T* reprezinta

concatenarea tuturor sirurilor din multimea T

Compilatorul este

o aplicatie software care translateaza un program scris intr un limbaj de programare intr-o forma executabila de catre calculator

Identificatorii utilizati in programul-sursa sunt stocati

in tabela de simboluri

Atomii lexicali sunt caracterizati prin

tip si valoare

Arborele de derivare este

reprezentarea grafica a unei secvente de derivari

Outputul unui compilator consta din

fisier(e)-obiect sau erori

Analiza sintactica

extrage componentele gramaticale ale codului sursa si constuieste arborele de derivare (parse tree)

Daca masina destinatie(cea pentru care compilatorul trebuie sa genereze cod) poate sa difere de masina-sursa (cea pe care se executa compilatorul), procesul se numeste

cross-compiling

In definirea gramaticilor P din tupla G(N,T,P,S) reprezinta

setul finit de reguli de productie

O expresie regulata este

o secventa de caractere cu rol de sablon(pattern)

Expresiile regulate permit

extragerea tuturor subsirurilor care au aceeasi structura cu sablonul definit

"Revenirea din eroare" semnifica

tratarea erorilor detectate si continuarea procesului de compilare(daca e posibil)

In teoria limbajelor formale, "*" reprezinta

operatorul Kleene

Optimizarea codului reprezinta

o etapa a compilatorului

Limbajul unei gramatici formale va defini

toate sirurile care pot fi generate cu simbolul de start dat si aplicand regulile de productie specificate, pana la epuizarea tuturor simbolurilor neterminate

Un program translator este:

un program care converteste programele scrise de utilizatori intr-un limbaj de programare in alt limbaj de programare

O gramatica se poate utiliza pentru

determinarea apartenentei unui sir la un limbaj dat

Iesirea analizatorului lexical consta din

un sir de atomi lexicali

Gramatica reprezinta

o descriere precisa a unui limbaj, definita cu ajutorul unui alfabet

Un alfabet reprezinta

o multime finita de simboluri(caractere si cifre)

In definirea gramaticilor S din tupla G(N,T,P,S) reprezinta

simbol de start pentru gramatica

Care din urmatoarele faze nu fac parte din structura unui compilator

JIT(just-in-time)

Arborele de derivare se poate construi

top-down sau bottom-up

O gramatica specifica

setul de siruri ce pot fi considerate ca fiind cuvinte sau declaratii valide in limbajul respective

Sintaxa generala a comenzii “flex” este:

-flex [opțiune][fis]

Fisierul de reguli flex sunt compilate folosind

utilitarul " flex"

”Flex” este

un compilator ce genereaza output in limbajul C

Un fișier de specificatii gramaticale bison va contine:

mai multe variante ale producțiilor gramaticale

Iesirea analizatorului lexical este

un sir de atomi lexicali

Un fișier de specificatii bison va avea secțiunile:

declarații C, declarații bison, reguli gramaticale, cod utilizator C

În urma compilării implicite a unui fișier de reguli flex, va rezulta:

fișier text "lex.yy.c"

Secțiunile unui fișier de reguli flex sunt separate de simbolul:

"%%"

Operatorul regex "?" semnifică:

elementul precedent acestui operator este optional

Utilitarul flex este folosit pentru:

analiza lexicală;

Operatorul regex "*" semnifica

substringul se poate repeta de mai multe ori, inclusiv de 0 ori

Valoarea semantică a unui simbol neterminal (care primește valori) se reprezintă prin:

simbolul “$$”

Compilarea fișierelor de specificații “bison” generează:

fișiere .tab.c

Analiza sintactica din cadrul unui parser “bison” se face prin:

funcția “yyparse()”

Fisierul de reguli flex poate contine operatori regex

Da

Programul C obținut la execuția aplicației “flex” rezidă într-o:

funcție yylex()

Care din comenzile urmatoare permit lansarea in executie a unui navigator de fișiere sub shell-ul Linux

Mc

Un fișier de specificatii bison va avea secțiunile separate prin:

simbolurile “%%”,”%{“ și “}%”

In utilitarul "flex", o declaratie o data definita:

se apeleaza folosindu-se acoladele "{}"

Secțiunea “Rutine auxiliare” a unui fișier de reguli flex:

este opțională

Daca un sir de intrare in tokenizator are mai multe potriviri(conform pattern-ului)

se genereaza eroare

Sintaxa generică a comenzii “bison” este:

bison [opțiuni] <fișier specificatii>

Utilitarul “flex” folosește parametrul “-i” pentru:

-pentru generarea unui analizor lexical de tipul case-insensitiv

Fisierul de intrare flex contine sectiunile

declaratii, reguli, rutine auxiliare

Secțiunea “Declarații” a unui fișier de reguli flex:

este optionala

Utilizarea operatorului “./” in shell-ul Linux permite

lansarea in executie a unui binar executabil

Fisierul de reguli flex sunt compilate folosind

compilatorul "gcc"

Valoarea semantică a unui simbol neterminal (care primește valori) se reprezintă prin:

simbolul “$$”

In utilitarul "flex", o declaratie o data definita:

se apeleaza folosindu-se acoladele "{}"

”Flex” este

un analizator lexical

În urma compilării implicite a unui fișier de reguli flex, va rezulta:

fișier text "lex.yy.c"

Operatorul regex "?" semnifică:

elementul precedent acestui operator este optional

Operatorul regex "*" semnifica

substringul se poate repeta de mai multe ori, inclusiv de 0 ori

Utilitarul “flex” folosește parametrul “-i” pentru:

-pentru generarea unui analizor lexical de tipul case-insensitiv

Programul C obținut la execuția aplicației “flex” rezidă într-o:

funcție yylex()

Utilitarul flex este folosit pentru:

analiza lexicală;

Un fișier de specificatii bison va avea secțiunile separate prin:

simbolurile “%%”,”%{“ și “}%”

Sintaxa generala a comenzii “flex” este:

-flex [opțiune][fis]

Secțiunile unui fișier de reguli flex sunt separate de simbolul:

"%%"

Analiza sintactica din cadrul unui parser “bison” se face prin:

funcția “yyparse()”

Un fișier de specificatii gramaticale bison va contine:

mai multe variante ale producțiilor gramaticale

Iesirea analizatorului lexical este

un sir de atomi lexicali

Sintaxa generică a comenzii “bison” este:

bison [opțiuni] <fișier specificatii>

Daca un sir de intrare in tokenizator are mai multe potriviri(conform pattern-ului)

se alege potrivirea de lungime minima

Utilizarea operatorului “./” in shell-ul Linux permite

lansarea in executie a unui binar executabil

Secțiunea “Declarații” a unui fișier de reguli flex:

este optionala

Fisierul de reguli flex poate contine operatori regex

Da

Fisierul de intrare flex contine sectiunile

declaratii, reguli, rutine auxiliare

Care din comenzile urmatoare permit lansarea in executie a unui navigator de fișiere sub shell-ul Linux

Mc

Un fișier de specificatii bison va avea secțiunile:

declarații C, declarații bison, reguli gramaticale, cod utilizator C

Compilarea fișierelor de specificații “bison” generează:

fișiere .c

Secțiunea “Rutine auxiliare” a unui fișier de reguli flex:

este opțională

O gramatica se poate utiliza pentru

determinarea apartenentei unui sir la un limbaj dat

Daca masina destinatie(cea pentru care compilatorul trebuie sa genereze cod) poate sa difere de masina-sursa (cea pe care se executa compilatorul), procesul se numeste

cross-compiling

T* reprezinta

multimea tuturor sirurilor finite din T

In definirea gramaticilor P din tupla G(N,T,P,S) reprezinta

setul finit de reguli de productie

Atomii lexicali sunt caracterizati prin

tip si valoare

Iesirea analizatorului lexical consta din

un sir de atomi lexicali

O gramatica specifica

setul de siruri ce pot fi considerate ca fiind cuvinte sau declaratii valide in limbajul respective

O expresie regulata este

o secventa de caractere cu rol de sablon(pattern)

Limbajul unei gramatici formale va define

toate sirurile care pot fi generate cu simbolul de start dat si aplicand regulile de productie specificate, pana la epuizarea tuturor simbolurilor neterminate

Optimizarea codului reprezinta

o etapa a compilatorului

Un alfabet reprezinta

o multime finita de simboluri(caractere si cifre)

Outputul unui compilator consta din

fisier(e)-obiect sau erori

Gramatica reprezinta

o descriere precisa a unui limbaj, definita cu ajutorul unui alfabet

Arborele de derivare este

reprezentarea grafica a unei secvente de derivari

"Revenirea din eroare" semnifica

tratarea erorilor detectate si continuarea procesului de compilare(daca e posibil)

Analiza sintactica

extrage componentele gramaticale ale codului sursa si constuieste arborele de derivare (parse tree)

Un program translator este:

un program care converteste programele scrise de utilizatori intr-un limbaj accesibil calculatorului (cod-masina)

In teoria limbajelor formale, "*" reprezinta

operatorul Kleene

In definirea gramaticilor S din tupla G(N,T,P,S) reprezinta

simbol de start pentru gramatica

Identificatorii utilizati in programul-sursa sunt stocati

in tabela de simboluri

Arborele de derivare se poate construe

top-down sau bottom-up

Care din urmatoarele faze nu fac parte din structura unui compilator

JIT(just-in-time)

Expresiile regulate permit

extragerea tuturor subsirurilor care au aceeasi structura cu sablonul definit

Compilatorul este

o aplicatie software care translateaza un program scris intr un limbaj de programare intr-o forma executabila de catre calculator

Utilizarea operatorului “./” in shell-ul Linux permite

lansarea in executie a unui binar executabil

Utilitarul flex este folosit pentru:

analiza lexicală;

Iesirea analizatorului lexical este

un sir de atomi lexicali

Secțiunea “Rutine auxiliare” a unui fișier de reguli flex:

este opțională

Analiza sintactica din cadrul unui parser “bison” se face prin:

funcția “yyparse()”

Sintaxa generala a comenzii “flex” este:

-flex [opțiune][fis]

Operatorul regex "*" semnifica

substringul se poate repeta de mai multe ori, inclusiv de 0 ori

Un fișier de specificatii gramaticale bison va contine:

mai multe variante ale producțiilor gramaticale

Fisierul de reguli flex sunt compilate folosind

compilatorul "gcc"

Compilarea fișierelor de specificații “bison” generează:

fișiere .c

Care din comenzile urmatoare permit lansarea in executie a unui navigator de fișiere sub shell-ul Linux

Mc

În urma compilării implicite a unui fișier de reguli flex, va rezulta:

fișier text "lex.yy.c"

Secțiunea “Declarații” a unui fișier de reguli flex:

este optionala

Un fișier de specificatii bison va avea secțiunile:

declarații C, declarații bison, reguli gramaticale, cod utilizator C

Operatorul regex "?" semnifică:

elementul precedent acestui operator este optional

In utilitarul "flex", o declaratie o data definita:

se apeleaza folosindu-se acoladele "{}"

Utilitarul “flex” folosește parametrul “-i” pentru:

-pentru generarea unui analizor lexical de tipul case-insensitiv

Secțiunile unui fișier de reguli flex sunt separate de simbolul:

"%%"

Un fișier de specificatii bison va avea secțiunile separate prin:

simbolurile “%%”,”%{“ și “}%”

Fisierul de reguli flex poate contine operatori regex

Da

Fisierul de intrare flex contine sectiunile

declaratii, reguli, rutine auxiliare

Valoarea semantică a unui simbol neterminal (care primește valori) se reprezintă prin:

simbolul “$$”

Sintaxa generică a comenzii “bison” este:

bison [opțiuni] <fișier specificatii>

Programul C obținut la execuția aplicației “flex” rezidă într-o:

funcție yylex()

Daca un sir de intrare in tokenizator are mai multe potriviri(conform pattern-ului)

se alege potrivirea de lungime minima

”Flex” este

un analizator lexical