obowiązywania zmiennych

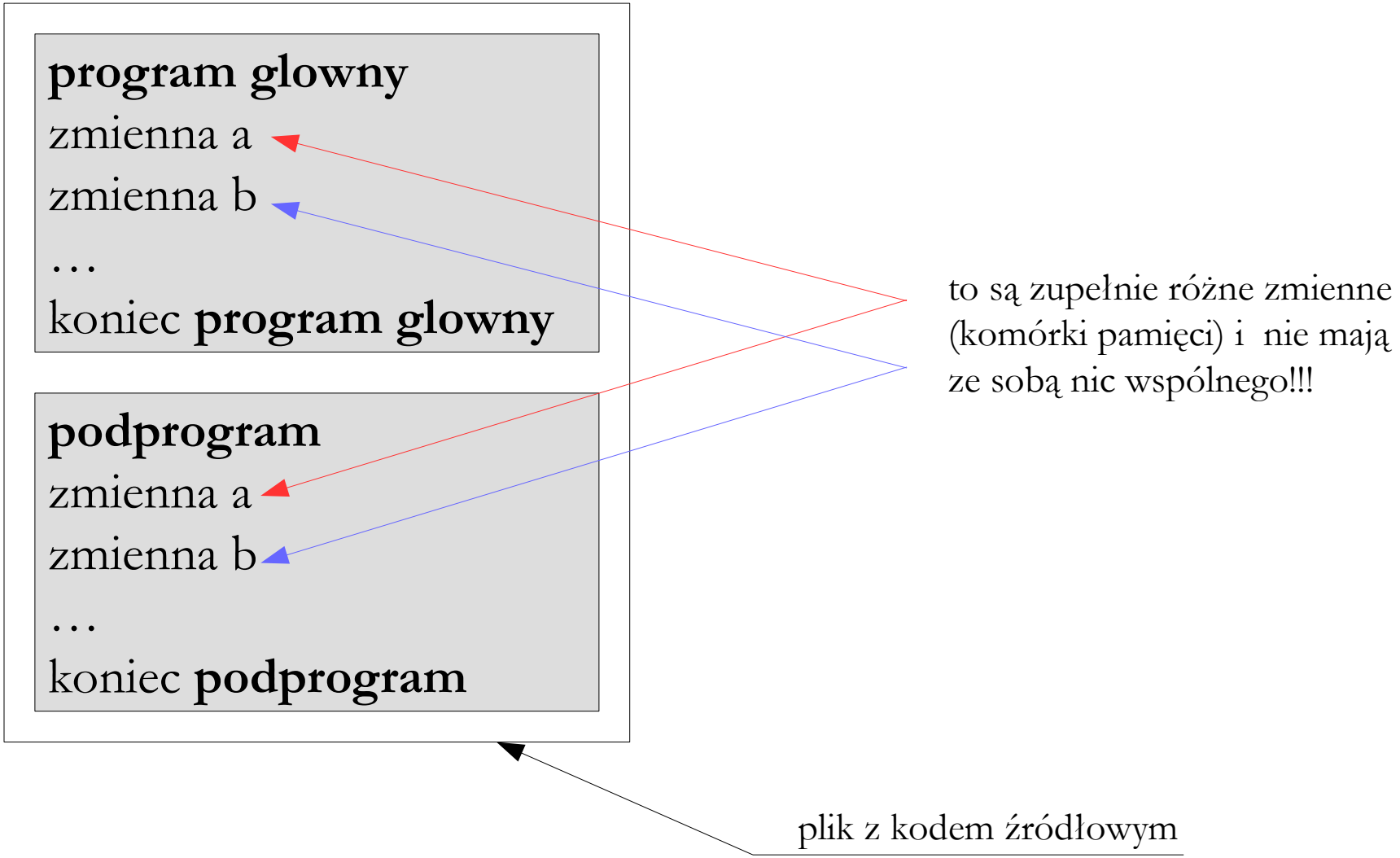
Potrzeba wymiany danych między różnymi modułami (podprogramami, funkcjami i procedurami).

Sposób wymiany danych (wartości zmiennych) zależy w dużej mierze od organizacji i struktury konkretnego języka programowania.

Generalnie rozróżnia się (jest tu pewne uproszczenie):

- zmienne globalne – widziane w całym programie
- zmienne lokalne – widziane w bieżącym module

Zakresy obowiązywania zmiennych



Działania na plikach

Potrzeba wykonywania działań na plikach.

Rodzaje działań:

- zapis i odczyt
- kasowanie, przenoszenie, zmiana nazwy

```
open(1,file='dane.txt')
read(1,*) x, y, z
close(1)
```

czytanie z pliku (1 zamiast *)

```
open(1,file='dane.txt')
write(1,*) x, y, z
close(1)
```

zapis od pliku (1 zamiast *)

```
call system('rename dane.txt dane.dat')
```

zmiana nazwy pliku

Komunikacja z systemem operacyjnym

Potrzeba komunikacji z systemem operacyjnym.

Przydatnym elementem języka programowania jest możliwość korzystania w funkcji systemu operacyjnego. Jest to zazwyczaj możliwe na dwa sposoby:

- język posiada własne instrukcję odpowiadające poleceniom systemu operacyjnego (ale i tak przeważnie w jakiś sposób z nich korzysta)
- język posiada mechanizm wywoływania poleceń systemowych i ich parametrów

```
call system('rename dane.txt dane.dat')
```

Zarządzenie pamięcią

Potrzeba zarządzania pamięcią operacyjną komputera.

Zarządzanie statyczne – przypisywanie rozmiaru zmiennych indeksowanych w kodzie źródłowym (przed kompilacją). Nie ma możliwości zmiany rozmiaru zmiennej w trakcie działania programu.

```
real(kind=4)                    ::    xs(1:5)
```

Zarządzanie dynamiczne – przypisywanie rozmiaru zmiennych indeksowanych w trakcie działania programu (przed kompilacją).

```
real(kind=4),allocatable        ::    xd(:)

allocate(xd(1:n))          }  n ustala się w trakcie działania programu

 . . .

deallocate(xd)             }  przypisaną pamięć należy na końcu zwolnić
```

Zmienne i stałe

Zmienna (prosta) – symboliczna nazwa komórki w pamięci operacyjnej komputera, służąca do zapisu wartości liczbowej, tekstowej lub logicznej. Zmienne są podstawowym elementem algorytmu obliczeniowego. W czasie działania programu wartości przypisane zmiennym mogą się dowolnie zmieniać, zależnie od wykonywanych operacji. Zmienne są rozróżniane po nazwach, czyli tzw. identyfikatorach. Nadanie wartości zmiennym odbywa się poprzez instrukcję przypisania – bezpośrednio podczas deklaracji typu lub później (zależy to od języka programowania).

```
real(kind=4)        ::    x
integer(kind=4)     ::    n
logical(kind=4)     ::    l
character(len=8)    ::    s
```

przykłady deklaracji zmiennych prostych w języku Fortran

Zmienne i stałe

Zmienna indeksowana (tablica) – zmienna zawierająca nie jedną wartość lecz cały ich zbiór. W przypadku zmiennych indeksowanych oprócz deklaracji typu należy podać jej wymiar. Służy do zapisywania list, wektorów i tablic. Maksymalna ilość wymiarów zmiennej indeksowanej jest zależna od języka programowania.

```
real(kind=4)        ::    x(1:10)
integer(kind=4)     ::    n(1:10)
logical(kind=4)     ::    l(1:10)
character(len=8)    ::    s(1:10)
```

przykłady deklaracji zmiennych indeksowanych w języku Fortran

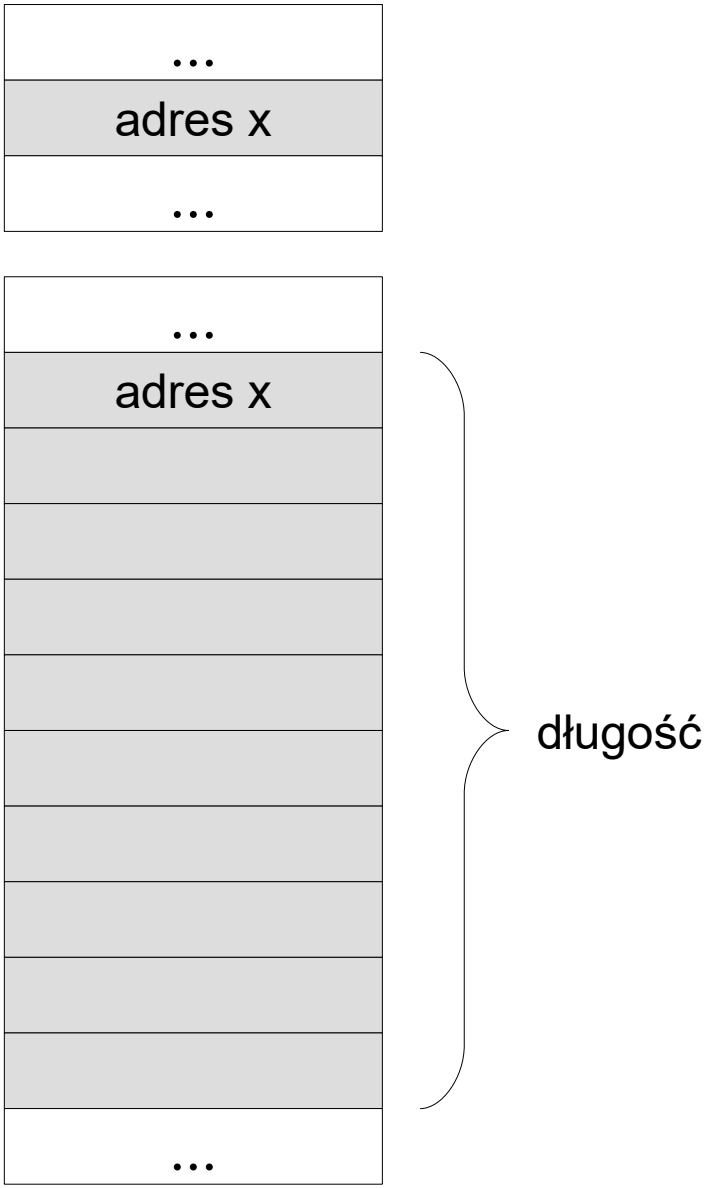
Zmienne i stałe

```
real(kind=4)   :: x
```

```
real(kind=4)   :: x(1:10)
```

• do deklaracji zmiennej prostej potrzebny jest adres komórki pamięci

• do deklaracji zmiennej indeksowanej potrzebny jest adres komórki pamięci oraz długość bloku pamięci



Zmienne i stałe

czy to wyrażenie jest poprawne?

$$x = x + 1$$

Zmienne i stałe

pobranie wartości z komórki pamięci:

$$x = x + 1$$

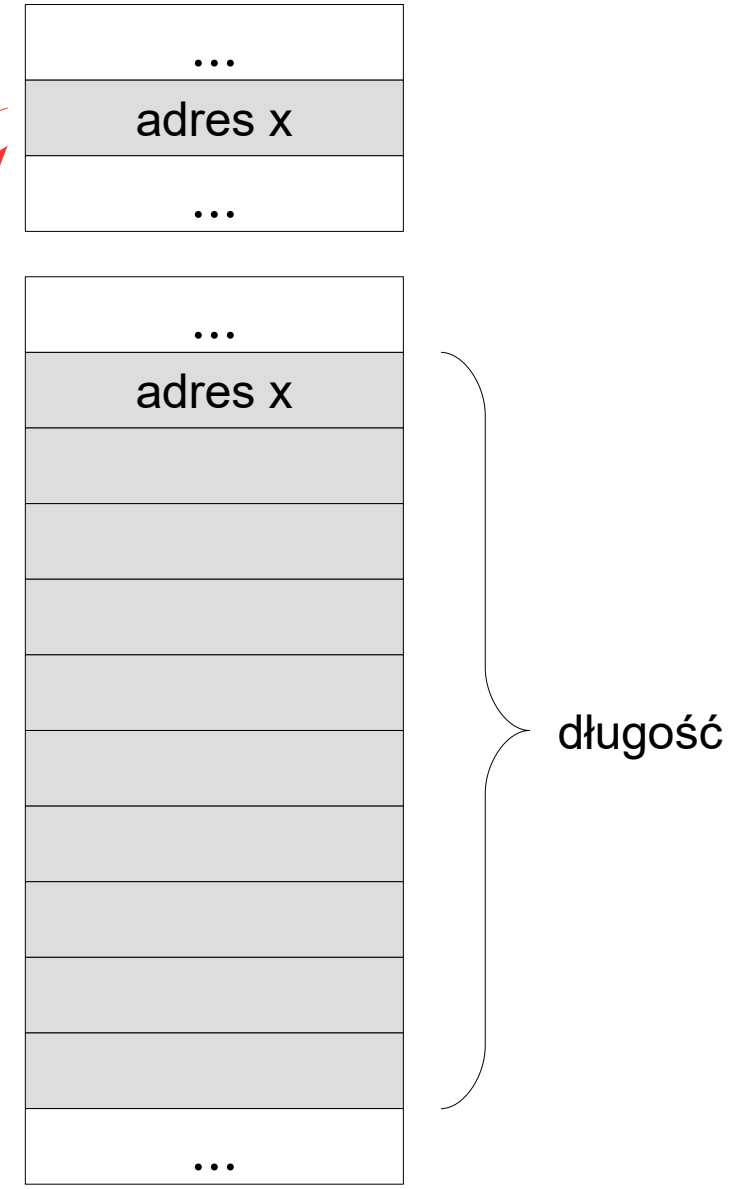
obliczenie prawej strony:

$$x = x + 1$$

zapisanie wyniku w komórce pamięci:

$$x = x + 1$$

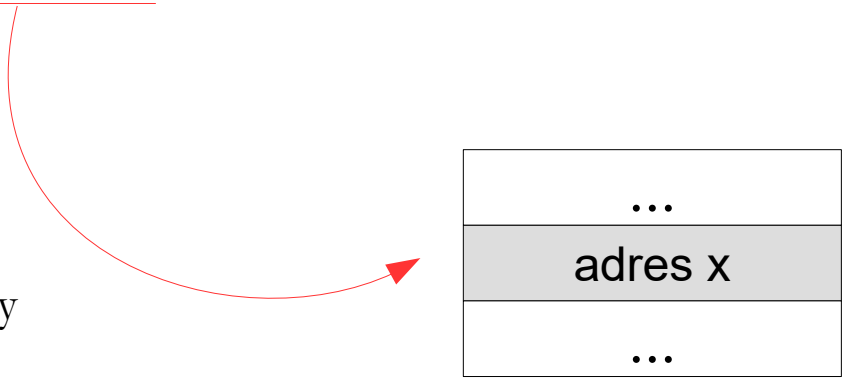
to jest operator przypisania (nadania wartości), a nie operator relacji (porównania)



Zmienne i stałe

Stała – symboliczna nazwa komórki w pamięci operacyjnej komputera służąca do zapisu określonej wartości. Różnica pomiędzy zmienną a stałą jest taka, że wartości stałej nie można zmienić podczas wykonywania obliczeń. Deklaracja stałej i nadanie wartości realizuje się odpowiednią instrukcją.

```
real(kind=4),parameter :: pi = 3.141592
```



blokada możliwości zmiany zawartości komórki

Zmienne i stałe

Podział zmiennych i stałych ze względu na typ:

- zespolone
- rzeczywiste
- całkowite
- logiczne
- walutowe
- daty lub czasu
- tekstowe

Podział zmiennych i stałych ze względu na sposób deklaracji:

- deklarowane jawnie
- deklarowane niejawnie

Zmienne i stałe

Podział zmiennych i stałych ze względu na dokładność (dotyczy zmiennych zespolonych i rzeczywistych):

- o pojedynczej precyzji
- o podwójnej precyzji

Podział zmiennych i stałych ze względu na strukturę:

- prosty
- indeksowany (wektory i tablice)

Zmienne i stałe

Podział zmiennych i stałych ze względu na ilość zajmowanej pamięci:

- o określonej ilości bajtów (1, 2, 4, 6, 8 lub 10 bajtów)
- o dowolnej ilości bajtów

Podział zmiennych i stałych ze względu na sposób przydziału pamięci:

- statyczne
- dynamiczne

Wyjątki

Potrzeba obsługi błędów (wyjątków).

Wyjątek – zdarzenie niezgodne z oczekiwaniami.

Obsługa wyjątków jest konieczna ze względu na dwa podstawowe czynniki:

- niemożność przewidzenia wszystkich działań użytkownika
- możliwość wystąpienia awaryjnych stanów pracy komputera (np. uszkodzenie pliku, nośnika danych, systemu operacyjnego)

Język Fortran posiada ubogie możliwości obsługi błędów.

```
(err=10)
```

numer etykiety, do której program ma przeskoczyć w przypadku zaistnienia błędu

Wyrażenia i operatory

Potrzeba wykonywania działań na zmiennych.

Działania na zmiennych wymagają stosowania odpowiednich:

- **wyrażeń** (arytmetycznych, logicznych, tekstowych)
- **operatorów** (arytmetycznych, logicznych, tekstowych)

Jeżeli w jakimś wyrażeniu występują jednocześnie operatory arytmetyczne, relacji i logiczne, to kolejność wykonywania działań jest następująca:

- operacje arytmetyczne
- operacje relacji
- operacje logiczne

Wyrażenia arytmetyczne

Wyrażenia arytmetyczne – wyrażenia służce do obliczania wartości zmiennych liczbowych na podstawie wzorów matematycznych. Priorytety: potęgowanie, mnożenie i dzielenie oraz dodawanie i odejmowanie.

W wyrażeniach arytmetycznych mogą wystąpić następujące elementy:

- stałe arytmetyczne
- odwołania do zmiennych
- odwołania do elementów tablic
- wywołania funkcji wewnętrznych i zewnętrznych

```
x = 1.
y(2) = 1.
z = sin(x+1)*cos(y(2)/2.)
```

Operatory arytmetyczne

Operator arytmetyczny – operator dostępny w określonym języku programowania, który działając na podanych argumentach reprezentujących wartości liczbowe, w wyniku zwraca również wyznaczoną wartość liczbową, realizując podstawowe operacje arytmetyki.

Fortran
+ - * / **

BASIC
+ - * / ↑ (**)

C++
+ - * / * % ++ -- << >>

Pascal
+ - * / ** div mod shl shr

przykłady operatorów arytmetycznych w wybranych językach programowania

Wyrażenia logiczne

Wyrażenia logiczne – wyrażenia służące do wyznaczania wartości logicznej typu PRAWDA lub FAŁSZ.

W wyrażeniach logicznych mogą wystąpić następujące elementy:

- stałe i zmienne logiczne
- elementy tablic logicznych
- wywołania funkcji logicznych
- operatory logiczne i operatory relacji

```
if (.TRUE.) print *, 'prawda'
```

Operatory logiczne

Operator logiczny – operator dostępny w określonym języku programowania, który działając na argumentach reprezentujących wartości logiczne, w wyniku zwraca również wartość logiczną, realizując podstawowe operacje algebry Boole'a.

A	B	negacja (zmienna A)	koniunkcja	alternatywa	tożsamość	nie tożsamość
1	1	0	1	1	1	0
1	0	0	0	1	0	1
0	1	1	0	1	0	1
0	0	1	0	0	1	0

tabela wyrażeń logicznych

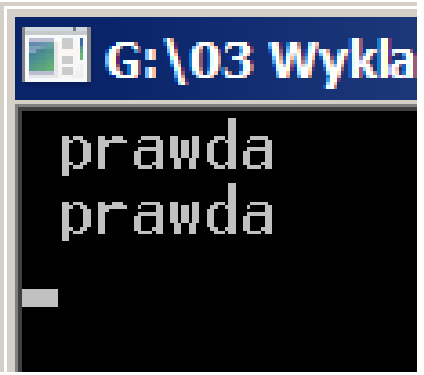
Operatory relacji

Operator relacji – operator dostępny w określonym języku programowania, który działając na podanych argumentach, w wyniku zwraca wartość logiczną, określającą spełnienie bądź nie spełnienie reprezentowanej przez ten operator relacji zachodzącej między zapodanymi argumentami.

Wynikiem działania operatora relacji jest wartość reprezentująca zgodnie z zasadami obowiązującymi w składni danego języka programowania jedną z wartości logicznych: prawdę (**TRUE**) lub fałsz (**FALSE**).

```
if (1 < 2) print *, 'prawda'
if (.TRUE.) print *, 'prawda'
```

należy zapytać CZY wyrażenie jest prawdziwe: TAK lub NIE



Operatory relacji

Fortran 77	opis	Fortran 90/95
.LT.	mniejsze (Less Than)	<
.EQ.	mniejsze równe (Less EQual)	<=
.GT.	większe (Greater Than)	>
.GE.	większe niż (Greater or Equal)	>=
.EQ.	równe (Equal)	==
.NE.	nie równe (Not Equal)	/=

nie mylić operatora relacji z operatorem przypisania, tzn. nadania danej zmiennej określonej wartości!

	Fortran 77	Fortran 90/95	Pascal	Basic
przypisanie	=	=	:=	=
relacja	.EQ.	==	=	=

Wyrażenia tekstowe

Wyrażenia tekstowe – wyrażenia służące do przetwarzania zmiennych tekstowych (np. tworzenie podłańcuchów) oraz do zbierania określonych informacji (np. określanie długości łańcucha, położenia określonego znaku w łańcuchu, itp.).

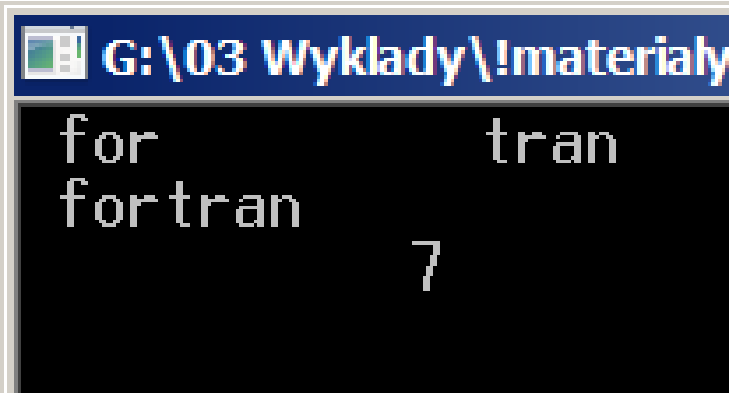
W wyrażeniach logicznych mogą wystąpić następujące elementy:

- stałe i zmienne tekstowe
- podłańcuchy znaków
- wywołania funkcji tekstowych

Wyrażenia tekstowe

```
character(len=12) :: s1 = 'for '
character(len=12) :: s2 = 'transport'

print *, s1//s2(:4)
print *, trim(s1)//trim(s2(:4))
print *, len(trim(trim(s1)//trim(s2(:4))))
```



przykład działań na łańcuchach tekstowych

Wydział Nauk Technicznych

UNIVERSITY OF WARMIA AND MAZURY IN OLSZTYN
The Faculty of Technical Sciences
POLAND, 10-957 Olsztyn, M. Oczapowskiego 11
tel.: (48)(89) 5-23-32-40, fax: (48)(89) 5-23-32-55
URL: http://www.uwm.edu.pl/edu/sobieski/ (in Polish)

Dziękuję

Wojciech Sobieski

Olsztyn, 2001-2021