Fakulteten för ekonomi, kommunikation och IT

Programspråk

Laboration 1

Datum: 2007-09-03
Namn: Henrik Bäck
Mathias Andersson
Kurs: DAV C02

Innehållsförteckning

Parsing av Pascalprogram

Lexer

Lexerns uppgift är att serva Parsern med tokens från den aktuella programkodskällan. Lexern öppnar programkoden från en fil och läser därefter ut varje token eller ID när parsern ber om detta.

Symboltabell

Det finns en symboltabell där information om alla dataobjekt samt programmet i sig är lagrat. I tabellen finns även information om vilka datatyper som finns i språket. Eftersom alla dataobjekt som påträffas under deklarationen i programkoden läggs till i den här tabellen kan den användas för att kontrollera ifall alla dataobjekt som används i programkoden är deklarerade. Detta görs varje gång ett dataobjekt returneras till parsern.

Typkontroll

Utöver symboltabellen finns en så kallas operationstabell. Denna tabell innehåller samtliga tillåtna operatorer samt deras operationer. I denna tabell specificeras även vilka datatyper operatorerna kan operera på samt vilken resulterande datatyp detta ger.

Parsern

Genom att omforma grammatikreglerna för Pascal så att de kan användas med högerrekursion blir skapandet av själva parsern mycket simpel. När detta var gjort är det möjligt att bygga upp funktioner som motsvarar enskilda delar i språket. Det här leder i sin tur till att programkoden för varje funktion blir mycket liten och varje funktion består i stort sett av ett antal anrop till andra funktioner. I slutänden används en funktion för att jämföra det förväntade värdet mot det värde som erhålls från lexern. På så sätt upptäcks eventuella avvikelser i det inlästa programmet från den korrekta grammatiken.

Parsning av ett program

Parsern börjar med att kontrollera så att programhuvudet motsvarar det förväntade utseendet för ett programhuvud. Detta gör den genom att fråga lexern efter ett nytt token och se ifall det stämmer över ens med det som bör stå på den platsen. Detta upprepas ett antal gånger för att kontrollera hela huvudet.

När detta är klart så går den över till att arbeta med variabler för programmet. När en variabel (id) returneras av lexern lägger densamma även upp denna variabel med namn på en kö och i en tabell över alla variabler i programmet. När senare en datatyp påträffas kommer programmet att ta varje element i kön och sätta deras datatyp i symboltabellen. Detta är viktigt för att senare kunna kontrollera så att alla datatyper som kombineras är tillåtna.

Under själva programkodsdelen kommer parsern att fortsätta kontrollera så att språkets uppbyggnad stämmer, precis på samma sätt som tidigare. Varje gång en variabel påträffas i språket här kommer lexern för det första kontrollera så att denna finns. Om variabeln används för tilldelning eller beräkning kommer den även att läggas upp på en kö. När senare ett ; eller ett end påträffas i programkoden kommer hela denna kö att arbetas igenom. Nu kontrolleras varje del i uttrycket så att det är korrekt i jämförelse med den operationstabell som finns uppsatt. Alla beräkningar måste vara tillåtna enligt denna.

Det sista som sker är att programmets fot kontrolleras. Här måste det förekomma ett end följt av en punkt (.). När en punkt (.) har kontrollerats så är kontrollen av programmet över och parsen avslutas. Om det skulle finnas extra tecken efter denna så kommer detta så kommer dessa att ignoreras.

Bilaga – Programkod

tokenDef.h

```
#ifndef _tokenDEF1234567890_
#define _tokenDEF1234567890_

        #define start     256
        #define id        start    +1
        #define number    id       +1
        #define assign    number   +1

        #define program   assign   +1
        #define input     program  +1
        #define output    input    +1
        #define var       output   +1
        #define begin_    var      +1
        #define end_      begin_   +1
        #define integer   end_     +1
        #define boolean   integer  +1
        #define unknown    boolean  +1
        #define real_      unknown  +1

#endif
```

lexer.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <ctype.h>
#include <list>
#include <map>
#include <iomanip>
#include "tokenDef.h"
#include "tokenInfo.cpp"

using namespace std;
string programBuffer;

typedef  struct tab {
   char *text;
   int   token;
   } tab;

typedef struct symbolTable
{
        string role;
        string type;
        int size;
        int addr;

} symbolTable;

typedef struct operatorTable
{
        string opr;
        int arg1;
        int arg2;
        int result;

} operatorTable;

map<string,symbolTable> symbTable;
map<int,operatorTable> oprTable;

tab tokentab[] = {
             "id",       id,
             "number",   number,
             ":=",       assign,
             "",         0
```

```
            };

tab keywordtab[] = {
            "program",  program,
            "input",    input,
            "output",   output,
            "var",      var,
            "begin",    begin_,
            "end",      end_,
            "integer",  integer,
            "boolean",  boolean,
            "real",     real_,
            "",         0
            };

//Add the std datatypes to the ST
void fillSymbTable()
{
        symbolTable inten;
        inten.role = "type";
        inten.type = "_predef";
        inten.size = 4;
        inten.addr = 0;
        symbTable["integer"] = inten;

        symbolTable boolen;
        boolen.role = "type";
        boolen.type = "_predef";
        boolen.size = 1;
        boolen.addr = 0;
        symbTable["boolean"] = boolen;

        symbolTable realen;
        realen.role = "type";
        realen.type = "_predef";
        realen.size = 16;
        realen.addr = 0;
        symbTable["real"] = realen;

}

//Add information to the operation table.
void fillOprTable()
{
        operatorTable toAdd;

        toAdd.opr = "+";
        toAdd.arg1 = integer;
        toAdd.arg2 = integer;
        toAdd.result = integer;
        oprTable[1] = toAdd;

        toAdd.opr = "+";
        toAdd.arg1 = real_;
        toAdd.arg2 = integer;
        toAdd.result = real_;
        oprTable[2] = toAdd;

        toAdd.opr = "+";
        toAdd.arg1 = integer;
        toAdd.arg2 = real_;
        toAdd.result = real_;
        oprTable[3] = toAdd;

        toAdd.opr = "+";
        toAdd.arg1 = real_;
        toAdd.arg2 = real_;
        toAdd.result = real_;
        oprTable[4] = toAdd;

        toAdd.opr = "*";
        toAdd.arg1 = integer;
        toAdd.arg2 = integer;
        toAdd.result = integer;
        oprTable[5] = toAdd;
```

```
toAdd.opr = "*";
toAdd.arg1 = real_;
toAdd.arg2 = integer;
toAdd.result = real_;
oprTable[6] = toAdd;

toAdd.opr = "*";
toAdd.arg1 = integer;
toAdd.arg2 = real_;
toAdd.result = real_;
oprTable[7] = toAdd;

toAdd.opr = "*";
toAdd.arg1 = real_;
toAdd.arg2 = real_;
toAdd.result = real_;
oprTable[8] = toAdd;

toAdd.opr = "+";
toAdd.arg1 = integer;
toAdd.arg2 = number;
toAdd.result = integer;
oprTable[9] = toAdd;

toAdd.opr = "+";
toAdd.arg1 = number;
toAdd.arg2 = integer;
toAdd.result = integer;
oprTable[10] = toAdd;

toAdd.opr = "+";
toAdd.arg1 = number;
toAdd.arg2 = integer;
toAdd.result = real_;
oprTable[11] = toAdd;

toAdd.opr = "+";
toAdd.arg1 = integer;
toAdd.arg2 = number;
toAdd.result = real_;
oprTable[12] = toAdd;

toAdd.opr = "+";
toAdd.arg1 = real_;
toAdd.arg2 = number;
toAdd.result = real_;
oprTable[13] = toAdd;

toAdd.opr = "+";
toAdd.arg1 = number;
toAdd.arg2 = real_;
toAdd.result = real_;
oprTable[14] = toAdd;

toAdd.opr = "+";
toAdd.arg1 = number;
toAdd.arg2 = number;
toAdd.result = number;
oprTable[15] = toAdd;

      toAdd.opr = "*";
toAdd.arg1 = integer;
toAdd.arg2 = number;
toAdd.result = integer;
oprTable[16] = toAdd;

toAdd.opr = "*";
toAdd.arg1 = number;
toAdd.arg2 = integer;
toAdd.result = integer;
oprTable[17] = toAdd;

toAdd.opr = "*";
toAdd.arg1 = number;
```

```
        toAdd.arg2 = integer;
        toAdd.result = real_;
        oprTable[18] = toAdd;

        toAdd.opr = "*";
        toAdd.arg1 = integer;
        toAdd.arg2 = number;
        toAdd.result = real_;
        oprTable[19] = toAdd;

        toAdd.opr = "*";
        toAdd.arg1 = real_;
        toAdd.arg2 = number;
        toAdd.result = real_;
        oprTable[20] = toAdd;

        toAdd.opr = "*";
        toAdd.arg1 = number;
        toAdd.arg2 = real_;
        toAdd.result = real_;
        oprTable[21] = toAdd;

        toAdd.opr = "*";
        toAdd.arg1 = number;
        toAdd.arg2 = number;
        toAdd.result = number;
        oprTable[22] = toAdd;

}

void printSymbolTable()
{

  map<string,symbolTable>::iterator it;

  cout << endl << "                    Symbol Table";
  cout << endl << "                    ============" << endl << endl;

  cout << setiosflags(ios::left) << setw(20) << "Name" << setw(10)
       << "Role" << setw(10) << "Type" << setw(5) << "Size" << setw(10) << "Address"
<< endl << "--------------------------------------------------" << endl;

  // show content:
  for (it=symbTable.begin(); it != symbTable.end(); it++ )
    cout  <<  setiosflags(ios::left)  <<  setw(20)  <<  (*it).first  <<  setw(10)  <<
(*it).second.role << setw(10) << (*it).second.type << setw(5) << (*it).second.size <<
setw(10) << (*it).second.addr << endl;

        cout << endl << endl;

}

void printOprTable()
{

  map<int,operatorTable>::iterator it;

  cout << endl << "                    Operator Table";
  cout << endl << "                    ==============" << endl << endl;

  cout << setiosflags(ios::left) << setw(10) << "Operator" << setw(10)
       << "Arg1" << setw(10) << "Arg2" << setw(10) << "Result" << endl << "----------
----------------------------------------" << endl;

  // show content:
  for (it=oprTable.begin(); it != oprTable.end(); it++ )
    cout  <<  setiosflags(ios::left)  <<  setw(10)  <<  (*it).second.opr  <<  setw(10)  <<
(*it).second.arg1 << setw(10) << (*it).second.arg2 << setw(10) << (*it).second.result
<< endl;

        cout << endl << endl;

}

//Read the program source code from the filebuffer
```

```
bool readInProgram(char fileName[], string& programText)
{
	string line;
	ifstream source(fileName);

	if (source.is_open())
	{
		while(!source.eof())
		{
			getline(source,line);
			programText.append("\t");
			programText.append(line);
			programText.append("\n");
		}
		source.close();

		return true;
	}
	else
	{
		cout << "*** File '" << fileName << "' not found or file corrupt..." <<
endl << endl;
		return false;
	}

}

//Match a keyword in the string at position with one in the keywordtable
int matchKeyword(string &word, int position)
{
	if(keywordtab[position].token == 0 || keywordtab[position].text == word)
		return keywordtab[position].token;
	else
		matchKeyword(word, position + 1);
}

//Get the length of a alfanumric word in a string, starting at position
int getWordLen(string &buffer, int position)
{
	return	isalpha(buffer[position])	||	isdigit(buffer[position])	?
getWordLen(buffer, position + 1) : position;
}

//Get the length of a number in a string, startinga at position
int getNumLen(string &buffer, int position)
{
	return	isdigit(buffer[position])	?	getNumLen(buffer,	position	+	1)	:
position;
}

//Checks if an expression has the correct datatypes
bool checkExpr(list<string> &expr)
{
	bool wentOk = true;
	string a, b, c;
	int pre, cType;

	a = expr.front(); expr.pop_front();

	if(symbTable.find(a) != symbTable.end())
		pre = matchKeyword(symbTable[a].type,0);
	else if(a == "number")
		pre = number;
	else
		pre = unknown;

	while(expr.size() >= 2)
	{
		b = expr.front(); expr.pop_front();
		c = expr.front(); expr.pop_front();
		map<int,operatorTable>::iterator oprIt;

		if(symbTable.find(c) != symbTable.end())
			cType = matchKeyword(symbTable[c].type,0);
		else if(c == "number")
```

```
                cType = number;
            else
                cType = unknown;

            bool found = false;

            if( b == ":=")
            {
                if((pre == cType || pre == number) && cType != unknown)
                    found = true;
            }
            else
            {
                for (oprIt=oprTable.begin(); oprIt != oprTable.end() && !found;
oprIt++ )
                    if((*oprIt).second.opr == b)
                    {
                        if((*oprIt).second.arg1 == pre && (*oprIt).second.arg2
== cType)
                        {
                            pre = (*oprIt).second.result;
                            found = true;
                        }
                    }
            }

            if(!found)
            {
                cout << endl << "*** Mismatch types for " << c;
                wentOk = false;
            }
        }

        return wentOk;
}

//Returns a token and information about if getToken was sucessfully to the parser, it
does also add symbols to the ST
pair<int,bool> getTokenArg(string &buffer)
{
        static int position = 0;
        static int pgmPart;
        static string pgmName;
        static string varType;
        static list<string> unSetVar;
        static int addrOffset = 0;
        static list<string> expr;
        int lexLen = 0, token;
        bool wentOk = true;
        string tempBuffer;
        pair<map<string,symbolTable>::iterator,bool> ret;

        if(isalpha(buffer[position]))
        {
            lexLen = getWordLen(buffer,position) - position;
            tempBuffer = buffer.substr(position,lexLen);
            position += lexLen;
            token = matchKeyword(tempBuffer,0);

            switch(token)
            {
                case program: pgmPart = program; break;
                case var: pgmPart = var; break;
                case begin_: pgmPart = begin_; break;
            };

            if(!token)
            {
                symbolTable add;

                if(pgmPart == program)
                {
                    pgmName = tempBuffer;
                    add.role = "program";
                    add.type = "_predef";
```

```
                        add.size = 0;
                        add.addr = 0;
                  }
                  if(pgmPart == var)
                  {
                        add.role = "variable";
                        add.type = "unknown";
                        add.size = 0;
                        add.addr = 0;
                        unSetVar.push_front(tempBuffer);
                  }
                  if(pgmPart == var || pgmPart == program)
                  {
                        ret                     =                     symbTable.insert
(pair<string,symbolTable>(tempBuffer,add));
                        if(ret.second == false)
                        {
                              cout << endl << endl << "*** Variable " << tempBuffer
<< " already defined." <<  endl;
                              unSetVar.pop_back();
                              wentOk = false;
                        }
                  }

                  if(pgmPart == begin_)
                  {
                        if(symbTable.find(tempBuffer) == symbTable.end())
                        {
                              cout << endl  << endl << "*** Variable " << tempBuffer
<< " is not defined." <<  endl;
                              wentOk = false;
                        }

                        if(!token)
                              expr.push_front(tempBuffer);
                  }
            }

            if(token == integer || token == boolean || token == real_)
            {

                  while(!unSetVar.empty())
                  {
                        string currentVar = unSetVar.back();
                        unSetVar.pop_back();
                        varType = tempBuffer;

                        symbTable[currentVar].size = symbTable[varType].size;
                        symbTable[currentVar].addr = addrOffset;
                        symbTable[currentVar].type = varType;

                        addrOffset += symbTable[varType].size;

                        symbTable[pgmName].size = addrOffset;

                  }

            }
            pair<int, bool> retur((token ? token : id),wentOk);
            return retur;
      }
      else if(isdigit(buffer[position]))
      {
            lexLen = getNumLen(buffer,position) - position;
            expr.push_front("number");
            position += lexLen;
            pair<int, bool> retur(number,wentOk);
            return retur;
      }
      else
      {
            int tempPos = position;

            if(pgmPart == begin_ && expr.size() > 2 && (buffer[tempPos] == 13 ||
buffer[tempPos] == 10 || buffer[tempPos] == ';'))
```

```
                wentOk = checkExpr(expr);
            else if(pgmPart == begin_ && expr.size() <= 2 && (buffer[tempPos] == 13
|| buffer[tempPos] == 10 || buffer[tempPos] == ';'))
                expr.clear();

            if(buffer[tempPos] == ':' && buffer[tempPos+1] == '=')
            {
                position += 2;
                expr.push_front(":=");
                pair<int, bool> retur(assign,wentOk);
                return retur;
            }
            else  if(buffer[tempPos]  ==  '  '  ||  buffer[tempPos]  ==  13  ||
buffer[tempPos] == 10 || buffer[tempPos] == 9 )
            {
                position += 1;
                pair<int,bool> returen;
                returen = getTokenArg(buffer);
                if(returen.second == false)
                    wentOk = false;
                pair<int, bool> retur(returen.first,wentOk);
                return retur;
            }
            else
            {
                if(buffer[position] == '+' || buffer[position] == '*')
                {
                    string toInsert;
                    toInsert = buffer.substr(position,1);
                    expr.push_front(toInsert);
                }
                position += 1;
                pair<int, bool> retur(buffer[tempPos],wentOk);
                return retur;
            }
        }
}

//The lexer was originally built without non-local dataobjekts
//Insted of re-writing the orginal function we just wrap it in.
pair<int,bool> getToken()
{
        return getTokenArg(programBuffer);
}
```

tokenInfo.cpp

```
#ifndef _tokenInfoDEF1234567890_
#define _tokenInfoDEF1234567890_

void coutTokenInfo(int ascii)
{
        using namespace std;

        if(ascii == '$')
                cout << "EOF";
        else if(ascii < start)
                cout << ((char) ascii);
        else
                {
                    switch (ascii)
                    {

                        case 256: cout << "start"; break;
                        case 257: cout << "id"; break;
                        case 258: cout << "number"; break;
                        case 259: cout << ":="; break;
                        case 260: cout << "program"; break;
                        case 261: cout << "input"; break;
                        case 262: cout << "output"; break;
                        case 263: cout << "var"; break;
                        case 264: cout << "begin"; break;
                        case 265: cout << "end"; break;
```

```
                        case 266: cout << "type"; break;
                        case 267: cout << "type"; break;
                        case 268: cout << "type"; break;
                        case 269: cout << "type"; break;
                        default: break;
                    };

            }

}


#endif
```

parser.cpp

```
#include <iostream>
#include <cstring>
#include "lexer.cpp"
#include "tokenDef.h"
#include "tokenInfo.cpp"
#define DEBUG false

using namespace std;

int lookahead;

//Matches a expected token with the next token in the program
bool match(int token)
{

      pair<int,bool> returen;

      if(DEBUG)
      {
            cout << endl << "*   In 'match'. Token infomration; expected: ";
            coutTokenInfo(token);
            cout << ", found: ";
            coutTokenInfo(lookahead);
      }

      if (lookahead == token)
      {
            returen = getToken();
            lookahead = returen.first;
            return returen.second;
      }
      else
      {
            cout << endl << "*** Unexpected Token: expected:  ";
            coutTokenInfo(token);
            cout << endl << "                     found:     ";
            coutTokenInfo(lookahead);
            returen = getToken();
            lookahead = returen.first;
            cout << endl << "    -> (in match)" << endl;
            return false;
      }

}

// [prog header] ::= program id ( input , output );
bool programHeader()
{
      if(DEBUG)
            cout << endl << "*   In 'programHeader'";

      return (match(program) & match(id) & match('(') & match(input) & match(',') &
match(output) & match(')') & match(';')) ? true : false;
}

//[id list] ::= id | [id list] , id
bool idList()
```

```
{
	if(DEBUG)
		cout << endl << "*   In 'idList'";

	bool returnVal1 = true, returnVal2 = true;

	returnVal1 = match(id);
	if(lookahead == ',')
		returnVal2 = match(',') & idList();

	return returnVal1 & returnVal2;
}

//[type] ::= integer | boolean
bool type()
{
	if(DEBUG)
		cout << endl << "*   In 'type'";

	if(lookahead == integer)
		return match(integer);
	else if(lookahead == boolean)
		return match(boolean);
	else
		return match(real_);

}

//[var dec]::= [id list] : [type]  ;
bool varDec()
{
	if(DEBUG)
		cout << endl << "*   In 'varDec'";

	return (idList() & match(':') & type() & match(';')) ? true : false;
}

//[var dec list] ::= [var dec] | [var dec list] [var dec]
bool varDecList()
{
	if(DEBUG)
		cout << endl << "*   In 'varDecList'";

	bool returnVal1 = true, returnVal2 = true;

	returnVal1 = varDec();
	if(lookahead != begin_ && returnVal1)
		returnVal2 = varDecList();

	return returnVal1 & returnVal2;
}

//[var part] ::= var [var dec list]
bool varPart()
{
	if(DEBUG)
		cout << endl << "*   In 'varPart'";

	return match(var) & varDecList();
}

//[operand]::= id | number
bool operand()
{
	if(DEBUG)
		cout << endl << "*   In 'operand'";

	return (lookahead == id) ? match(id) : match(number);

}

//[operator] ::=  + | *
bool operatorn()
{
```

```
	if(DEBUG)
		cout << endl << "*   In 'operatorn'";

	return (lookahead == '*') ? match('*') : match('+');
}

//[expr] ::=  [operand] | [expr] [operator] [operand]
bool expr()
{
	if(DEBUG)
		cout << endl << "*   In 'expr'";

	bool returnVal1 = true, returnVal2 = true;

	returnVal1 = operand();
	if(lookahead == '*' || lookahead == '+')
		returnVal2 = operatorn() & expr();

	return returnVal1 & returnVal2;
}

//[assign stat] ::= id := [expr]
bool assignStat()
{
	if(DEBUG)
		cout << endl << "*   In 'assignStat'";

	return match(id) & match(assign) & expr();
}

//[stat] ::= [assign stat]
bool stat()
{
	if(DEBUG)
		cout << endl << "*   In 'stat'";

	return assignStat();
}

//[stat list] ::= [stat] | [stat list]  ;  [stat]
bool statList()
{
	if(DEBUG)
		cout << endl << "*   In 'statList'";

	bool returnVal1 = true, returnVal2 = true;

	returnVal1 = stat();
	if(lookahead == ';')
	returnVal2 =  match(';') & statList();

	return returnVal1 & returnVal1;
}

//[stat part] ::= begin [stat list] end .
bool statPart()
{
	if(DEBUG)
		cout << endl << "*   In 'statPart'";

	return match(begin_) & statList() & match(end_) & match('.');
}

//[prog] ::=  [prog header] [var part] [stat part]
bool prog()
{
	pair<int,bool> returen;

	if(DEBUG)
		cout << endl << "*   In 'prog'";

	//Fill the Operator-table
	fillOprTable();

	//Fill the symbol table
```

```
	fillSymbTable();

	returen = getToken();

	lookahead = returen.first;

	return programHeader() & varPart() & statPart() & returen.second;
}

//Run the program, read the input data and read the fil in to the buffer
//Start the parser
int main(int argc, char *argv[])
{
	if(argc == 2)
	{
		int token;
		if(readInProgram(argv[1], programBuffer))
		{
			cout << "*    --- Program '" << argv[1] << "' is ---" << endl <<
endl;
			cout << programBuffer << endl;
			programBuffer.append("$");
			cout << "*   --- Start parsing ---" << endl;
			prog() ? (cout << endl << endl << "*   --------- Parse Successful! -
--------    *" << endl << endl) : (cout << endl << endl << "*    --------- Parse
failed! ---------   *" << endl << endl);
			printSymbolTable();
			//printOprTable();
		}
	}
	else
		cout << "*** Wrong number of arguments supplied..." << endl << "*
Usage: " << argv[0] << " file_to_check" << endl << endl;


	return 0;

}
```

Bilaga – Körning

```
*   --- Program 'fun1.pas' is ---
program fun1(input, output);
var A, B, A: integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*** Variable A already defined.
*** Variable C is not defined.
*** Mismatch types for C
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    4
B                    variable  integer   4    0
boolean              type      _predef   1    0
fun1                 program   _predef   8    0
integer              type      _predef   4    0
real                 type      _predef   16   0


*   --- Program 'fun2.pas' is ---
program fun2(input, output);
var A, B, C: integer;
begin
3A := B + C * 2
end.
```

```
*   --- Start parsing ---
*** Unexpected Token: expected:  id
                      found:     number
*** Unexpected Token: expected:  :=
                      found:     id
*** Unexpected Token: expected:  number
                      found:     :=
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     +
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
fun2                 program   _predef   12   0
integer              type      _predef   4    0
real                 type      _predef   16   0

*   --- Program 'fun3.pas' is ---
program fun3(input, output);
var A, B, C: integer;
begin
A :
= B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  :=
                      found:     :
*** Unexpected Token: expected:  number
                      found:     =
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     +
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
fun3                 program   _predef   12   0
integer              type      _predef   4    0
real                 type      _predef   16   0

*   --- Program 'fun4.pas' is ---
program fun4(input, output);
var A, B, C: integer;
D, E, F: integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*   --------- Parse Successful! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
D                    variable  integer   4    12
E                    variable  integer   4    16
F                    variable  integer   4    20
```

```
boolean             type      _predef   1    0
fun4                program   _predef   24   0
integer             type      _predef   4    0
real                type      _predef   16   0

*   --- Program 'fun5.pas' is ---
program fun5(input, output);
var A, B, C: integer;
begin
A := B + C * 2
end.
OOOOPS!!!
*   --- Start parsing ---
*** Variable OOOOPS is not defined.
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                Role      Type      Size Address
--------------------------------------------------
A                   variable  integer   4    0
B                   variable  integer   4    4
C                   variable  integer   4    8
boolean             type      _predef   1    0
fun5                program   _predef   12   0
integer             type      _predef   4    0
real                type      _predef   16   0

*   --- Program 'sem1.pas' is ---
program sem1(input, output);
var A, B, C: integer;
begin
A := D + C * 2
end.
*   --- Start parsing ---
*** Variable D is not defined.
*** Mismatch types for D
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                Role      Type      Size Address
--------------------------------------------------
A                   variable  integer   4    0
B                   variable  integer   4    4
C                   variable  integer   4    8
boolean             type      _predef   1    0
integer             type      _predef   4    0
real                type      _predef   16   0
sem1                program   _predef   12   0

*   --- Program 'sem2.pas' is ---
program sem2(input, output);
var A, B, A: integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*** Variable A already defined.
*** Variable C is not defined.
*** Mismatch types for C
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                Role      Type      Size Address
--------------------------------------------------
A                   variable  integer   4    4
B                   variable  integer   4    0
boolean             type      _predef   1    0
integer             type      _predef   4    0
real                type      _predef   16   0
sem2                program   _predef   8    0

*   --- Program 'sem3.pas' is ---
```

```
program sem3(input, output);
var A, B, C: double;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  type
                      found:     id
*** Mismatch types for C
*** Mismatch types for B
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  unknown   0    0
B                    variable  unknown   0    0
C                    variable  unknown   0    0
boolean              type      _predef   1    0
double               variable  unknown   0    0
integer              type      _predef   4    0
real                 type      _predef   16   0
sem3                 program   _predef   0    0

*   --- Program 'sem4.pas' is ---
program sem4(input, output);
var A, B, C: integer;
D, E, F: real;
begin
F := B + C * 2
end.
*   --- Start parsing ---
*** Mismatch types for F
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
D                    variable  real      16   12
E                    variable  real      16   28
F                    variable  real      16   44
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
sem4                 program   _predef   60   0

*   --- Program 'sem5.pas' is ---
program sem5(input, output);
var A, B, C: integer;
D, E, F: real;
begin
A := D + E * 2
end.
*   --- Start parsing ---
*** Mismatch types for A
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
D                    variable  real      16   12
E                    variable  real      16   28
F                    variable  real      16   44
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
```

```
sem5                program   _predef   60   0

*   --- Program 'testa.pas' is ---
*   --- Start parsing ---
*** Unexpected Token: expected:  program
                     found:     EOF
*** Unexpected Token: expected:  id
                     found:
*** Unexpected Token: expected:  (
                     found:
*** Unexpected Token: expected:  input
                     found:
*** Unexpected Token: expected:  ,
                     found:
*** Unexpected Token: expected:  output
                     found:
*** Unexpected Token: expected:  )
                     found:
*** Unexpected Token: expected:  ;
                     found:
*** Unexpected Token: expected:  var
                     found:
*** Unexpected Token: expected:  id
                     found:
*** Unexpected Token: expected:  :
                     found:
*** Unexpected Token: expected:  type
                     found:
*** Unexpected Token: expected:  ;
                     found:
*** Unexpected Token: expected:  begin
                     found:
*** Unexpected Token: expected:  id
                     found:
*** Unexpected Token: expected:  :=
                     found:     □
*** Unexpected Token: expected:  number
                     found:
*** Unexpected Token: expected:  end
                     found:
*** Unexpected Token: expected:  .
                     found:
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                Role      Type      Size Address
--------------------------------------------------
boolean             type      _predef   1    0
integer             type      _predef   4    0
real                type      _predef   16   0

*   --- Program 'testb.pas' is ---
testb(input, output);
var A, B, C: integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  program
                     found:     id
*** Unexpected Token: expected:  id
                     found:     (
*** Unexpected Token: expected:  (
                     found:     input
*** Unexpected Token: expected:  input
                     found:     ,
*** Unexpected Token: expected:  ,
                     found:     output
*** Unexpected Token: expected:  output
                     found:     )
*** Unexpected Token: expected:  )
                     found:     ;
*** Unexpected Token: expected:  ;
                     found:     var
```

```
*** Unexpected Token: expected:  var
                      found:     id
*** Unexpected Token: expected:  id
                      found:     ,
*** Unexpected Token: expected:  :
                      found:     id
*** Unexpected Token: expected:  type
                      found:     ,
*** Unexpected Token: expected:  ;
                      found:     id
*** Unexpected Token: expected:  begin
                      found:     :
*** Unexpected Token: expected:  id
                      found:     type
*** Unexpected Token: expected:  :=
                      found:     ;
*** Unexpected Token: expected:  number
                      found:     begin
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     :=
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
12   3148656
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0

*   --- Program 'testc.pas' is ---
program (input, output);
var A, B, C: integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  id
                      found:     (
*** Unexpected Token: expected:  (
                      found:     input
*** Unexpected Token: expected:  input
                      found:     ,
*** Unexpected Token: expected:  ,
                      found:     output
*** Unexpected Token: expected:  output
                      found:     )
*** Unexpected Token: expected:  )
                      found:     ;
*** Unexpected Token: expected:  ;
                      found:     var
*** Unexpected Token: expected:  var
                      found:     id
*** Unexpected Token: expected:  id
                      found:     ,
*** Unexpected Token: expected:  :
                      found:     id
*** Unexpected Token: expected:  type
                      found:     ,
*** Unexpected Token: expected:  ;
                      found:     id
*** Unexpected Token: expected:  begin
                      found:     :
*** Unexpected Token: expected:  id
                      found:     type
*** Unexpected Token: expected:  :=
                      found:     ;
*** Unexpected Token: expected:  number
                      found:     begin
```

```
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     :=
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
--------------------------------------------------
12   3148656
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0

*   --- Program 'testd.pas' is ---
program testd input, output);
var A, B, C: integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  (
                      found:     input
*** Unexpected Token: expected:  input
                      found:     ,
*** Unexpected Token: expected:  ,
                      found:     output
*** Unexpected Token: expected:  output
                      found:     )
*** Unexpected Token: expected:  )
                      found:     ;
*** Unexpected Token: expected:  ;
                      found:     var
*** Unexpected Token: expected:  var
                      found:     id
*** Unexpected Token: expected:  id
                      found:     ,
*** Unexpected Token: expected:  :
                      found:     id
*** Unexpected Token: expected:  type
                      found:     ,
*** Unexpected Token: expected:  ;
                      found:     id
*** Unexpected Token: expected:  begin
                      found:     :
*** Unexpected Token: expected:  id
                      found:     type
*** Unexpected Token: expected:  :=
                      found:     ;
*** Unexpected Token: expected:  number
                      found:     begin
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     :=
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
--------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testd                program   _predef   12   0

*   --- Program 'teste.pas' is ---
program teste ( , output);
```

```
var A, B, C: integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  input
                      found:     ,
*** Unexpected Token: expected:  ,
                      found:     output
*** Unexpected Token: expected:  output
                      found:     )
*** Unexpected Token: expected:  )
                      found:     ;
*** Unexpected Token: expected:  ;
                      found:     var
*** Unexpected Token: expected:  var
                      found:     id
*** Unexpected Token: expected:  id
                      found:     ,
*** Unexpected Token: expected:  :
                      found:     id
*** Unexpected Token: expected:  type
                      found:     ,
*** Unexpected Token: expected:  ;
                      found:     id
*** Unexpected Token: expected:  begin
                      found:     :
*** Unexpected Token: expected:  id
                      found:     type
*** Unexpected Token: expected:  :=
                      found:     ;
*** Unexpected Token: expected:  number
                      found:     begin
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     :=
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
teste                program   _predef   12   0

*   --- Program 'testf.pas' is ---
program testf (input  output);
var A, B, C: integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  ,
                      found:     output
*** Unexpected Token: expected:  output
                      found:     )
*** Unexpected Token: expected:  )
                      found:     ;
*** Unexpected Token: expected:  ;
                      found:     var
*** Unexpected Token: expected:  var
                      found:     id
*** Unexpected Token: expected:  id
                      found:     ,
*** Unexpected Token: expected:  :
                      found:     id
*** Unexpected Token: expected:  type
                      found:     ,
*** Unexpected Token: expected:  ;
```

```
                      found:     id
*** Unexpected Token: expected:  begin
                      found:     :
*** Unexpected Token: expected:  id
                      found:     type
*** Unexpected Token: expected:  :=
                      found:     ;
*** Unexpected Token: expected:  number
                      found:     begin
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     :=
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testf                program   _predef   12   0

*   --- Program 'testg.pas' is ---
program testg (input, );
var A, B, C: integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  output
                      found:     )
*** Unexpected Token: expected:  )
                      found:     ;
*** Unexpected Token: expected:  ;
                      found:     var
*** Unexpected Token: expected:  var
                      found:     id
*** Unexpected Token: expected:  id
                      found:     ,
*** Unexpected Token: expected:  :
                      found:     id
*** Unexpected Token: expected:  type
                      found:     ,
*** Unexpected Token: expected:  ;
                      found:     id
*** Unexpected Token: expected:  begin
                      found:     :
*** Unexpected Token: expected:  id
                      found:     type
*** Unexpected Token: expected:  :=
                      found:     ;
*** Unexpected Token: expected:  number
                      found:     begin
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     :=
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testg                program   _predef   12   0
```

```
*   --- Program 'testh.pas' is ---
program testh (input, output ;
var A, B, C: integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  )
                      found:     ;
*** Unexpected Token: expected:  ;
                      found:     var
*** Unexpected Token: expected:  var
                      found:     id
*** Unexpected Token: expected:  id
                      found:     ,
*** Unexpected Token: expected:  :
                      found:     id
*** Unexpected Token: expected:  type
                      found:     ,
*** Unexpected Token: expected:  ;
                      found:     id
*** Unexpected Token: expected:  begin
                      found:     :
*** Unexpected Token: expected:  id
                      found:     type
*** Unexpected Token: expected:  :=
                      found:     ;
*** Unexpected Token: expected:  number
                      found:     begin
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     :=
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                  Role      Type      Size Address
-----------------------------------------------------
A                     variable  integer   4    0
B                     variable  integer   4    4
C                     variable  integer   4    8
boolean               type      _predef   1    0
integer               type      _predef   4    0
real                  type      _predef   16   0
testh                 program   _predef   12   0

*   --- Program 'testi.pas' is ---
program testi (input, output)
var A, B, C: integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  ;
                      found:     var
*** Unexpected Token: expected:  var
                      found:     id
*** Unexpected Token: expected:  id
                      found:     ,
*** Unexpected Token: expected:  :
                      found:     id
*** Unexpected Token: expected:  type
                      found:     ,
*** Unexpected Token: expected:  ;
                      found:     id
*** Unexpected Token: expected:  begin
                      found:     :
*** Unexpected Token: expected:  id
                      found:     type
*** Unexpected Token: expected:  :=
                      found:     ;
*** Unexpected Token: expected:  number
                      found:     begin
```

```
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     :=
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testi                program   _predef   12   0

*   --- Program 'testj.pas' is ---
program testj (input, output);
A, B, C: integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  var
                      found:     id
*** Unexpected Token: expected:  id
                      found:     ,
*** Unexpected Token: expected:  :
                      found:     id
*** Unexpected Token: expected:  type
                      found:     ,
*** Unexpected Token: expected:  ;
                      found:     id
*** Unexpected Token: expected:  begin
                      found:     :
*** Unexpected Token: expected:  id
                      found:     type
*** Unexpected Token: expected:  :=
                      found:     ;
*** Unexpected Token: expected:  number
                      found:     begin
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     :=
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    program   _predef   0    0
B                    program   _predef   0    0
C                    program   _predef   0    0
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testj                program   _predef   0    0

*   --- Program 'testk.pas' is ---
program testk (input, output);
var , B, C: integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  id
                      found:     ,
*** Unexpected Token: expected:  :
                      found:     id
*** Unexpected Token: expected:  type
                      found:     ,
*** Unexpected Token: expected:  ;
```

```
                      found:     id
*** Unexpected Token: expected:  begin
                      found:     :
*** Unexpected Token: expected:  id
                      found:     type
*** Unexpected Token: expected:  :=
                      found:     ;
*** Unexpected Token: expected:  number
                      found:     begin
*** Variable A is not defined.
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     :=
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
--------------------------------------------------
B                    variable  integer   4    0
C                    variable  integer   4    4
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testk                program   _predef   8    0

*   --- Program 'testl.pas' is ---
program testl (input, output);
var A  B, C: integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  :
                      found:     id
*** Unexpected Token: expected:  type
                      found:     ,
*** Unexpected Token: expected:  ;
                      found:     id
*** Unexpected Token: expected:  begin
                      found:     :
*** Unexpected Token: expected:  id
                      found:     type
*** Unexpected Token: expected:  :=
                      found:     ;
*** Unexpected Token: expected:  number
                      found:     begin
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     :=
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
--------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testl                program   _predef   12   0

*   --- Program 'testm.pas' is ---
program testm (input, output);
var A, B, C  integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  :
                      found:     type
```

```
*** Unexpected Token: expected:  type
                      found:     ;
*** Unexpected Token: expected:  ;
                      found:     begin
*** Unexpected Token: expected:  begin
                      found:     id
*** Unexpected Token: expected:  id
                      found:     :=
*** Unexpected Token: expected:  :=
                      found:     id
*** Unexpected Token: expected:  number
                      found:     +
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     *
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testm                program   _predef   12   0

*   --- Program 'testn.pas' is ---
program testn (input, output);
var A, B, C: ;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  type
                      found:     ;
*** Unexpected Token: expected:  ;
                      found:     begin
*** Unexpected Token: expected:  begin
                      found:     id
*** Unexpected Token: expected:  id
                      found:     :=
*** Unexpected Token: expected:  :=
                      found:     id
*** Unexpected Token: expected:  number
                      found:     +
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     *
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  unknown   0    0
B                    variable  unknown   0    0
C                    variable  unknown   0    0
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testn                program   _predef   0    0

*   --- Program 'testo.pas' is ---
program testo (input, output);
var A, B, C: integer
begin
A := B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  ;
```

```
                      found:     begin
*** Unexpected Token: expected:  begin
                      found:     id
*** Unexpected Token: expected:  id
                      found:     :=
*** Unexpected Token: expected:  :=
                      found:     id
*** Unexpected Token: expected:  number
                      found:     +
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     *
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testo                program   _predef   12   0

*   --- Program 'testok1.pas' is ---
program testok1(input, output);
var A, B, C: integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*   --------- Parse Successful! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testok1              program   _predef   12   0

*   --- Program 'testok2.pas' is ---
program testok2(input, output);
var A, B, C: integer;
D, E, F: integer;
G, H, I: integer;
begin
A := B + C * 2;
D := E + F * 2;
G := H + I * 2
end.
*   --- Start parsing ---
*   --------- Parse Successful! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
D                    variable  integer   4    12
E                    variable  integer   4    16
F                    variable  integer   4    20
G                    variable  integer   4    24
H                    variable  integer   4    28
I                    variable  integer   4    32
boolean              type      _predef   1    0
```

```
integer             type      _predef   4    0
real                type      _predef   16   0
testok2             program   _predef   36   0

*   --- Program 'testok3.pas' is ---
program testok3(input, output);
var A, B, C: integer;
begin
A := 2 + 6 * 8;
B := 3 + 9 * 1;
C := 4 + 1 * 1
end.
*   --- Start parsing ---
*   --------- Parse Successful! ---------   *

Symbol Table
============
Name                Role      Type      Size Address
--------------------------------------------------
A                   variable  integer   4    0
B                   variable  integer   4    4
C                   variable  integer   4    8
boolean             type      _predef   1    0
integer             type      _predef   4    0
real                type      _predef   16   0
testok3             program   _predef   12   0

*   --- Program 'testok4.pas' is ---
program testok4(input, output);
var A, B, C: integer;
DD, EE, FF: integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*   --------- Parse Successful! ---------   *

Symbol Table
============
Name                Role      Type      Size Address
--------------------------------------------------
A                   variable  integer   4    0
B                   variable  integer   4    4
C                   variable  integer   4    8
DD                  variable  integer   4    12
EE                  variable  integer   4    16
FF                  variable  integer   4    20
boolean             type      _predef   1    0
integer             type      _predef   4    0
real                type      _predef   16   0
testok4             program   _predef   24   0

*   --- Program 'testok5.pas' is ---
program testok5(input, output);
var x, y, z: integer;
A: integer;
B: integer;
C: integer;
begin
x := y + z * 2;
A := B + C * 2
end.
*   --- Start parsing ---
*   --------- Parse Successful! ---------   *

Symbol Table
============
Name                Role      Type      Size Address
--------------------------------------------------
A                   variable  integer   4    12
B                   variable  integer   4    16
C                   variable  integer   4    20
boolean             type      _predef   1    0
integer             type      _predef   4    0
real                type      _predef   16   0
```

```
testok5             program   _predef   24   0
x                   variable  integer   4    0
y                   variable  integer   4    4
z                   variable  integer   4    8

*   --- Program 'testok6.pas' is ---
program testok6(input, output);
var id, number, assign: integer;
begin
id := number + assign * 2
end.
*   --- Start parsing ---
*   --------- Parse Successful! ---------   *

Symbol Table
============
Name                Role      Type      Size Address
--------------------------------------------------
assign              variable  integer   4    8
boolean             type      _predef   1    0
id                  variable  integer   4    0
integer             type      _predef   4    0
number              variable  integer   4    4
real                type      _predef   16   0
testok6             program   _predef   12   0

*   --- Program 'testok7.pas' is ---
program testok7(input, output);
var
A: integer;
B: integer;
C: integer;
begin
A := B + C * 2
end.
*   --- Start parsing ---
*   --------- Parse Successful! ---------   *

Symbol Table
============
Name                Role      Type      Size Address
--------------------------------------------------
A                   variable  integer   4    0
B                   variable  integer   4    4
C                   variable  integer   4    8
boolean             type      _predef   1    0
integer             type      _predef   4    0
real                type      _predef   16   0
testok7             program   _predef   12   0

*   --- Program 'testp.pas' is ---
program testp (input, output);
var A, B, C: integer;
A := B + C * 2
end.
*   --- Start parsing ---
*** Variable A already defined.
*** Unexpected Token: expected:  begin
                      found:     id
*** Unexpected Token: expected:  id
                      found:     :=
*** Variable B already defined.
*** Unexpected Token: expected:  :=
                      found:     id
*** Unexpected Token: expected:  number
                      found:     +
*** Variable C already defined.
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     *
*   --------- Parse failed! ---------   *

Symbol Table
============
```

```
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testp                program   _predef   12   0

*   --- Program 'testq.pas' is ---
program testq (input, output);
var A, B, C: integer;
begin
:= B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  id
                      found:     :=
*** Unexpected Token: expected:  :=
                      found:     id
*** Unexpected Token: expected:  number
                      found:     +
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     *
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testq                program   _predef   12   0

*   --- Program 'testr.pas' is ---
program testr (input, output);
var A, B, C: integer;
begin
A = B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  :=
                      found:     =
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testr                program   _predef   12   0

*   --- Program 'tests.pas' is ---
program tests (input, output);
var A, B, C: integer;
begin
A : B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  :=
                      found:     :
*   --------- Parse failed! ---------   *
```

```
Symbol Table
============
Name                 Role      Type      Size Address
-------------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
tests                program   _predef   12   0

*   --- Program 'testt.pas' is ---
program testt (input, output);
var A, B, C: integer;
begin
A  B + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  :=
                      found:     id
*** Unexpected Token: expected:  number
                      found:     +
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     *
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-------------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testt                program   _predef   12   0

*   --- Program 'testu.pas' is ---
program testu (input, output);
var A, B, C: integer;
begin
A := + C * 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  number
                      found:     +
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     *
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-------------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testu                program   _predef   12   0

*   --- Program 'testv.pas' is ---
program testv (input, output);
var A, B, C: integer;
begin
A := B  C * 2
end.
*   --- Start parsing ---
```

```
*** Unexpected Token: expected:  end
                      found:     id
*** Unexpected Token: expected:  .
                      found:     *
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testv                program   _predef   12   0

*   --- Program 'testw.pas' is ---
program testw (input, output);
var A, B, C: integer;
begin
A := B + C - 2
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  end
                      found:     -
*** Unexpected Token: expected:  .
                      found:     number
*** Mismatch types for +
*** Mismatch types for :=
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testw                program   _predef   12   0

*   --- Program 'testx.pas' is ---
program testx (input, output);
var A, B, C: integer;
begin
A := B + C * 2 ;
end.
*   --- Start parsing ---
*** Unexpected Token: expected:  id
                      found:     end
*** Unexpected Token: expected:  :=
                      found:     .
*** Unexpected Token: expected:  number
                      found:     EOF
*** Unexpected Token: expected:  end
                      found:
*** Unexpected Token: expected:  .
                      found:
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                 Role      Type      Size Address
-----------------------------------------------------
A                    variable  integer   4    0
B                    variable  integer   4    4
C                    variable  integer   4    8
boolean              type      _predef   1    0
integer              type      _predef   4    0
real                 type      _predef   16   0
testx                program   _predef   12   0
```

```
*   --- Program 'testy.pas' is ---
program testy (input, output);
var A, B, C: integer;
begin
A := B + C * 2
.
*   --- Start parsing ---
*** Unexpected Token: expected:  end
                      found:     .
*** Unexpected Token: expected:  .
                      found:     EOF
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                  Role      Type      Size Address
-----------------------------------------------------
A                     variable  integer   4    0
B                     variable  integer   4    4
C                     variable  integer   4    8
boolean               type      _predef   1    0
integer               type      _predef   4    0
real                  type      _predef   16   0
testy                 program   _predef   12   0

*   --- Program 'testz.pas' is ---
program testz (input, output);
var A, B, C: integer;
begin
A := B + C * 2
end
*   --- Start parsing ---
*** Unexpected Token: expected:  .
                      found:     EOF
*   --------- Parse failed! ---------   *

Symbol Table
============
Name                  Role      Type      Size Address
-----------------------------------------------------
A                     variable  integer   4    0
B                     variable  integer   4    4
C                     variable  integer   4    8
boolean               type      _predef   1    0
integer               type      _predef   4    0
real                  type      _predef   16   0
testz                 program   _predef   12   0
```