Katedra Informatyki Stosowanej
Politechnika Łódzka

Lingwistyka Matematyczna
Analiza składniowa

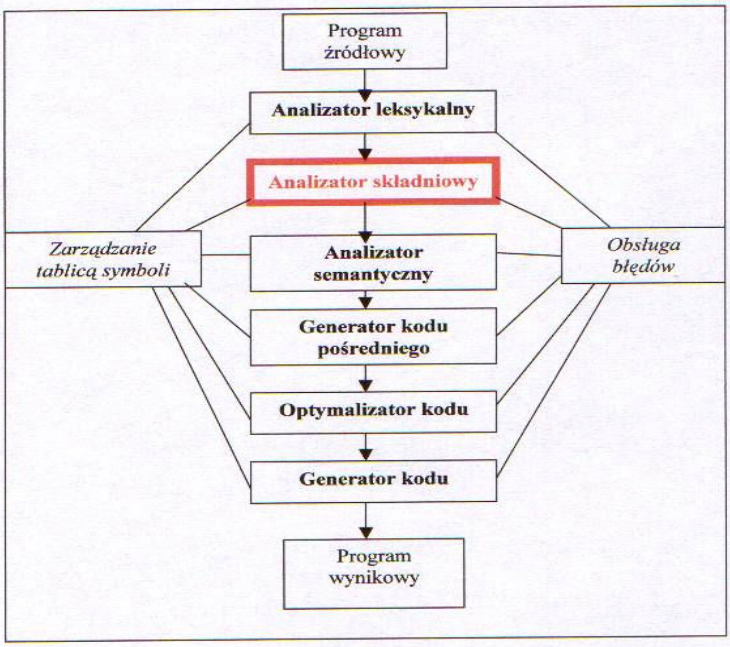
© dr hab. inż. Lidia Jackowska - Strumiłło

ANALIZATOR SKŁADNIOWY

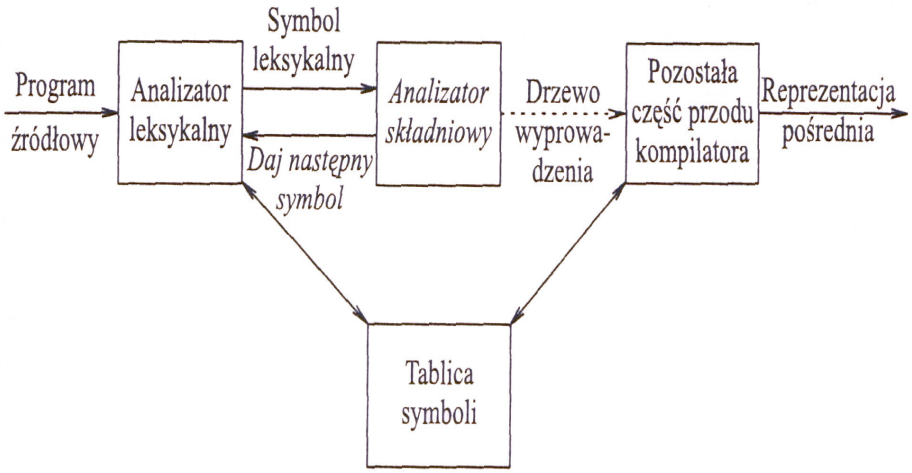
***Analizator składniowy*, inaczej** analizator syntaktyczny lub **parser** (ang. *syntax analyser, syntactic analyser, parser*) jest to część translatora dokonująca rozbioru gramatycznego programu lub samodzielnej jednostki kompilacji (modułu).

Ważnym elementem procesu translacji jest zgłoszenie użytkowi komunikatów o ewentualnych błędach w programie źródłowym

Etapy Procesu Kompilacji



Umiejscowienie analizatora składniowego w kompilatorze



KLASYFIKACJA ANALIZATORÓW SKŁADNIOWYCH

ZE WZGLĘDU NA:

- **gramatykę:**
  - **analizatory LR,**
  - **analizatory LL;**
- **kolejność przeglądania:**
  - **analizatory wstępujące,**
  - **analizatory zstępujące,**
  - **analizatory przewidujące;**
- **sposób reprezentacji składni:**
  - **analizator składniowy dla zadanej gramatyki,**
  - **analizator składniowy sterowany składnią.**

DIAGRAMY SKŁADNIOWE

Diagramy składniowe to graficzny sposób przedstawienia gramatyki równoważny notacji BNF.

Cechy diagramów:

- wygodna forma zapisu języka
- dają zwięzły i przejrzysty obraz struktury języka
- pozwalają zrozumieć proces rozbioru
- postać opisu właściwa przy projektowaniu języka

Reguły konstrukcji diagramu składni

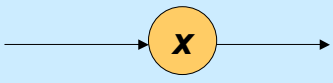
1. Każdy symbol pomocniczy A, dla którego istnieje produkcja:

   $$A ::= \xi_1 \mid \xi_2 \mid \ldots \mid \xi_n$$

   zostaje przekształcony na diagram symbolu A o strukturze określonej przez prawą stronę produkcji zgodnie z regułami 2 - 6.

Reguły konstrukcji diagramu składni

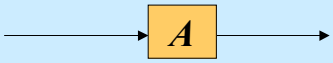
2. Symbol końcowy x oznaczany jest symbolem:



   Każde wystąpienie symbolu końcowego x w ciągu ξ_i odpowiada instrukcji rozpoznającej ten symbol i pobierającej kolejny symbol z ciągu wejściowego.

Reguły konstrukcji diagramu składni

3. Symbol pomocniczy B oznaczany jest symbolem:



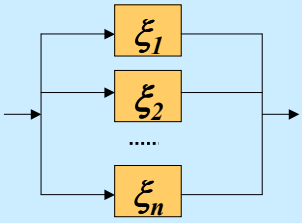
Każde wystąpienie symbolu pomocniczego B w ciągu ξ_i odpowiada uaktywnieniu podprogramu opisanego diagramem dla symbolu B.

Reguły konstrukcji diagramu składni

4. Produkcja postaci:

$$A ::= \xi_1 \mid \xi_2 \mid \ldots \mid \xi_n$$

zostaje przekształcona na diagram:



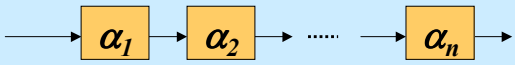
gdzie każde  jest utworzone przez stosowanie reguł 2 – 6 dla ξ_i .

Reguły konstrukcji diagramu składni

5. Ciąg ξ postaci:

$$\xi = \alpha_1 \alpha_2 \ldots \alpha_m$$

zostaje przekształcony na diagram:



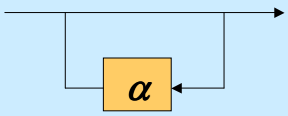
gdzie każde α_i jest utworzone przez stosowanie reguł 2 – 6 dla α_i .

Reguły konstrukcji diagramu składni

6. Ciąg ξ postaci:

$$\xi = \{ \alpha \}$$

zostaje przekształcony na diagram:



gdzie α jest utworzone z α przez stosowanie reguł 2 – 6.

Przykład

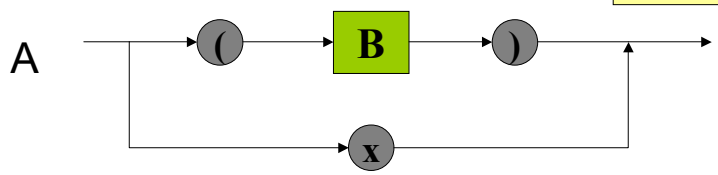
Zbuduj diagramy dla gramatyki:

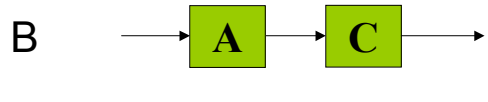
A::= x | (B)
B::= AC
C::= {+A}

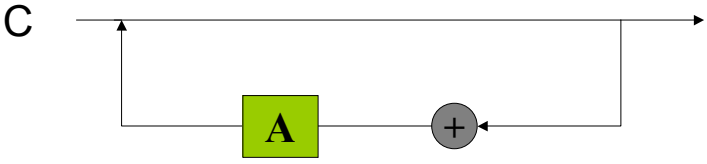
i uprość je do optymalnej postaci.

Przykład

A::= x | (B)
B::= AC
C::= {+A}



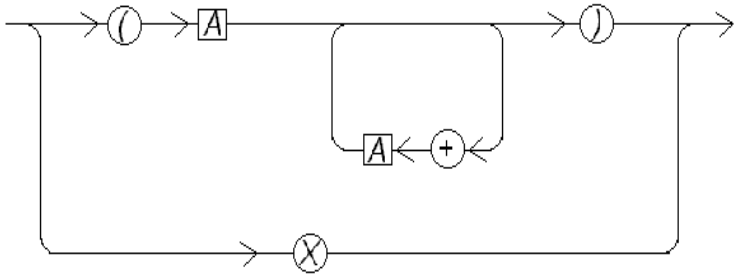




Przykład

Zredukowany diagram składni:

A



ANALIZATOR SKŁADNIOWY DLA ZADANEJ GRAMATYKI

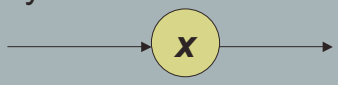
Przy budowie analizatora dla konkretnej gramatyki stosowane są ***reguły przejścia od deterministycznego diagramu składni do programu***.

Diagram stanowi schemat blokowy działania programu.

Celem ***analizy składniowej programu*** jest sprawdzenie poprawności składni programu i wygenerowanie komunikatów o błędach. ***Wszystkie błędy składniowe powinny zostać zasygnalizowane***.

Reguły przejścia od diagramu składni do programu

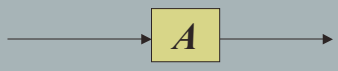
1. Zredukuj, na ile to możliwe, liczbę diagramów stosując odpowiednie podstawienia.
2. Każdy diagram zastąp deklaracja procedury zgodnie z regułami 3 - 7.
3. Element diagramu oznaczający symbol końcowy x



zastąp instrukcją:
if *ch = 'x'* **then** *read(ch)* **else** *błąd*;

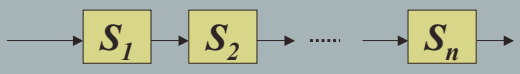
Reguły przejścia od diagramu składni do programu

4. Element diagramu oznaczający inny diagram



zastąp instrukcją wywołania procedury A.

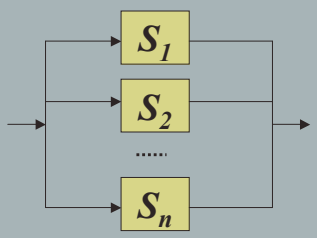
5. Ciąg elementów postaci



zastąp instrukcją złożoną:
begin $T(S_1)$; $T(S_2)$; ...; $T(S_n)$ **end**

Reguły przejścia od diagramu składni do programu

6. Diagram alternatywny zastąp instrukcją wyboru lub instrukcja warunkową:

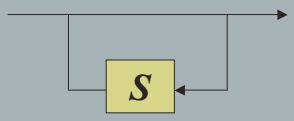



```
case ch of
        L1: T(S1);
        L2: T(S2);
        ...
        Ln: T(Sn)
end
```

if *ch* **in** L_1 **then** $T(S_1)$ **else**
if *ch* **in** L_2 **then** $T(S_2)$ **else**
…
if *ch* **in** L_n **then** *T(Sn)* **else** *błąd*;

Reguły przejścia od diagramu składni do programu

7. Pętlę postaci:



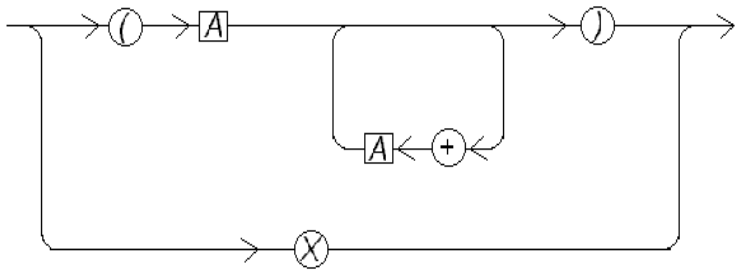
zastąp instrukcją:

while *ch* **in** *L* **do** *T(S);*

gdzie: *T(S)* otrzymano z *S* zgodnie z regułami 3 - 7, a zbiór *L* = *pierw(S)*

Przykład

Napisz program analizatora dla gramatyki:
A::= x | (B)
B::= AC
C::= {+A}




```
Program analizator;
var ch: char;
Procedure A;
begin
   if ch ='x' then read (ch) else
   if ch ='(' then
                  begin
                    read (ch); A;
                    while ch ='+' do
                                  begin
                                    read (ch); A;
                                  end;
                    if ch =')' then read (ch) else blad
                  end     else blad
end;
```

```
{program główny}
begin
   read (ch); A
end.
```

ANALIZATOR SKŁADNIOWY STEROWANY SKŁADNIĄ

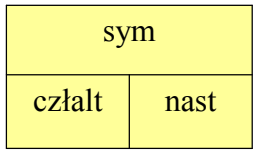
Przy budowie analizatora sterowanego składnią trzeba zaprojektować ***ogólny program rozbioru odpowiedni dla każdej gramatyki***.

Dla konkretnej gramatyki tworzona jest jej reprezentacja w postaci ***struktury danych dynamicznych***.

Przy ***analizie składniowej programu*** ogólny program rozbioru składniowego jest sterowany utworzoną strukturą danych dynamicznych.

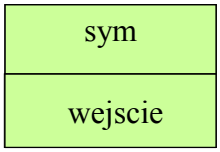
Węzły struktury danych

```
Type wskaznik =↑wezel;
   wezel = record
        nast,czlalt: wskaznik;
        case koncowy:boolean of
                true: (ksym: char);
                false: (psym:poczwsk)
   end
```



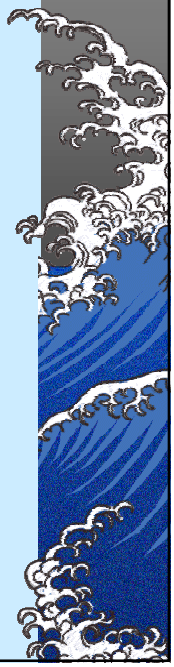
Węzeł początkowy

```
Type poczwsk =↑poczatek;
poczatek = record
   wejscie: wskaznik;
   sym: char
end
```



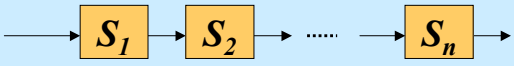
Reguły przejścia od diagramu składni do dynamicznej struktury danych

1. Zredukuj, na ile to możliwe, liczbę diagramów stosując odpowiednie podstawienia.
2. Każdy diagram zastąp odpowiednią strukturą danych zgodnie z regułami 3 - 6.
3. Element diagramu oznaczający symbol końcowy lub pomocniczy zastąp odpowiednim węzłem struktury danych.

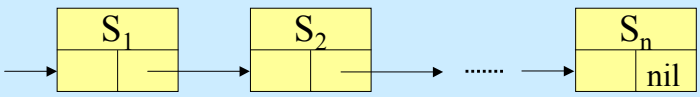


Reguły przejścia od diagramu składni do dynamicznej struktury danych

4. Ciąg elementów postaci

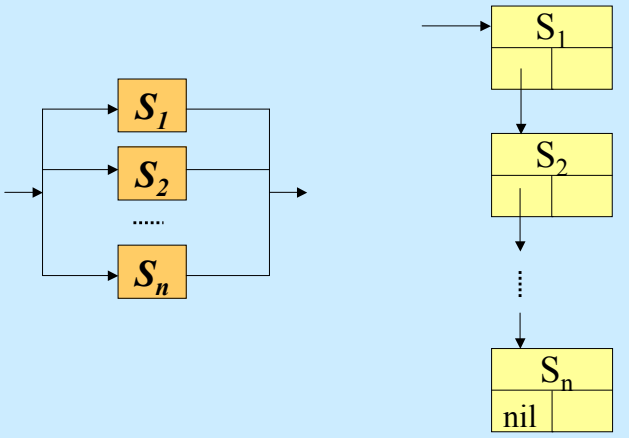


zastąp listą węzłów danych:



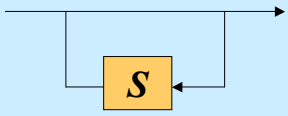
Reguły przejścia od diagramu składni do dynamicznej struktury danych

5. Diagram alternatywny zastąp strukturą danych:

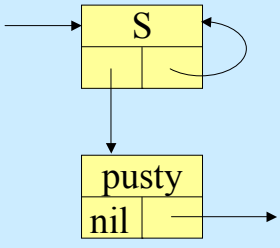


Reguły przejścia od diagramu składni do dynamicznej struktury danych

6. Pętlę postaci:

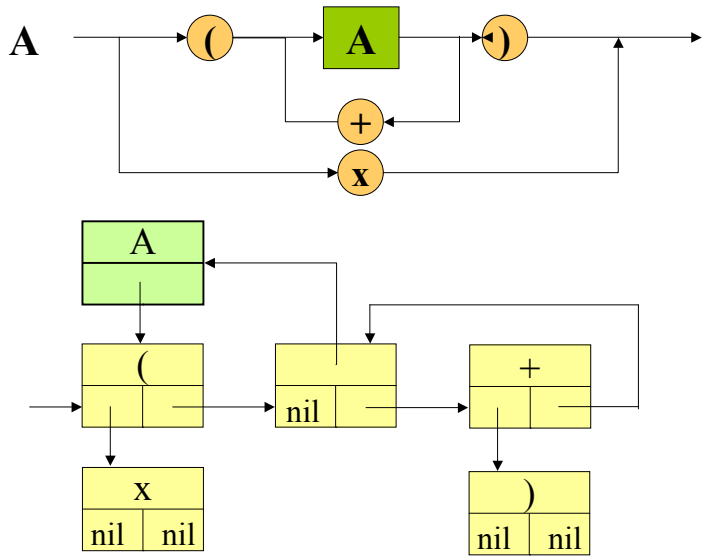


zastąp strukturą:



Przykład

Przekształć diagram w strukturę danych



Notacja RBNF

BNF	RBNF
::=	=
|	,
{	[
}	]

Każda produkcja w notacji RBNF zakończona jest kropką.

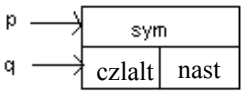
Gramatyka języka produkcji składniowych

Program analizatora akceptuje produkcje w notacji RBNF.

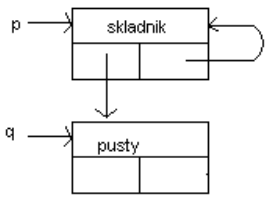
<produkcja> ::= <symbol> = <wyrażenie>.
<wyrażenie> ::= <składnik> {,<składnik>}
<składnik> ::= <czynnik> {<czynnik>}
<czynnik> ::= <symbol> | [<składnik>]

CZYNNIKI

<symbol>

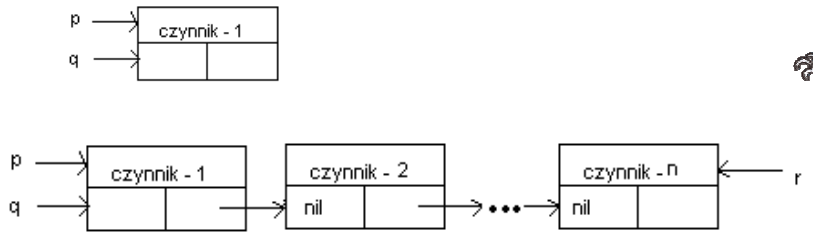


Połączenie symboli [<skladnik>]

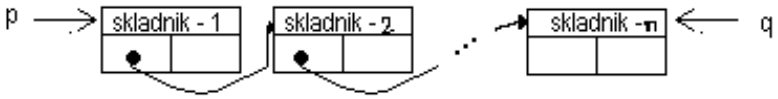


SKŁADNIKI

<czynnik-1>...<czynnik-n>



Wyrażenia



Analizator składniowy dla języka PL/0

Symbol pomocniczy	pierw (X)	nast (X)
Blok	const var procedure ident call begin if while	. ;
Instrukcja	ident call begin if while	. ; end
Warunek	odd + - (ident liczba	then do
Wyrażenie	+ - (ident liczba	. ; end then do R)
Składnik	(ident liczba	. ; end then do R) + -
Czynnik	(ident liczba	. ; end then do R) + - * /

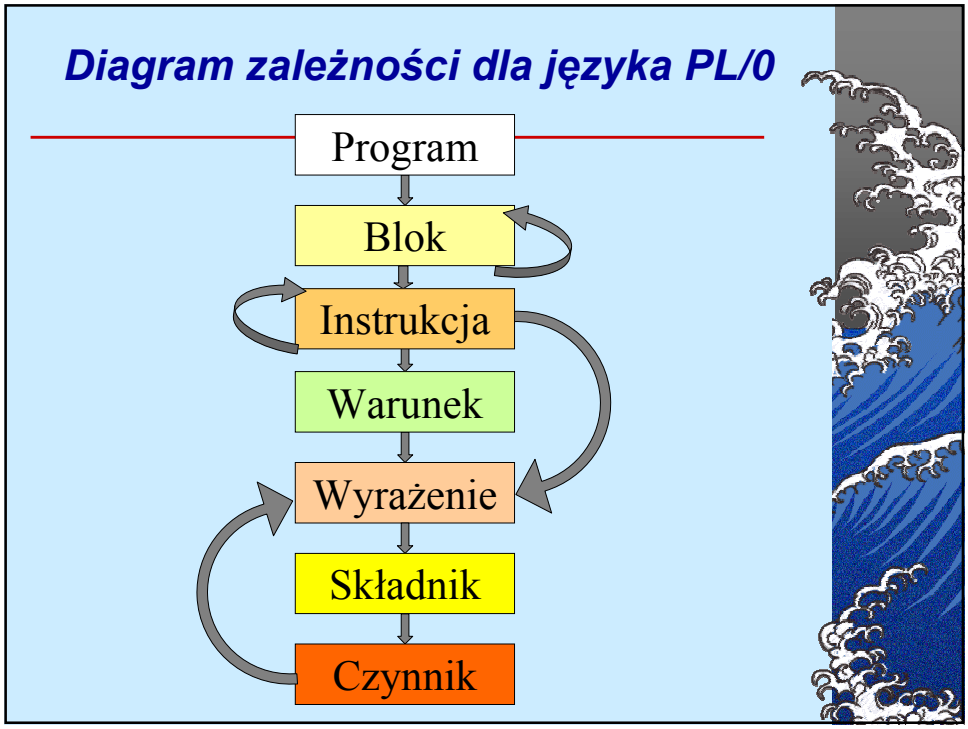
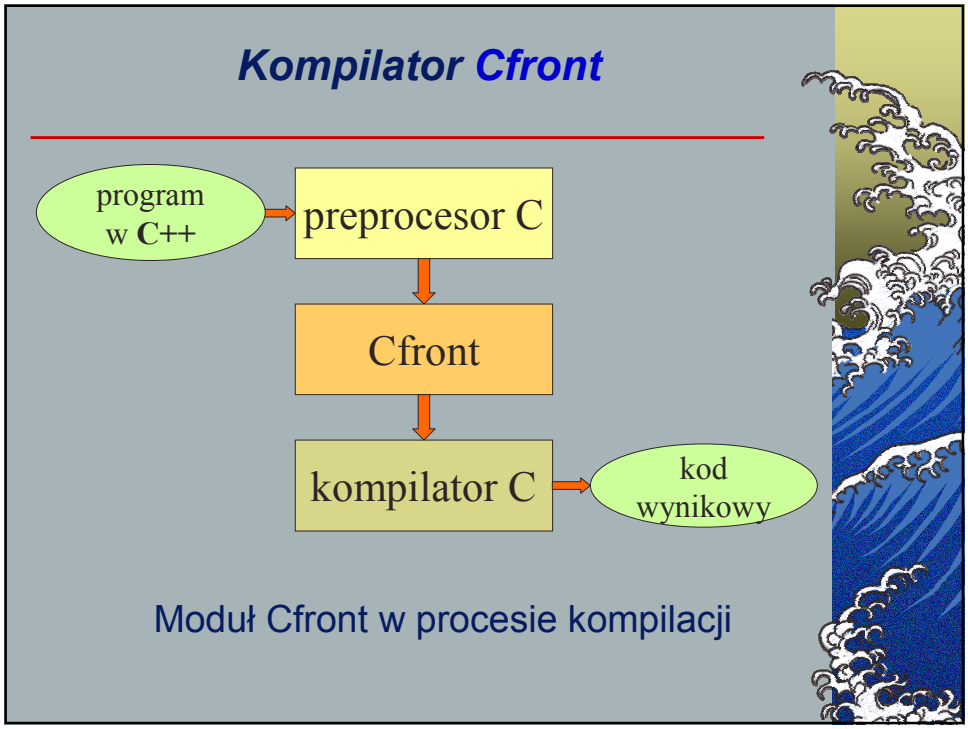
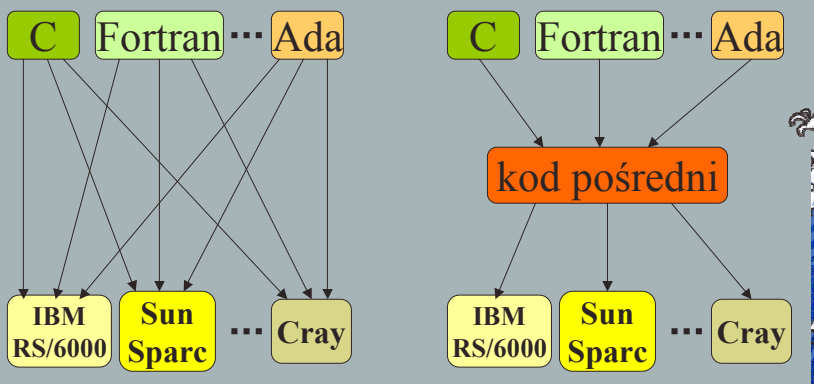


Diagram zależności dla języka PL/0
Program
Blok
Instrukcja
Warunek
Wyrażenie
Składnik
Czynnik


Kompilator Cfront
program
w C++
preprocesor C
Cfront
kompilator C
kod
wynikowy
Moduł Cfront w procesie kompilacji

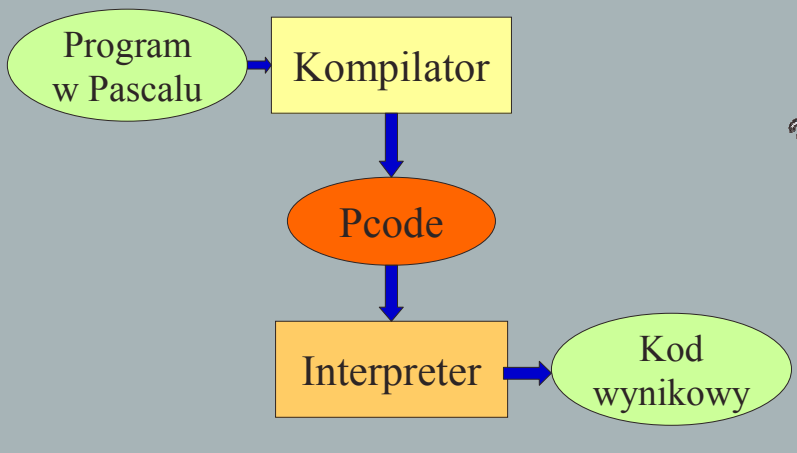
Kod pośredni w procesie kompilacji



Wybrane kody pośrednie

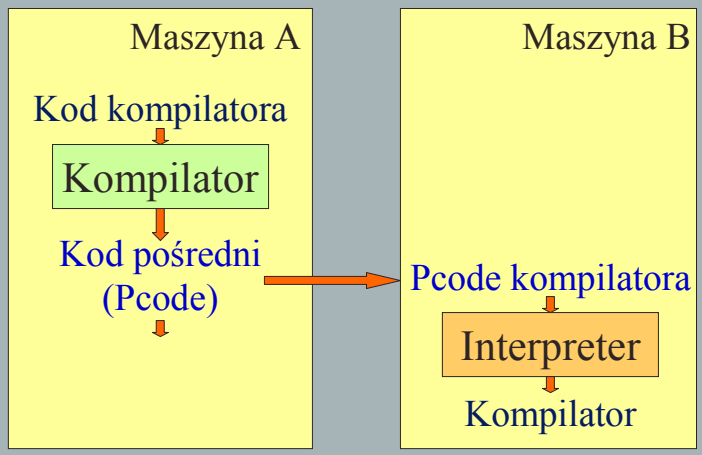
Kod źródłowy	Kod pośredni
Pascal	Pcode
Java	Java VM
Ada	Diana

Kod pośredni w procesie kompilacji



Kompilator Pascala używający Pcode

Przenośność kodu kompilatora



Przeniesienie kompilatora Pascala na inną platformę sprzętową przy użyciu Pcode

Gramatyki z translacją

- Gramatyka z translacją nazywamy gramatykę bezkontekstową, której zbiór terminali został uzupełniony dodatkowymi symbolami, nazywanymi symbolami translacji.
- Symbole translacji generują dodatkowe słowo wyjściowe obok słowa otrzymanego z gramatyki.

Przykład 1

Gramatyka generująca wyrażenia arytmetyczne

$E ::= T\ E_1$

$E_1 ::= +T\ E_1 \mid -T\ E_1 \mid \varepsilon$

$T ::= F\ T_1$

$T_1 ::= *\ F\ T_1 \mid /\ F\ T_1 \mid \varepsilon$

$F ::= -\ F \mid (E) \mid id$

$id ::= a \mid b \mid c$

Przykład 2

Gramatyka generująca wyrażenia arytmetyczne uzupełniona o translację na ONP

$E ::= T\ E_1$

$E_1 ::= +T\ \{+\}\ E_1 \mid -T\ \{-\}\ E_1 \mid \varepsilon$

$T ::= F\ T_1$

$T_1 ::= *\ F\ \{*\}\ T_1 \mid /\ F\ \{/\}\ T_1 \mid \varepsilon$

$F ::= -\ F\ \{-\} \mid (E) \mid id\ \{id\}$

$id ::= a \mid b \mid c$

PRZYKŁADY WYRAŻEŃ W NOTACJI WROSTKOWEJ I PRZYROSTKOWEJ (ONP – Odwrotna Notacja Polska)

Notacja wrostkowa	Notacja przyrostkowa (ONP)
x+y	*xy+*
(x–y)+z	*xy–z+*
x– (y+z)	*xyz+–*
x(y+z)*w*	*xyz+*w**