Language

* c

Objective

* Assignment
* Function interface
* if-else

Source:

Page 177.

<https://stackoverflow.com/questions/7828357/building-an-expression-tree-in-prolog/7828855#7828855>

C BNF:

<https://cs.wmich.edu/~gupta/teaching/cs4850/sumII06/The%20syntax%20of%20C%20in%20Backus-Naur%20form.htm>

<http://marvin.cs.uidaho.edu/Teaching/CS445/c-Grammar.pdf>

Working Visual Prolog 5.2:

<https://sourceforge.net/projects/ezop-project/files/Visual%20Prolog%205.2/>

Book:

<https://studylib.net/doc/8096752/visual-prolog-5.2>

Sample output from sample programming language:

program(

[

assign("b",int(2)),

if\_then\_else(var("b"),

assign("a",int(1)),

assign("a",int(2))

),

while(var("a"),

assign("a",minus(var("a"),int(1)))

)

]

)

C BNFs to implement:

<function-definition> ::= <declaration-specifier> <declarator> {<declaration>}\* <compound-statement>

<declaration-specifier> ::= <type-specifier>

<type-specifier> ::= char

| int

| float

<declarator> ::= <direct-declarator>

<direct-declarator> ::= <identifier>

| ( <declarator> )

| <direct-declarator> [ {<constant-expression>}? ]

| <direct-declarator> ( <parameter-type-list> )

| <direct-declarator> ( {<identifier>}\* )

<expression> ::= <assignment-expression>

<equality-expression> ::= <relational-expression>

| <equality-expression> == <relational-expression>

| <equality-expression> != <relational-expression>

<relational-expression> ::= <shift-expression>

| <relational-expression> < <shift-expression>

| <relational-expression> > <shift-expression>

| <relational-expression> <= <shift-expression>

| <relational-expression> >= <shift-expression>

/\*

<assignment-expression> ::= <conditional-expression>

| <unary-expression> <assignment-operator> <assignment-expression>

<assignment-operator> ::= =

\*/

<declaration> ::= <declaration-specifier> <init-declarator> ;

<init-declarator> ::= <declarator>

| <declarator> = <initializer>

<initializer> ::= <assignment-expression>

<compound-statement> ::= { {<declaration>}\* {<statement>}\* }

<statement> ::= <expression-statement>

| <compound-statement>

| <selection-statement>

| <iteration-statement>

<selection-statement> ::= if ( <expression> ) <statement>

| if ( <expression> ) <statement> else <statement>

<iteration-statement> ::= while ( <expression> ) <statement>

Simple test C program

int max(int ch, int nm);

char x;

x = ‘a’;

int y = 7;

int a = 3;

if (y < a) {

x = ‘b’;

}

else if (y > a) {

x = ‘c’;

} else {

x = ‘d’;

}

while (1) {

x = ‘e’;

}

Should return:

program(

[

declaration(function, type(int), “max”,

parameter([type(int),type(int)])

),

declaration(variable, type(char), “x”),

assign(“x”,char(‘a’)),

declare\_init()

assign("b",int(2)),

if\_then\_else(var("b"),

assign("a",int(1)),

assign("a",int(2))

),

while(var("a"),

assign("a",minus(var("a"),int(1)))

)

]

)

Simple source.