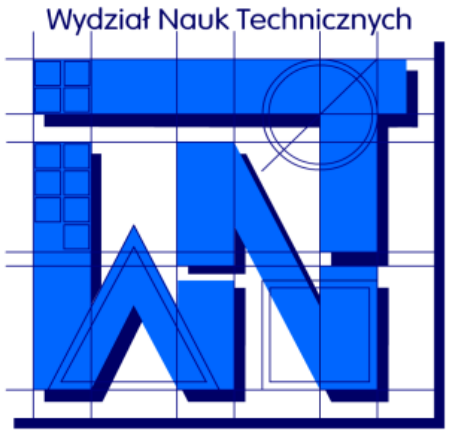


UNIVERSITY OF WARMIA AND MAZURY IN OLSZTYN
The Faculty of Technical Sciences
POLAND, 10-957 Olsztyn, M. Oczapowskiego 11
tel.: (48)(89) 5-23-32-40, fax: (48)(89) 5-23-32-55
URL: http://www.uwm.edu.pl/edu/sobieski/ (in Polish)

Języki Programowania

Dobre praktyki

Wojciech Sobieski

Olsztyn, 2001-2021

Dobre praktyki

Dobre praktyki – zasady pisania kodu źródłowego:

- poprawiające czytelność
- ułatwiające wprowadzenie zmian i modyfikacji (nawet po dłuższym czasie)
- ułatwiające wykorzystywanie tych samych rozwiązań (nawet fragmentów kodu), w różnych programach
- przyspieszające pisanie programów
- pozytywnie wpływające na końcową jakość programów
- ...

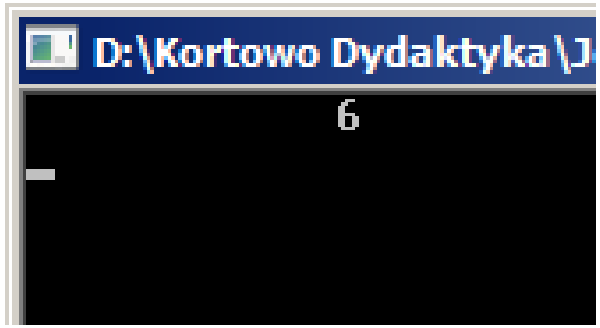
Im więcej aspektów się kontroluje, tym lepsza jest jakość programu!

> **„Skromny, działający program jest bardziej użyteczny niż program imponujący, ale niedokończony”**

Dobre praktyki

- **jawna deklaracja typów zmiennych**

```
program test

x = 2.45
y = 3.78
n = x + y

print *, n

read *
end
```



przykład błędnego wyniku (6 zamiast 6.23), będącego skutkiem niejawnej deklaracji typu – aby uniknąć podobnych sytuacji należy stosować instrukcję IMPLICIN NONE

Dobre praktyki

- **kontrola typu wyniku**

```
program test
implicit none

integer(kind=4) :: n = 12

print *, n/3.
print *, n/3

read *
end
```

przykład wpływu sposobu zapisu liczby na typ wyniku: może prowadzić do błędów

Dobre praktyki

- **nawiasy**

```
program test
implicit none

real(kind=4)     :: x = 1.
real(kind=4)     :: y = 2.
real(kind=4)     :: z = 3.

print *, x/y*z
print *, x/(y*z)

read *
end
```

przykład błędnych obliczeń wynikających z braku nawiasów

Dobre praktyki

- **przemyślana i logiczna struktura nazw**
- **nazewnictwo adekwatne do dziedziny (np. F, l, zamiast a, b)**

```
program test
implicit none

real(kind=4) ::     a
real(kind=4) ::     b
real(kind=4) ::     c
real(kind=4) ::     d
real(kind=4) ::     e

a = 100.
b = 10.
c = 2.e-6
d = 2.08e11
e = (a*b)/(c*d)

print '(A,F6.2)', 'wynik : ', e*1000.

read *
end
```

przykład złego nazewnictwa identyfikatorów: trudno odgadnąć, co liczy ten program i jakie jest znaczenie poszczególnych zmiennych (w rzeczywistości program liczy wydłużenie pręta na podstawie prawa Hooka)

Dobre praktyki

- **możliwie krótkie nazwy zmiennych (styl wielbłąda)**

```
program test
implicit none

real(kind=4) :: MaksymalnaTemperaturaRoczna
real(kind=4) :: MinimalnaTemperaturaRoczna

MaksymalnaTemperaturaRoczna = 32
MinimalnaTemperaturaRoczna = -28

print *, 'zakres = ', MaksymalnaTemperaturaRoczna -
MinimalnaTemperaturaRoczna

read *
end
```

przykład zbyt długich nazw, utrudniających pisanie wyrażeń – lepiej by było:
T_Rok_Max, T_Rok_Min lub T_max, T_min

Dobre praktyki

- **jedna instrukcja w jednym wierszu**

wersja 1:

```
program test;implicit none;real(kind=4)::x(1:20,1:30);
integer(kind=4)::i,j;do,i=1,20;do,j=1,30;x(i,j)=0.;enddo;enddo;read *;end
```

wersja 2:

```
program test
implicit none

real(kind=4)     :: x(1:20,1:30)
integer(kind=4)  :: i,j

do, i = 1, 20
  do, j = 1, 30
    x(i,j)=0.
  end do
end do

read *
end
```

obie wersje są poprawne,
ale druga jest znacznie czytelniejsza

Dobre praktyki

- **analogiczna symbolika w różnych programach realizujących podobne algorytmy (np. lx, ly, nx, ny, dx, dy, t, dt, n_iter)**

```
real(kind=8)     :: lx = 10.0   !dlugosc na kierunku x
real(kind=8)     :: ly = 10.0   !dlugosc na kierunku y

integer(kind=8)  :: nx = 50     !liczba komorek siatki na kierunku x
integer(kind=8)  :: ny = 50     !liczba komorek siatki na kierunku y

real(kind=8)     :: dx          !krok na kierunku x
real(kind=8)     :: dy          !krok na kierunku y
```

patrząc na ten fragment kodu, widać, że w programie będzie generowana dwuwymiarowa siatka obliczeniowa (fragment programu Dyfuzja2D)

Dobre praktyki

- **komentowanie znaczenia zmiennych (najlepiej wraz z jednostką)**

```
!wczytanie parametrow zadania:
open(unit=1,file='dane/input.dat')
read(1,*) ro          !gestosc
read(1,*) mi          !wspolczynnik lepkosci dynamicznej
read(1,*) lref        !dlugosc charakterystyczna dla liczby Reynoldsa
read(1,*) uref        !predkosc charakterystyczna dla liczby Reynoldsa
read(1,*) alx         !aktualna dlugosc domeny w kierunku x
read(1,*) aly         !aktualna dlugosc domeny w kierunku y
…
...
close(1)
```

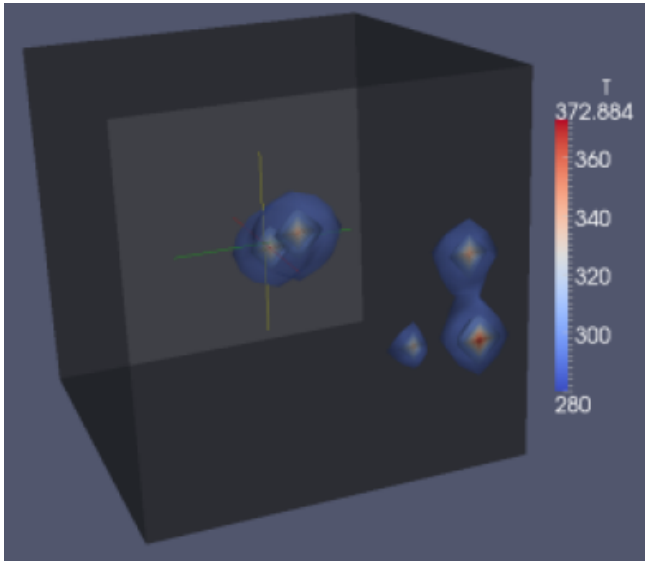
przykład stosowania opisów zmiennych w miejscu ich odczytywania z pliku:
zrobiono tak, gdyż deklaracje zmiennych zapisane są w osobnym pliku tekstowym, dołączanym instrukcją INCLUDE i opisu zmiennych nie widać w głównym kodzie źródłowym
(fragment programu Mac)

Dobre praktyki

- **grupowanie nazw wg logicznych kryteriów**

```
integer(kind=8)                :: sc_n      !liczba zrodel
integer(kind=8),allocatable    :: sc_i(:)   !indeks x komorki, w ktorej ma dzialac zrodlo
integer(kind=8),allocatable    :: sc_j(:)   !indeks y komorki, w ktorej ma dzialac zrodlo
integer(kind=8),allocatable    :: sc_k(:)   !indeks z komorki, w ktorej ma dzialac zrodlo
real(kind=8),allocatable       :: sc_t_start(:)!poczatkowy czas dzialania zrodla
real(kind=8),allocatable       :: sc_t_stop(:) !koncowy czas dzialania zrodla
real(kind=8),allocatable       :: sc_f(:)      !wartosc zrodla
integer(kind=8)                :: sc_l         !licznik zrodel
```

przykład grupowania nazw: wszystkie zmienne dotyczące tzw. źródeł (tu: źródeł ciepła) rozpoczynają się od tych samych liter (fragment programu Dyfuzja3D)



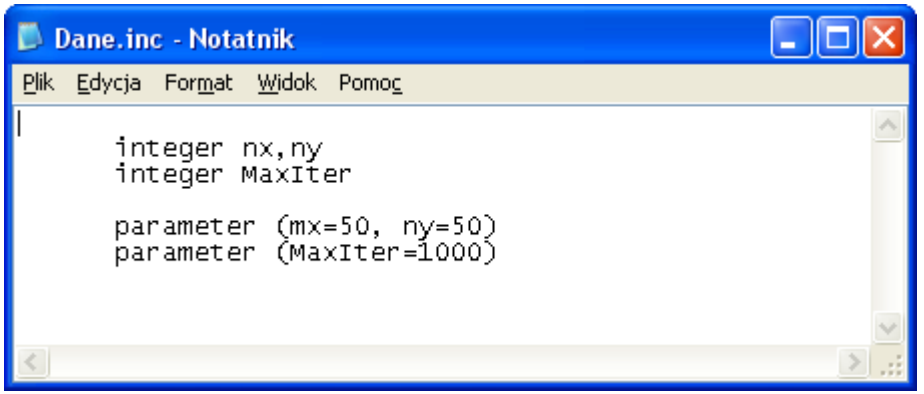
Dobre praktyki

- **deklarowanie zmiennych w blokach tematycznych (np. zmienne dotyczące siatki obliczeniowej, warunków brzegowych, warunków początkowych, metod rozwiązywania równań itp.)**

```
integer(kind=4)              :: n            !licznba punktow na wykresach
integer(kind=4)              :: i,j          !liczniki petli

real(kind=4)                 :: Re_min       !poczatek zakresu liczb Reynoldsa
real(kind=4)                 :: Re_max       !koniec zakresu liczb Reynoldsa

real(kind=4),allocatable     :: x(:)         !tablica punktow x
real(kind=4),allocatable     :: y(:)         !tablica punktow y

real(kind=4)                 :: dx           !krok przestrzenny

real(kind=4)                 :: f1,f2,f3,f4  !funkcja pomocnicze (Almedeij)
real(kind=4)                 :: a            !funkcja pomocnicza (FlammerBlanks)

real(kind=4)                 :: dana         !bieazaca wartosc Re
real(kind=4)                 :: wynik(15)    !tablica wynikow dla danego Re
```

Dobre praktyki

- **deklarowanie stałych w jednym miejscu programu (najlepiej na samym początku lub w pliku dołączanym)**



przykład zastosowania pliku dołączanego, służącego do definicji podstawowych parametrów programu (tu stosowane jest statyczne przypisanie pamięci)

Dobre praktyki

- **parametryzacja programu (np. rozmiaru zadania)**

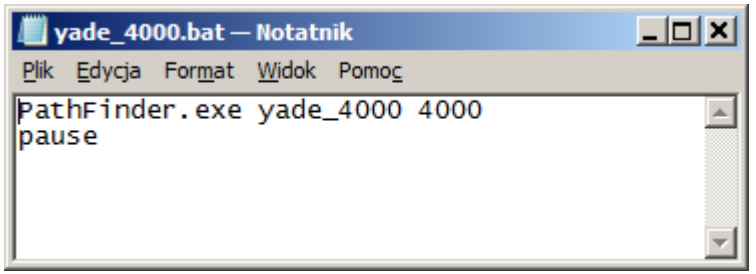
```
program test
implicit none

real(kind=4)     :: x(1:20,1:30)
integer(kind=4)  :: i,j

do, i = 1, 20
  do, j = 1, 30
    x(i,j) = 0.
  end do
end do

read *
end
```

przykład niepoprawnej budowy kodu źródłowego: zmiana rozmiaru siatki obliczeniowej wymaga korekty wszystkich pętli w programie: należy wprowadzić dodatkowe parametry (np. nx i ny)

Dobre praktyki

- **stosowanie parametrów wywołania**

```
!saving arguments of the software calling:
if (command_argument_count() == 2) then
  call getarg(1,dir)
  dir = trim(dir)
  call getarg(2,tmp)
  read(tmp,*) ns
else
  print '(/a$)',    ' No arguments ... '
  read *
  stop
end if
```



przykład zastosowania parametrów wywołania:
przekazanie nazwy przykładu, który ma być analizowany oraz jego rozmiaru
(fragment programu PathFinder)

Dobre praktyki

- **dynamiczne zarządzanie pamięcią**

```
integer(kind=8)                 :: ns     !the total number of spheres in the bed
integer(kind=8),allocatable     :: n(:)   !the numbers of all spheres
real(kind=8),allocatable        :: x(:)   !the X-coordinates of sphere centers
real(kind=8),allocatable        :: y(:)   !the Y-coordinates of sphere centers
real(kind=8),allocatable        :: z(:)   !the Z-coordinates of sphere centers
real(kind=8),allocatable        :: d(:)   !the diameters of all spheres
logical(kind=4),allocatable     :: flag(:) !the particle "flag"
```

przykład zastosowania dynamicznej alokacji pamięci:
rozmiar układu został określony przez parametr wywołania i nie trzeba kompilować programu oddzielnie dla każdego przypadku obliczeniowego
(fragment programu PathFinder)

Dobre praktyki

- **komentarze**

```
!poczatek 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
```

przykład komentarzy
w kodzie źródłowym
(fragment programu Drgania)

Dobre praktyki

- **komentarze nagłówkowe**

```
!-----------------------------------------------------------------------
!     autor:                        wojciech sobieski
!     kontakt:                      wojciech.sobieski@uwm.edu.pl
!     data utworzenia:              2005-09-14
!     ostatnia modyfikacja:         2016-09-19
!-----------------------------------------------------------------------
      program powierzchnia_swobodna

      implicit none

      integer :: n                      !liczba wezlow
      integer :: m                      !liczba iteracji w cyklu

      real,allocatable :: x(:)          !tablica wspolrzednych x
      real,allocatable :: z(:)          !tablica wspolrzednych z
      real,allocatable :: zr(:)         !tablica wspolrzednych z dla rysunku
```

przykład komentarza nagłówkowego w kodzie źródłowym (fragment programu Powierzchnia)

Dobre praktyki

- **komentarze nagłówkowe**

```
!-----------------------------------------------------------------------
! Program Fourier v. 1.4 (C) Sobieski & Trykozko 2014
!
! ver. 1.0 (W.S.)       scianka jednowarstwowa (rozwiazanie klasyczne),
!                       siatka regularna, wizualizacja Gnuplot
! ver. 1.1 (W.S.,A.T.) scianka jednowarstwowa (rozwiazanie macierzowe),
!                       siatka regularna, wizualizacja Gnuplot
! ver. 1.2 (W.S.,A.T.) scianka wielowarstwowa (rozwiazanie macierzowe),
!                       siatka regularna, wizualizacja Gnuplot
! ver. 1.3 (W.S.,A.T.) scianka wielowarstwowa (rozwiazanie macierzowe),
!                       siatka nieregularna, wizualizacja Gnuplot i ParaView
! ver. 1.4 (W.S.)       scianka wielowarstwowa (rozwiazanie macierzowe),
!                       siatka nieregularna, wizualizacja Gnuplot I
!                       ParaView, obliczenia ciepla, sledzenie zbieznosci
!                       obliczen, rozne metody rozwiazywania ukladow rownan
!
! ostatnia zmiana:      21 grudnia 2015
!-----------------------------------------------------------------------
```

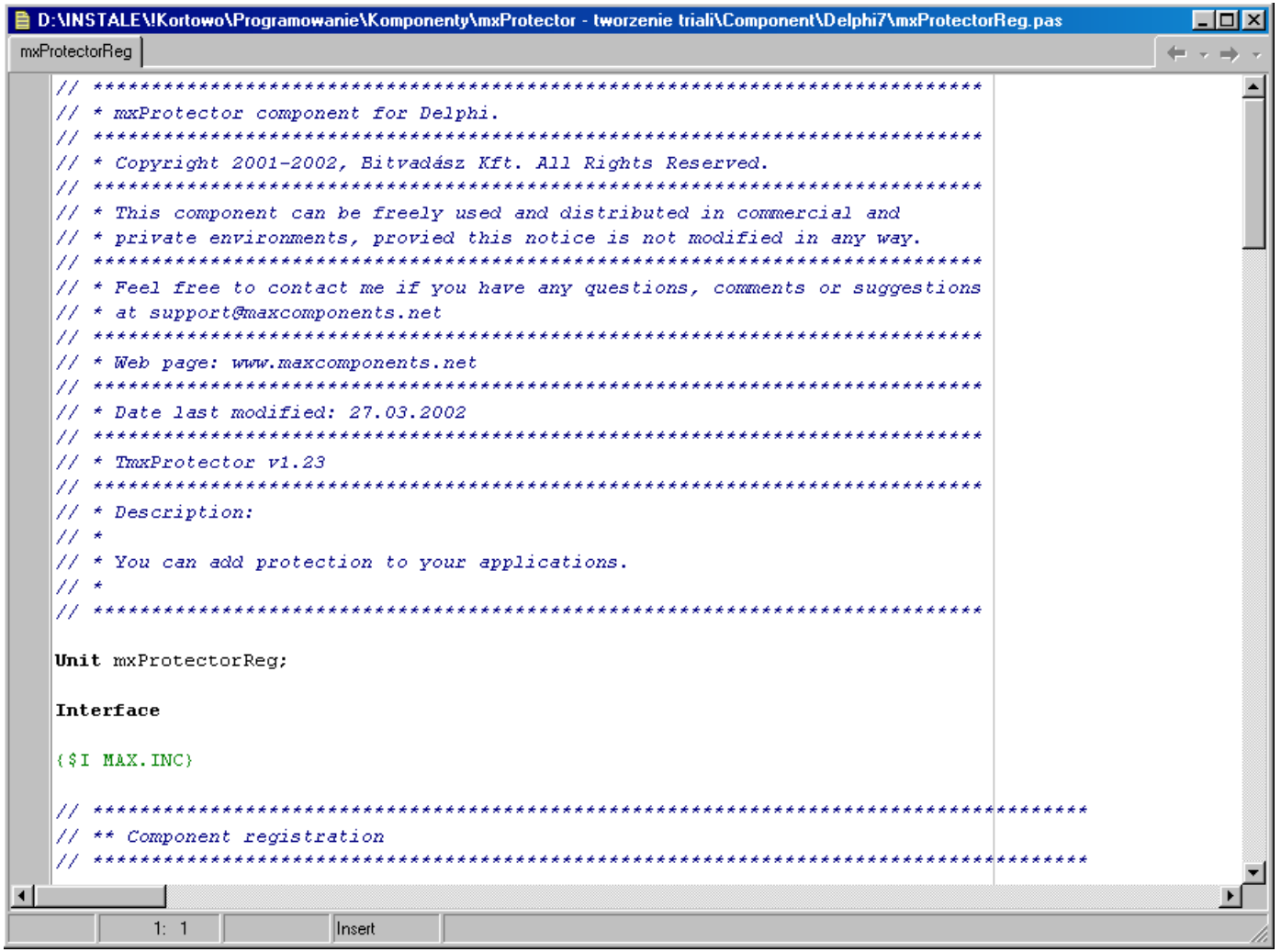
przykład komentarzy w kodzie źródłowym (fragment programu Fourier 1D, FVM)

Dobre praktyki

Komentarze nagłówkowe bywają mocno rozbudowane i zawierają:

- opis działania programu
- sposób użycia – jak wywołać program
- listę i opis ważniejszych zmiennych
- opis plików WE/WY
- nazwy używanych podprogramów
- nazwy wszelkich specjalnych metod, które zostały użyte, wraz ze wskazaniem, gdzie można znaleźć dalsze informacje
- informacje o czasie działania (jeśli ma to znaczenie)
- wymagania sprzętowe i systemowe
- opis specjalnych poleceń dla operatora / użytkownika
- informacje o autorach i kontaktach
- datę napisania (ew. datę ostatniej modyfikacji lub numer wersji)

Dobre praktyki



przykład rozbudowanego komentarza wstępnego (komponent mxProtector do środowiska Delphi)

Dobre praktyki

- **komentarze podprogramów**

```
!-----------------------------------------------------------------------
! funkcja liczy pole powierzchni pod krzywa:
!-----------------------------------------------------------------------
      real function calka(x,n,dx)

      implicit none

      real :: x(n)
      real :: dx                    !odstep miedzy punktami na kierunku x
      integer :: n                  !liczba przedzialow dyskretyzacji
      integer :: i                  !licznik petli

      calka=0
      do i=2,n
        calka = calka+((x(i)+x(i-1))/2.)*dx
      end do

      end function calka
```

przykład opisu funkcji: łatwiej ją znaleźć w kodzie źródłowym i od razu wiadomo, jakie jest jej zadanie (fragment programu Powierzchnia)

Dobre praktyki

- **odstępy pionowe**

```
program test
implicit none
real(kind=4) ::     l           !dlugosc [m]
real(kind=4) ::     dl          !zmiana dlugosci [m]
real(kind=4) ::     F           !sila [N]
real(kind=4) ::     A           !pole przekroju [m^2]
real(kind=4) ::     E           !modul Younga [N/m^2]
l = 10.
dl = 0.
F = 100.
A = 2.e-6
E = 2.08e11
dl = (F*l)/(A*E)
print '(A,F6.2,A)', 'Pret wydluzy sie o : ', dl*1000., ' [mm]'
read *
end
```

program byłby czytelniejszy, gdyby dodać puste linie pomiędzy fragmentami realizującymi różne zadania

Dobre praktyki

- **wcięcia i odstępy poziome**

```
!nalozenie biezacych zrodel: ------------------------------------------------
do i = 1, nx
  do j = 1, ny
    do k = 1, nz
      do sc_l = 1, sc_n
        if (sc_i(sc_l) == i .and. sc_j(sc_l) == j .and. sc_k(sc_l) == k) then
          if (t > sc_t_start(sc_l) .and. t < sc_t_stop(sc_l)) f(i,j,k) = sc_f(sc_l)
        end if
      end do
    end do
  end do
end do
```

dzięki wcięciom wyraźniej widać gdzie
jest początek i koniec danej pętli
(fragment programu Dyfuzja 3D)

Dobre praktyki

- **etykiety (rosnące ze skokiem > 1)**

```
      !parametry domyslne ukladu (jak zawiedzie plik):
20    continue
      g  = 9.81
      m  = 0.5
      l  = 2.
      ax = pi/8.          !8
      ay = pi/12.         !12
      dt = 0.01
      print '(a)', ' przyjeto wartosci domyslne: '

      !wyswietlenie parametrow:
30    continue
      print '(/a35,F12.6)', '   przyspieszenie ziemskie [m/s^2] :', g
      print '(a35,F12.6)' , '   masa ciezarka [kg]              : ', m
      print '(a35,F12.6)' , '   dlugosc liny [m]                : ', l
      print '(a35,F12.6)' , '   poczatkowe wychylenie na x [m]  : ', ax
      print '(a35,F12.6)' , '   poczatkowe wychylenie na y [m]  : ', ay
```

przykład poprawnego nadawania etykiet: jeśli zajdzie potrzeba, można wstawić etykietę między dwa inne bloki (fragment programu Lissajous)

Dobre praktyki

- **podział algorytmu na niezależne od siebie części elementarne** (najlepiej z wykorzystaniem funkcji i procedur)

```
!definicja siatki:
call define_grid(lx,ly,gn_x,gn_y,gc_x,gc_y)
!definicja warunkow brzegowych:
call define_boundary(gc_x,gc_y,s,b_s,b_n,b_e,b_w)
!definicja dodatkowych zrodel:
call define_source(s,sx_min,sx_max,sy_min,sy_max,source)

!pobranie liczby iteracji:
print '(/A$)', '  podaj liczbe iteracji: '
read (*,*,err=10) iter
print *, ''

!uruchomienie obliczen:
call solver_mrs(gc_x,gc_y,s,b_s,b_n,b_e,b_w,iter,sx_min,sx_max,sy_min,sy_max,source)
```

przykład budowy kodu z podziałem na jednostkowe zadania
(fragment programu Poisson – jest to większość programu głównego,
reszta zdefiniowana jest w odseparowanych procedurach)

Dobre praktyki

- **rozwiązywanie danego zadania tylko raz**
  (najlepiej z wykorzystaniem funkcji i procedur)

```
program test
implicit none

include 'hook_types.inc'

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

include 'hook_values.inc'

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

read *
end
```

przykład zastosowania funkcji lokalnej do obliczania wydłużenia pręta zgodnie z prawem Hooka: gdyby zaszła potrzeba modyfikacji wzoru obliczeniowego, wystarczy zmienić funkcję

Dobre praktyki

- **tworzenie uniwersalnych rozwiązań**
  (najlepiej z wykorzystaniem funkcji i procedur)

```
      ...
      call gaussj(AC,BT,T_nowe,n_c-2)
      ...
!-----------------------------------------------------------------------
!     Metoda Gauss-Jordan
!-----------------------------------------------------------------------
      subroutine gaussj(a,b,x,n)
      implicit none
      include 'fourier.prec'       !plik definiujacy precyzje obliczen
      ...

      integer(kind=prec)            :: n
      real(kind=prec)               :: a(n,n)
      real(kind=prec)               :: b(n)
      real(kind=prec)               :: x(n)
```

przykład procedury rozwiązującej układ równań liniowych metodą bezpośrednią Gaussa-Jordana – posiada ona uniwersalne nazwy zmiennych, odwołujące się do symboliki matematycznej: a – macierz współczynników, b – wektor prawej strony, x – wektor wartości szukanych, n – rozmiar układu (procedura użyta w programie Profil, Mac oraz Fourier 1D)

Dobre praktyki

- **przewidywanie wszystkich możliwych sytuacji**

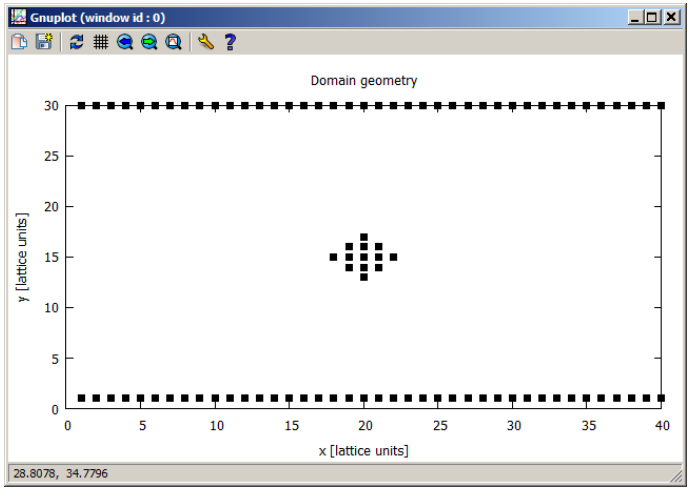
```
program test
implicit none

real(kind=4) :: delta = -4

if (delta > 0) then
  print *, ' dwa miejsca zerowe'
else if (delta == 0) then
  print *, ' jedno miejsce zerowe'
end if

read *
end
```

przykład błędnej instrukcji warunkowej (wyobraźmy sobie, że program wylicza miejsca zerowe funkcji): gdy delta jest mniejsza od zera to nie ma rozwiązania, ale program powinien jakoś sensownie zareagować – tu będzie wykonywał się dalej

Dobre praktyki

- **ciągłe testowanie kodu**

```
! Definicja wezlow tworzacych scianki:
is_solid_node = 0
do i = 1, lx
  is_solid_node(i,1)  = 1
  is_solid_node(i,ly) = 1
end do

! Definicja wezlow tworzacych przeszkode:
is_solid_node(int(lx/2),int(ly/2))     = 1
is_solid_node(int(lx/2-1),int(ly/2))   = 1
is_solid_node(int(lx/2),int(ly/2-1))   = 1
is_solid_node(int(lx/2-1),int(ly/2-1)) = 1
is_solid_node(int(lx/2+1),int(ly/2))   = 1
is_solid_node(int(lx/2),int(ly/2+1))   = 1
is_solid_node(int(lx/2+1),int(ly/2+1)) = 1
is_solid_node(int(lx/2-1),int(ly/2+1)) = 1
is_solid_node(int(lx/2+1),int(ly/2-1)) = 1
is_solid_node(int(lx/2-2),int(ly/2))   = 1
is_solid_node(int(lx/2+2),int(ly/2))   = 1
is_solid_node(int(lx/2),int(ly/2-2))   = 1
is_solid_node(int(lx/2),int(ly/2+2))   = 1
```



przykład sprawdzenia, czy dany fragment programu działa poprawnie: po zdefiniowaniu węzłów odbijających, tablica **is_solid_node** została zapisana do pliku i przedstawiona za pomocą środowiska Gnuplot – ponieważ efekt jest zgodny z oczekiwaniami, program można pisać dalej

Dobre praktyki

- **wieloplatformowość rozwiązań (Fortran)**

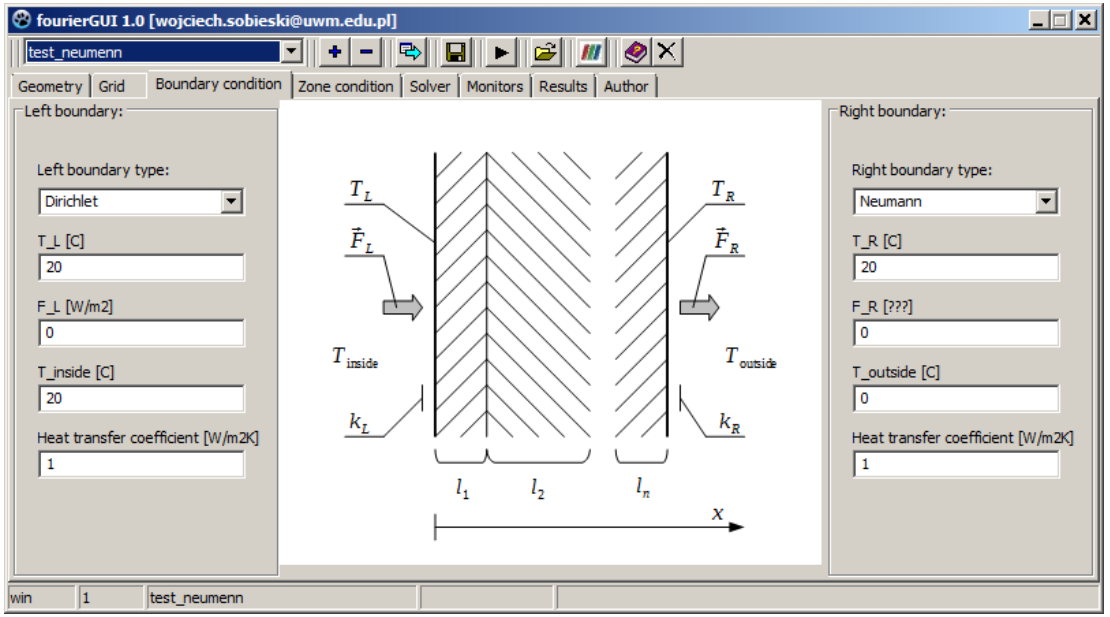
```
!pobiera sciezke biezacego katalogu:
call getcwd(tmp)
if (trim(tmp(:1)) == '/') then
  print *, 'Linux'
  sign = '/'
  call system('clear')
else
  print *, 'Windows'
  sign = '\'
  call system('cls')
end if
```

przykład programu uwzględniającego możliwość kompilacji i działania w różnych systemach operacyjnych: w systemach Windows oraz UNIX\Linux inne są znaki łączące katalogi i pliki w ścieżkach oraz inne są polecenia systemowe
(fragment programu PathFinder)

Dobre praktyki

- **wieloplatformowość rozwiązań (Lazarus)**

```
{$IFDEF Windows}
WriteLn(tmp,'win');
sign := '\';
{$ENDIF Windows}
{$IFDEF Unix}
WriteLn(tmp,'lin');
sign := '/';
{$ENDIF Unix}
```



przykład programu wykorzystującego specjalne dyrektywy, których realizacja zależy od rodzaju systemu operacyjnego, w którym program jest wykonywany (fragment programu fourierGUI napisanego w środowisku Lazarus)

Dobre praktyki

- **wieloplatformowość rozwiązań (Gnuplot)**

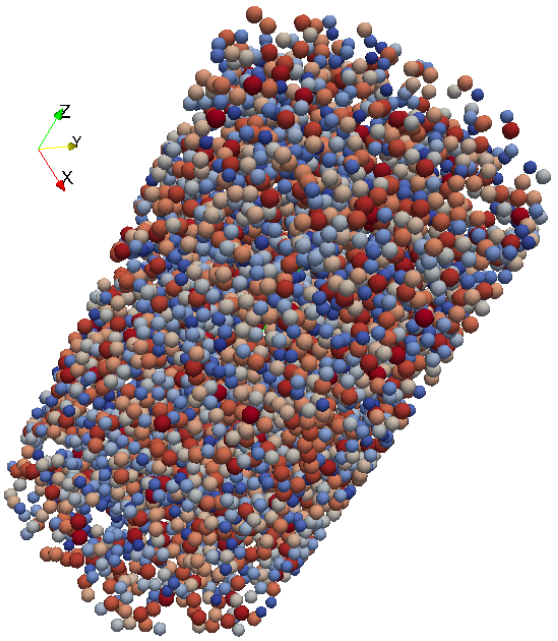
```
set title 'Wykres zbieznosci - porownanie'
set xlabel 'Numer iteracji'
set ylabel 'x[n+1] - x[n]'
set yrange [0.0:0.0001]
set grid
plot 'zbieznosc-jacobi.txt' title 'Jacobi' with lines lt 1, \
     'zbieznosc-gauss-seidel.txt' title 'Gauss-Seidel' with lines lt 2, \
     'zbieznosc-sor.txt' title 'SOR' with lines lt 3
set terminal png
set output '!zbieznosc-suma.png'
replot
pause mouse 'Wykres zapisano do pliku...'
exit gnuplot
```

przykład skryptu Gnuplota działającego tak samo w różnych systemach operacyjnych
(fragment programu Profil)
UWAGI: unikać terminala windows; sposób wyświetlania (rodzaje linii, kolory) mogą być różne w różnych systemach operacyjnych

Dobre praktyki

- **wieloplatformowość rozwiązań (VTK)**

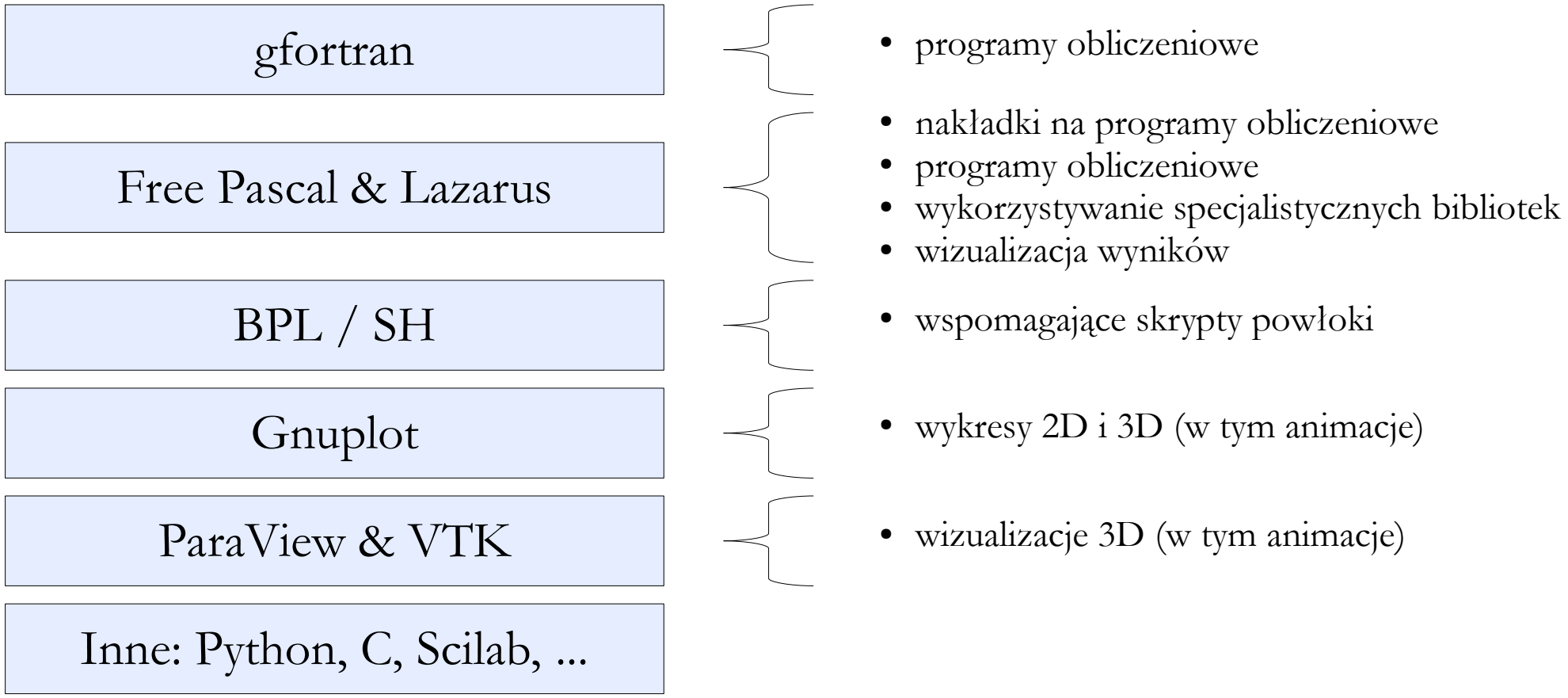
```
!saving the input data in VTK format for ParaView:
open(1,file=trim(dir)//sign//trim(dir)//'.in.vtk')
write(1,'(A)') '# vtk DataFile Version 2.1'
write(1,'(A)') 'The bed'
write(1,'(A)') 'ASCII'
write(1,'(A)') ''
write(1,'(A)') 'DATASET UNSTRUCTURED_GRID'
write(unit=tmp,fmt='(I6)') ns
write(1,'(A)') 'POINTS '//trim(tmp)//' double'
do i = 1, ns
  write(1,'(3F16.8)') x(i),y(i),z(i)
end do
write(1,'(A)') 'POINT_DATA '//trim(tmp)
write(1,'(A)') 'SCALARS Diameter double'
write(1,'(A)') 'LOOKUP_TABLE default'
do i = 1, ns
  write(1,'(F16.8)') d(i)
end do
close(1)
```



przykład zapisu danych do standardowego pliku VTK
(fragment programu PathFinder)

Dobre praktyki

- **„otwartość” stosowanych narzędzi**



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