Notacje

Do opisu składni języków formalnych często używa się notacji zaproponowanej przez J. Backusa do opisu języka Fortran i zastosowanej na dużą skalę przez P. Naura w opisie języka Algol 60. Od tych dwu osób nazywana jest notacją Backusa-Naura w skrócie BNF. Istnieje kilka odmian tej notacji – poniżej opisana będzie odmiana pierwotna.

Metajęzyk tej notacji zawiera następujące symbole:

::= równe z definicji
| lub
< > nawiasy obejmujące symbole nieterminalne
[ ] wydzielanie sekwencji powtarzanych 1 lub 0 razy
{ } wydzielanie sekwencji powtarzanych n razy (także 0 razy)

Przykład definicji składni prostego języka wyrażeń arytmetycznych

<A> ::= x | (<B>) A = ”x” | ”(” B ”)”
<B> ::= <A> <C> B = A B
<C> ::= { + <A> } C = { ”+” A }

Konstrukcje dopuszczalne w języku:

x
(x) ((x+x)+x)
(x+x)

Konstrukcji: ((x+x)x) nie da się wywieźć

Wykorzystanie BNF do opisu języka pascal:

<cyfra> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<ciag cyfr> ::= <cyfra> {<cyfra>}
<LCBZ> ::= <ciąg cyfr>
<liczba całkowita> ::= <LCBZ> | + <LCBZ> | - <LCBZ>

definicja równoważna ciągu cyfr oparta na rekurencji:

<ciąg cyfr> ::= <cyfra> | <ciąg cyfr> <cyfra>

<litera> ::= a | b |c ... z | A | B |... | Z
<identyfikator> ::= <litera> { <litera> | <cyfra> }

Zmodyfikowana wersja notacji BNF ma zestaw symboli metajęzykowych nieco zmieniony:

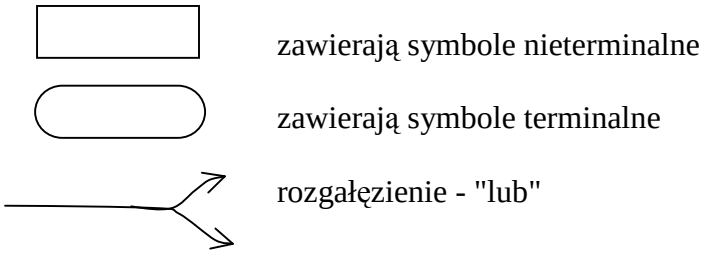
= z definicji. Po lewej stronie podajemy nazwę definiowanego pojęcia, a po prawej jego definicję
„x” cudzysłowy ograniczają symbole terminalne (znaki alfabetu). Symbole nieterminalne występują bez użycia żadnych ograniczników
[x] nawiasy kwadratowe oznaczają opcjonalne wystąpienie zapisu (jednokrotne lub brak)

{x} nawiasy klamrowe oznaczają wielokrotne wystąpienie zapisu, w tym także brak
| alternatywa. Po obu stronach kreski występują zapisy z których można wybrać tylko jeden
(x|y|...|z) nawiasy okrągłe ograniczają zakres wyboru spośród wielu wariantów
. kropka kończy definicję

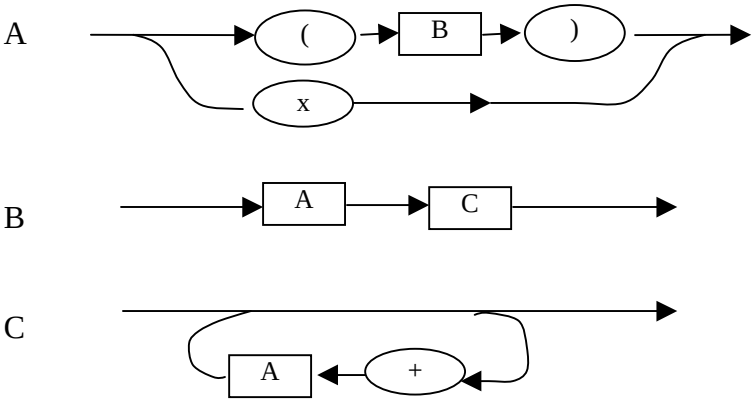
Przy użyciu tego nieznacznie tylko wzbogaconego systemu opisu można prościej opisać b. złożone zapisy. Na przykład w definicji identyfikatora można łatwo uwzględnić fakt, że w Turbo Pascalu zamiast każdej litery może także występować znak podkreślenia:

identyfikator = (litera | „_”) { (litera | cyfra | „_”) }

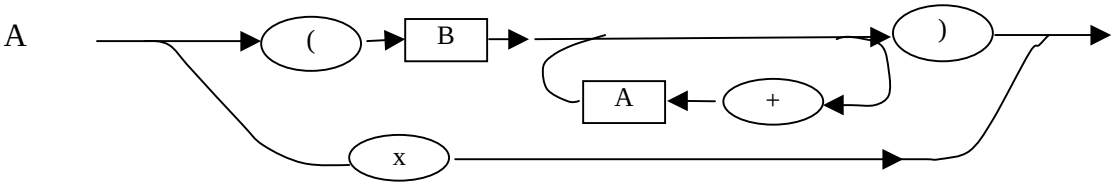
Inny sposób formalnego opisu języka to diagramy składniowe:



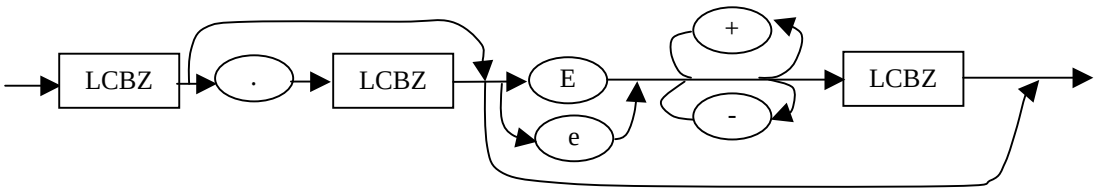
Przykład – język wyrażeń jak poprzednio:



Trzy diagramy można zastąpić jednym:



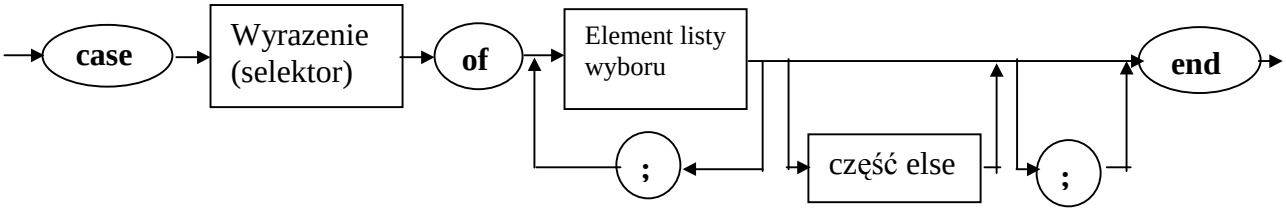
Definicja Liczby bez znaku:



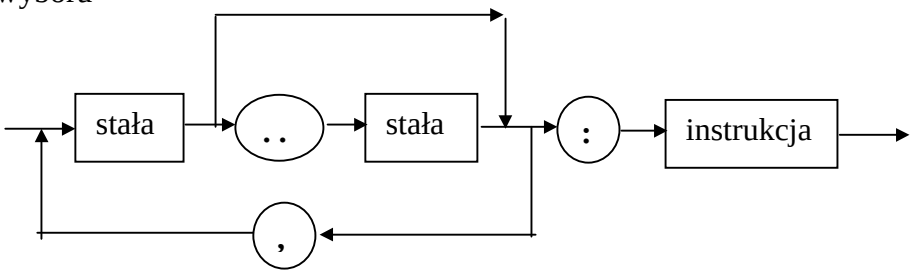
INSTRUKCJA WYBORU CASE

Jest to instrukcja warunkowa, w której podstawą podjęcia decyzji nie musi być wyrażenie logiczne, lecz może być wyrażenie innego typu porządkowego. Instrukcje ta stosuje się wtedy, gdy wykonanie rożnych operacji uzależnione jest od wartości wyrażenia nazywanego selektorem.

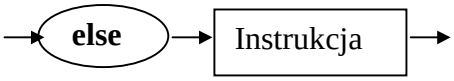
instrukcja_wyboru



element_listy_wyboru



czesc else



Selektor musi być wyrażeniem typu porządkowego.
Wszystkie stale w elementach listy wyboru (zwane etykietami wyboru) musza być parami rożne i musza być typu porządkowego, zgodnego z typem selektora.
Instrukcje występujące w elementach listy wyboru oraz instrukcje po słowie kluczowym **else** mogą być dowolnymi instrukcjami języka (prostymi lub strukturalnymi - wiec w szczególności może to być instrukcja złożona).
Działanie instrukcji wyboru jest następujące: W sekwencji elementów listy wyboru wyszukiwana jest instrukcja poprzedzona stałą wyboru równą obliczonej wcześniej wartości selektora (lub przedziałem ograniczenie_dolne **..** ograniczenie_gorne, w zakresie którego mieści się wartość selektora) i ta właśnie instrukcja jest wykonywana, po czym sterowanie przekazywane jest do instrukcji występującej po słowie **end** kończącym instrukcje wyboru. Jeżeli żadna z etykiet wyboru nie jest równa wartości selektora, a w instrukcji wyboru

występuje człon **else**, to zostanie wykonana instrukcja podana po słowie else; jeśli członu **else** nie m, wykonana zostanie instrukcja pusta. W obu przypadkach sterowanie zostanie przekazane do instrukcji występującej po instrukcji **case**.

przykładowo:

```
case operator of
  plus:  x := x + y;
  minus: x := x - y;
  razy:  x := x * y;
end;

case i of
  0,2,4,6,8 : writeln ('liczby parzyste');
  1,3,5,7,9 : writeln ('liczby nieparzyste');
  10.. 100  : writeln ('miedzy 10 i 100);
else
  writeln('ujemne lub wieksze od 100);
end;

case miesiac of
  1,3,5,7,8,10,12: dni := 31;
  2              : dni := 28;
  4,6,9,11       : dni := 30;
end;
```

Jeżeli w ostatnim przykładzie założymy, że zmienna miesiąc przyjmuje tylko wartości całkowite z przedziału <1,12>, czyli np. zadeklarowana jest następująco:

```
var
  miesiac: 1.. 12;
```

to instrukcja **case** może mieć postać:

```
case miesiac of
   2       : dni := 28;
   4,6,9,11: dni := 30;
 else        dni := 31;
end;
```