Tým xhricma00 varianta vv-BVS

Marek Hric	Mikuláš Lešiga	Roman Andraščík	Adam Veselý
xhricma00	xlesigm00	xandrar00	xvesela00

December 4, 2024

Rozdelenie bodov

xhricma00: 25%
xlesigm00: 25%
xandrar00: 25%
xvesela00: 25%

Rozšírenia

ORELSE
UNREACHABLE
BOOLTHEN
FOR
WHILE
FUNEXP

Rozdelenie prace :

Marek Hric :

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Mikuláš Lešiga :

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

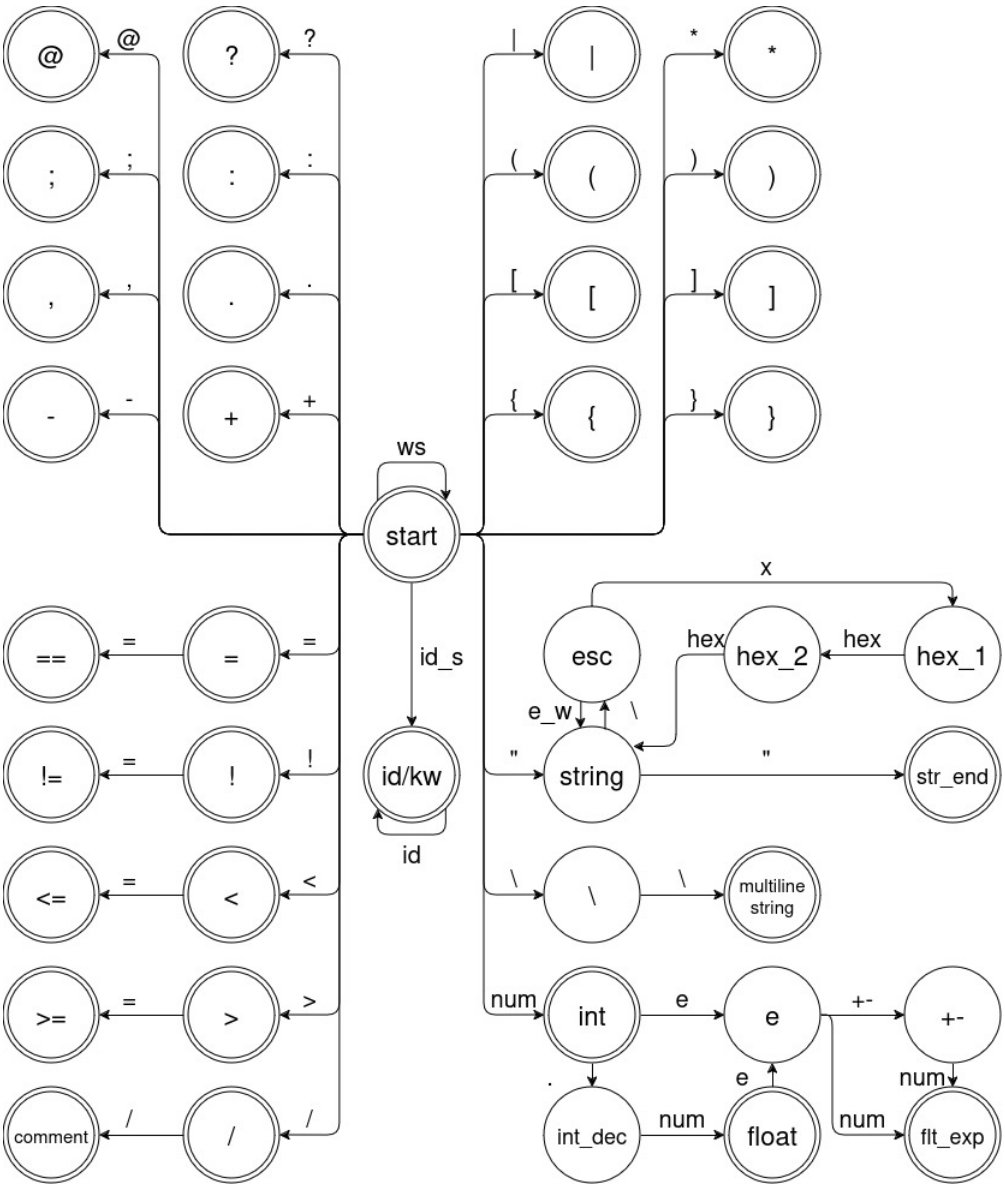
Roman Andraščík :

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Adam Veselý :

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

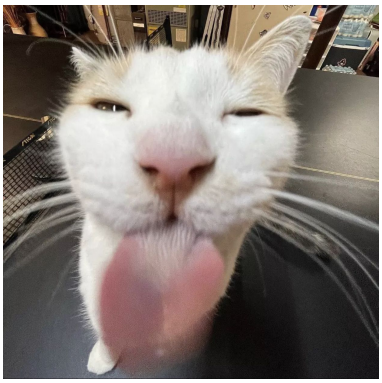
Diagram konečného automatu :



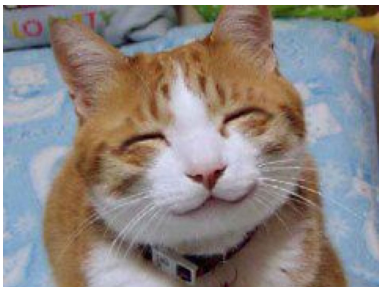
LL-gramatika :

1. aaaa
2. bbbb
3. ccccc
4. ddddddddd

LL-tabulka :



Precendecna-tabulka :



Lexikalna analyza

Riešenie lexikálnej analýzy sme začali vytvorením diagramu deterministického konečného automatu. Následne sme na jeho základe začali vypracovávať implementáciu. Implementácia sa nachádza v súbore *scanner.c,* ktorý pracuje s tokenmi deklarovanymi v súbore *token.h*. Hlavnou funkciou *scanner.c* je funkcia *get_token*. Pre uľahčenie práce a prehľadnosti kódu sme si deklarovali niekoľko makier, ktoré sú extensivne používané v hlavnej funkcii. Funkcia *get_token* berie postupne znaky zo štandardného vstupu a vytvára token. Tokenu je priradeny jeho typ a hodnota, ktorá mu odpovedá. Funkcia začína určovaním jednoznakovych tokenov, ktoré vie určite hneď na začiatku. Pokračuje identifikáciou komentárov, ktoré následne ignoruje. Po identifikácii komentárov zisťuje či sa jedna o ID alebo Keyword, pri kľúčových slovách sa následne určuje aj ich typ. Ak sa nejedna ani o jedno pokračuje kontrolou dátových typov, pri ktorých ukladá aj ich hodnoty.

Syntakticka analyza

Riešenie syntaktickej analýzy sme započali vytvorením LL gramatiky, LL tabuľky a precendecnej tabuľky. Následne na ich základe sme vypracovali súbor *parser.c a exp_parser.c*.Tieto súbory pracujú s uzlami deklarovanymi v súbore *ast.h*. Spustenie syntaktickej analýzy započne zavolanim funkcie *Parse()*. Tato funkcia postupne prechádza cez tokeny a priradzuje ich do uzlov, pomocou ktorých postupne tvorí abstraktný syntakticky strom na základe LL gramatiky. Súbor *parser.c* ďalej riadi aj precedencnu analýzu volaním funkcii zo súboru *exp_parser.c*. Tento súbor vytvorí strom výrazov, ktorý je následne pripojení do syntaktického stromu.

Semanticka analyza

Sémantická analýza je implementovana v súboroch *sem_anal.c , symtable.c, sem_anal.h a symtable.h*. Spustenie sémantickej analýzy započne zavolanim funkcie *analyse()*. Sémantická analýza je založená na rekurzivnom prechode AST stromu, ktorý je prevzatí od funkcie *parse()*, ktorá ho vygenerovala. Funkcia ďalej využíva globálne deklarovany AST strom, v ktorom sa nachádzajú built in funkcie. Pri generácií nového AST stromu sa do neho vkladajú built in funkcie práve z tohto globálneho stromu. Funkcia *analyse()* po spustení hľadá *main* a následne postupne rekurzivne prechádza cez AST strom, kde kontroluje validitu dátových typov uložených v ASTNode štruktúrach. Po tejto kontrole započne aj kontrola navratovych hodnôt. Po úspešnej validacii dát predáva nový AST strom funkcii *codegen()*, ktorá začína generaciu kódu. Ak validacia neprebehne

úspešne, vyhlási sematicku chybu.

Symtable, ktorý tato funkcia využíva je implementovany ako AVL strom.

Generovanie kodu

Generátor je implementovany v súboroch *codegen_priv.h. codegen.h a codegen.c* .Spustenie generácie kódu započne zavolanim funkcie *codegen()*. Kód je generovaný na základe rekurzivneho prechadzania abstraktivneho syntaktického stromu podľa dátového typu uloženého v štruktúre *ASTNode*. Kód ďalej využíva aj pomocný Linked List na ukladanie deklarovanych premenných v danej funkcii.

Datove struktury

Circular Buffer

Implementovane v súboroch *circ_buff.c circ_buff.h*.

Implementácia Circular Buffer je využitá hlavne v časti Scanner, kde slúži na bezpreblemove získavanie dať a ich následnú validaciu. Na prácu so scannerom ho neskôr využívajú aj časti Parser a Expression Parser. Štruktúra obsahuje klasické funkcie *circ_buff_init, circ_buff_free, circ_buff_enqueue, circ_buff_dequeue, circ_buff_is_empty*.

Dynamic String

Implementovane v súboroch *dyn_str.c, dyn_str.h*.

Implementácia dynamického reťazca je využitá hlavne v Scanner časti programu, kde sprostredkuvava validaciu a uschovavanie dať, neskôr je použitá aj v časti Codegen, kde slúži na uľahčenie validacie dať. Štruktúra dynamického reťazca obsahuje klasické funkcie *dyn_str_init, dyn_str_grow, dyn_str_append, dyn_str_append_str a dyn_str_free*.

Stack

Implementovane v súboroch *stack.c, stack.h*.

Implementáciu nášho zásobníku využívame v Expression Parser časti programu. Štruktúra zásobníku je implementovana s klasickými funkciami *stackInit, stackPush, stackPop, stackIsEmpty, stackClear a stackGetTop*. Zásobník sme zvolili pre jeho optimálny prístup k dátam a zachovanie jednoduchosti kódu.