Wprowadzenie do programowania w języku FORTRAN

Wprowadzenie

Fortran jest językiem programowania zorientowanym głównie na zagadnienia matematyczne i zastosowania inżynierskie. Nazwa jest akronimem słów FORmula TRANslator. Istnieje kilka standardów tego języka programowania. Powszechnie używanym jest Fortran77. Późniejsze wersje np. Fortran90 czy Fortran95 dają większe możliwości związane m.in. z wykorzystaniem pamięci operacyjnej, obliczeniami równoległymi czy grafiką. Zgromadzone ogromne biblioteki programów i procedur zapisane w Fortranie77 powodują, że ten standard jest wciąż aktualny i intensywnie użytkowany. Z powodu licznych zalet, Fortran jest najdłużej wykorzystywanym językiem programowania w historii i ciągle popularnym.

Fortran jest dominującym językiem programowania w zastosowaniach inżynierskich. Rozumienie kodu programu napisanego w Fortranie i umiejętność jego modyfikowania jest zatem cenną i pożądaną umiejętnością absolwenta studiów technicznych.

Struktura programu

Program w języku Fortran ma budowę segmentową – składa się on z tzw. jednostek programowych, którymi są:

- program główny,
- procedury, (subroutines)
- funkcje, (functions)

Struktura programu głównego jest następująca:

```
PROGRAM nazwa
Deklaracje
Instrukcje wykonawcze
END
```

W programie najważniejszy i konieczny jest segment główny. Wykonywanie algorytmu zapisanego w Fortranie zaczyna się zawsze od pierwszej instrukcji wykonawczej programu głównego. Kolejność, w jakiej zapisane są poszczególne jednostki w kodzie źródłowym nie ma znaczenia.

Reguły zapisu kodu programu

Wielkość liter w instrukcjach Fortranu nie ma znaczenia i nie jest rozróżniana przez kompilator. Rozróżnienie to występuje tylko w stałych tekstowych. Spacje są ignorowane. Tradycyjnym formatem zapisu plików źródłowych w języku programowania FORTRAN 77 jest format stały. Dopuszcza on zapis programu w pierwszych 72 kolumnach tekstu, zgodnie z następującymi regułami

Kolumna 1 : znak C, c lub * oznacza linię komentarza
Kolumny 1-5 : etykieta
Kolumna 6 : dowolny znak oznacza kontynuację poprzedniej linii
Kolumny 7-72 : instrukcje fortranu

W nowszym standardzie wykorzystywanym np. przez kompilator FORCE 2.0 dopuszcza się ponadto:

Znak ! w dowolnej kolumnie 1 – 72 z wyjątkiem 6 oznacza, że tekst na prawo od niego jest traktowany jako komentarz

Znak ; służy do oddzielania instrukcji zapisanych w jednej linii

Deklaracje i typy zmiennych i stałych

Zmienne w fortranie są identyfikowane przez ich nazwę. Nazwa jest zaczynającym się od litery ciągiem składającym się z: liter, cyfr oraz znaku _.

Składnia

```
typ lista_zmiennych oddzielonych przecinkami
```

Przykład deklaracji typu:

```
INTEGER I, J23, LONG, PRO1
```

Typy zmiennych i stałych:		przykładowy zapis stałych
INTEGER	typ całkowity	12, -5, 456745
REAL	typ rzeczywisty	12.0, -.3, 1.35E-1
DOUBLE PRECISION	typ podwójnej precyzji	3.54D0, 35.4D-1
COMPLEX	typ zespolony	(-4.6, 5.3)
LOGICAL	typ logiczny	.TRUE., .FALSE.
CHARACTER	typ znakowy	‘Ala’

Informacje o typach zmiennych i stałych

Typ	Liczba bajtów	Zakres wartości	Dokładność
INTEGER	4	od -2 147483648 do +2 147483647	Dokładna reprezentacja
REAL	4	od $-3.4028\ 10^{38}$ do $-1.1755\ 10^{-38}$ od $+1.1755\ 10^{-38}$ do $+3.4028\ 10^{38}$	6-7 pozycji znaczących
DOUBLE PRECISION	8	od $-\ 1.7977\ 10^{308}$ do $-2.2251\ 10^{-308}$ od $+2.2251\ 10^{-308}$ do $+\ 1.7977\ 10^{308}$	15 – 16 pozycji znaczących
COMPLEX	8	tak jak REAL	Tak jak REAL
LOGICAL	4	.TRUE. lub .FALSE.	
CHARACTER	Liczba znaków	znaki ASCI	

Każda zmienna powinna być zdefiniowana w początkowym segmencie programu (deklaracje). Jeśli zmienna nie zostanie zadeklarowana to Fortran77 przyjmie domyślnie jej typ w zależności od pierwszej litery w nazwie. Nazwy zaczynające się od i, j, k, l, m, lub n zostaną automatycznie przypisane do typu całkowitego, natomiast pozostałe do typu rzeczywistego.

Wyrażenia arytmetyczne

Wyrażenie arytmetyczne służy do wyznaczenia, według zadanej formuły (wzoru matematycznego), wartości liczbowej. Operacje są przeprowadzane na danych liczbowych które mogą mieć postać

- stałych liczbowych,
- nazw stałych liczbowych lub zmiennych liczbowych prostych,
- wywołań funkcji typu liczbowego,
- odwołań do elementu tablicy liczbowej.

Operacje arytmetyczne Fortranu

	Symbol	Rodzaj operacji	Priorytet
Operacje dwuargumentowe	**	potęgowanie	1
	/	dzielenie	2
	*	mnożenie	
	+	dodawanie	3
	-	odejmowanie	
Operacje jednoargumentowe	+	znak liczby	3
	-		

Kolejność wykonywania operacji w wyrażeniach arytmetycznych ustalona jest zgodnie z ich priorytetem, przy czym najwyższy priorytet oznaczono liczbą 1. Kolejność działań można zmienić przy pomocy nawiasów okrągłych. Wyrażenia w nawiasie są wykonywane w pierwszej kolejności.

Operacje logiczne

Wyrażenia logiczne mogą przyjmować wartość .TRUE. lub .FALSE. Można tworzyć je przy pomocy operatorów relacji np. poprzez porównywanie wyrażeń arytmetycznych.
Wszystkie relacje są dwuargumentowe.

Operatory relacji w Fortranie

Operator		znaczenie
.LT.	<	mniejszy (less than)
.LE.	<=	mniejszy lub równy (less than or equal to)
.EQ.	= =	równy (equal)
.NE.	/=	różny (not equal to)
.GE.	>=	większy lub równy (greater than or equal to)
.GT.	>	większy (greater than)

Wyrażenia logiczne mogą być rozbudowane przy pomocy operatorów logicznych zestawionych w poniższej tabeli

Operacje logiczne Fortranu

	Symbol	Rodzaj operacji	Priorytet
Operacje jednoargumentowe	.NOT.	negacja	1
Operacje dwuargumentowe	.AND.	koniunkcja	2
	.OR.	alternatywa	3
	.EQV.	równoważność	4
	.NEQV.	nierównoważność	

Instrukcje Fortranu

Z uwagi na znaczenie wyróżniamy:

- instrukcje wykonawcze (executable statements)
- instrukcje niewykonawcze (non-executable statements).

Instrukcje wykonawcze służą do zapisywania operacji algorytmu. Ciąg instrukcji wykonawczych składa się na część wykonawczą jednostki programowej.

Instrukcje niewykonawcze służą do opisu danych, struktury programu, postaci danych w instrukcjach wejścia/wyjścia.

Instrukcje wykonawcze

Instrukcje wykonawcze Fortranu podzielić możemy na trzy grupy:

1. instrukcje przypisania,
2. instrukcje sterujące,
3. instrukcje wejścia/wyjścia.

Instrukcje przypisania powodują nadanie zmiennej określonej wartości.

Instrukcje sterujące decydują o przepływie sterowania w programie tzn. określają kolejność wykonywanych operacji algorytmu. Instrukcje wejścia/wyjścia służą do przesyłania danych z lub do urządzeń wejścia/wyjścia (klawiatura, monitor, pamięć dyskowa, drukarka, ploter itp.).

Instrukcja przypisania

Instrukcja przypisania jest podstawową instrukcją wykonawczą języka programowania. Nazywana jest również niekiedy instrukcją podstawienia. W Fortranie postać tej instrukcji jest następująca:

```
zmienna = wyrażenie
```

gdzie:

wykonanie instrukcji podstawienia składa się z etapów według następującej kolejności:

1. obliczenie wartości wyrażenia po prawej stronie instrukcji,
2. ewentualna konwersja typu wartości wyrażenia,
3. nadanie zmiennej wartości wyrażenia.

Instrukcje sterujące

W Fortranie instrukcje wykonywane są zwykle w takiej kolejności, w jakiej zostały zapisane w kodzie źródłowym, to znaczy od góry ku dołowi i od strony lewej do prawej. Najpierw wykonywana jest pierwsza instrukcja wykonawcza programu głównego. Ten naturalny porządek może zostać zmieniony przez instrukcje sterujące.

Wśród instrukcji sterujących możemy wyróżnić:

- instrukcje warunkowe,
- instrukcje cyklu (pętli),
- instrukcję skoku i pozostałe

Instrukcje warunkowe

Instrukcje warunkowe uzależniają dalszy przebieg algorytmu od spełnienia określonego warunku. Instrukcja warunkowa IF może mieć kilka postaci.

Najprostsza z nich IF logiczny ma postać

```
IF (warunek logiczny) instrukcja_wykonawcza
```

Jeśli wyrażenie logiczne przyjmie wartość prawda (.TRUE.) wówczas wykonana zostanie instrukcja wykonawcza stojąca za warunkiem. Powyższa postać umożliwia wykonanie jednej instrukcji w przypadku spełnienia warunku logicznego. W przypadku kiedy wyrażenie logiczne przyjmuje wartość fałsz (.FALSE.) – program przechodzi do następnej linii. Jeśli więcej instrukcji ma zostać wykonanych przy spełnieniu warunku wówczas postać instrukcji warunkowej jest następująca

```
IF (warunek logiczny) THEN
  Instrukcja 1
  Instrukcja 2
  :
  :
  Instrukcja n
ENDIF
```

Najbardziej ogólna postać instrukcji warunkowej to:

```
IF (warunek logiczny 1) THEN
    ciąg instrukcji
ELSEIF (warunek logiczny 2) THEN
    ciąg instrukcji
  :
  :
ELSE
    instrukcje które zostaną wykonane w przypadku gdy żaden wcześniejszy warunek logiczny nie został spełniony
ENDIF
```

W powyższej instrukcji kolejno od gory sprawdzane są warunki logiczne. Po napotkaniu na warunek którego wartością jest .TRUE., zostają wykonane nastepujące po nim instrukcje, po czym blok IF zostaje opuszczony i wykonuje się następna instrukcja po ENDIF. Instrukcja ELSE nie jest wymagana. Jeśli ona wystąpi to jeden z ciągów instrukcji na pewno zostanie wykonany.

Pętle

Pętle wykorzystuje się gdy zachodzi potrzeba wielokrotnego wykonania określonych instrukcji. Pętla jest instrukcją złożoną. Wystepuje w dwóch postaciach:

Postać 1

```
DO etykieta zmienna=wartość_początkowa,wartość_końcowa[, skok]
 ciąg instrukcji
etykieta instrukcja_wykonawcza
```

Przykład: zastosowanie pętli do obliczania silni z liczby n

```
      silnia=1
      DO 100 i=2,n
      silnia=silnia*i
100   CONTINUE
```

W powyższym przykładzie, instrukcja podstawienia znajdująca się między DO i CONTINUE zostanie wykonana n-1 razy, przy czym zmienna I będzie przyjmować kolejno wartości 2, 3, … n.

Instrukcją wykonawczą kończącą pętle DO jest najczęściej instrukcja CONTINUE. Powoduje ona przejście do następnej linii kodu. Jeśli parametr `skok` jest pominięty to zmienna sterująca przyrasta w kolejnych cyklach o wartość 1.

Postać 2

```
DO zmienna_sterująca=wartość_początkowa,wartość_końcowa[,skok]
   ciąg instrukcji
END DO
```

Przykład: obliczanie sumy liczb parzystych z przedziału <2,n>

```
suma=1
DO  i=1,n,2
suma=suma+i
ENDDO
WRITE(*,*) 'suma liczb naturalnych parzystych =', suma
```

Instrukcja skoku

W celu zmiany domyślnej kolejności wykonywania instrukcji programu, wykorzystuje się instrukcję skoku. Służy ona do skierowania programu do linii kodu oznaczonego wskazana etykietą. Składnia instrukcji skoku jest następująca:

```
GOTO etykieta
```

Instrukcje sterujące: CONTINUE, STOP, PAUSE

1. Instrukcja CONTINUE nie powoduje wykonania żadnej operacji. Najczęściej używana jest w postaci z etykietą po lewej stronie jako instrukcja kończąca pętlę
2. Instrukcja STOP powoduje natychmiastowe zakończenie wykonywania programu. Może wystepować z komunikatem który zostanie wyświetlony na ekranie po zakończeniu działania programu.

Przykład: (zastosowanie instrukcji IF, GOTO i STOP)

```
      delta = b*b-4.0*a*c
      IF(delta.LT.0)  GOTO 12
      pd = SQRT(delta)
      x1=0.5*(-b + pd )/a
      x2=0.5*(-b – pd )/a
      WRITE(*,*) ‘Pierwiastki:       x1=’,x1,’      x2=’,x2
      STOP
12    WRITE(*,*) ‘Rownanie nie ma pierwiastkow rzeczywistych’
      STOP ‘brak pierwiastkow’
```

3. Instrukcja PAUSE powoduje chwilowe zawieszenie wykonywania programu. Może występować z komunikatem. Np.

```
PAUSE  'wykonanie programu zostalo chwilowo wstrzymane'
```

Uwaga! Aby wznowić działanie programu skompilowanego przez Force 2.0 należy wpisać z klawiatury – „go".

Instrukcje wejścia/wyjścia

Wprowadzanie danych do programu oraz wyprowadzanie wyników jego działania odbywa się przy pomocy tzw. instrukcji wejścia/wyjścia (Input/Output). Poniżej zebrano je z krótkim opisem.

CLOSE zamknięcie pliku
Składnia
`CLOSE(NR_URZĄDZENIA)`
Np. `CLOSE(3)`

OPEN otwarcie pliku
Składnia
`OPEN(NR_URZĄDZENIA,FILE=NAZWA_PLIKU,STATUS=STAN)`
Np.
`OPEN(3,FILE='DANE.DAT',STATUS='OLD')`

PRINT wypisanie na monitorze wartości wyszczególnionych na liście wyjścia
Składnia
`PRINT *,`
Np. `PRINT *, 'DELTA =',DEL`

READ czytanie z klawiatury lub z pliku wartości wyszczególnionych na liście wejścia
Składnia
`READ(*,*)` format swobodny

WRITE wypisanie na ekranie lub do pliku wartości wyszczególnionych na liście wyjścia
Składnia
`WRITE(*,*)` format swobodny
`WRITE(NR_URZĄDZENIA,ETYKIETA)` redagowanie formatowane

Np.
`WRITE(*,*) 'WYNIK OBLICZEŃ = ', Y`

Przesyłanie formatowane

Instrukcje wejścia/wyjścia umożliwiają ścisłe zdefiniowanie postaci danych. Przesyłanie formatowane zwykle stosuje się w przypadku instrukcji wyjścia, co oczywiście nie wyklucza zastosowań w połączeniu z instrukcją wejścia.

Do zdefiniowania postaci danych w instrukcjach READ i WRITE służy drugi parametr występujący w nawiasie. Może on przyjmować postać:

- * – gdy korzystamy z tzw. formatu swobodnego (zapis domyślny dla danego typu),
- etykiety instrukcji FORMAT, w której określono postać przesyłanych danych,
- stałej tekstowej zawierającej wzorzec zapisu

Składnia:

```
        WRITE(*, etykieta) lista danych oddzielonych przecinkami
etykieta FORMAT wzorzec redagowania
```

Opisy pól danych

Opisy pól danych służą do formatowania wyrażeń z odpowiedniej listy wejścia/wyjścia. Każdy opis pola danych zawiera informacje o:

- typie przesyłanej wartości,
- wielkości zajmowanego przez nią pola,
- sposobie zapisu wartości.

Typ elementu na liście wejścia/wyjścia:

A – tekst
D – liczba typu DOUBLE PRECISION, zapis wykładniczy
E – liczba typu REAL, zapis wykładniczy
F – liczba typu REAL, zapis stałoprzecinkowy
I – liczba typu INTEGER
X – odstęp poziomy (spacja)
/ – odstęp pionowy (przejście do nowej linii)

Ogólna postać zapisu dla pola F to F*w.d*, gdzie *w* i *d* są stałymi całkowitymi oznaczającymi odpowiednio długość zajmowanego pola oraz liczbę cyfr znaczących. Analogicznie zapisuje się typ D i E.

Dla liczb całkowitych określa się tylko długość zajmowanego pola stąd postać – I*w*. Gdy liczba zajmuje mniej pozycji niż „*w*”, spacje są wyprowadzane od lewej strony. Podobnie dla pól tekstowych zapis ogólny ma postać A*w*. Jeśli pominięte zostanie „*w*” w opisie pola tekstowego, to zostanie ono wyprowadzone na tylu pozycjach ile dokładnie zajmuje.

Dla wartości zespolonych (typu COMPLEX) opis pola składa się z dwóch opisów pól rzeczywistych, przy czym opisy te nie muszą być jednakowe.

Umieszczając przed opisem pola całkowitą dodatnią liczbę *n*, definiujemy tzw. opisy powtarzalne np.:

*n*I*w* *n*F*w.d* *n*Ew.d *n*Dw.d *n*Lw *n*Aw.

Jeśli n jest pominięte to domyślnie n=1.

W celu uzyskania odstępu n spacji należy zapisać – nX.

Przykład:

```
      x=2134.5678
      i=12
      WRITE(*, 800) i, x
800   FORMAT (I4,2X,F8.3)
```

Powyższy fragment kodu spowoduje wypisanie na ekran:

```
  12  1234.568
```

Po lewej stronie są dwie spacje, gdyż liczba 12 zajęła tylko 2 miejsca a specyfikator formatu przewidział na nią 4 miejsca. Następnie są dwie spacje zgodnie z formułą – 2X, po czym

wyprowadzona jest liczba rzeczywista na 8 miejscach z dokładnością do 3 miejsc po przecinku. Ostatnia cyfra wynika z zaokrąglenia. Cyfry 0 – 4 są zaokrąglane w dół, natomiast 5 – 9 w górę.

Do jednej instrukcji FORMAT może odwoływać się wiele instrukcji WRITE. Jest to szczególnie korzystne w sytuacji gdy chcemy aby wyniki były zapisane w równych kolumnach.

Przykładowe sposoby wyprowadzenia na ekran stałej tekstowej i zmiennej typu REAL

```
      x = 0.025
      WRITE (*,'(A, F8.3)') 'Wartosc zmiennej x = ', x
      WRITE (*,990) 'Wartosc zmiennej x = ', x
      WRITE (*,999) x
990   FORMAT (A, F8.3)
999   FORMAT ('Wartosc zmiennej x = ', F8.3)
```

Każda z trzech powyższych instrukcji WRITE wyświetli wynik w taki sam sposób.
W celu wyprowadzenia 4 liczb typu INTEGER oddzielonych podwójną spacją można użyć formatu:

```
      FORMAT (2X, I3, 2X, I3, 2X, I3, 2X, I3)
```

lub równoważnie

```
      FORMAT (4(2X, I3))
```

Zmienne tablicowe

W wielu przypadkach istnieje potrzeba użycia zmiennych w postaci wektorów lub macierzy. Takie zmienne określa się mianem *tablic*. Tablica jest skończonym ciągiem elementów tego samego typu. Pojedyncze elementy tablicy są zmiennymi skalarnymi określonego typu, jednak z uwagi na to, że należą do tablicy nazywamy je *zmiennymi indeksowanymi*. Uporządkowanie elementów tablicy odbywa się za pomocą indeksów. Odwołanie do pojedynczego elementu tablicy wymaga podania indeksu (lub indeksów) tego elementu. Liczba indeksów elementu tablicy jest równa liczbie wymiarów tablicy.

Delkaracja zmiennych tablicowych:

sposób 1

```
      Typ nazwa_zmiennej(wymiar)
```

Przykład:

```
      REAL a(10), b(3,5)
      INTEGER K(0:19)
```

Sposób 2: przy pomocy instrukcji DIMENSION

Przykład:

```
      REAL a,b
      INTEGER k
      DIMENSION a(10), b(3,5), k(0:19)
```

W powyższym przykładzie a jest tablicą jednowymiarową 10 elementową czyli wektorem, przy czym kolejne elementy tego wektora są liczbami typu rzeczywistego indeksowane w porządku od 1 do 10. Tablica rzeczywista b jest dwuwymiarowa o liczbie elementów 3*5=15. W tym przypadku pierwszy indeks oznacza nr wiersza a drugi, nr kolumny. Wektor typu całkowitego k jest 20 elementowy przy czym numeracja indeksów przebiega od 0 do 19.

Każdy element tablicy można traktować jako oddzielną zmienną. Odwołanie do elementu tablicy „a” dokonuje się poprzez podanie nazwy tablicy oraz numeru elementu np. a(5).

Przykład: zapamiętanie w tablicy kwadratów 10 kolejnych liczb naturalnych, począwszy od 1

```
      INTEGER i, square(20)
      DO 100 i = 1, 10
      square(i) = i**2
100   CONTINUE
```

Uwaga! W Fortranie77 nie ma możliwości dynamicznego alokowania tablic, stąd deklarowany rozmiar nie zawsze jest wykorzystany. W powyższym przykładzie nadano wartości tylko 10 pierwszym elementom wektora square. Nie oznacza to, że pozostałe jego elementy są zerami.

Fortran pozwala deklarować zmienne tablicowe maksymalnie 7 wymiarowe. Ich deklaracje i użycie są analogiczne do zmiennych dwuwymiarowych.

Funkcje i procedury

Jak wspomniano na początku rozdziału kod programu ma budowę modułową. Modułem podstawowym jest program główny jednak z uwagi na czytelność oraz na zwiększenie elastyczności programów część zadań jest wykonywana w podprogramach.

Zadaniem podprogramu jest wykonywanie wybranych fragmentów algorytmu. Może on być wywoływany z danej jednostki programowej (np. z programu głównego) wielokrotnie z różnym zestawem danych. Dzięki podprogramom, kod źródłowy staje się czytelniejszy i krótszy, gdyż unika się powtarzania zapisu tych samych operacji wielokrotnie. Poza tym język programowania jest bardziej „elastyczny", ponieważ raz napisany podprogram może być wykorzystany w różnych aplikacjach.

W Fortranie występują dwa rodzaje podprogramów: *funkcje* i *procedury*.

Funkcje

Funkcja jest podprogramem który dla określonych parametrów z którymi jest wywoływana oblicza wartość liczbową określonego typu. Można wyróżnić funkcje standardowe oraz funkcje definiowane przez programistę.

Funkcje standardowe

Przykład użycia funkcji standardowej:

```
x = COS(pi/3.0)
```

Lista często używanych matematycznych funkcji standardowych; Oznaczenia typu: D – double precision, I – integer, R – real, C – complex

Nazwa funkcji	Typ argumentu x	Typ wyniku	Opis
ABS(x)	D I R C	D I R	Wartość bezwzględna
SQRT(x)	D R C	D R C	Pierwiastek kwadratowy
SIN(x)	D R C	D R C	Sinus
COS(x)	D R C	D R C	Cosinus
TAN(x)	D R C	D R C	Tangens
ATAN(x)	D R C	D R C	Arcus tangens
EXP(x)	D R C	D R C	eksponens e^x
LOG(x)	D R C	D R C	Logarytm naturalny
LOG10(x)	D R C	D R C	Logarytm dziesiętny
MIN(x1, x2,..)	D I R	D I R	Wartość najmniejsza
MAX(x1, x2,..)	D I R	D I R	Wartość największa

W przypadku funkcji podanych w powyższej tabeli kompilator automatycznie dobiera odpowiedni typ funkcji zgodnie z typem argumentu.

Funkcje definiowane przez programistę

Postać:

```
Typ FUNCTION nazwa(lista argumentów)
deklaracje
część wykonawcza
nazwa = wyznaczona wartość   !element konieczny!
RETURN
END
```

Przykład użycia i definiowania funkcji obliczającej wyróżnik równania kwadratowego

```
X1=(-b-SQRT(delta(a,b,c)))/(2*a)
X2=(-b+SQRT(delta(a,b,c)))/(2*a)

REAL FUNCTION delta(a,b,c)
REAL a,b,c
delta=b**2-4*a*c
RETURN
END
```

Funkcja jest wywoływana poprzez podanie jej nazwy wraz z listą parametrów ujętych w nawias okrągły. Typ funkcji musi być zadeklarowany w jednostce programowej w której jest używana. Instrukcja RETURN powoduje zakończenie wykonania podprogramu i przekazanie sterowania do segmentu, z którego nastąpiło jego wywołanie.

Procedury

Omówione wyżej funkcje mogą zwracać tylko jedną wartość. Często zachodzi potrzeba wyznaczenia większej liczby wartości lub wykonania innych operacji. Do takich celów służy procedura.

Ogólna postać procedury wygląda następująco:

```
SUBROUTINE nazwa (lista argumentów)
deklaracje
część wykonawcza
RETURN
END
```

W przeciwieństwie do funkcji, procedura nie ma określonego typu i nie jest deklarowana w jednostce programowej z której ją wywołujemy.

Przykład: procedura, która zamienia wartości dwóch zmiennych całkowitych.

```
      SUBROUTINE swap (a, b)
      INTEGER a, b
C zmienna lokalna
      INTEGER tmp
```

```
tmp = a
a = b
b = tmp
RETURN
END
```

W pierwszej części procedury są zadeklarowane zmienne wejścia/wyjścia `a` i `b` oraz zmienna `tmp` o charakterze lokalnym. Nazwa zmiennej lokalnej jest wykorzystywana tylko w danej procedurze i może być użyta ponownie w innym znaczeniu w innej jednostce programowej np. w programie głównym.

Wywołanie procedury odbywa sie przy pomocy instrukcji `CALL`.

Przykład wywołania procedury `swap` z programu głównego:

```
PROGRAM zamiana
INTEGER m, n
m = 1
n = 2
CALL swap(m, n)
WRITE(*,*) m, n
STOP
END
```

Jako wynik zostanie wyświetlona para liczb "2 1". Wartości przekazywane są poprzez parametry między procedurą a jednostką programową z której jest wywoływana. Konieczne jest zachowanie odpowiedniej kolejności parametrów w procedurze i przy jej wywołaniu, gdyż to kolejność decyduje jakie wartości się pod nimi kryją. W powyższym przykładzie zmienne `m`, `n` pełnią rolę zmiennych wejściowych i jednocześnie wyjściowych z procedury.

Na liście parametrów mogą znajdować się zmienne, stałe, tablice lub elementy tablic.

Literatura uzupełniająca oraz źródła internetowe

- Anna Trykosko, Ćwiczenia z języka Fortran, MIKOM, Warszawa 1999.
- Janusz R. Piechna, Programowanie w języku Fortran 90 i 95, OWPW, Warszawa 2000.
- http://www-classes.usc.edu/engr/ce/108/text/fbkindex.htm
- http://www.livephysics.com/computational-physics/fortran/
- http://en.wikibooks.org/wiki/Fortran
- http://www.obliquity.com/computer/fortran/
- http://folk.uio.no/steikr/doc/f77/tutorial/
- http://www.fortran.com/F77_std//rjcnf0001.html