Typowe konstrukcje językowe

(na przykładzie PASCALA)

STRUKTURA PROGRAMU

- **część opisowa**

  zawiera definicje i deklaracje *„obiektów”*

  (np. zmiennych lub typów zmiennych)

- **część wykonawcza**

  zawiera sekwencję instrukcji realizujących algorytm

definicje, deklaracje, instrukcje - zdania języka

***semantyka* – dostępne symbole do budowy wyrażeń**

W PASCALU zdania zbudowane z *“wyrazów”*, a te z elementarnych *“cegiełek”* - znaków;

- **słowa kluczowe,**
- **identyfikatory „obiektów języka”**
- **stałe**
- **inne elementy konstrukcji zdań**

***syntaktyka* – zasady budowy wyrażeń i zdań**
(w celu kodowania informacji)

PASCAL – zdania oddzielone ;
Fortran – zdania w oddzielnych wierszach

zmienna posiada: (*mat*. Wielkość mogąca przyjmować rozmaite wartości)

atrybuty (w informatyce)

- **nazwę (identyfikator)**
- **jest określonego typu (tzn. przyjmuje wartości tylko z jednego zbioru o określonej długości i określonym sposobie reprezentacji binarnej)**
- **miejsce w PAO na przechowywanie aktualnej wartości = adres**

typ:

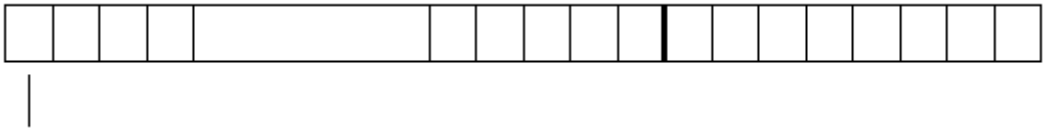
zbiór możliwych wartości (niekoniecznie liczbowych) o jednakowej reprezentacji binarnej

PASCAL

- **typy standardowe (predefiniowane)**
- **możliwość tworzenia własnych typów (typów programisty) – duże bogactwo możliwości ....**

przykłady (najbardziej typowe):

nazwa typu	opis typu
char	**wszystkie znaki ('A', '%', 's',...) 1 bajt**
boolean	**zb. wartości logicznych {true, false} 1b**
integer	**zb. liczb całkowitych representowanych na 2 bajtach (2^{15}-1, -2^{15})**
byte	**liczby całkowite dodatnie na 1 bajcie**
real	**przybliżenia do liczb rzeczywistych na 6 bajtach (48 bitów)**



identyfikator –

(dotyczy zmiennych, funkcji, podprogramów...)
PASCAL:
nazwa ----> dowolny ciąg max. 63 znaków zaczynający się od litery (nie zawierający pewnych znaków specjalnych np. spacji)

w PASCALU

type *nazwa*=opis typu lub nazwa innego typu;

w FORTRANIE definiuje się zmienne określonego typu
jako *explicit* lub *implicit*

implicit double precision (a-h,o-z)

lub

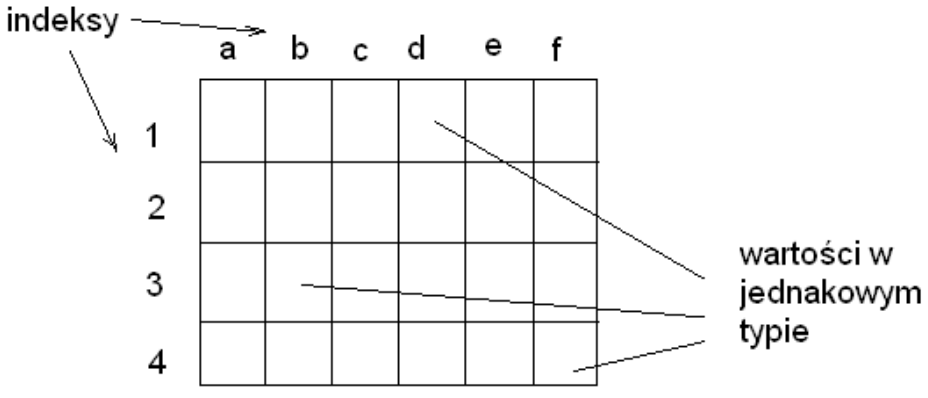
implicit none
double precision zmienna_1, zmienna_2, ...
integer liczba, n_liczba, ...
character*24 alfa,beta, ...

w zastosowaniach matematyczno-fizycznych fundamentalne są typy i zmienne indeksowane (tablicowe)

w PASCALU

typ tablicowy = struktura danych zbudowana z elementów jednego (jednakowego) typu; (macierze)
dostęp do pojedynczego elementu (macierzy) poprzez indeksy;

```
type ala2=array[char,21..30,1..15] of real;
type ala3=array[1..10,boolean] of ala2;
```



mogą być wielowymiarowe

deklaracje zmiennych mogą odbywać się bez poprzedzania definicją typu

w FORTRANIE

w deklaracji zmiennych:	**integer a(100,200)**	**- statycznie**
	integer a(:,:)	**- dynamicznie**
- w części wykonawczej po wcześniejszej deklaracji	allocate (a(n,m))	
	allocatable a	
lub po deklaracji typu ***(stat lub dyn)***	**dimension a(20,30)**	

deklaracja zmiennych:

(w PASCALU w części opisowej)

`var lista nazw zmiennych:typ;`

możliwość sztywnej alokacji zmiennych

`var nazwa : opis_typu  absolute  adres : offset;`

lub

`var nazwa : opis_typu  absolute  nazwa_innej_zm;`

podstawowe sposoby odwołania do zmiennej

(do jej wartości lub wartości elementu dla zmiennych zadeklarowanych w typach złożonych)

- **zmienne proste przez nazwę**
- **zmienne indeksowane – nazwa[lista indeksów]**
- **zmienne rekordowe – nazwa.pole ....**

WYRAŻENIA -

sensowna kombinacja operatorów argumentów i nawiasów

(argumenty: stałe, zmienne, inne wyrażenia, nazwy funkcji są częścią składową instrukcji)

operatory (w-g priorytetu)

- **jednoargumentowe**
- **multiplikatywne**
- **addytywne**
- **relacyjne**
- **logiczne**

ZDANIA - INSTRUKCJE

PROSTE (PASCAL)

- **przypisania** **a:=(zesp.re + sin(zesp.im))/2;**
- **wywołania procedury** **writeln(a);**
- **skoku** **goto label1;**

FORTRAN

```
a=(z+20.d0)/(1-x**2)
call calkuj(x1,x2,n,a)
goto 2001
```

(w FORTRANIE symbolem relacji równoważności jest „ .eq. „)

STRUKTURALNE

PASCAL

1. **złożona (bloku) – begin ..... end;**
2. **warunkowe (decyzyjne)**

- **if *wyrażenie* then *instrukcja1*** [**else *instrukcja2***];

- **case *wyrażenie* of**

  lista_instrukcji_wyboru
  [else *instrukcja*]
  end;

gdzie

lista_instrukcji_wyboru:

lista_wartości_w_typie_*wyrażenia* : *instrukcja*;

FORTRAN

```
if(a.eq.b) then
      lista instrukcji
else
      lista instrukcji
endif
```

lub

...select case ....

PĘTLE

1. **for *nazwa_zmiennej* := w1 to [downto] w2 do instrukcja**

   w1, w2 - wyrażenia zgodne z typem *zmiennej*

   ***zmienna* kontrolna pętli**

2. **while *wyrażenie_l* do *instrukcja*;**

3. **repeat**

   lista_instrukcji
   until *wyrażenie_l*;

wyrażenie_l **- o wartości logicznej**

FORTRAN

do zm_kontr=wyr1,wyr2[,krok] [..] – *ozn. opcjonalność*
lista instrukcji
enddo

operacje wejścia / wyjścia realizowane są w PASCALU przez

standardowe procedury

- **read / readln (lista_nazw_zmiennych);**
- **write / writeln (lista_elementów_wyjścia);**

*** przy czytaniu ze standardowego wejścia (klawiatura)**
i pisaniu na standardowe wyjście (ekran)

w FORTRANIE – realizacja przez instrukcje

read (*numer pliku,format[,end=etykieta]*) lista

write (*numer pliku,format*) lista

standardowe funkcje matematyczne są w PASCAL-u i innych językach dostępne pod standardowymi skrótami, np.:
sin(x), sqr(x), exp(x) ...

FUNKCJE I PROCEDURY

(tylko na przykładzie PASCALA)

wydzielona część programu wykonująca wielokrotnie powtarzany fragment algorytmu

stanowi oddzielną całość;
może współpracować z różnymi programami

przykłady (trywialne)

1. **przy mnożeniu 2 macierzy wielokrotnie wykonujemy mnożenie wiersza przez kolumnę (mnożenie 2 wektorów)**

2. **przy numerycznym całkowaniu funkcji (pole pod krzywą złożone z trapezów) odwołujemy się do wartości funkcji (ten sam przepis) dla wielu różnych wartości argumentu**

definicja procedury:

procedure *nazwa_procedury* (lista_parametrów_formalnych);

część opisowa

begin
zestaw_instrukcji <--- ”ciało procedury”
end;

definicja procedury lub funkcji musi pojawić się w części opisowej programu (lub w module)

lista_parametrów_formalnych - opcjonalna

parametry formalne powinny wystąpić w ciele procedury,

stanowią argumenty dla operacji wykonywanych w procedurze

wywołanie procedury (w programie poprzez nazwę procedury) odbywa się z parametrami aktualnymi ;

w ten sposób przekazuje się wartości zmiennych, stałych i wyrażeń między programem (głównym) a procedurą

dwa sposoby przekazywania parametrów

1. przez wartość

*** w procedurze tworzona jest lokalna zmienna o nazwie takiej jak parametr formalny, której początkowa wartość = wartość parametru aktualnego**

*** operacje wewnątrz procedury nie zmieniają wartości parametru aktualnego (na zewnątrz procedury)**

2. przez zmienną

*** pod nazwą parametru lokalnego kryje się adres nielokalnej zmiennej (parametru aktualnego)**

.....

- **zmienne zadeklarowane w części opisowej programu / modułu traktowane są jako globalne = dostępne (również dla zmian wartości) wewnątrz procedur i funkcji**

FUNKCJE

```
function nazwa (lista_param_form) : typ_wyniku ;
      część opisowa
begin
      sekwencja instrukcji
end;
```

*** powinna (ale nie musi) wystąpić instrukcja nadania wartości**

zmiennej o *nazwie* funkcji

wywołanie funkcji -

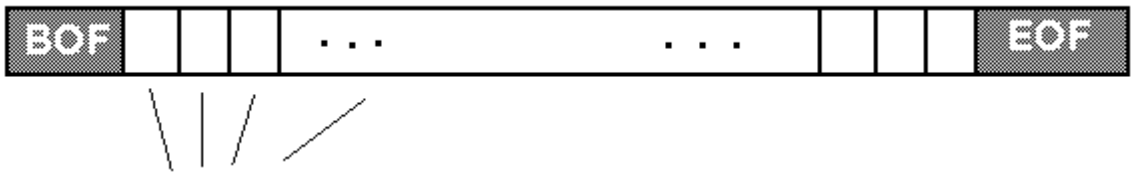
przez podstawienie nazwy funkcji (z argumentami aktualnymi) w miejsce zmiennej w dowolnym wyrażeniu

MODUŁY, biblioteki procedur, funkcji, typów, zmiennych...

grupowanie procedur i funkcji w biblioteki

MODUŁY STANDARDOWE

postać plików zewnętrznych



elementy pliku (porcje zapisu)
„wiersze" dla plików tekstowych

DYNAMICZNY PRZYDZIAŁ PAMIĘCI

typ wskazujący (wskaźnikowy)

type nazwa = ^nazwa_typu_wskazywanego(bazowego);

zbiór wartości możliwych adresów dla zmiennych

dynamicznych typu bazowego

utworzenie zmiennej dynamicznej (w heap'ie - "stercie")
(funkcje z modułu system)

new (nazwa_zmiennej_wskaźnikowej)

getmem (zmienna_wskaźnikowa , rozmiar)
w bajtach

przykład:

```
type xxx=record
                napiecie : real;
                stan       : boolean;
              end;
type tab=array[1..3,boolean] of  xxx;
var  nowa1 : ^tab;
     nowa2 : ^xxx;
 ...
new (nowa1);
getmem(nowa2, 7);
 ...
nowa1^[3,false].stan := nowa2^.stan;
```

zwalnianie obszaru pamięci dynamicznej:

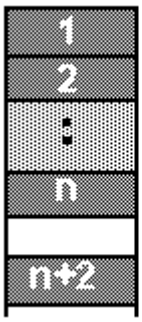
dispose (zm_wskaźnikowa)

freemem (zm_wskaźnikowa, rozmiar)

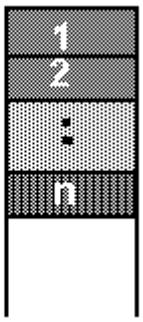
release (zm_wskaźnikowa)

mark (zm_wskaźnikowa)

- przypisanie aktualnego adresu *szczytu* stosu



dispose zmiennej n+1



release zmiennej n+1

W FORTRANIE 90-95

bardzo nowoczesna konstrukcja

1. **deklaracja nie-statyczna**

   dimension a(:,:),b(:)

2. **deklaracja dynamicznego przypisywania miejsca**

   allocatable a,b

3. **przypisywanie miejsca w pamięci**

   allocate (a(100,20))

4. **zwalnianie miejsca w pamięci**

   deallocate (a)
   allocate (b(50))
   .....
   deallocate (b)