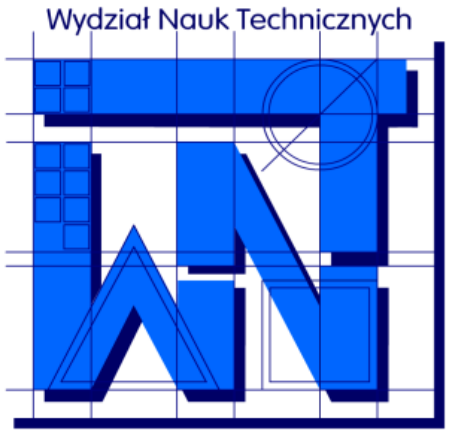


UNIVERSITY OF WARMIA AND MAZURY IN OLSZTYN
The Faculty of Technical Sciences
POLAND, 10-957 Olsztyn, M. Oczapowskiego 11
tel.: (48)(89) 5-23-32-40, fax: (48)(89) 5-23-32-55
URL: http://www.uwm.edu.pl/edu/sobieski/ (in Polish)

Języki Programowania

Instrukcje języków programowania

Wojciech Sobieski

Olsztyn, 2001-2021

Instrukcje

Instrukcja – w programowaniu jest to najmniejszy samodzielny element imperatywnego języka programowania.

Instrukcja niskiego poziomu – instrukcja napisana w asemblerze (np. mov ax, bx), która po przetłumaczeniu na kod maszynowy nadaje się do uruchomienia przez procesor.

Instrukcja wysokiego poziomu – instrukcja napisana w języku wysokiego poziomu (np. if (a==b) goto 10), która jest zazwyczaj tłumaczona na kilka instrukcji niskiego poziomu.

Program jest tworzony jako zbiór różnych instrukcji.

Instrukcja może zawierać wewnętrzne komponenty (np. wyrażenia).

Instrukcje

Podział instrukcji:

- **instrukcje czynne** – instrukcje związane bezpośrednio z tokiem obliczeń i wpływające na wartości poszczególnych zmiennych

- **instrukcje bierne** – instrukcje pomocnicze, służące do organizacji kodu źródłowego i określenia właściwości poszczególnych obiektów.

Czasami instrukcje służące do deklaracji typów zmiennych oraz segmentów nazywa się **deklaratorami**.

Instrukcje

Instrukcje czynne:

- instrukcja przypisania
- instrukcje sterujące:
  skoku, zatrzymania, wstrzymania, końca, powrotu, warunkowa, wyboru, powtórzeń (pętli), wejścia-wyjścia
- instrukcje obsługi wyjątków

Instrukcje bierne:

- specyfikacje segmentów i ich wejść
- specyfikacje cech obiektów
- specyfikacje formatu danych
- instrukcje funkcji i procedur
- instrukcje inicjowania danych

Instrukcje

```
program p_12_14

include 'p_12_14_types.inc'

oblicz(F,l,A,E) = (F*l)/(A*E)

include 'p_12_14_values.inc'

dl = oblicz(F,l,A,E)
print '(A,F6.2,A)',  'Pret wydluzy sie o : ',dl*1000,' [mm]'

read(*,*)
end
```

obszary zawierające instrukcje czynne

Instrukcje

```
program p_12_05

real    x,y,z

x = 1.
y = 1.

write(*,*) oblicz(x,y)

read(*,*)
end

real function oblicz(x,y)
oblicz = x+y
end
```

obszary zawierające instrukcje czynne

Instrukcje

```
program p_08_10

character      key

do while(.TRUE.)
   print '(A$)', 'Wcisnij klawisz [Ctrl+C - koniec]: '
   read(*,*) key
   print *, 'Numer ASCII = ',ichar(key)
end do

read(*,*)
end
```

obszary zawierające instrukcje czynne

Instrukcje czynne

Instrukcja przypisania (podstawienia) – instrukcja pozwalająca nadać zmiennej określoną wartość.

Instrukcja przypisania ma postać:

zmienna = *wyrażenie*

gdzie *zmienna* oznacza zmienną prostą lub element tablicy, zaś *wyrażenie* jest dowolną wartością liczbową, logiczną lub tekstową (w zależności od typu zmiennej) lub odpowiednio wyrażeniem algebraicznym logicznym lub tekstowym.

Instrukcje czynne

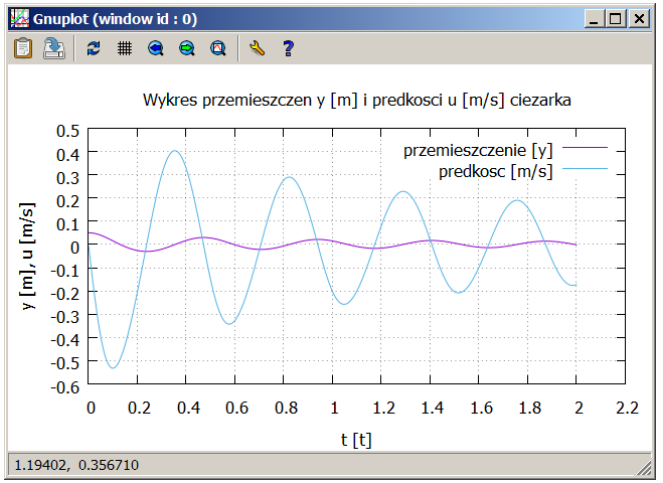
```
!rozpoczecie petli po czasie:
do i=1,n

!wybor stalej tlumienia
 !w zaleznosci od kierunku ruchu:
  if (u.gt.0) then
    c=cd
  else
    c=cg
  end if

!obliczenie pochodnych:
dydt=u
dudt=-g-(k/m)*(y-w0)-c*u*abs(u)/m

!obliczenie nowych wartosci zmiennych:
y=y+dydt*dt
u=u+dudt*dt
t=t+dt

!zakonczenie petli po czasie:
end do
```



obliczanie nowych wartości zmiennej na podstawie wartości starej oraz jej przyrostu (fragment programu Drgania)

Instrukcje czynne

Instrukcja skoku – instrukcja służąca do przeniesienia punktu sterowania w inne, wcześniej zdefiniowane, miejsce programu. Instrukcja skoku nie jest zalecana przez teoretyków programowania, jednak często znacznie ułatwia realizację algorytmu. Instrukcja skoku ma ogólną postać logiczną:

IDŹ DO *nazwa_miejsca*

gdzie *nazwa_miejsca* musi być już jednoznacznie zdeklarowana: może to być etykieta liczbowa (Fortran, Basic) lub tekstowa („label”) (Pascal).

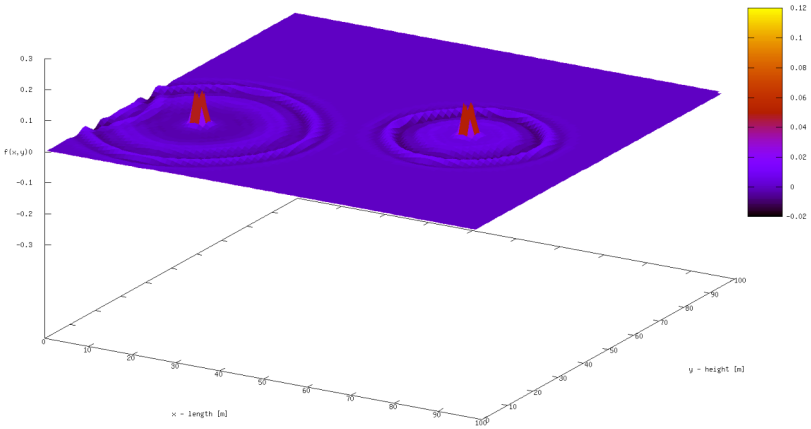
Fortran:

GO TO *etk.*

Instrukcje czynne

```
      !odczyt danych z pliku:
      open(1,file='fala2d.dat')
      read(1,*,err=20) nx
      read(1,*,err=20) ny
      read(1,*,err=20) c
      read(1,*,err=20) lx
      read(1,*,err=20) ly
      read(1,*,err=20) dt
      read(1,*,err=20) eps
      read(1,*,err=20) min
      read(1,*,err=20) max
      read(1,*,err=20) zmax
      close(1)

      goto 30
20    print '(A$)', 'Blad odczytu danych '
30    continue
```



przykład zastosowania instrukcji **GOTO** do obsługi błędów operacji wejścia (fragment programu Fale2D)

Instrukcje czynne

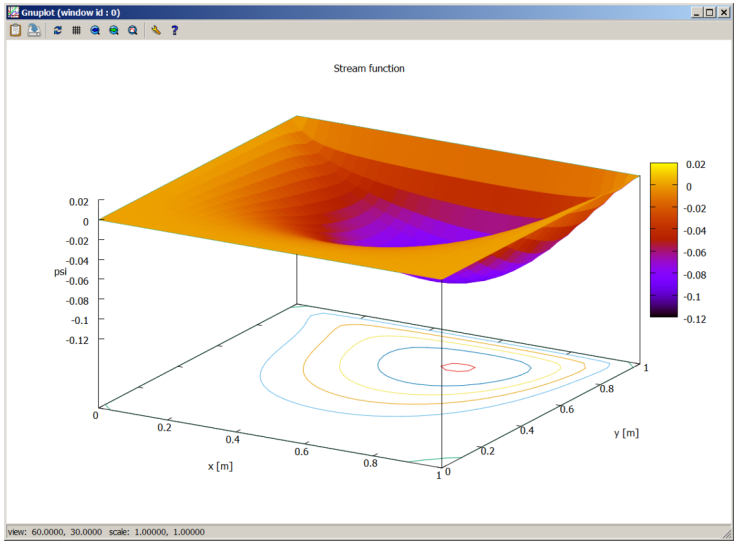
Instrukcja zatrzymania – instrukcja służąca do bezwarunkowego zakończenia wykonywanego programu. Instrukcja często spotykana w blokach podejmowania decyzji, uruchamiana gdy nie są spełnione warunki do wykonywania dalszych działań.

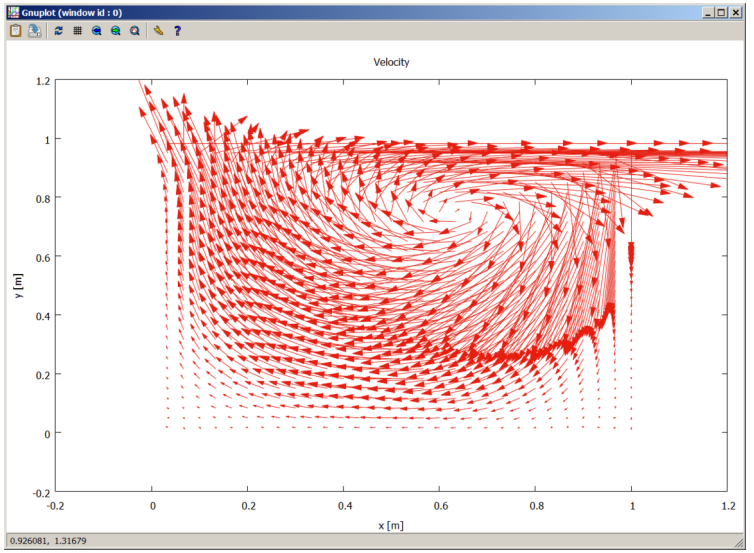
Fortran:

STOP ['*zmienna_tekstowa*']

Instrukcje czynne

```
!sprawdzenie poprawnosci czasu koncowego:
if (trans) then
  if (tfinal.lt.dt) then
    write(*,'(A$)') 'Koncowy czas nie moze byc mniejszy od
                     kroku czasowego...'
    stop
  endif
endif
```





przykład zastosowania instrukcji **STOP** do wyjścia z programu przy błędnie zdefiniowanym kroku czasowym (fragment programu Mac)

Instrukcje czynne

Instrukcja wstrzymania – instrukcja służąca do chwilowego zatrzymania wykonywania programu. Kontynuacja może być podjęta wskutek działania użytkownika bądź też po upływie określonego czasu.

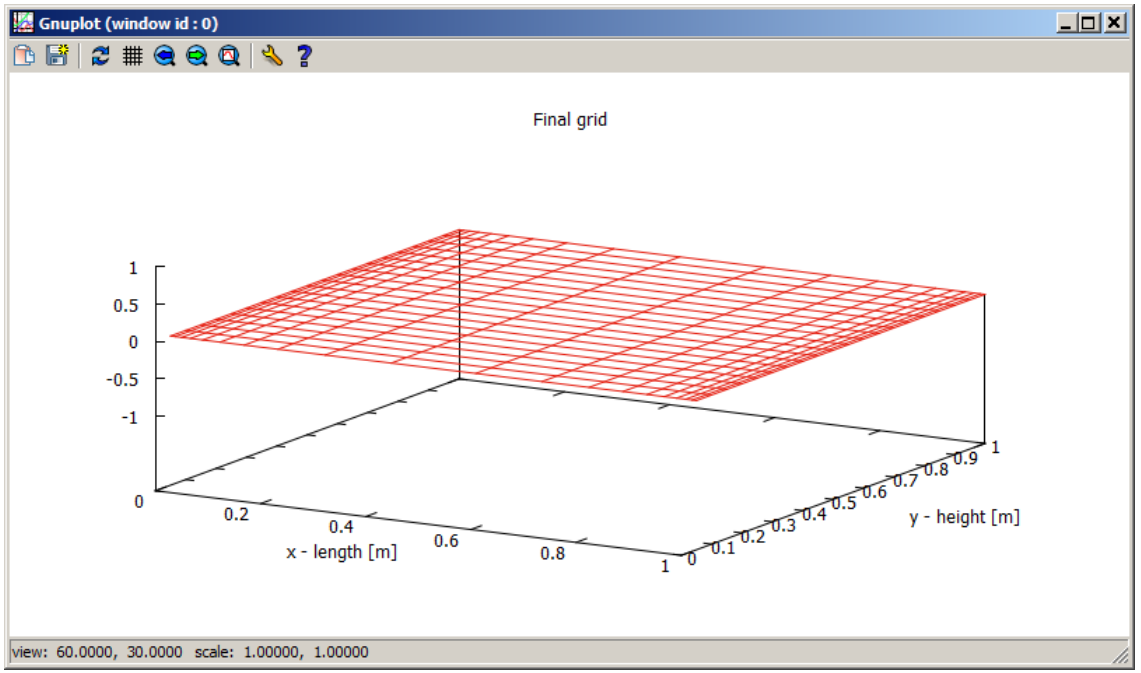
Fortran:

PAUSE ['*zmienna_tekstowa*']

Instrukcje czynne

```
set terminal wxt size 800, 400
set title 'Final grid'
set xlabel 'x - length [m]'
set ylabel 'y - height [m]'
set nokey
splot 'siatka_wezly.txt' notitle with lines lt 7
pause -1 'Enter'
exit gnuplot
```

przykład zastosowania instrukcji **PAUSE** do chwilowego zatrzymania wykonywania skryptu Gnuplota (fragment programu GridMaker)



Instrukcje czynne

Instrukcja powrotu – instrukcja służąca do wyjścia z podprogramu, procedury, funkcji lub też do wskazania ponownego wykonania pętli.

Fortran:

RETURN [*n*]

Instrukcje czynne

Instrukcja końca – instrukcja służąca do określenia końca programu, procedury, funkcji, bloku deklaracji, bloku instrukcji bądź bloku inicjacji danych.

Fortran:

END

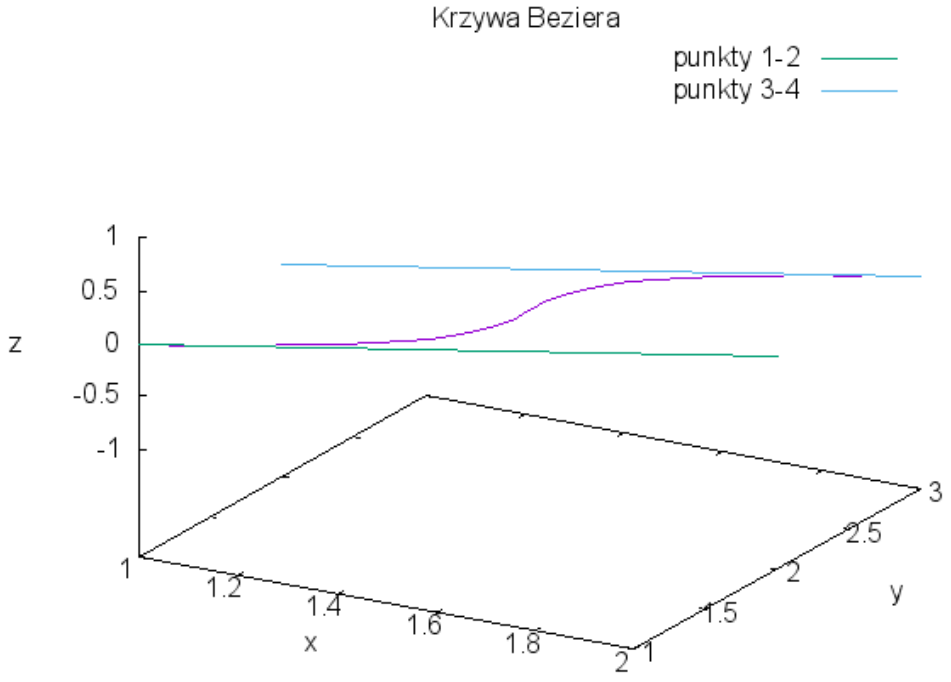
END IF

END DO

END SELECT

Instrukcje czynne

```
!zapis wspolrzednych krzywej:
open(1,file='bezier_xyz.txt')
do i = 1, n
  write(1,'(3F12.6)') x(i), y(i), z(i)
end do
close(1)
```



przykład zastosowania instrukcji **END DO** do określenia końca pętli (fragment programu Bezier)

Instrukcje czynne

Instrukcja warunkowa – instrukcja służąca do podejmowania decyzji w zależności od postawionych warunków:

JEŻELI *warunek* TO *instrukcja*

lub

JEŻELI *warunek_1* TO *instrukcja_1*
W INNYCH PRZYPADKACH *instrukcja_2*

gdzie *warunek* jest dowolnym wyrażeniem arytmetycznym, relacją lub wyrażeniem logicznym, a *instrukcja* dowolnym blokiem poleceń, wykonywanym gdy warunek jest prawdziwy.

Instrukcje czynne

Często zachodzi konieczność rozważenia większej liczby możliwości, wówczas stosuje się inną konstrukcję logiczną:

JEŻELI *warunek_1* TO *instrukcja_1*
JEŻELI ZAŚ *warunek_2* TO *instrukcja_2*
JEŻELI ZAŚ *warunek_3* TO *instrukcja_3*
JEŻELI ZAŚ *warunek_4* TO *instrukcja_4*
……
W INNYCH PRZYPADKACH *instrukcja_N*

Zależnie od języka programowania budowa instrukcji warunkowych może być nieco inna, mogą też występować odmiany instrukcji. Tworząc instrukcję warunkową należy uwzględnić wszystkie możliwe przypadki – zaniedbanie może spowodować błędne działanie programu lub też jego zawieszenie.

Instrukcje czynne

Fortran:

IF (wyrażenie) instrukcja

} **IF** logiczne

IF *(zmienna) etk._1, etk._2, etk._3*

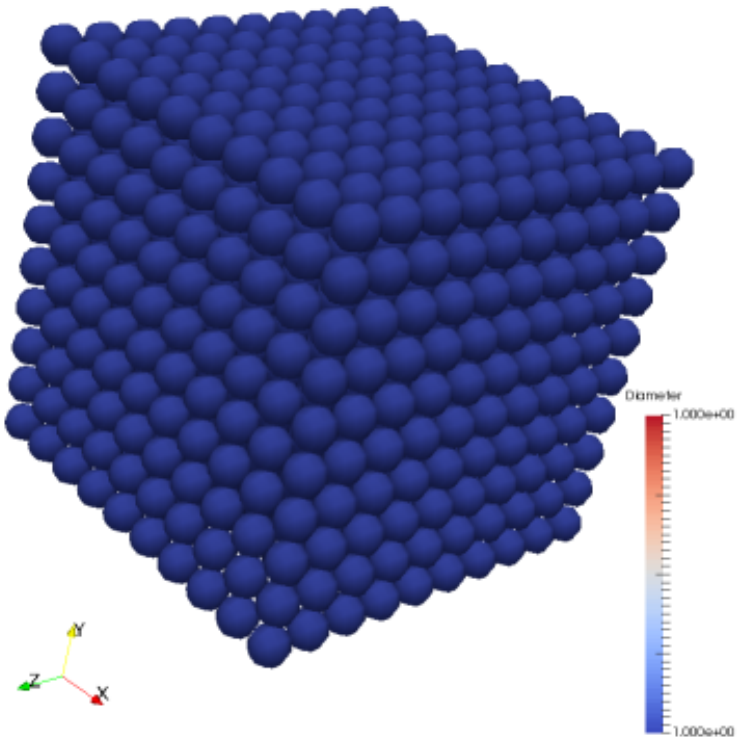
} **IF** arytmetyczne

```
IF (wyrażenie_1) THEN
        {...}
ELSE IF (wyrażenie_2) THEN
        {...}
ELSE
        {...}
END IF
```

} **IF** blokowe

Instrukcje czynne

```
if (2*int(i/2) == i) then !
  x = x_min-d/2.+i*d + x_imp
else
  x = x_min-d/2.+i*d - x_imp
end if
if (2*int(j/2) == j) then !
  y = y_min-d/2.+j*d + y_imp
else
  y = y_min-d/2.+j*d - y_imp
end if
if (2*int(j/2) == j) then !
  z = z_min-d/2.+k*d + z_imp
else
  z = z_min-d/2.+k*d - z_imp
end if
```



przykład zastosowania instrukcji **IF (**blokowego) do obliczania współrzędnych środków sfer w regularnym złożu cząstek (fragment programu Regular bed – dodatku do pakietu PathFinder)

Instrukcje czynne

Instrukcja wyboru – instrukcja służąca do podejmowania decyzji w zależności od postawionych warunków:

WYBIERZ ZALEŻNIE OD WARTOŚCI *zmienna*
JEŻELI (*lista_przypadków_1*) TO *instrukcja_1*
JEŻELI (*lista_przypadków_2*) TO *instrukcja_2*
...
DOMYŚLNIE *instrukcja_N*

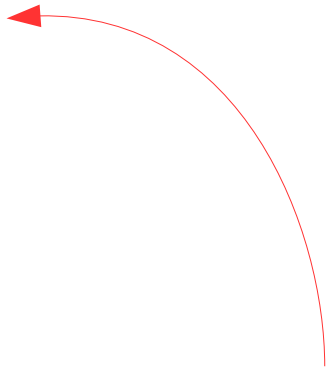
gdzie *zmienna* jest identyfikatorem zmiennej, zaś *lista_przypadków* zawiera oczekiwane wartości, dla których mają być wykonane odpowiednie instrukcje.

Instrukcja wyboru może być realizowana tylko dla zmiennej całkowitej lub logicznej.

Instrukcje czynne

Fortran:

SELECT CASE *(zmienna)*
CASE (*lista_przypadków_1*)
{...}
CASE (*lista_przypadków_2*)
{...}
...
{...}
CASE DEFAULT
{...}
END SELECT

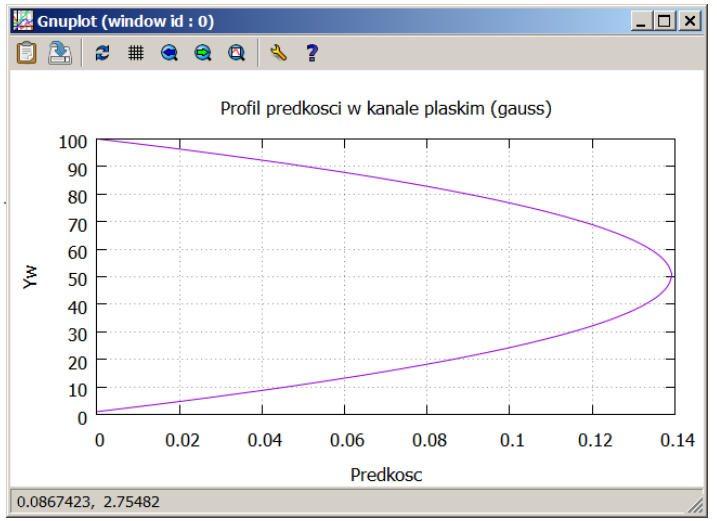


warianty lista przypadków:

CASE(1)
CASE(1,2,7,8)
CASE(1,2,5:8)

Instrukcje czynne

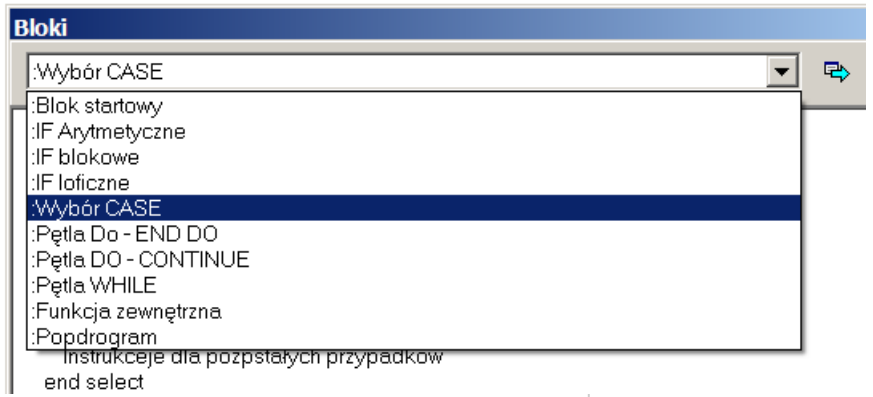
```
!zapis wartosci predkosci w kolejnych  wezlach do pliku:
select case (metoda)
case(1)
  open(1,file='profil-gauss-jordan.txt')
case(2)
  open(1,file='profil-gauss.txt')
case(3)
  open(1,file='profil-lu.txt')
case(4)
  open(1,file='profil-jacobi.txt')
case(5)
  open(1,file='profil-gauss-seidel.
case(6)
  open(1,file='profil-sor.txt')
end select
do i=1, nkom
  write(1,*) v(i),i
end do
close(1)
```

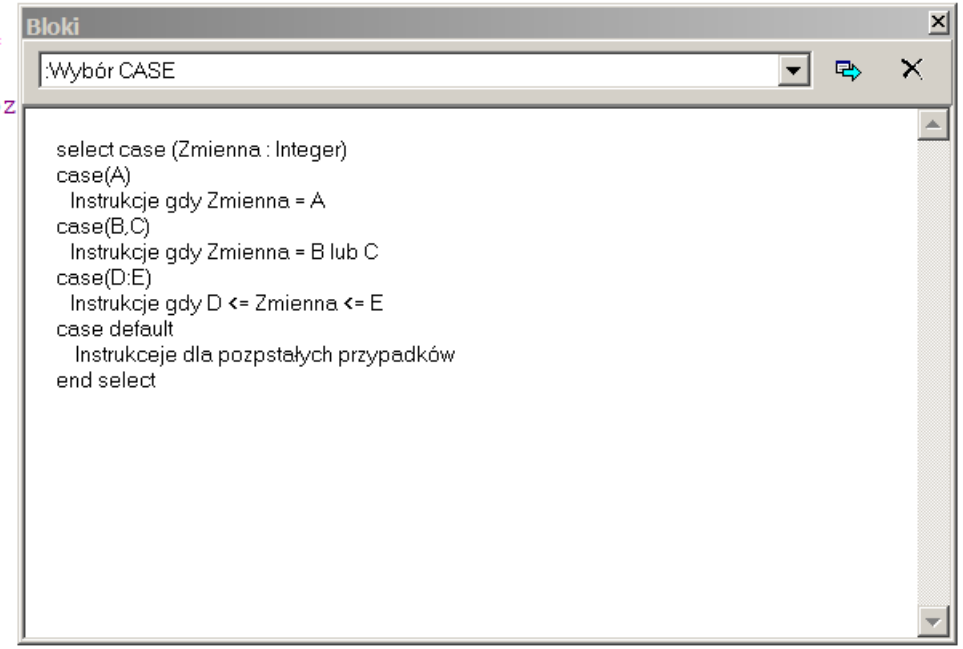


przykład zastosowania konstrukcji **SELECT CASE** do określenia nazwy pliku (fragment programu Profil)

Instrukcje czynne

```
select case (Zmienna : Integer)
case(A)
   Instrukcje gdy Zmienna = A
case(B,C)
   Instrukcje gdy Zmienna = B lub C
case(D:E)
   Instrukcje gdy D <=
case default
    Instrukceje dla poz
end select
```





w środowisku Edi (**Ctrl+B**) dostępne jest narzędzie wspomagające korzystanie z niektórych konstrukcji – tu: schemat konstrukcji **SELECT CASE** (po wciśnięciu klawisza ze strzałką, schemat zostanie wstawiony do kodu źródłowego)

Instrukcje czynne

Instrukcje powtórzeń (pętle) – instrukcje służące do wielokrotnego wykonania tego samego fragmentu programu.

Rozróżnia się dwa podstawowe typy instrukcji powtórzeń:

- o znanej z góry liczbie powtórzeń:
  - realizowane "od dołu"
  - realizowane "od góry"

- o nieznanej z góry liczbie powtórzeń:
  - z warunkiem "na początku"
  - z warunkiem "na końcu"

Instrukcje czynne

Ogólna postać logiczna pętli o znanej liczbie powtórzeń wygląda następująco:

POWTARZAJ OD *wartość_1* DO *wartość_2*
{
instrukcje
}

gdzie *wartość_1* i *wartość_2* są liczbami typu całkowitego, a *instrukcje* dowolnym zestawem poleceń. Zestaw poleceń musi posiadać wskazanie początku i końca bloku. Czasami możliwe jest również określenie kroku przyrostu – licznik pętli nie musi bowiem wzrastać lub maleć zawsze o 1.

Instrukcje czynne

Fortran:

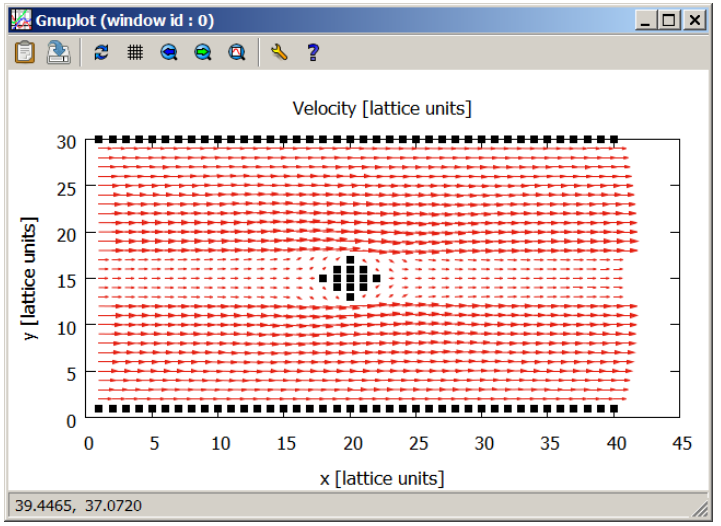
DO *licznik* = początek, koniec, krok
{...}
END DO

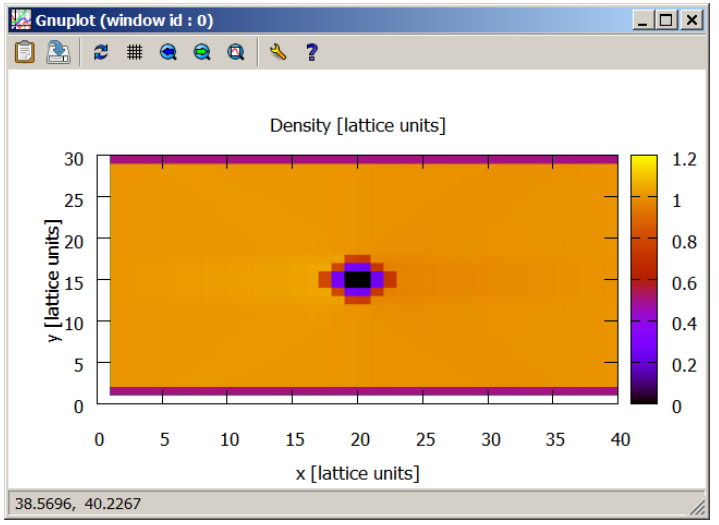
DO *etk.* *licznik* = początek, koniec, krok
{...}
etk. **CONTINUE**

DO *etk.* *licznik* = początek, koniec, krok
etk. {...}

Instrukcje czynne

```
! Poczatkowa gestosc gazu sieciowego:
rho=1.
do j=1,ly
  do i=1,lx
    f(j,i,1) = (4./9. )*rho(j,i)
    f(j,i,2) = (1./9. )*rho(j,i)
    f(j,i,3) = (1./9. )*rho(j,i)
    f(j,i,4) = (1./9. )*rho(j,i)
    f(j,i,5) = (1./9. )*rho(j,i)
    f(j,i,6) = (1./36.)*rho(j,i)
    f(j,i,7) = (1./36.)*rho(j,i)
    f(j,i,8) = (1./36.)*rho(j,i)
    f(j,i,9) = (1./36.)*rho(j,i)
  end do
end do
```





przykład zastosowania podwójnej pętli **DO**
do zadania początkowej gęstości gazu
sieciowego w poszczególnych kierunkach sieci
(fragment programu LBM)

Instrukcje czynne

Ogólna postać logiczna pętli o nieznanej liczbie powtórzeń z warunkiem na początku wygląda następująco (tzw. warunek WHILE w PASCALU):

JEŻELI *warunek*
= *prawda* WYKONAJ *instrukcje* I POWRÓĆ
= *fałsz* WYJDŹ Z PĘTLI

Należy podkreślić, że w przypadku, gdy warunek jest od razu fałszywy blok *instrukcje* nie zostanie wykonany ani razu.

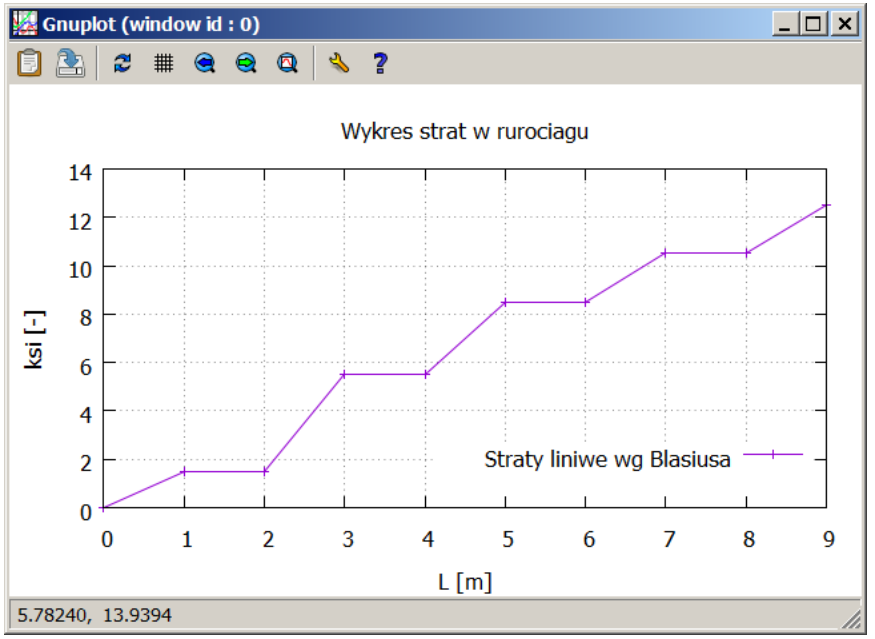
Fortran:

DO WHILE (warunek)
{...}
END DO

Instrukcje czynne

```
      open(4,file=trim(zadanie),err=60)
      blad = 0
      lp=0
      do while(blad == 0)
        read(4,*,err=60,iostat=blad) te
        lp = lp+1
      end do
60    close(4)
```

przykład zastosowania pętli **DO WHILE** do sprawdzenia liczby wierszy w pliku konfiguracyjnym (fragment programu Straty)



Instrukcje czynne

Ogólna postać logiczna pętli o nieznanej liczbie powtórzeń z warunkiem na końcu wygląda następująco (tzw. warunek REPEAT-UNTIL w PASCALU):

WYKONAJ *instrukcje* I JEŻELI warunek
= *prawda* WYJDŹ Z PĘTLI
= *fałsz* POWRÓĆ

W tym przypadku blok *instrukcje* wykonany zostanie przynajmniej raz.

Fortran:

```
etk     CONTINUE
        {...}
        IF (warunek) GOTO etk.
```

Instrukcje czynne

Instrukcje wejścia-wyjścia (I-O) – instrukcje służące do przesyłania danych pomiędzy różnymi elementami komputera, takimi jak monitor (domyślne wyjście), klawiatura (domyślne wejście), drukarka czy plik.

Rekord – elementarna porcja informacji, jaka może być przesyłana pomiędzy urządzeniami zewnętrznymi a pamięcią operacyjną. W przypadku monitora i klawiatury pojedynczy rekord odpowiada wierszowi znaków na ekranie monitora, w przypadku drukarki wierszowi poleceń.

Lista wejścia i wyjścia – lista określająca liczbę, typy i kolejność zmiennych przekazywanych w ramach operacji wejścia/wyjścia.

Instrukcje czynne

Fortran:

PRINT *, x, y, z

instrukcja służąca tylko do wyprowadzania danych na domyślne urządzenie wyjścia (monitor)

WRITE(*,*) x, y, z

instrukcja służąca tylko do **wyprowadzania** danych na **dowolne** urządzenie wyjścia

READ(*,*) x, y, z

instrukcja służąca tylko do **wprowadzania** danych z **dowolnego** urządzenia wejścia

druga gwiazdka oznacza wzorzec formatowania (dokładnie tak samo, jak w instrukcji **PRINT**)

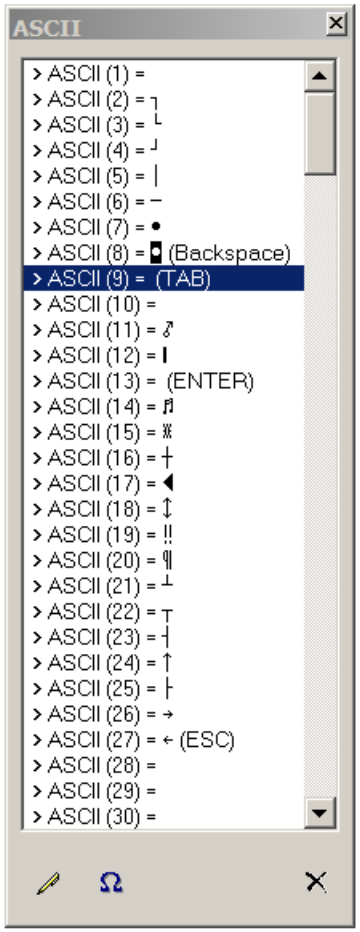
pierwsza gwiazdka oznacza numer urządzenia I/O (definiowany instrukcją **OPEN**)

Instrukcje czynne

w tym przypadku dwie zmienne zostaną rozdzielone znakiem tabulatora

```
write(1,*) x, Char(9), y
```

w pakiecie Edi dostępne jest narzędzie wspomagające korzystanie ze znaków ASCII - za pomocą dodatkowych klawiszy, do kodu źródłowego można wstawić wybrany znak ASCII bezpośrednio lub za pomocą jego numeru z tablicy znaków ASCII



Instrukcje czynne

Instrukcje obsługi wyjątków – instrukcje służące do sterowaniem tokiem obliczeń w przypadku zaistnienia błędu.

Ogólna struktura logiczna instrukcji obsługi wyjątków jest następująca

WYKONAJ
    instrukcje
A JEŚLI SIĘ COŚ NIE UDA
    instrukcje
KONIEC

Instrukcje czynne

lub

WYKONAJ
instrukcje
A JEŚLI SIĘ COŚ NIE UDA
GDY *błąd_1* TO *instrukcja_1*
GDY *błąd_2* TO *instrukcja_2*
GDY *błąd_3* TO *instrukcja_3*
...
KONIEC

Zależnie od języka programowania i jego implementacji obsługa błędów może dotyczyć dowolnej instrukcji w programie lub też wybranego zbioru instrukcji (np. tylko instrukcje wejścia-wyjścia). Pozostałe fragmenty programu muszą być wówczas zabezpieczane inaczej.

Instrukcje czynne

Object Pascal (Lazarus):

```
procedure TFG.WczytajDane;
begin
try
  AssignFile(Plik,'tmp.txt');
  Reset(Plik);
  ReadLn(Plik,x,y);
finally
  CloseFile(Plik);
  DeleteFile('tmp.txt');
end;
end;
```

przykład zastosowania konstrukcji **try-fianally-end** do zamknięcia pliku w każdym przypadku – jeżeli plik pozostanie otwarty, to system operacyjny zablokuje do niego dostęp

inną możliwą konstrukcją jest **try-except-end**

Instrukcje bierne

Specyfikacje segmentów i ich wejść – instrukcje służące do wyodrębniania określonych bloków programu stanowiących pewną logiczną całość.

Segmentem może by:

- **procedura**
- **funkcja**
- moduł
- blok inicjalizacji danych
- inne.

Segmenty mogą być wywoływane w głównym bloku programu lub też w innych segmentach (co zależy często od odpowiedniej deklaracji).

Instrukcje bierne

Procedura – samodzielny fragment programu, posiadający zazwyczaj budowę podobną do segmentu głównego: zawiera oznaczenie początku i końca, obszar deklaracji typów, bloki instrukcji, itd. Procedura może być wywoływana w segmencie głównym lub w innych procedurach. W całym programie może być zazwyczaj tylko jedna procedura o określonej nazwie, (ilość procedur jest nieograniczona).

Instrukcje bierne

Ogólna zasada logiczna stosowania procedur:

POCZĄTEK BLOKU s*egment*
instrukcje
WYWOŁAJ PROCEDURĘ *nazwa*
instrukcje
KONIEC BLOKU s*egment*

PROCEDURA *nazwa*
instrukcje
KONIEC PROCEDURY *nazwa*

gdzie *segment* oznacza główny segment programu lub inną procedurę, *instrukcje* – dowolny zestaw poleceń i instrukcji, *nazwa* – identyfikator procedury.

Instrukcje bierne

Fortran:

{...}
CALL *nazwa*(lista_parametrów_wywołania)
{...}

SUBROUTINE *nazwa* (lista_parametrów_wywołania)
{...}
RETURN
{...}
END

Instrukcje bierne

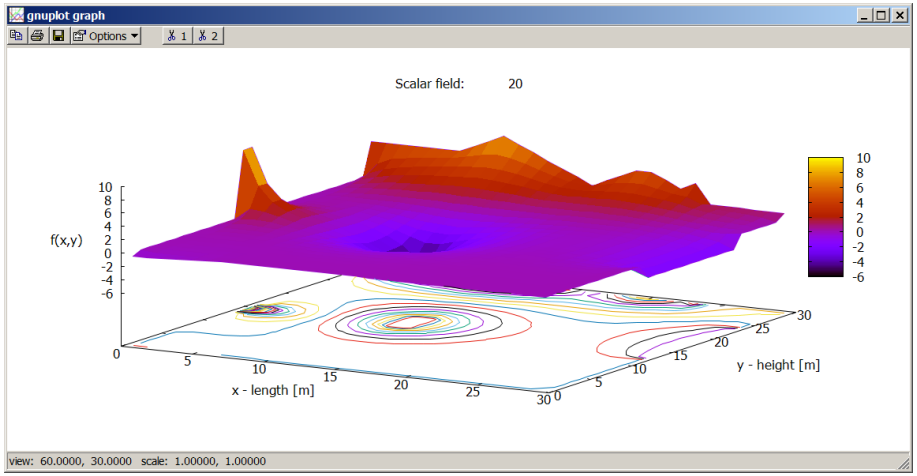
```
 ...
call save(gc_x,gc_y,s)
 ...

subroutine save(gc_x,gc_y,s)

implicit none
include 'poisson.inc'

!zapis pola skalarnego do pliku:
open(1,file='poisson.txt')
do i=1,nx-1
  do j=1,ny-1
     write(1,'(3F16.6)') gc_x(i,j), gc_y(i,j), s(i,j)
     if (j.eq.ny-1) write(1,'(a1)') ' '
 end do
end do
close(1)

end subroutine save
```



przykład zastosowania **procedury**
do zapisywania do pliku bieżących wyników
obliczeń (fragment programu Poisson)

Instrukcje bierne

W przypadku procedur sterowanych zdarzeniami w systemie, ogólna postać logiczna składni jest następująca:

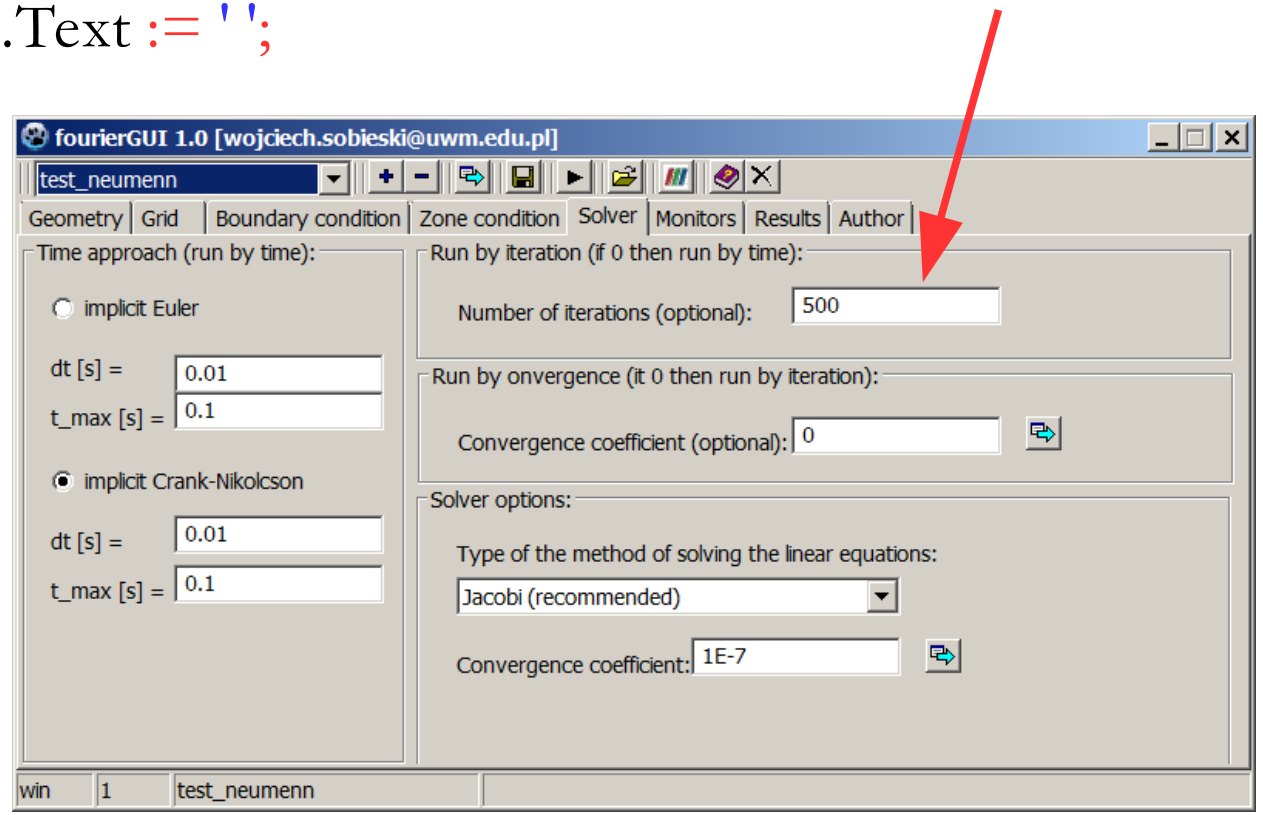
PROCEDURA *nazwa* JEŻELI *zdarzenie*
…
instrukcje
…
KONIEC PROCEDURY *nazwa*

Zdarzenie – dowolne działanie wykonane przez program lub użytkownika (zdarzeniem może być np. kliknięcie myszką, naciśnięcie klawisza, wybór elementu menu, najechanie kursorem na element formularza, aktywacja lub zamknięcie formularza, itd.).

Instrukcje bierne

```
procedure TFG.EditNumberOfIterationsChange(Sender: TObject);
begin
  try
    n_iter := StrToInt(EditNumberOfIterations.Text);
  except
    EditNumberOfIterations.Text := ' ';
    n_iter := 0;
  end;
end;
```

przykład **procedury** obsługi zdarzenia, tu: zmiany zawartości okna liczby iteracji (fragment programu FourierGUI)



Instrukcje bierne

Funkcja – samodzielny fragment programu, posiadający zazwyczaj budowę podobną do segmentu głównego: zawiera oznaczenie początku i końca, obszar deklaracji typów, bloki instrukcji, itd. W odróżnieniu od procedury, funkcja zawsze zwraca jakiś wynik. Funkcja wymaga zazwyczaj przekazania listy zmiennych o określonych typach. Ogólna postać logiczna funkcji jest następująca:

FUNKCJA *nazwa* (*lista_zmiennych*)
instrukcje
KONIEC FUNKCJI *nazwa*

gdzie *nazwa* jest identyfikatorem funkcji, *instrukcje* – dowolnym zestawem poleceń i instrukcji, zaś *lista_zmiennych* – uporządkowanym zbiorem nazw zmiennych, niezbędnych do obliczenia funkcji. W funkcjach mogą być wykorzystane zmienne dowolnych typów.

Instrukcje bierne

Fortran:

{…}
wynik = *nazwa*(lista_parametrów_wywołania)
{…}

TYP **FUNCTION** *nazwa_funkcji* (lista_argumentów_wywołania)
nazwa_funkcji = wyrażenie_arytmetyczne
END

Instrukcje bierne

```
program Hook
include 'variables.inc'

oblicz(F,l,A,E)=(F*l)/(A*E)

l=10.
dl=0.
F=100.
A=2.
A=A*1e-6
E=2.2e11

dl=oblicz(F,l,A,E)
print '(A,F8.5,A)',   ' Pret wydluzy
sie o    : ',dl*1000,' [mm]'

read(*,*)
end
```

przykład **funkcji lokalnej** oraz
funkcji zewnętrznej (program Hook)

```
program Hook
include 'variables.inc'

l=10.
dl=0.
F=100.
A=2.
A=A*1e-6
E=2.2e11

dl=oblicz(F,l,A,E)
print '(A,F8.5,A)',   ' Pret wydluzy
sie o    : ',dl*1000,' [mm]'

read(*,*)
end

real function oblicz(F,l,A,E)
include 'variables.inc'
oblicz=(F*l)/(A*E)
end
```

Instrukcje bierne

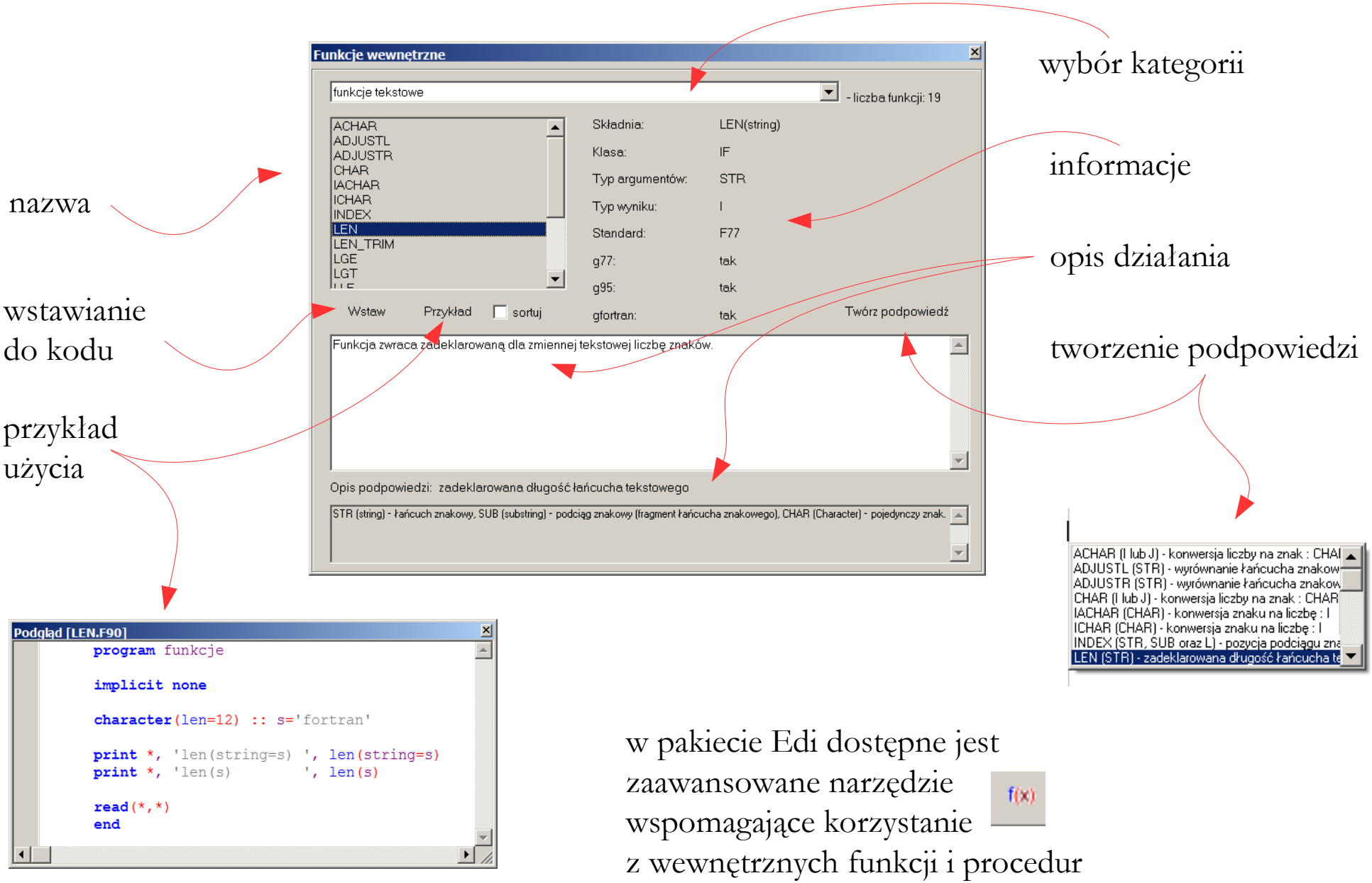
Cechy **procedury**:

- nie może być wywoływana w wyrażeniu
- może, ale nie musi pobierać argumentów
- nie musi zwracać wyniku (jeżeli zwraca to może to być wiele wartości zmiennych o różnych typach)

Cechy **funkcji**:

- jest wywoływana w wyrażeniu
- zazwyczaj musi być wywoływana z listą argumentów
- musi zawsze zwracać wynik (w postaci zmiennej o określonym typie)

Instrukcje bierne



w pakiecie Edi dostępne jest zaawansowane narzędzie wspomagające korzystanie z wewnętrznych funkcji i procedur

Instrukcje bierne

Specyfikacje cech obiektów – instrukcje służące do określenia pewnych właściwości zmiennych, procedur lub funkcji. Zazwyczaj jest to rodzaj definiowanego obiektu (zmienna czy stała), rozmiar zmiennej indeksowanej, obszar obowiązywania (obiekt "publiczny" czy "lokalny"), zasady organizacji pamięci, itd. Zależnie od języka programowania, jego możliwości w zakresie definicji cech obiektów mogą być bardzo zróżnicowane.

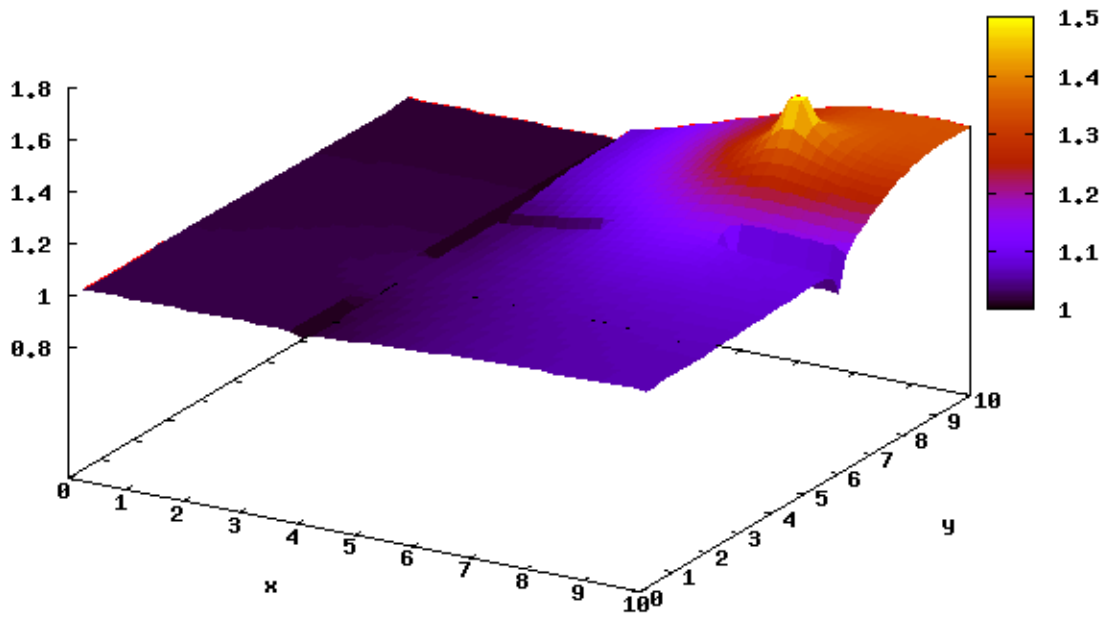
Fortran:

REAL, **ALLOCATABLE** :: x(:)

ELEMENTAL TYP **FUNCTION** *nazwa_funkcji* (lista_arg._wywołania)
nazwa_funkcji = wyrażenie_arytmetyczne
END

Instrukcje bierne

```
real(kind=8),allocatable :: f(:,:)   !funkcja obliczana
real(kind=8),allocatable :: s(:,:)   !zrodlo wartosci funkcji
real(kind=8),allocatable :: d(:,:)   !wspolczynnik dyfuzji
```



przykład zastosowania instrukcji biernej **ALLOCATABLE**
do zadeklarowania tablic dynamicznych (fragment programu Dyfuzja 2D)

Instrukcje bierne

Specyfikacje formatu danych – instrukcje służące do określania wyglądu zmiennych podczas wyprowadzania lub wprowadzania danych.

Fortran:

etk. **FORMAT**(wzorzec_formatowania)

Procesowi formatowania podlegają najczęściej następujące elementy:

Instrukcje bierne

- **całkowita liczba zajmowanych znaków**. W przypadku gdy liczba znaków jest krótsza od zadeklarowanej, pozostałe znaki uzupełniane są spacjami (z przodu lub na końcu). W przypadku zaś, gdy liczba znaków jest dłuższa, znaki ostatnie są obcinane lub też zgłaszany jest błąd wejścia-wyjścia. Spacja, znak wartości liczby (plus-minus) czy separator dziesiętny zajmują jeden znak.

				-	1	2	6	8	8	.	7	7	1	2	5
														2	2
P	O	N	I	E	D	Z	I	A	Ł	E	K				
1	2	0	0	0	4	7	6	.	1	6	7	7	7	7	7

- **znak separatora dziesiętnego** (dla liczb rzeczywistych) – przecinek lub kropka. Znak separatora może być definiowany w języku programowania lub też może być zmieniany poprzez zastosowanie własnoręcznie napisanej funkcji.

Instrukcje bierne

- **liczba cyfr po przecinku** (dla liczb rzeczywistych). Należy pamiętać, że zmiana formatu nie wpływa na zapisaną w pamięci komputera, bieżącą wartość danej ani też na jej dokładność. Przykład formatu zmiennej o wartości 1234.56789:

						1	2	3	4	.	5	6	7	8	9
			1	2	3	4	.	5	6	7	8	9	0	0	0
								1	2	3	4	.	5	6	8
												1	2	3	4

- **postać danej** (dla liczb rzeczywistych). Dostępne możliwości to postać prosta lub wykładnicza. W postaci wykładniczej można zazwyczaj określić całkowitą liczbę znaków oraz ilość cyfr wykładnika. Przykład formatu zmiennej o wartości 1234.56789 w postaci wykładniczej.

				.	1	2	3	4	5	7	E	+	0	0	4
1	2	3	4	.	5	6	7	8	9	0	0	E	+	0	4

Instrukcje bierne

- **sposób interpretacji znaku wartości liczby**. Domyślnie liczby dodatnie nie posiadają podczas wyświetlania na monitor (lub zapisu do pliku) znaku "+", można to jednak zmienić w większości języków programowania.

- **sposób interpretacji spacji** (dotyczy wprowadzania danych). Określa, czy podczas wprowadzania wartości liczbowych, znak spacji ma być ignorowany czy też nie.

- **sposób zapisu czasu i daty**. W przypadku czasu określa dokładność wyświetlania, (godzina, minuta, sekunda), zaś w przypadku daty określa kolejność składników oraz sposób wyświetlania roku (dwie lub cztery cyfry).

Instrukcje bierne

Instrukcje inicjowania danych – instrukcje służące do nadawania początkowych wartości zmiennych. Nadają się szczególnie do określania wartości wektorów i tablic – jest to zazwyczaj znacznie wygodniejsze i szybsze niż indywidualne przypisywanie wartości każdemu elementowi zmiennej indeksowanej.

Fortran:

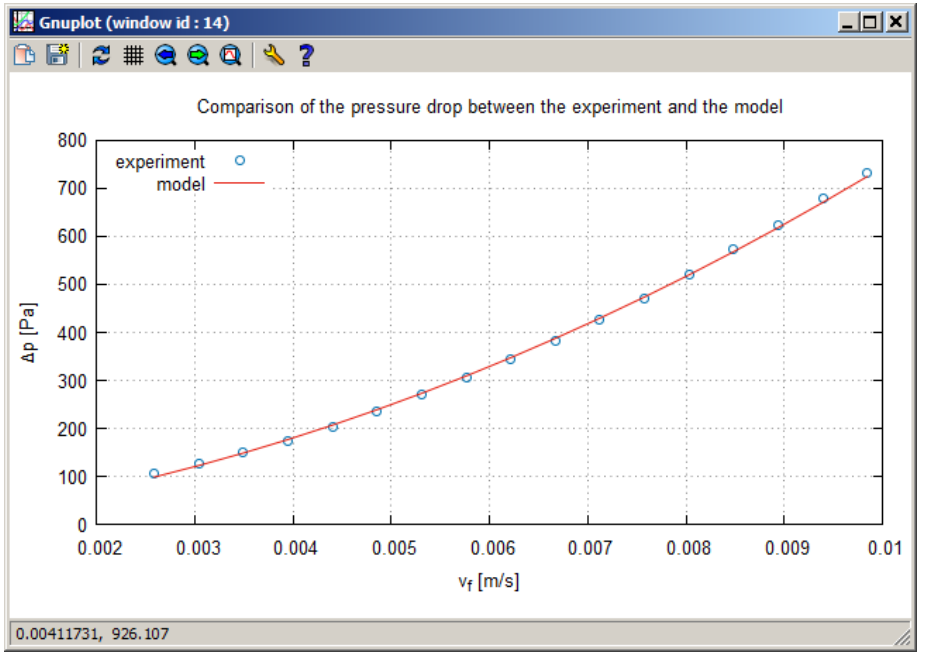
BLOCK DATA n*azwa*
deklaracja typów
definicja wspólnych bloków pamięci instrukcją **COMMON**
nadanie wartości początkowych zmiennym instrukcją **DATA**
END

Instrukcje bierne

```
!inicjalizacja tablic poczatkowych:
data t_w_t/270,272,274,276,278,280,282,284,286,288,290,292,294,296, &
298,300,302,304,306,308,310,312,314,316,318,320,322,324,326,328, &
330,332,334,336,338,340,342,344,346,348/

data ro_w_t/999.55,999.76,999.90,999.96,999.96,999.90,999.78,999.60, &
999.37,999.10,998.77,998.40,997.99,997.53,997.04,996.51,995.94, &
995.34,994.70,994.03,993.30,992.59,991.83,991.04,990.22,989.37, &
988.49,987.59,986.66,985.70,984.80,983.80,982.70,981.70,980.50, &
979.50,978.40,977.30,976.10,974.90/
```

przykład zastosowania instrukcji biernej **DATA** do przypisania początkowych wartości tablic (fragment programu Forchheimer)



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