Stefan Sokołowski

JĘZYKI FORMALNE I METODY KOMPILACJI

wykład 0

Inst. Informatyki Stosowanej, ANS
Elbląg, 2024/2025

JĘZYKI FORMALNE — reguły gry

Zasadnicze informacje na:

http://iis.ans-elblag.pl/~stefan/Dydaktyka/JezForm

JĘZYKI FORMALNE — reguły gry

Zasadnicze informacje na:

http://iis.ans-elblag.pl/~stefan/Dydaktyka/JezForm

między innymi:

- szkicowy **program** wykładu i laboratorium

JĘZYKI FORMALNE — reguły gry

Zasadnicze informacje na:

http://iis.ans-elblag.pl/~stefan/Dydaktyka/JezForm

między innymi:

- szkicowy **program** wykładu i laboratorium
- **slajdy** do wykładów — można (należy) robić notatki
  ale **nie warto przepisywać slajdów** z ekranu

JĘZYKI FORMALNE — reguły gry

Zasadnicze informacje na:

http://iis.ans-elblag.pl/~stefan/Dydaktyka/JezForm

między innymi:

- szkicowy **program** wykładu i laboratorium
- **slajdy** do wykładów — można (należy) robić notatki
  ale **nie warto przepisywać slajdów** z ekranu
- **zadania** z laboratorium

JĘZYKI FORMALNE — reguły gry

Zasadnicze informacje na:

http://iis.ans-elblag.pl/~stefan/Dydaktyka/JezForm

między innymi:

- szkicowy **program** wykładu i laboratorium
- **slajdy** do wykładów — można (należy) robić notatki
  ale **nie warto przepisywać slajdów** z ekranu
- **zadania** z laboratorium
- pomocnicze **programy**

JĘZYKI FORMALNE — reguły gry

Zasadnicze informacje na:

http://iis.ans-elblag.pl/~stefan/Dydaktyka/JezForm

między innymi:

- szkicowy **program** wykładu i laboratorium
- **slajdy** do wykładów — można (należy) robić notatki
  ale **nie warto przepisywać slajdów** z ekranu
- **zadania** z laboratorium
- pomocnicze **programy**
- spis **literatury**
- itd.

JĘZYKI FORMALNE — reguły gry

Zasadnicze informacje na:

http://iis.ans-elblag.pl/~stefan/Dydaktyka/JezForm

między innymi:

- szkicowy **program** wykładu i laboratorium
- **slajdy** do wykładów — można (należy) robić notatki
  ale **nie warto przepisywać slajdów** z ekranu
- **zadania** z laboratorium
- pomocnicze **programy**
- spis **literatury**
- itd.

Uwaga: przeczytanie tych materiałów nie wystarczy do zaliczenia!

JĘZYKI FORMALNE — reguły gry

Kontakt ze mną:

konsultacje: poniedziałki, 16:45–17:30, w pok. 322 (Wojska P.)

JĘZYKI FORMALNE — reguły gry

Kontakt ze mną:

konsultacje: poniedziałki, 16:45–17:30, w pok. 322 (Wojska P.)

e-mail: `s.sokolowski@ans-elblag.pl`

antyspam: unikać frikoprowajderów;

najlepiej używać konta mailowego w **ANS PWSZ**

JĘZYKI FORMALNE — reguły gry

Kontakt ze mną:

konsultacje: poniedziałki, 16:45–17:30, w pok. 322 (Wojska P.)

e-mail: `s.sokolowski@ans-elblag.pl`

antyspam: unikać frikoprowajderów;

najlepiej używać konta mailowego w **ANS PWSZ**

Wymaganie wstępne:

- zaliczone **Podstawy programowania**
- ogólna bystrość inżynierska

JĘZYKI FORMALNE — reguły gry

Zaliczenie przedmiotu

JĘZYKI FORMALNE — reguły gry

Zaliczenie przedmiotu:

laboratorium:

min. 56 pkt. na 100

JĘZYKI FORMALNE — reguły gry

Zaliczenie przedmiotu:

laboratorium:

min. 56 pkt. na 100 ← *niezapowiedziane krótkie sprawdziany*

JĘZYKI FORMALNE — reguły gry

Zaliczenie przedmiotu:

laboratorium:

min. 56 pkt. na 100 ← *niezapowiedziane krótkie sprawdziany ogłoszone zadania domowe*

JĘZYKI FORMALNE — reguły gry

Zaliczenie przedmiotu:

laboratorium:

min. 56 pkt. na 100 ← *niezapowiedziane krótkie sprawdziany ogłoszone zadania domowe*

wykład:

min. 56 pkt. na 100 ← *zal. pisemne w sesji*

JĘZYKI FORMALNE — reguły gry

Zaliczenie przedmiotu:

laboratorium:

min. 56 pkt. na 100 ← *niezapowiedziane krótkie sprawdziany ogłoszone zadania domowe*

wykład:

min. 56 pkt. na 100 ← *zal. pisemne w sesji*

---

ocena ostateczna:

$0.6 \times$ laboratorium $+ 0.4 \times$ wykład

JĘZYKI FORMALNE — reguły gry

Zaliczenie przedmiotu:

laboratorium:

min. 56 pkt. na 100 ← *niezapowiedziane krótkie sprawdziany ogłoszone zadania domowe*

wykład:

min. 56 pkt. na 100 ← *zal. pisemne w sesji*

---

ocena ostateczna:

$0.6 \times$ laboratorium $+ 0.4 \times$ wykład

trzeba zaliczyć osobno laboratorium i osobno wykład — dopiero wtedy oblicza się ocenę ostateczną

JĘZYKI FORMALNE — reguły gry

Zaliczenie przedmiotu:

laboratorium:

min. 56 pkt. na 100 ← *niezapowiedziane krótkie sprawdziany ogłoszone zadania domowe*

wykład:

min. 56 pkt. na 100 ← *zal. pisemne w sesji*

---

ocena ostateczna:

$0.6 \times$ laboratorium $+ 0.4 \times$ wykład

trzeba zaliczyć osobno laboratorium i osobno wykład — dopiero wtedy oblicza się ocenę ostateczną

NIE MA ZWOLNIEŃ!

JĘZYKI FORMALNE — reguły gry

Zaliczenie przedmiotu:

laboratorium:

min. 56 pkt. na 100 ← *niezapowiedziane krótkie sprawdziany ogłoszone zadania domowe*

wykład:

min. 56 pkt. na 100 ← *zal. pisemne w sesji*

---

ocena ostateczna:

$0.6 \times$ laboratorium $+ 0.4 \times$ wykład

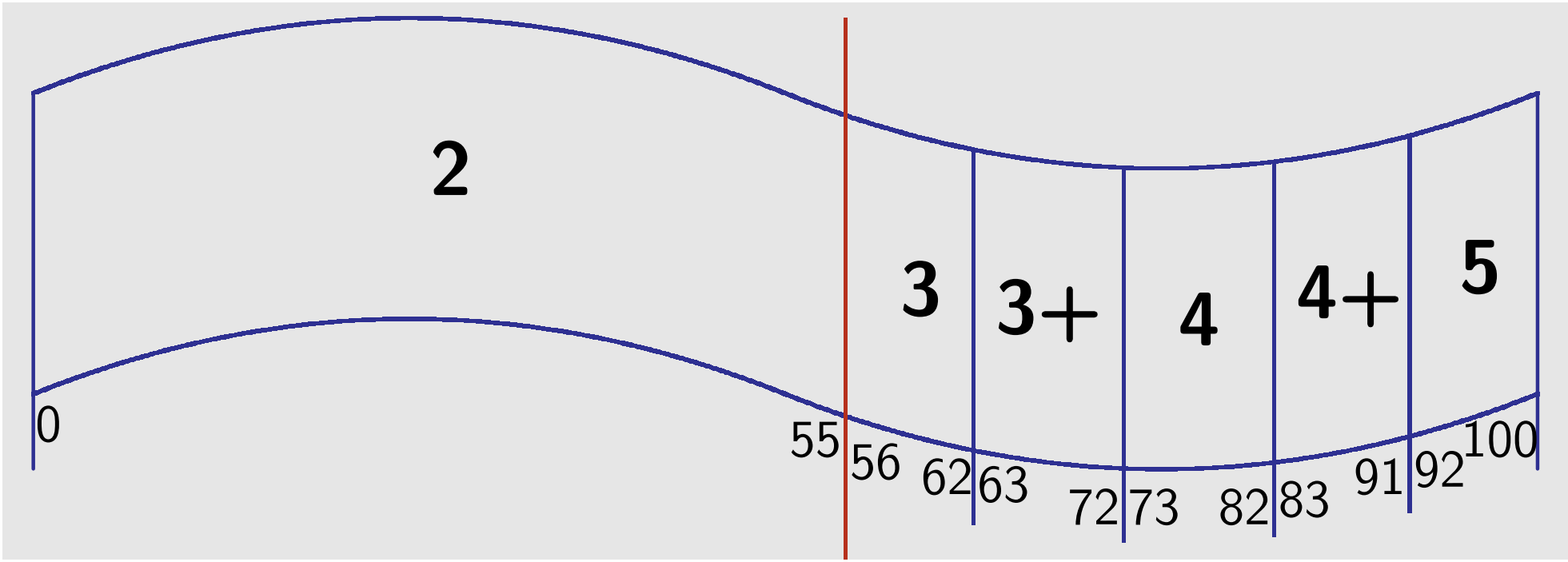
trzeba zaliczyć osobno laboratorium i osobno wykład — dopiero wtedy oblicza się ocenę ostateczną

NIE MA ZWOLNIEŃ!

— ale doliczam ew. punkty dodatkowe za aktywność, za znalezienie błędów w slajdach i za zadania dodatkowe

JĘZYKI FORMALNE — reguły gry

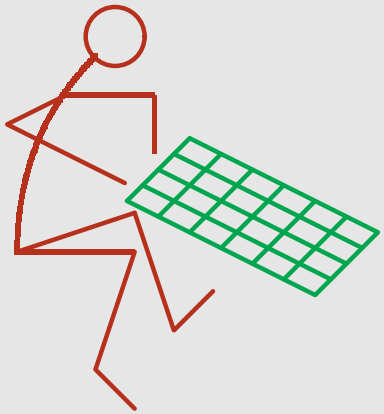
Skala ocen:



O czym będzie?

O czym będzie?

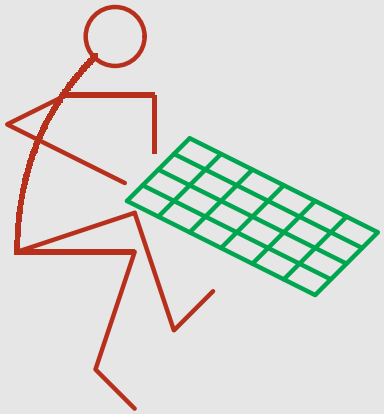
programista



pisze program w jęz.
wys. poziomu

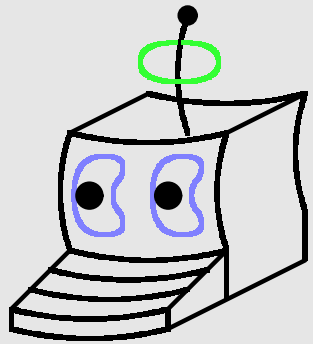
O czym będzie?

programista



pisze program w jęz.
wys. poziomu

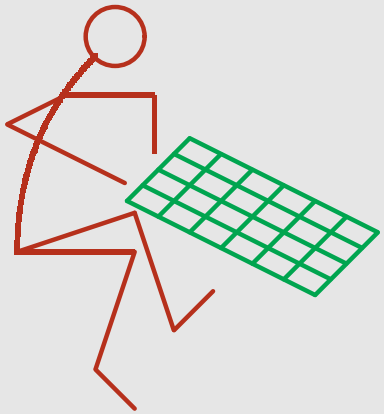
komputer



rozumie tylko swój
jęz. wewnętrzny

O czym będzie?

programista

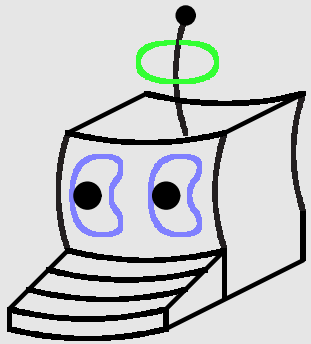


pisze program w jęz.
wys. poziomu

trzeba

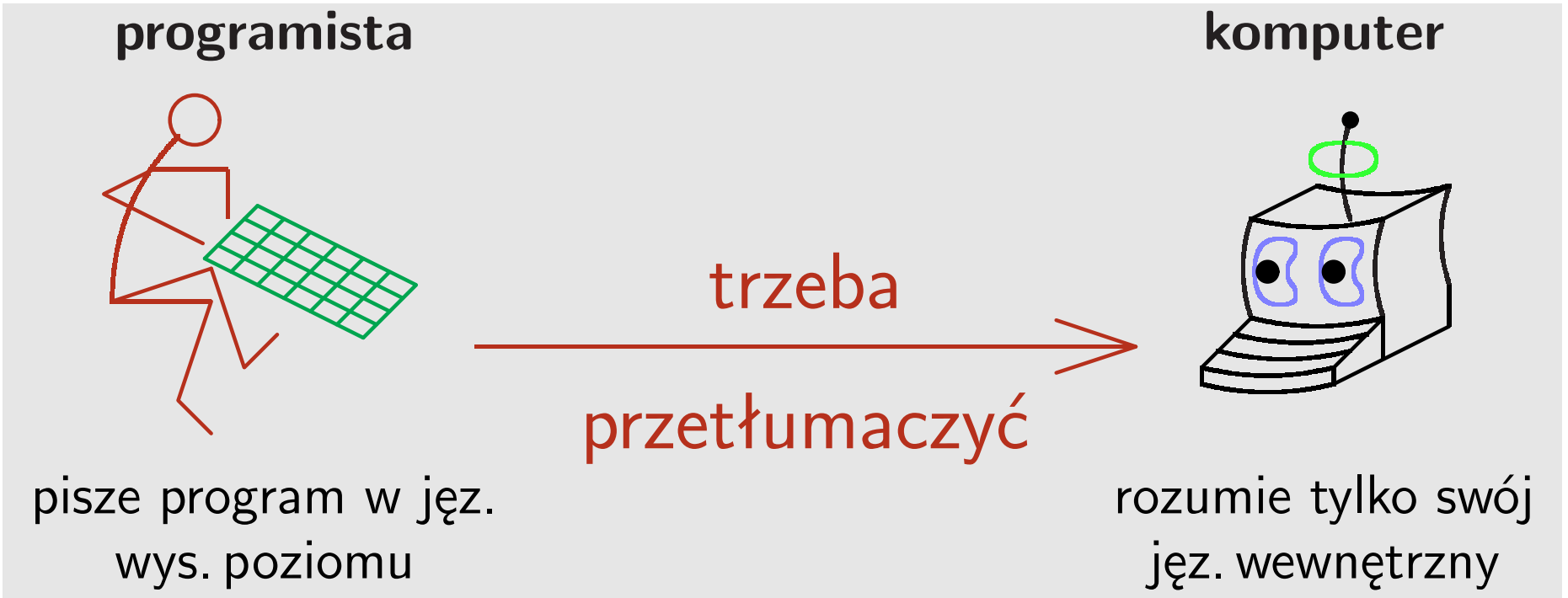
przetłumaczyć

komputer



rozumie tylko swój
jęz. wewnętrzny

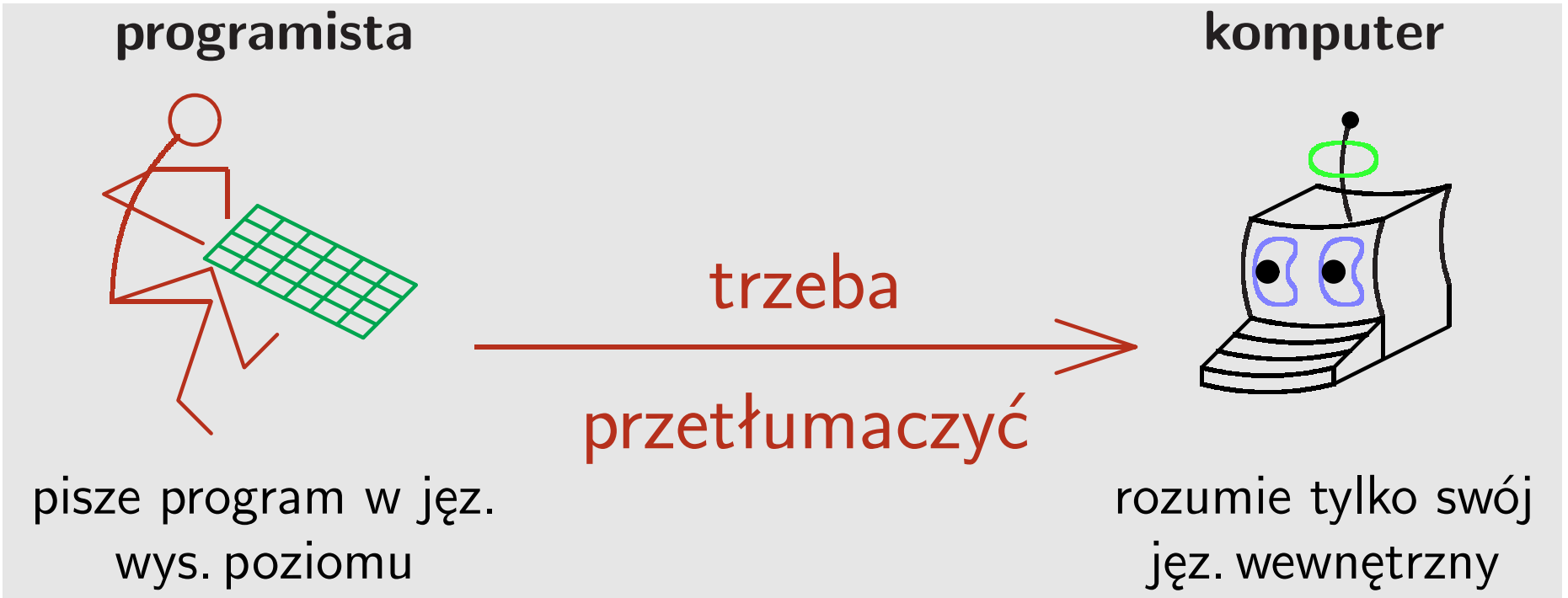
O czym będzie?



Kompilacja — automatyczny przekład programów

- z języka programowania *wysokiego poziomu* (np. C, Java, . . .)
- na język *wewnętrzny* komputera

O czym będzie?



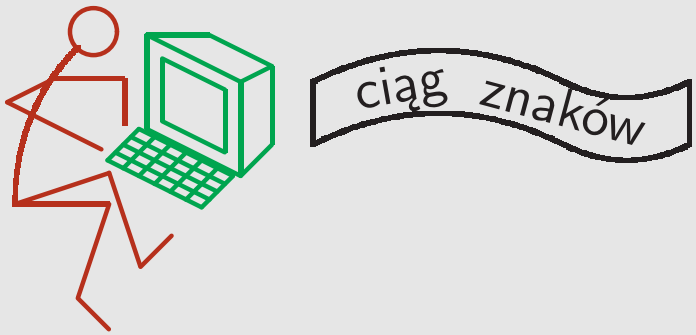
Kompilacja — automatyczny przekład programów

- z języka programowania *wysokiego poziomu* (np. C, Java, . . .)
- na język *wewnętrzny* komputera

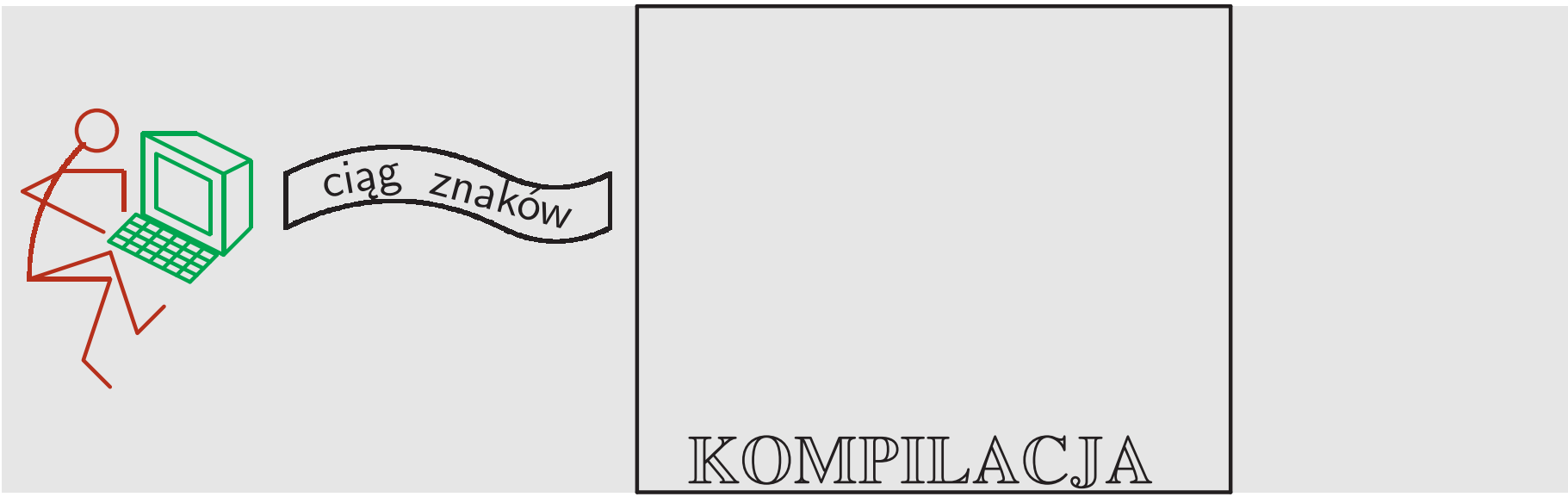
Kompilator musi „znać” oba języki. . .

Etapy kompilacji

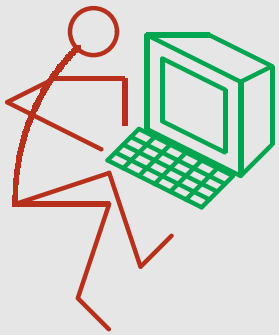
Etapy kompilacji



Etapy kompilacji

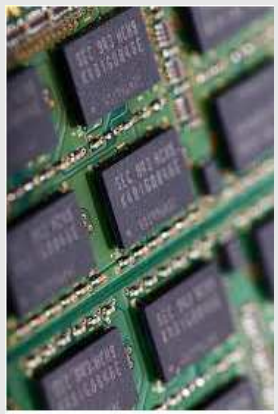


Etapy kompilacji



ciąg znaków

KOMPILACJA

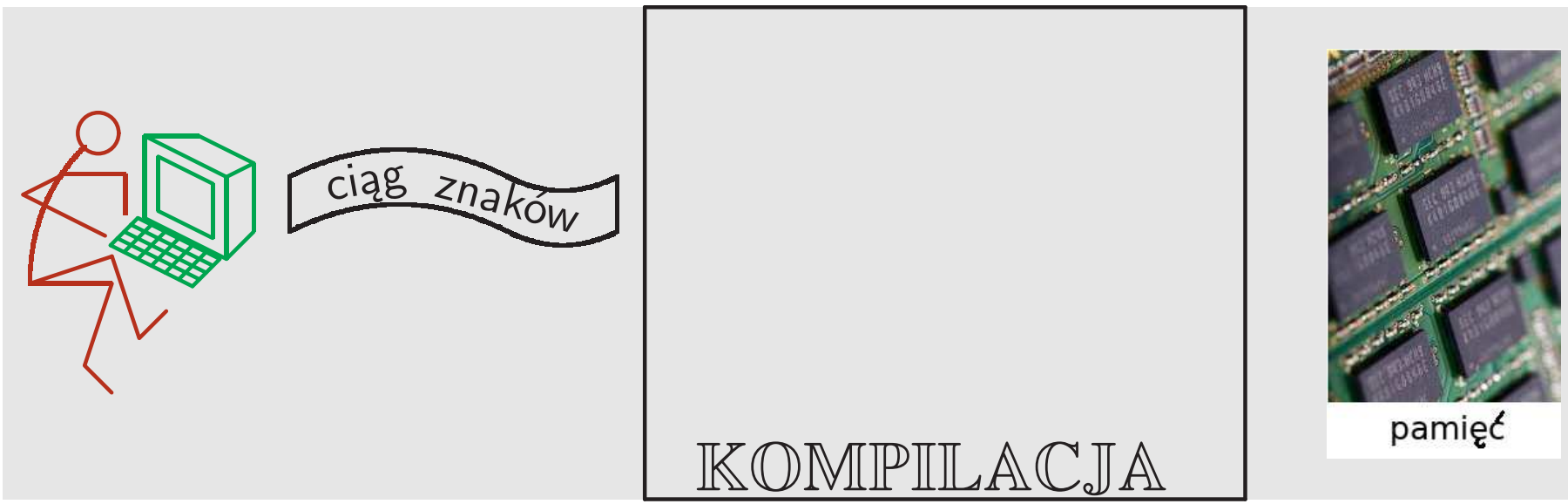


pamięć

Etapy kompilacji

Kompilator musi

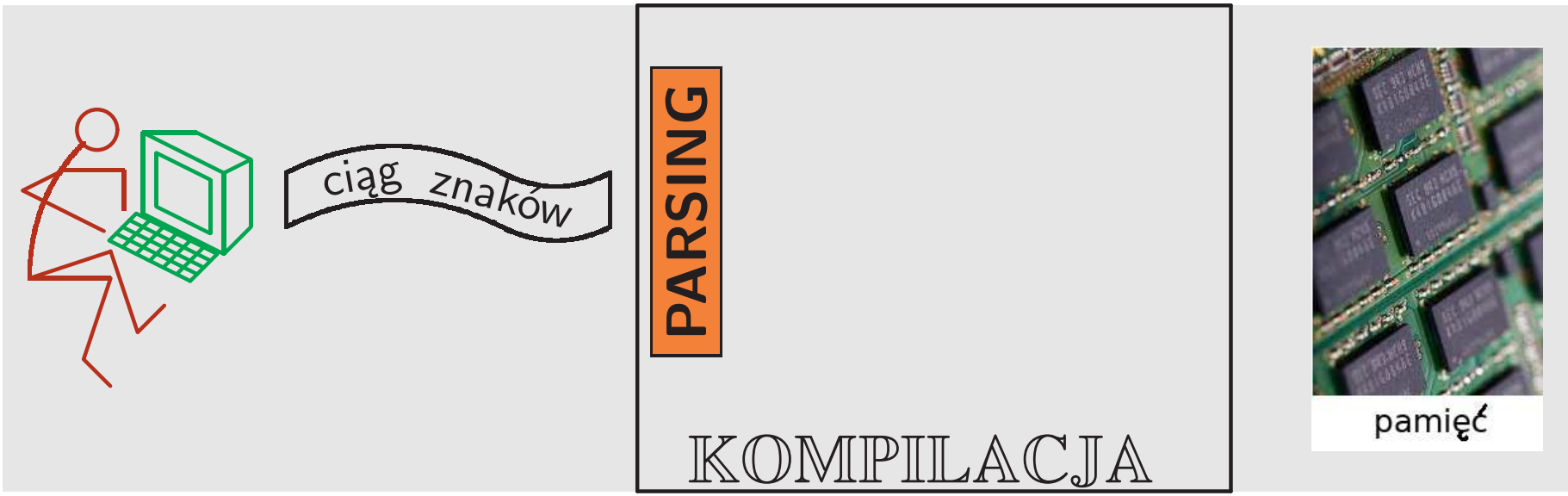
- „zrozumieć” tekst programu napisanego w jęz. wysokiego poziomu (oczyścić z el. nieistotnych, nadać strukturę, ustalić znaczenie części składowych)



Etapy kompilacji

Kompilator musi

- „zrozumieć” tekst programu napisanego w jęz. wysokiego poziomu (oczyścić z el. nieistotnych, nadać strukturę, ustalić znaczenie części składowych)

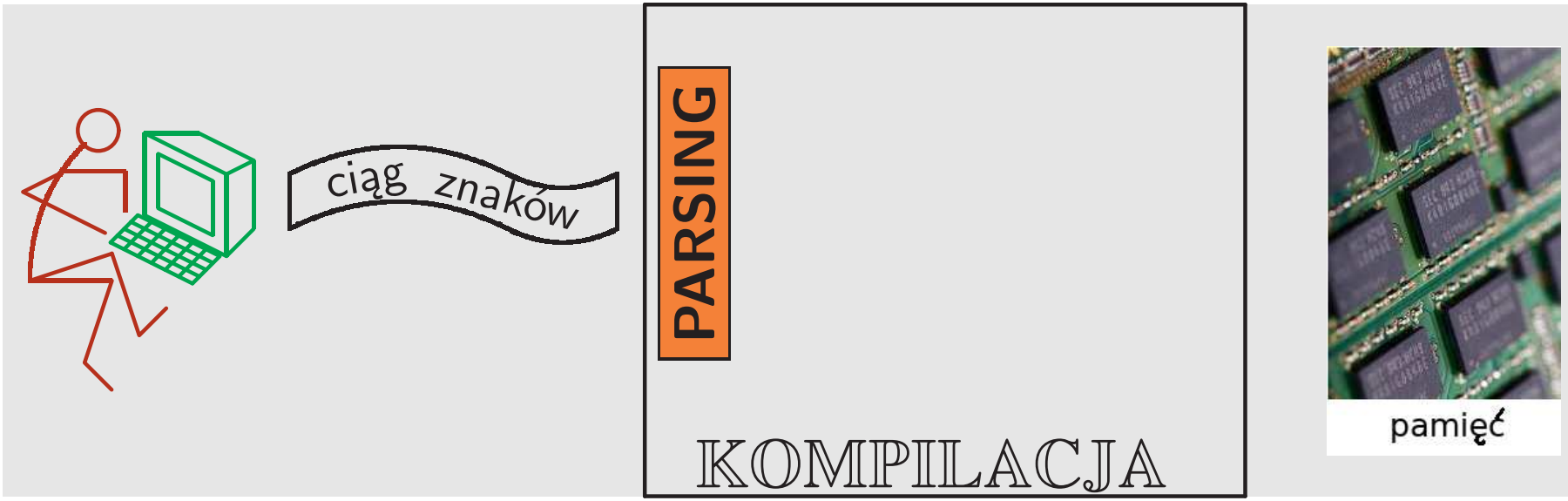


Etapy kompilacji

Kompilator musi

- „zrozumieć” tekst programu napisanego w jęz. wysokiego poziomu (oczyścić z el. nieistotnych, nadać strukturę, ustalić znaczenie części składowych)

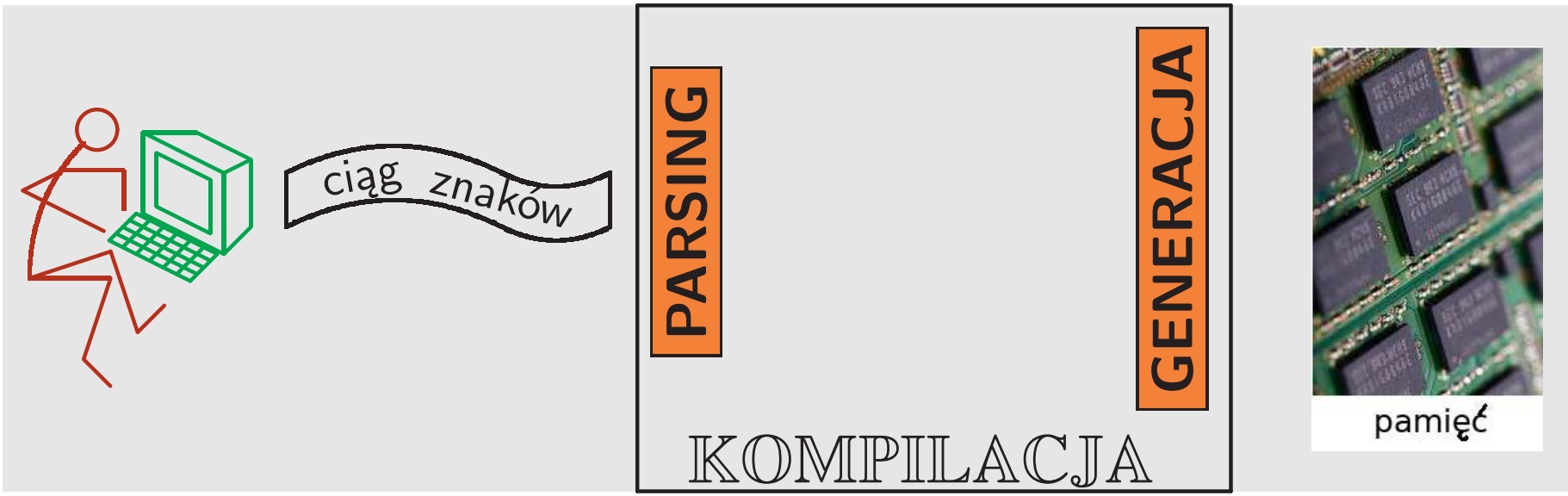
- zapisać w pamięci tak samo działający program w jęz. wewnętrznym



Etapy kompilacji

Kompilator musi

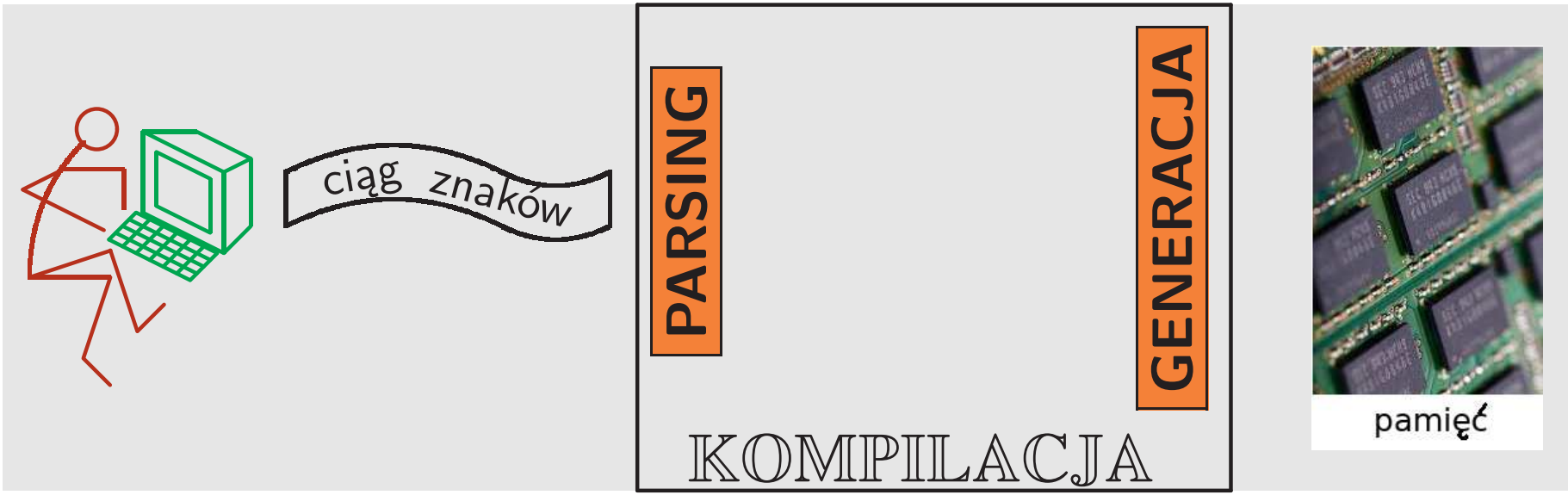
- „zrozumieć” tekst programu napisanego w jęz. wysokiego poziomu (oczyścić z el. nieistotnych, nadać strukturę, ustalić znaczenie części składowych)
- zapisać w pamięci tak samo działający program w jęz. wewnętrznym



Etapy kompilacji

Kompilator musi

- „zrozumieć” tekst programu napisanego w jęz. wysokiego poziomu (oczyścić z el. nieistotnych, nadać strukturę, ustalić znaczenie części składowych)
- zapisać w pamięci tak samo działający program w jęz. wewnętrznym



parser = analizator składniowy = analizator syntaktyczny

Etapy kompilacji

Kompilator musi

- „zrozumieć” tekst programu napisanego w jęz. wysokiego poziomu (oczyścić z el. nieistotnych, nadać strukturę, ustalić znaczenie części składowych)
- zapisać w pamięci tak samo działający program w jęz. wewnętrznym

parser = analizator składniowy = analizator syntaktyczny
— z 1-wymiarowego ciągu znaków tworzy *drzewo rozbioru*

Etapy kompilacji

Kompilator musi

- „zrozumieć” tekst programu napisanego w jęz. wysokiego poziomu (oczyścić z el. nieistotnych, nadać strukturę, ustalić znaczenie części składowych)
- zapisać w pamięci tak samo działający program w jęz. wewnętrznym

parser = analizator składniowy = analizator syntaktyczny

— z 1-wymiarowego ciągu znaków tworzy *drzewo rozbioru = drzewo wywodu*

Drzewo rozbioru

Drzewo rozbioru obrazuje strukturę logiczną programu — co z czym jest silniej związane, co jest nad czym nadrzędne, itp.

Drzewo rozbioru

Drzewo rozbioru obrazuje strukturę logiczną programu — co z czym jest silniej związane, co jest nad czym nadrzędne, itp.

Przykład:

wyrażenie arytmetyczne

$$7 + 6 * 12$$

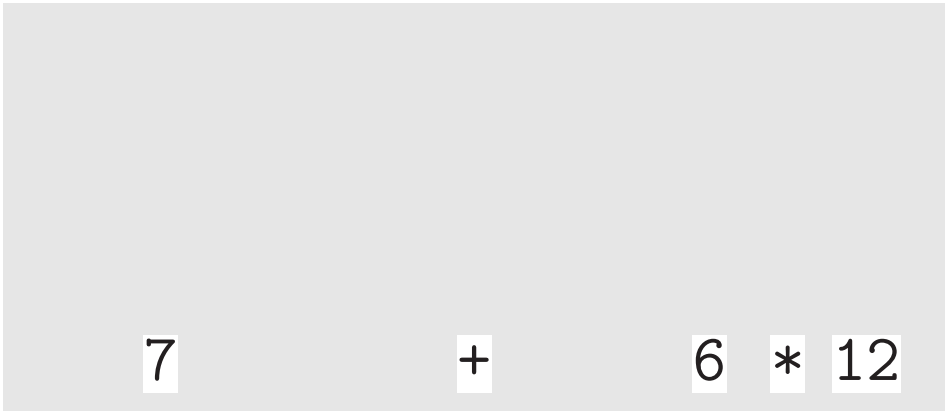
Drzewo rozbioru

Drzewo rozbioru obrazuje strukturę logiczną programu — co z czym jest silniej związane, co jest nad czym nadrzędne, itp.

Przykład:

wyrażenie arytmetyczne

$7 + 6 * 12$

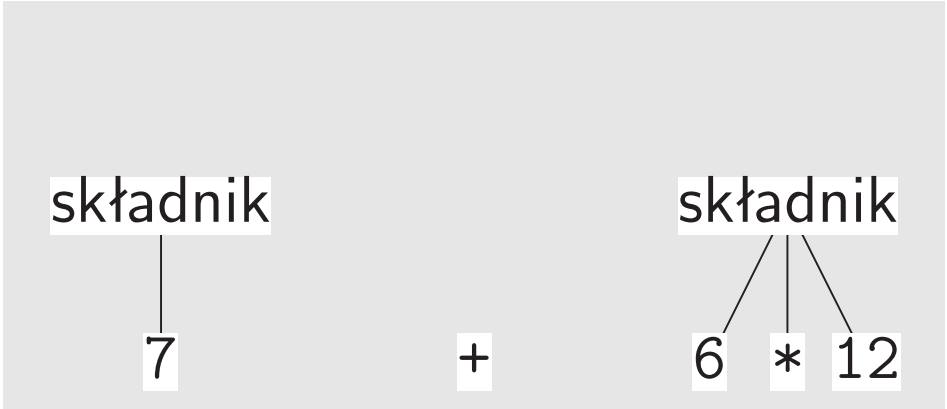


Drzewo rozbioru

Drzewo rozbioru obrazuje strukturę logiczną programu — co z czym jest silniej związane, co jest nad czym nadrzędne, itp.

Przykład:

wyrażenie arytmetyczne
$7 + 6 * 12$

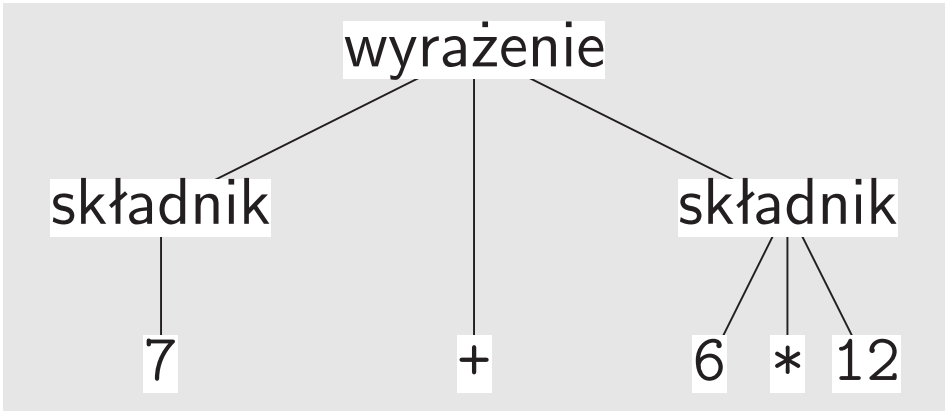


Drzewo rozbioru

Drzewo rozbioru obrazuje strukturę logiczną programu — co z czym jest silniej związane, co jest nad czym nadrzędne, itp.

Przykład:

wyrażenie arytmetyczne
$7 + 6 * 12$



Drzewo rozbioru

Drzewo rozbioru obrazuje strukturę logiczną programu — co z czym jest silniej związane, co jest nad czym nadrzędne, itp.

Przykład:

wyrażenie arytmetyczne
$7 + 6 * 12$

Przykład:

wyrażenie arytmetyczne
$(7 + 6) * 12$

Drzewo rozbioru

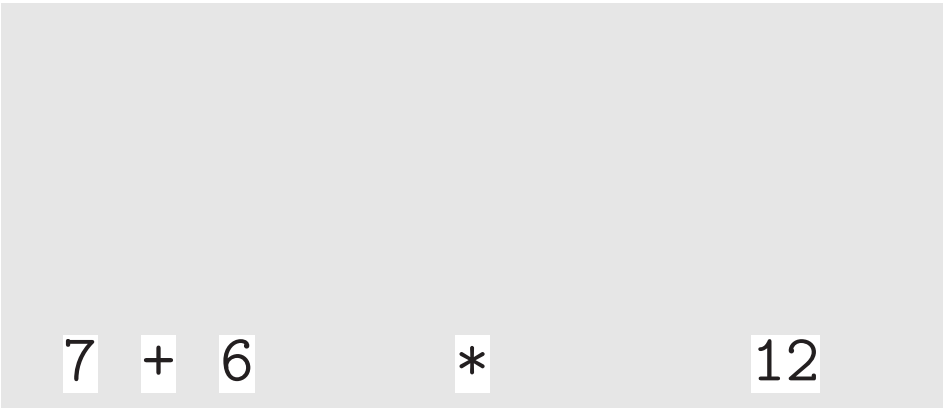
Drzewo rozbioru obrazuje strukturę logiczną programu — co z czym jest silniej związane, co jest nad czym nadrzędne, itp.

Przykład:

wyrażenie arytmetyczne
$7 + 6 * 12$

Przykład:

wyrażenie arytmetyczne
$(7 + 6) * 12$

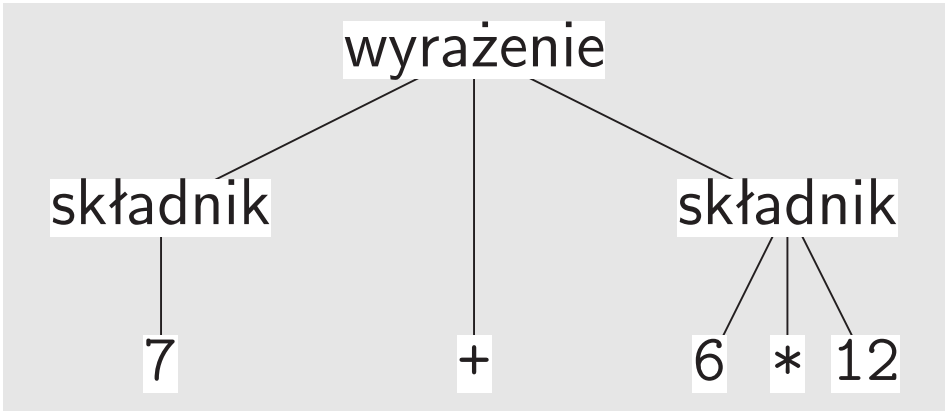


Drzewo rozbioru

Drzewo rozbioru obrazuje strukturę logiczną programu — co z czym jest silniej związane, co jest nad czym nadrzędne, itp.

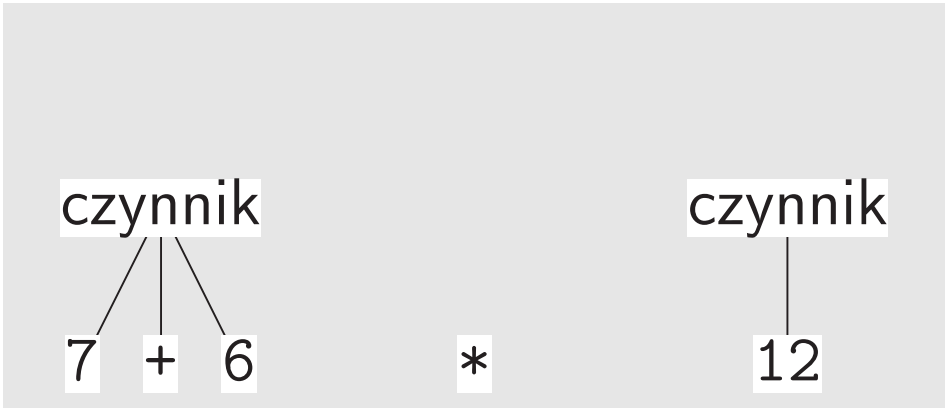
Przykład:

wyrażenie arytmetyczne
$7 + 6 * 12$



Przykład:

wyrażenie arytmetyczne
$(7 + 6) * 12$



Drzewo rozbioru

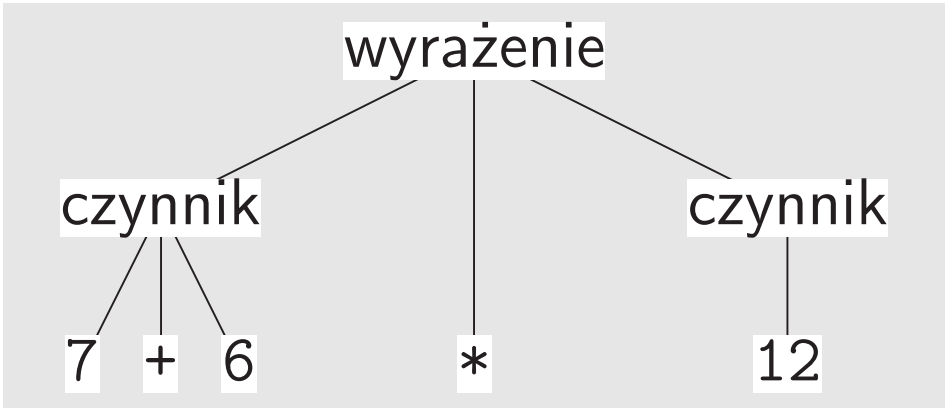
Drzewo rozbioru obrazuje strukturę logiczną programu — co z czym jest silniej związane, co jest nad czym nadrzędne, itp.

Przykład:

wyrażenie arytmetyczne
$7 + 6 * 12$

Przykład:

wyrażenie arytmetyczne
$(7 + 6) * 12$



Drzewo rozbioru

Budowanie drzewa rozbioru (parsing) jest najważniejszym etapem kompilacji. Parser jest głównym segmentem kompilatora.

to parse =
to examine closely,
esp. by breaking into components

Drzewo rozbioru

Budowanie drzewa rozbioru (parsing) jest najważniejszym etapem kompilacji. Parser jest głównym segmentem kompilatora.

to parse =
to examine closely,
esp. by breaking into components

Języki formalne — sposoby takiego przedstawienia języka, żeby umożliwić jego automatyczne przetwarzanie.

Drzewo rozbioru

Budowanie drzewa rozbioru (parsing) jest najważniejszym etapem kompilacji. Parser jest głównym segmentem kompilatora.

to parse =
to examine closely,
esp. by breaking into components

Języki formalne — sposoby takiego przedstawienia języka, żeby umożliwić jego automatyczne przetwarzanie.

Wiedza z języków formalnych przydaje się (oprócz kompilacji) do analizy danych tekstowych różnych rodzajów.

Uproszczony program wykładu

Uproszczony program wykładu

1. **PARSER** —
    - języki, dla których daje się zbudować

Uproszczony program wykładu

1. **PARSER** —
   - języki, dla których daje się zbudować
   - jak parser konstruuje drzewo

Uproszczony program wykładu

1. **PARSER** —
   - języki, dla których daje się zbudować
   - jak parser konstruuje drzewo

   } języki formalne

Uproszczony program wykładu

1. **PARSER** —
   - języki, dla których daje się zbudować
   - jak parser konstruuje drzewo
   - z czego parser się składa

} języki formalne

Uproszczony program wykładu

1. **PARSER** —
   - języki, dla których daje się zbudować
   - jak parser konstruuje drzewo

   } języki formalne

   - z czego parser się składa
   - rodzaje parserów

Uproszczony program wykładu

1. **PARSER** —
   - języki, dla których daje się zbudować
   - jak parser konstruuje drzewo
   - z czego parser się składa
   - rodzaje parserów

   } języki formalne

2. **JĘZYK WEWNĘTRZNY** —
   - co komputer umie sam z siebie

Uproszczony program wykładu

1. **PARSER** —
   - języki, dla których daje się zbudować
   - jak parser konstruuje drzewo

   } języki formalne
   - z czego parser się składa
   - rodzaje parserów
2. **JĘZYK WEWNĘTRZNY** —
   - co komputer umie sam z siebie
   - programowanie na najniższym poziomie

Uproszczony program wykładu

1. **PARSER** —
   - języki, dla których daje się zbudować
   - jak parser konstruuje drzewo
   - z czego parser się składa
   - rodzaje parserów

   } języki formalne

2. **JĘZYK WEWNĘTRZNY** —
   - co komputer umie sam z siebie
   - programowanie na najniższym poziomie

3. **PRZEKŁAD** —
   - zarządzanie pamięcią
   - programowanie przekładu

Uproszczony program wykładu

1. **PARSER** —
   - języki, dla których daje się zbudować
   - jak parser konstruuje drzewo

   } języki formalne
   - z czego parser się składa
   - rodzaje parserów
2. **JĘZYK WEWNĘTRZNY** —
   - co komputer umie sam z siebie
   - programowanie na najniższym poziomie
3. **PRZEKŁAD** —
   - zarządzanie pamięcią
   - programowanie przekładu
4. **NARZĘDZIA DO BUDOWY KOMPILATORÓW** —
   - FLEX
   - BISON