Metody kompilacji

Laboratorium 4

Piotr Błaszyński

18 marca 2019

Zadania (wyjaśnienie w dalszej części dokumentu):

- zmodyfikować pliki źródłowe i plik Makefile, tak, by analizator składniowy i cały kompilator był kompilowany przy pomocy kompilatora C++,
- zmodyfikować gramatykę tak, by można było kompilować program składający się z wielu wyrażeń (wielu linii),
- dopisać kod zapisujący trójki do pliku.

Modyfikacja w celu umożliwienia kompilacji przy pomocy C++:

- zmieniamy nazwę pliku def.y na def.yy (dzięki temu bison wygeneruje nam pliki def.tab.cc i def.tab.hh),
- w pliku Makefile:
  - zmieniamy odwołania do def.y na def.yy,
  - zmieniamy odwołania do def.tab.c na def.tab.cc,
  - dodajemy na początku nową zmienną: CPP=g++
  - zamieniamy w dwóch miejscach odwołanie do CC na CPP
    - kompilacja def.tab.cc
    - kompilacja całości (analizator leksykalny nadal kompilujemy przy pomocy kompilatora C),
- dodać opcję kompilacji -std=c++11 do Makefile (w obu miejscach gdzie używamy g++),
- do pliku def.yy dopisujemy w sekcji nagłówkowej:

  - extern "C" int yylex();
  - extern "C" int yyerror(const char *msg, ...);
  - using namespace std;
  - jeżeli mamy deklaracje *yyin* i *yyout* to dopisujemy do nich również *extern* (bez C),

- w pliku zx.l zmieniamy:
  - na początku: zmieniamy dołączany plik nagłówkowy (def.tab.h na def.tab.hh),
  - na początku: int yyerror(const char *msg, ...); (tak naprawdę zmieniamy nagłówek funkcji yyerror - dodajemy const)
  - na końcu: int yyerror(const char *msg, ...){ (tak naprawdę zmieniamy nagłówek funkcji yyerror - dodajemy const)

Dla zdefiniowania gramatyki do obsługi wielu wyrażeń można się posłużyć analogią do wyrażeń arytmetycznych, które również mogą być dowolnie długie.

Jako trójki będziemy na tym etapie traktować wyrażenia składające się ze zmiennej *result* (w przyszłości numerowanej) oraz dwóch argumentów i operatora. Na przykład dla wyrażenia $a = b + c * 9$; uzyskamy trzy trójki:

- $result = c\ 9\ *$
- $result = b\ result\ +$
- $result = a\ result\ =$

To tylko przykładowa forma zapisu, różne kompilatory robią to na swoje sposoby. Aby uzyskać taką formę zapisu należy liczby i identyfikatory odkładać na stosie (ten stos powinien przechowywać obiekty jakiejś struktury/klasy - oprócz wartości należy przechowywać również typ). Natomiast w akcjach semantycznych:

- dla operatorów arytmetycznych należy ściągać ze stosu odpowiednią liczbę argumentów (najczęściej 2),
- zapisać na stos zmienną przechowującą wynik,
- wszystkie te elementy (argumenty, operator i zmienna wyniku) zapisać do pliku.

Przypomnienie: klasa *std* :: *stack* ma metody:

- *push* - wstawienie wartości na szczyt stosu,
- *top* - pobranie wartości ze szczytu stosu,
- *pop* - usunięcie wartości ze szczytu stosu.