Студент: Ивенкова Л.В.
Группа: М8О-208Б-19
Номер по списку: 11

«СИСТЕМЫ ПРОГРАММИРОВАНИЯ»
Курсовая работа 2021.
Часть 2.

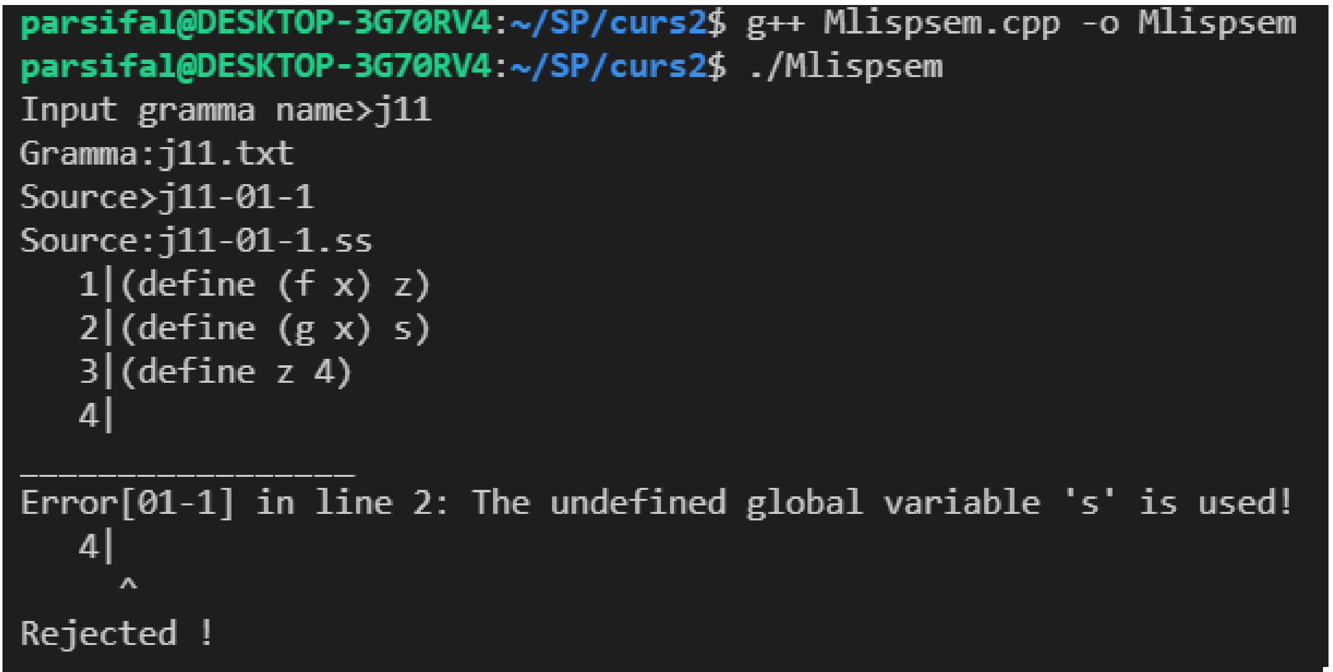
Для заданного в Лабораторной №8 диалекта языка МИКРОЛИСП разработайте семантический анализатор, применяя методику Лабораторной №10, Правила SemanticRules.rtf и MessageForms.rtf

Шаблон файла semantics.cpp создайте с помощью приложения Make-semantics.cpp .

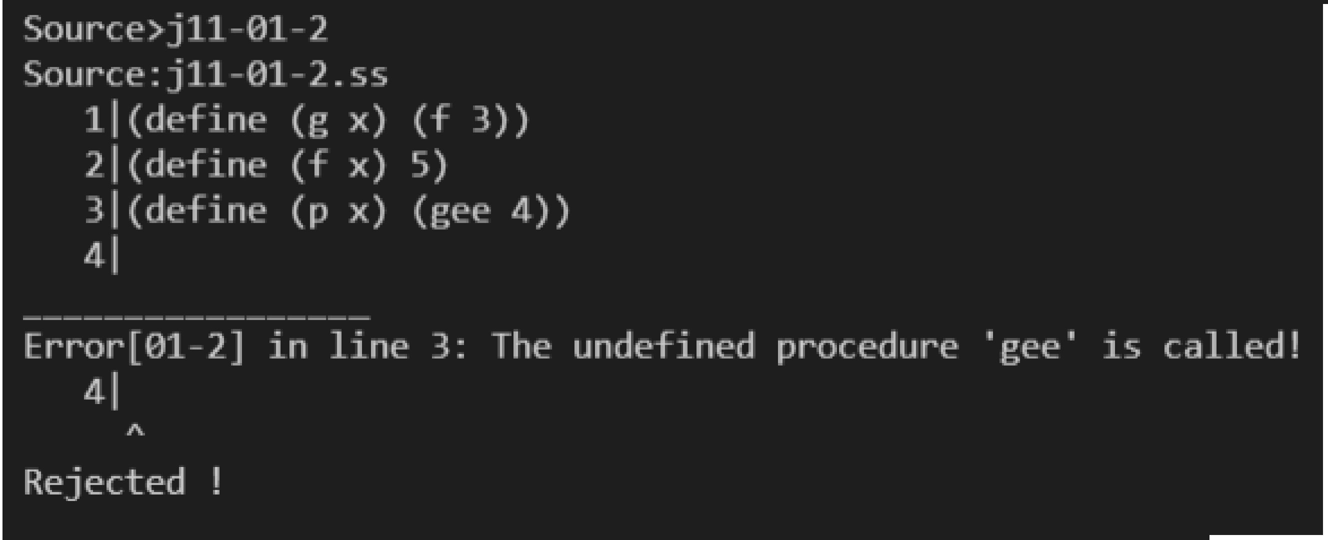
Разработайте сценарии тестирования алгоритмов анализа.

Перечень документов в отчете.
Вариант грамматики: j11

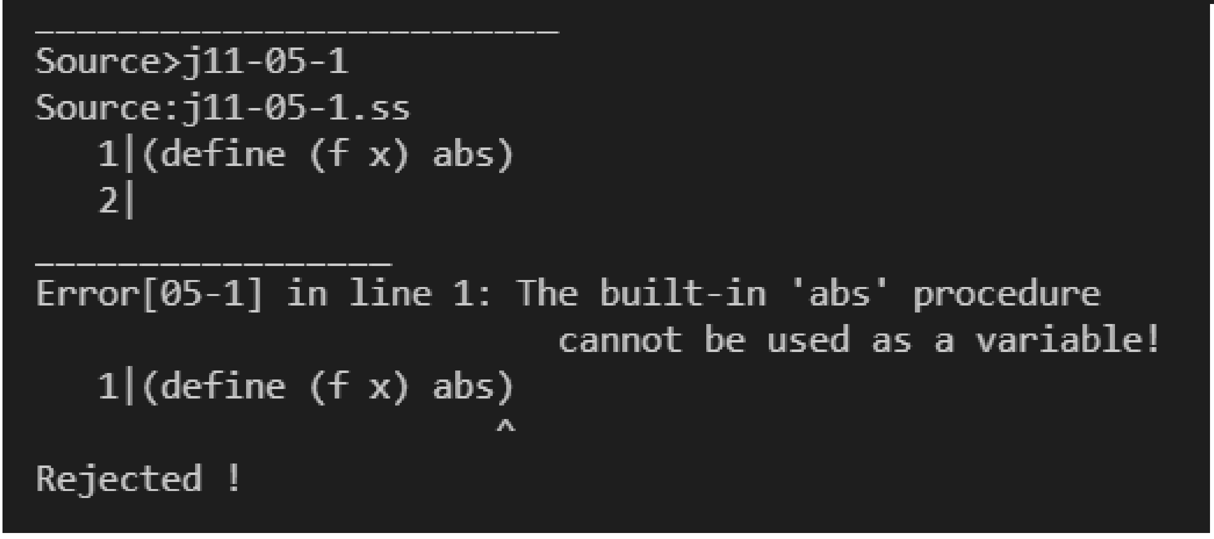
Скриншоты всех тестов, упорядоченные по номерам продукций и сообщений.
>



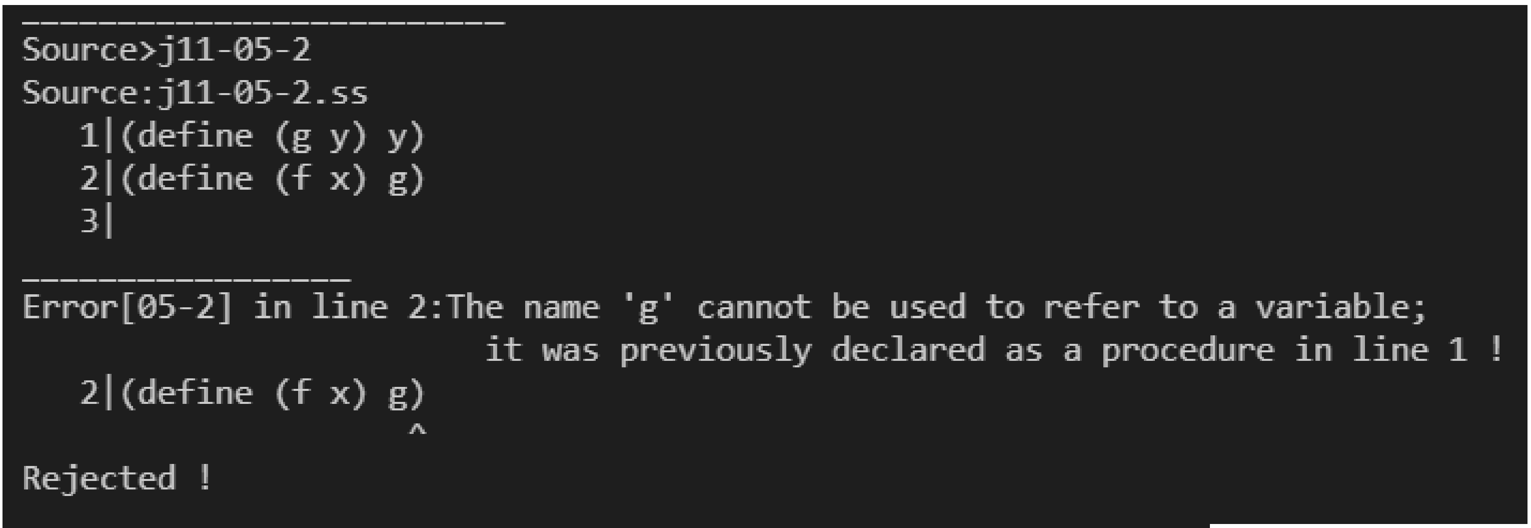
```
parsifal@DESKTOP-3G70RV4:~/SP/curs2$ g++ Mlispsem.cpp -o Mlispsem
parsifal@DESKTOP-3G70RV4:~/SP/curs2$ ./Mlispsem
Input gramma name>j11
Gramma:j11.txt
Source>j11-01-1
Source:j11-01-1.ss
   1|(define (f x) z)
   2|(define (g x) s)
   3|(define z 4)
   4|

_________________
Error[01-1] in line 2: The undefined global variable 's' is used!
   4|
     ^
Rejected !
```



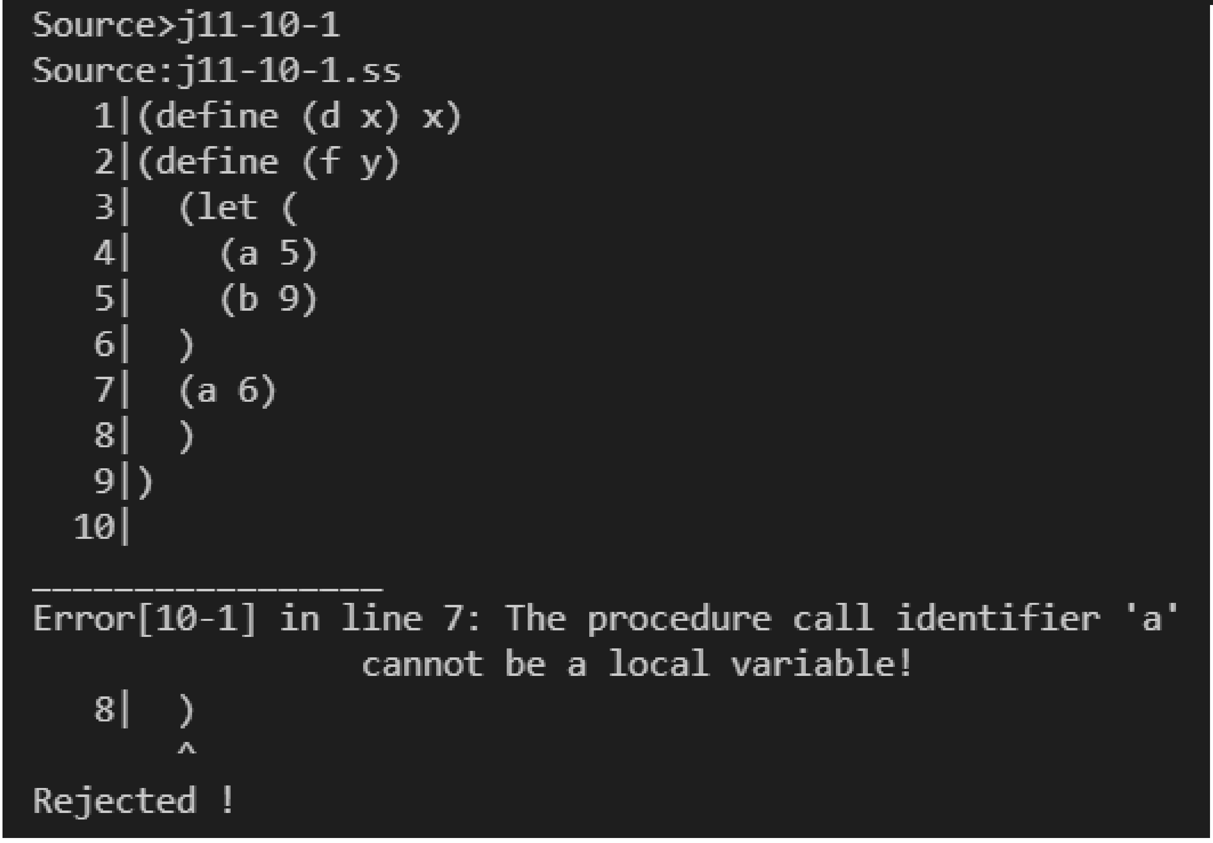
```
Source>j11-01-2
Source:j11-01-2.ss
   1|(define (g x) (f 3))
   2|(define (f x) 5)
   3|(define (p x) (gee 4))
   4|

_________________
Error[01-2] in line 3: The undefined procedure 'gee' is called!
   4|
     ^
Rejected !
```



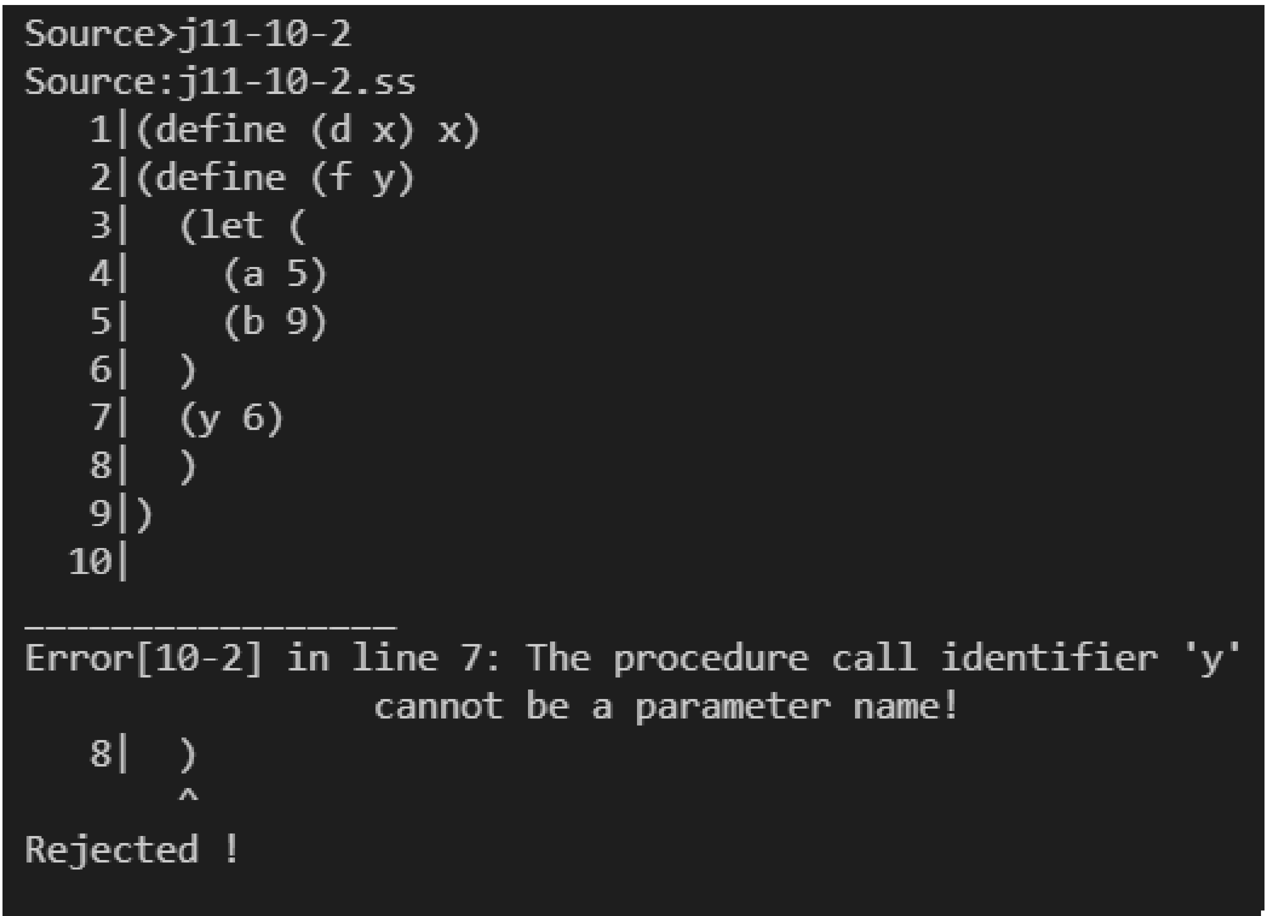
```
_________________________
Source>j11-05-1
Source:j11-05-1.ss
   1|(define (f x) abs)
   2|

_________________
Error[05-1] in line 1: The built-in 'abs' procedure
                         cannot be used as a variable!
   1|(define (f x) abs)
                     ^
Rejected !
```



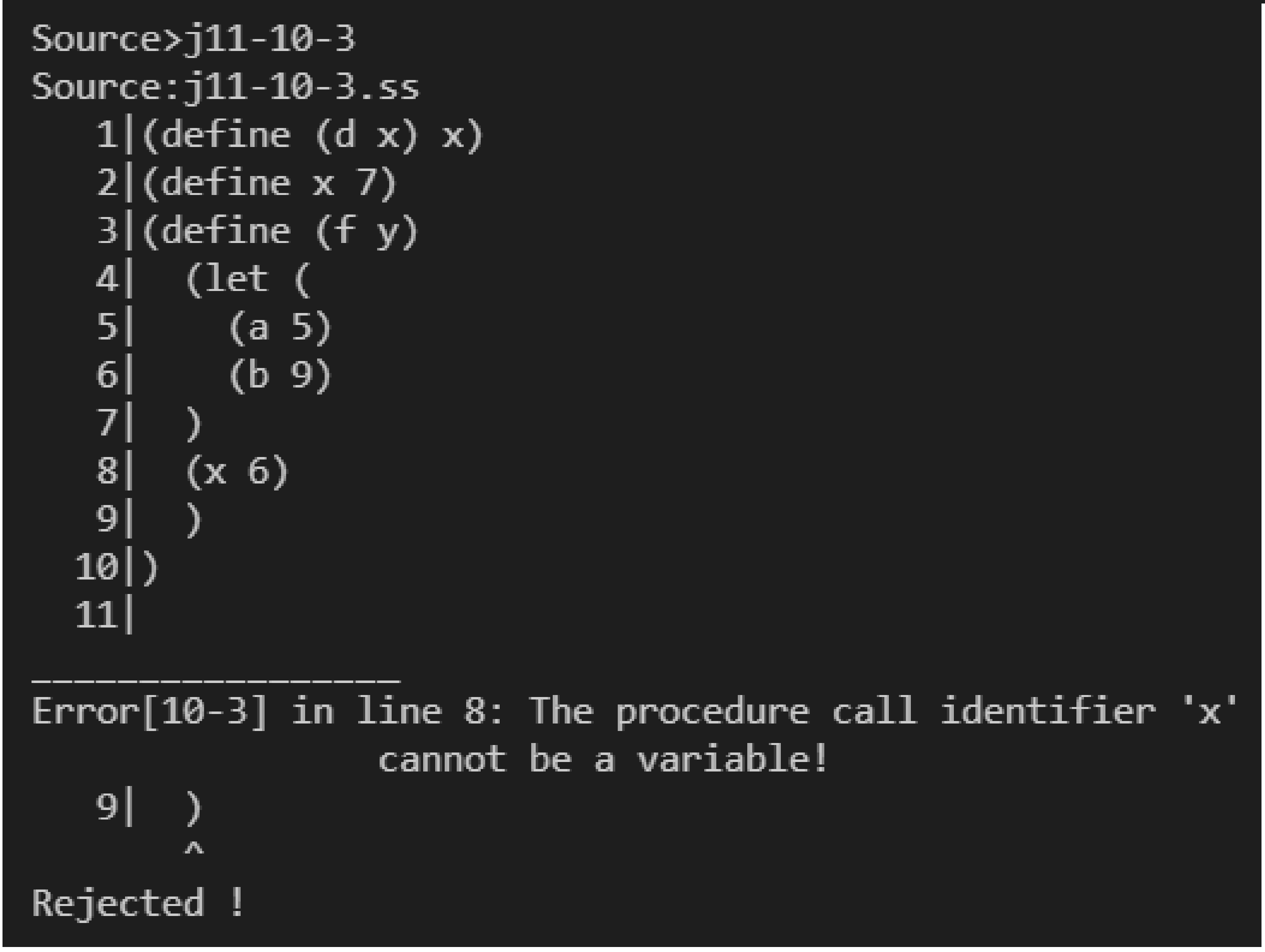
```
---------------------------
Source>j11-05-2
Source:j11-05-2.ss
   1|(define (g y) y)
   2|(define (f x) g)
   3|

------------------
Error[05-2] in line 2:The name 'g' cannot be used to refer to a variable;
                        it was previously declared as a procedure in line 1 !
   2|(define (f x) g)
                    ^
Rejected !
```



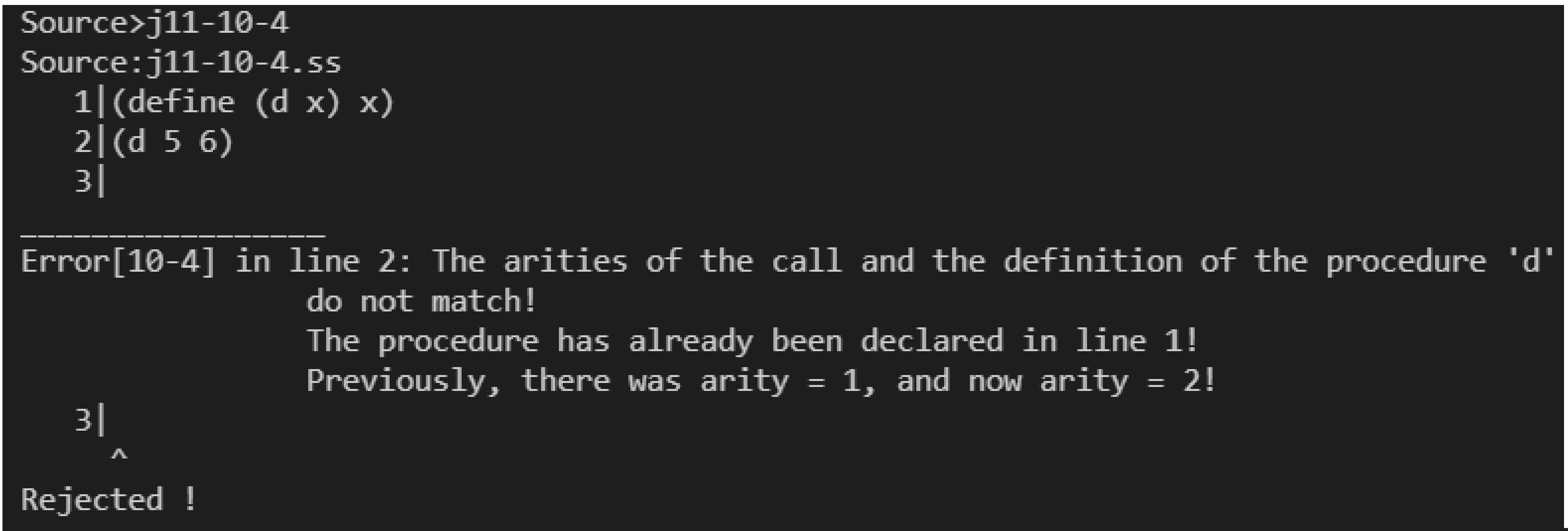
```
Source>j11-10-1
Source:j11-10-1.ss
   1|(define (d x) x)
   2|(define (f y)
   3|  (let (
   4|    (a 5)
   5|    (b 9)
   6|  )
   7|  (a 6)
   8|  )
   9|)
  10|

-----------------
Error[10-1] in line 7: The procedure call identifier 'a'
                cannot be a local variable!
   8|  )
       ^
Rejected !
```



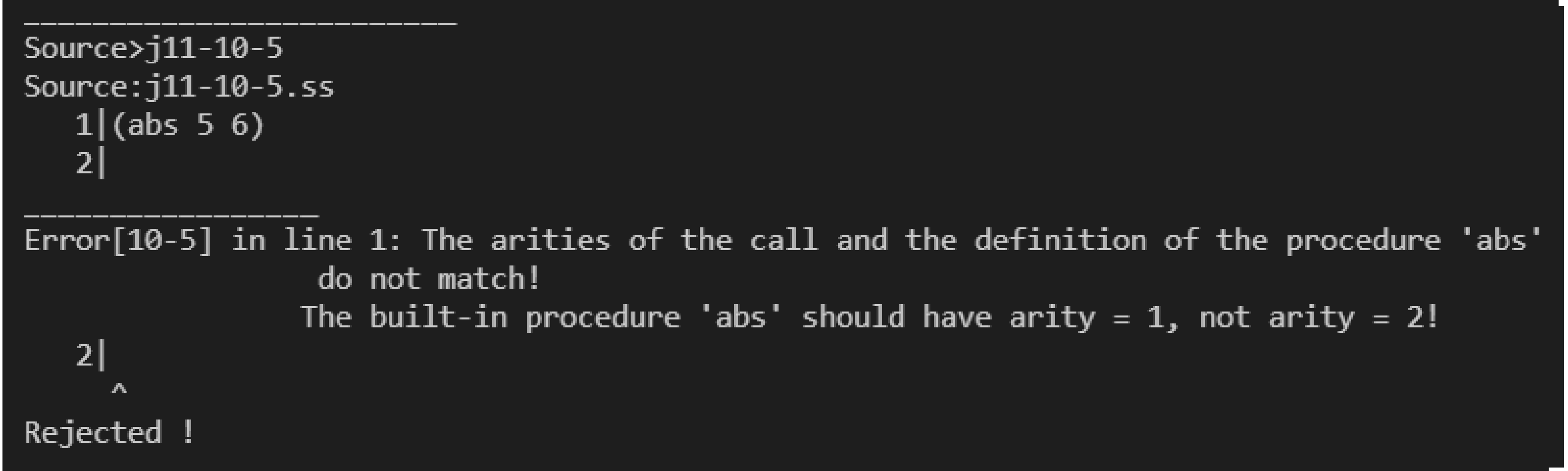
```
Source>j11-10-2
Source:j11-10-2.ss
   1|(define (d x) x)
   2|(define (f y)
   3|  (let (
   4|    (a 5)
   5|    (b 9)
   6|  )
   7|  (y 6)
   8|  )
   9|)
  10|

_________________
Error[10-2] in line 7: The procedure call identifier 'y'
                cannot be a parameter name!
   8|  )
       ^
Rejected !
```



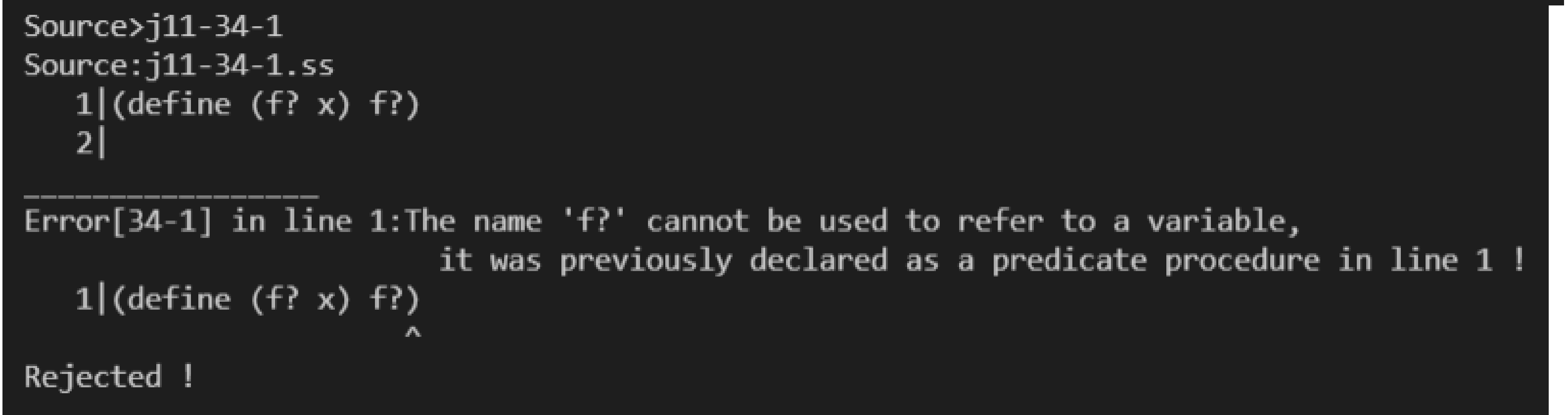
```
Source>j11-10-3
Source:j11-10-3.ss
   1|(define (d x) x)
   2|(define x 7)
   3|(define (f y)
   4|  (let (
   5|    (a 5)
   6|    (b 9)
   7|  )
   8|  (x 6)
   9|  )
  10|)
  11|

_________________
Error[10-3] in line 8: The procedure call identifier 'x'
                cannot be a variable!
   9|  )
       ^
Rejected !
```



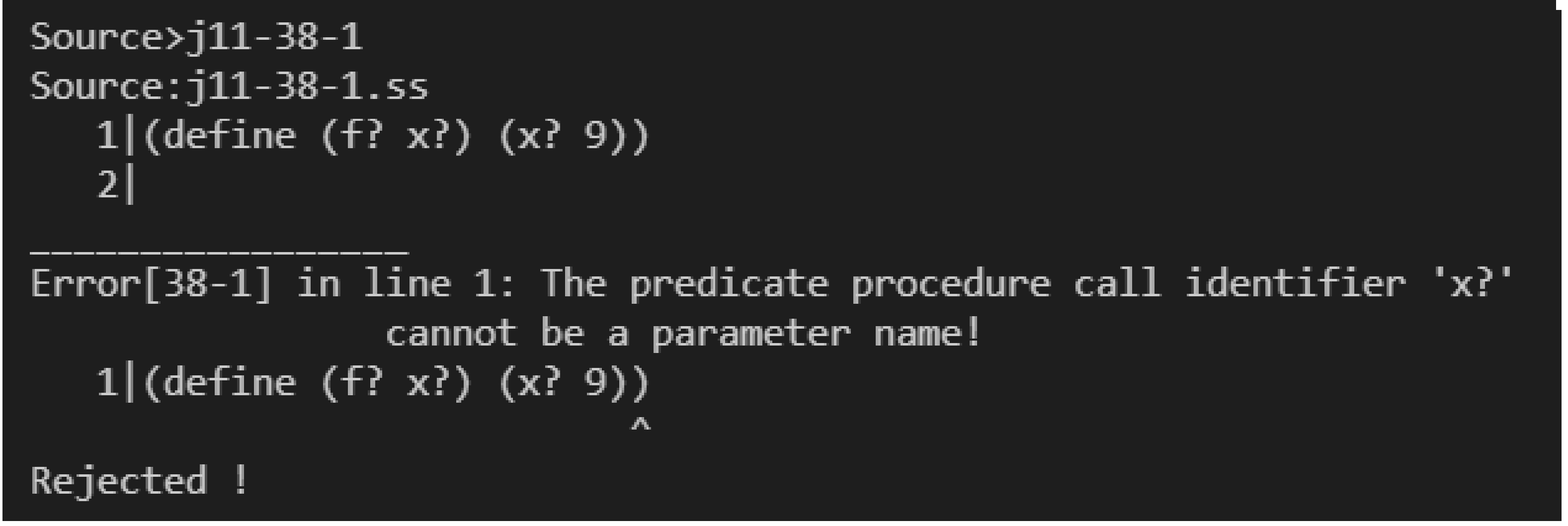
```
Source>j11-10-4
Source:j11-10-4.ss
   1|(define (d x) x)
   2|(d 5 6)
   3|

_________________
Error[10-4] in line 2: The arities of the call and the definition of the procedure 'd'
                do not match!
                The procedure has already been declared in line 1!
                Previously, there was arity = 1, and now arity = 2!
   3|
     ^
Rejected !
```



```
_________________________
Source>j11-10-5
Source:j11-10-5.ss
   1|(abs 5 6)
   2|

_________________
Error[10-5] in line 1: The arities of the call and the definition of the procedure 'abs'
                do not match!
               The built-in procedure 'abs' should have arity = 1, not arity = 2!
   2|
     ^
Rejected !
```



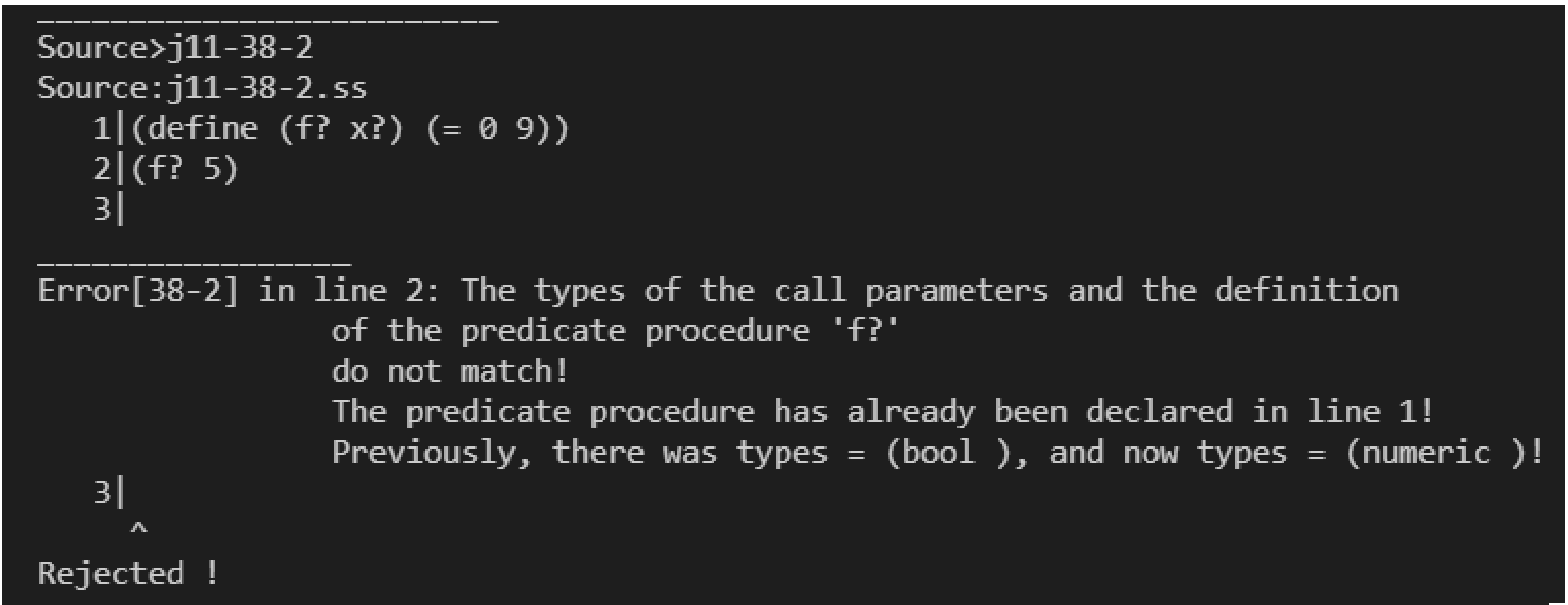
```
Source>j11-34-1
Source:j11-34-1.ss
   1|(define (f? x) f?)
   2|

_________________
Error[34-1] in line 1:The name 'f?' cannot be used to refer to a variable,
                       it was previously declared as a predicate procedure in line 1 !
   1|(define (f? x) f?)
                      ^
Rejected !
```



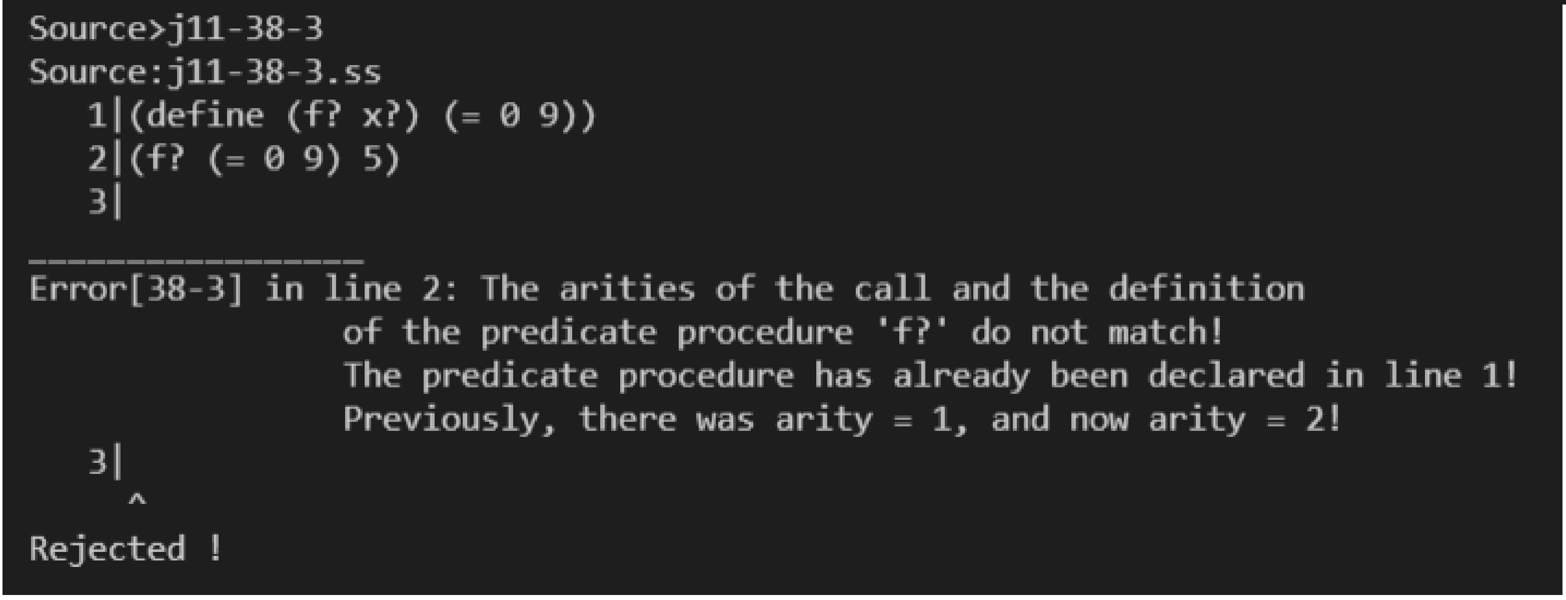
```
Source>j11-38-1
Source:j11-38-1.ss
   1|(define (f? x?) (x? 9))
   2|

_________________
Error[38-1] in line 1: The predicate procedure call identifier 'x?'
                cannot be a parameter name!
   1|(define (f? x?) (x? 9))
                          ^
Rejected !
```

/* При конфликтующих арностях я не смотрела на "лишние" параметры и сравнивала только общее число типов параметров (начиная с начала), и, если они не

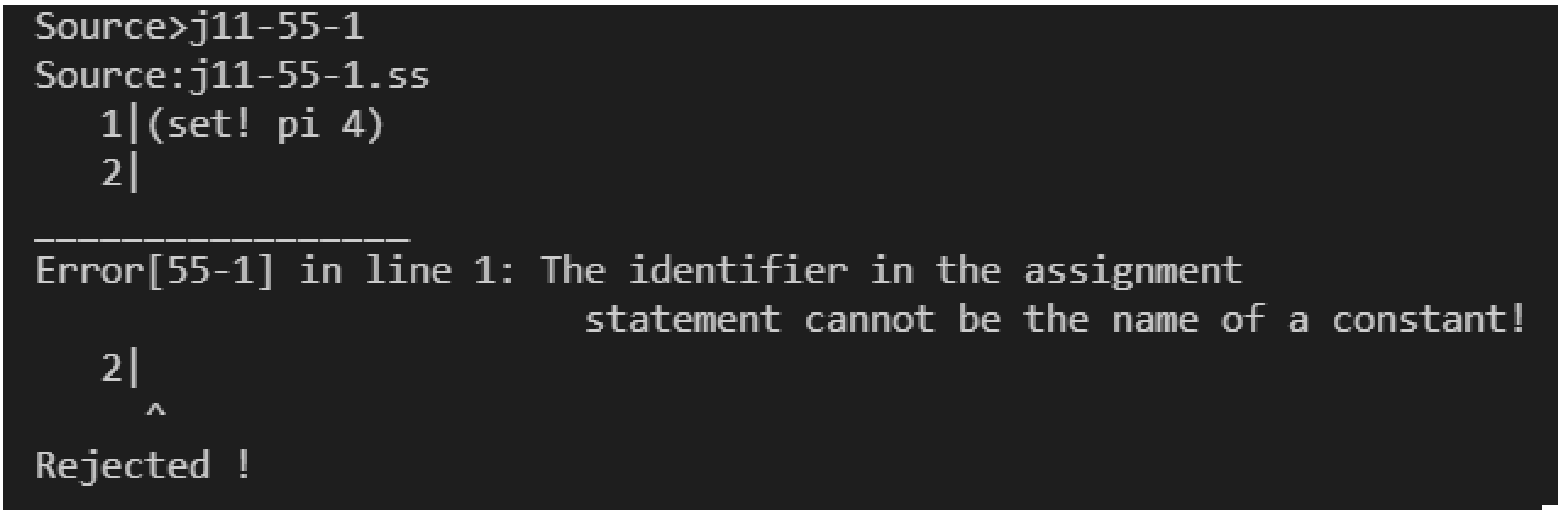
совпадали, то считала это несовпадением типов параметров.
Numeric – численный тип параметров. */



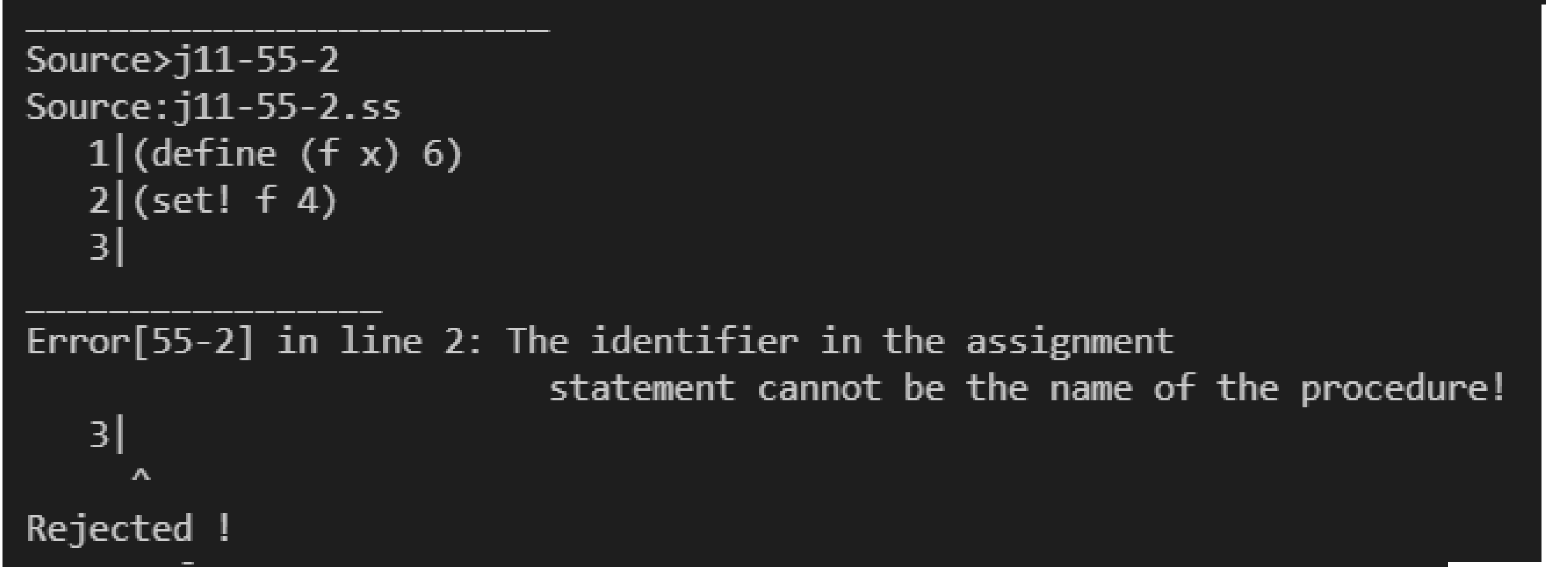
```
 -------------------------
 Source>j11-38-2
 Source:j11-38-2.ss
    1|(define (f? x?) (= 0 9))
    2|(f? 5)
    3|

 -----------------
 Error[38-2] in line 2: The types of the call parameters and the definition
                 of the predicate procedure 'f?'
                 do not match!
                 The predicate procedure has already been declared in line 1!
                 Previously, there was types = (bool ), and now types = (numeric )!
    3|
      ^
 Rejected !
```



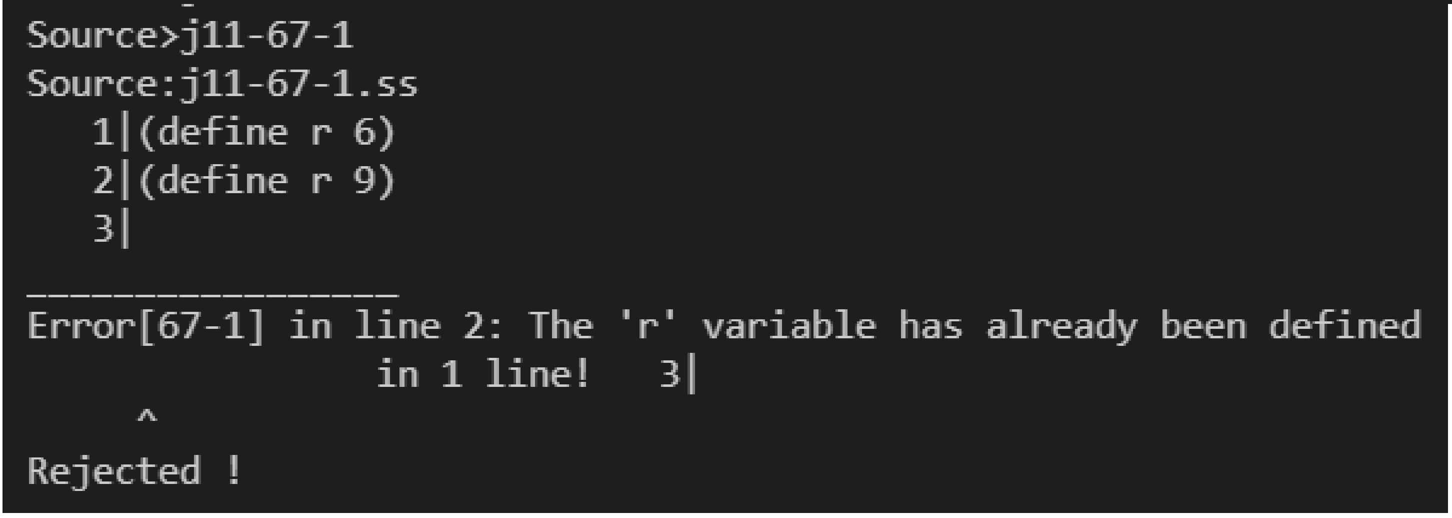
```
Source>j11-38-3
Source:j11-38-3.ss
   1|(define (f? x?) (= 0 9))
   2|(f? (= 0 9) 5)
   3|

------------------
Error[38-3] in line 2: The arities of the call and the definition
                of the predicate procedure 'f?' do not match!
                The predicate procedure has already been declared in line 1!
                Previously, there was arity = 1, and now arity = 2!
   3|
     ^
Rejected !
```



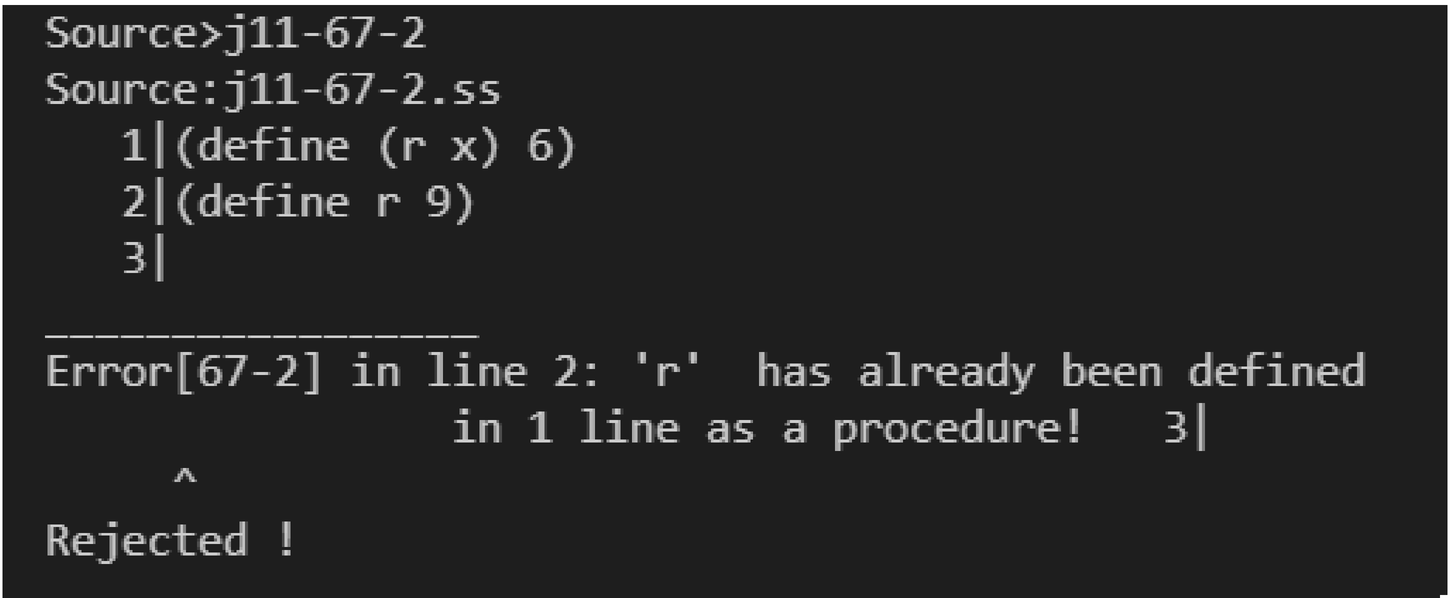
```
Source>j11-55-1
Source:j11-55-1.ss
   1|(set! pi 4)
   2|

_________________
Error[55-1] in line 1: The identifier in the assignment
                         statement cannot be the name of a constant!
   2|
     ^
Rejected !
```



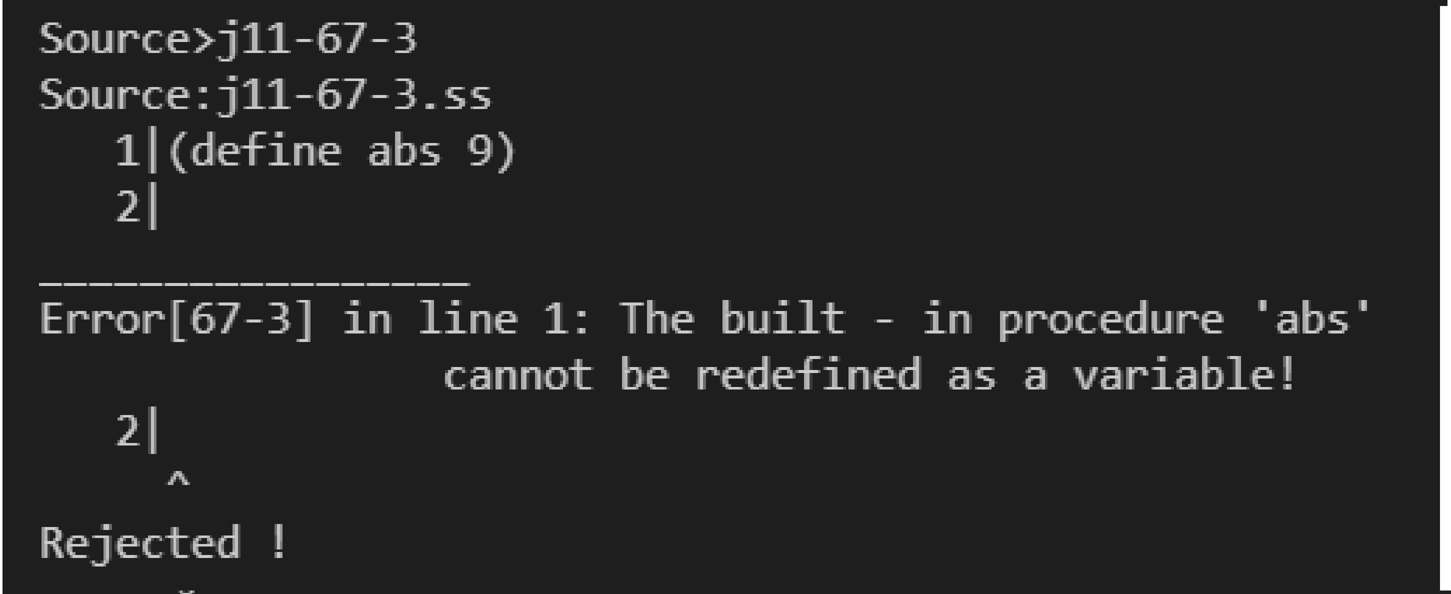
```
________________________
Source>j11-55-2
Source:j11-55-2.ss
   1|(define (f x) 6)
   2|(set! f 4)
   3|

_________________
Error[55-2] in line 2: The identifier in the assignment
                         statement cannot be the name of the procedure!
   3|
     ^
Rejected !
```



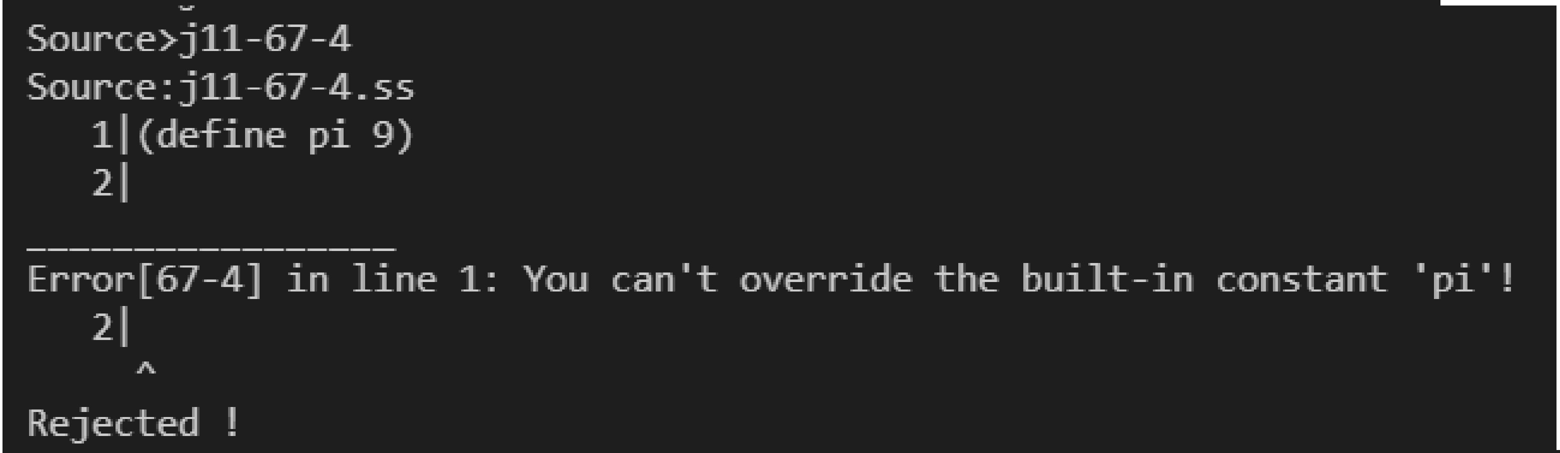
```
Source>j11-67-1
Source:j11-67-1.ss
   1|(define r 6)
   2|(define r 9)
   3|

_________________
Error[67-1] in line 2: The 'r' variable has already been defined
                 in 1 line!   3|
     ^
Rejected !
```



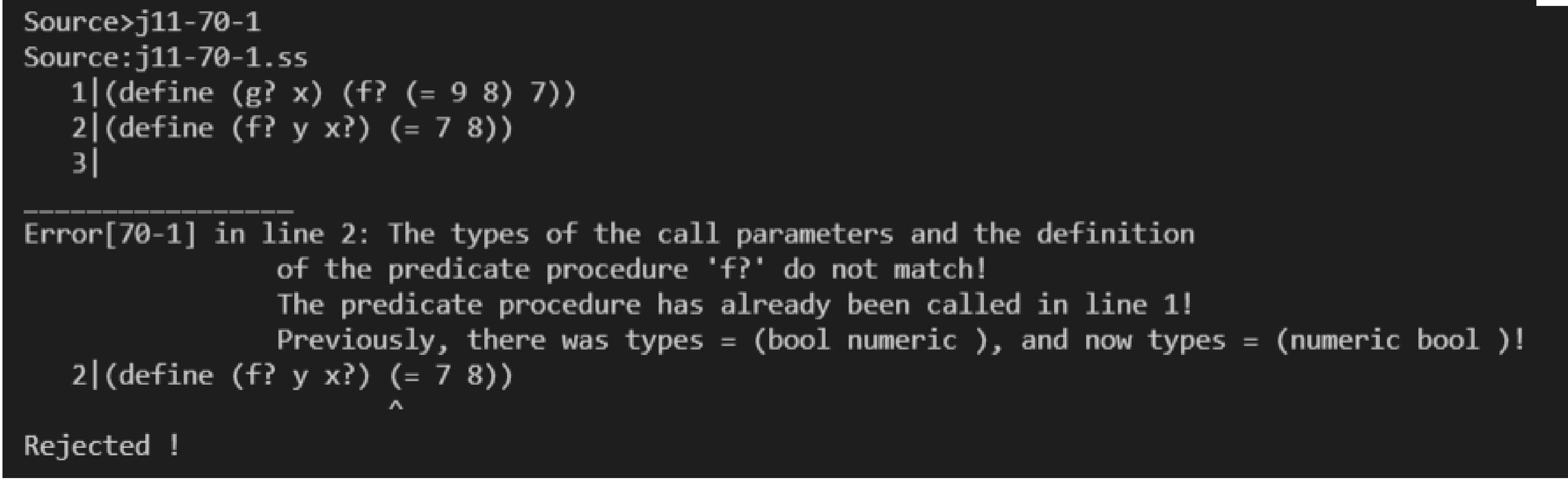
```
Source>j11-67-2
Source:j11-67-2.ss
   1|(define (r x) 6)
   2|(define r 9)
   3|

_________________
Error[67-2] in line 2: 'r'  has already been defined
                in 1 line as a procedure!   3|
     ^
Rejected !
```



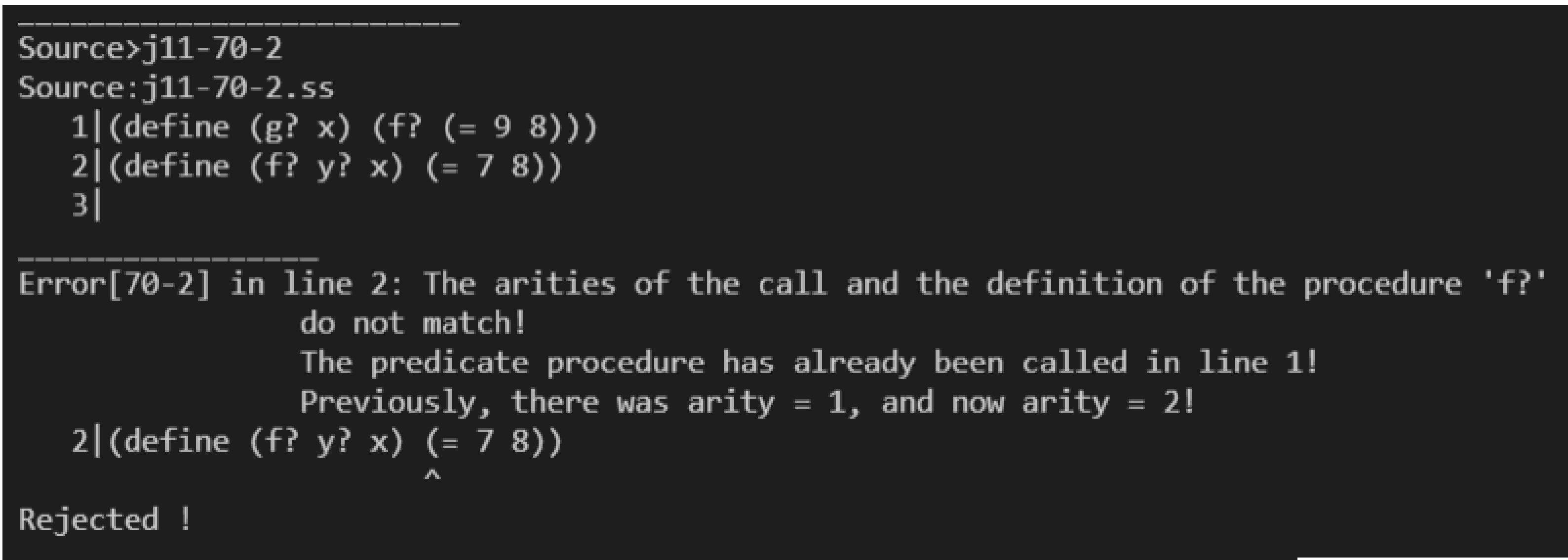
```
Source>j11-67-3
Source:j11-67-3.ss
   1|(define abs 9)
   2|

_________________
Error[67-3] in line 1: The built - in procedure 'abs'
                cannot be redefined as a variable!
   2|
     ^
Rejected !
```



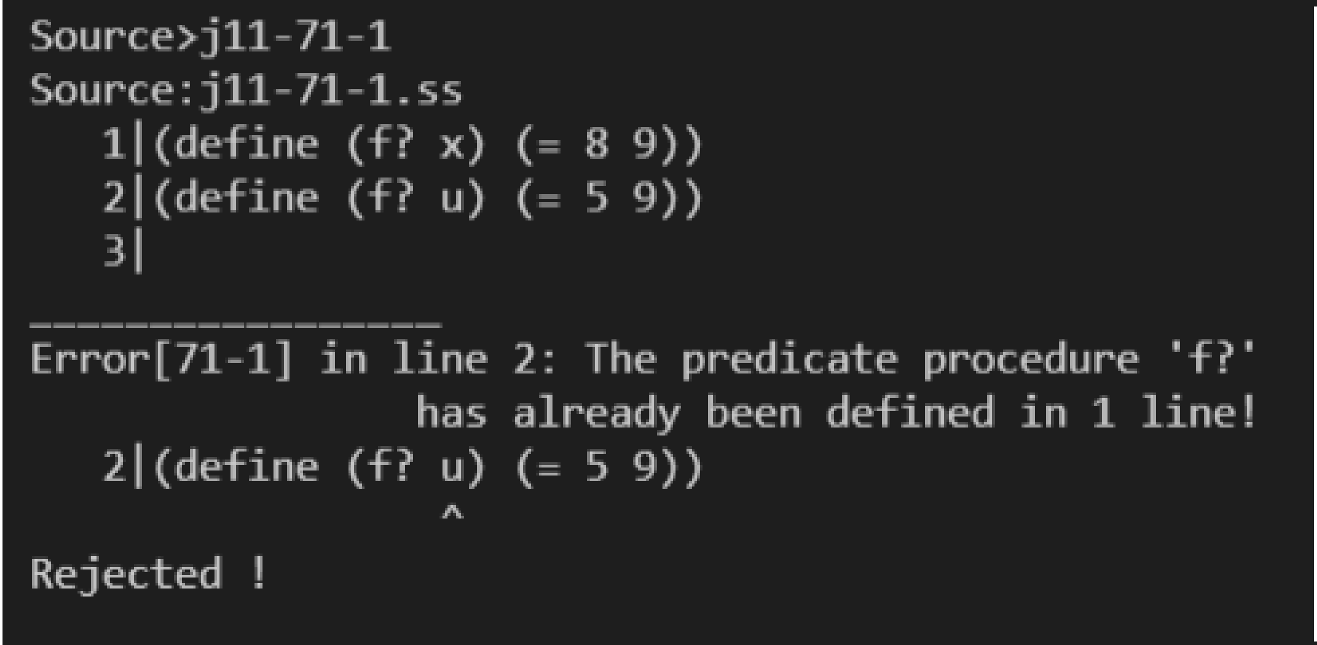
```
Source>j11-67-4
Source:j11-67-4.ss
   1|(define pi 9)
   2|

_________________
Error[67-4] in line 1: You can't override the built-in constant 'pi'!
   2|
     ^
Rejected !
```



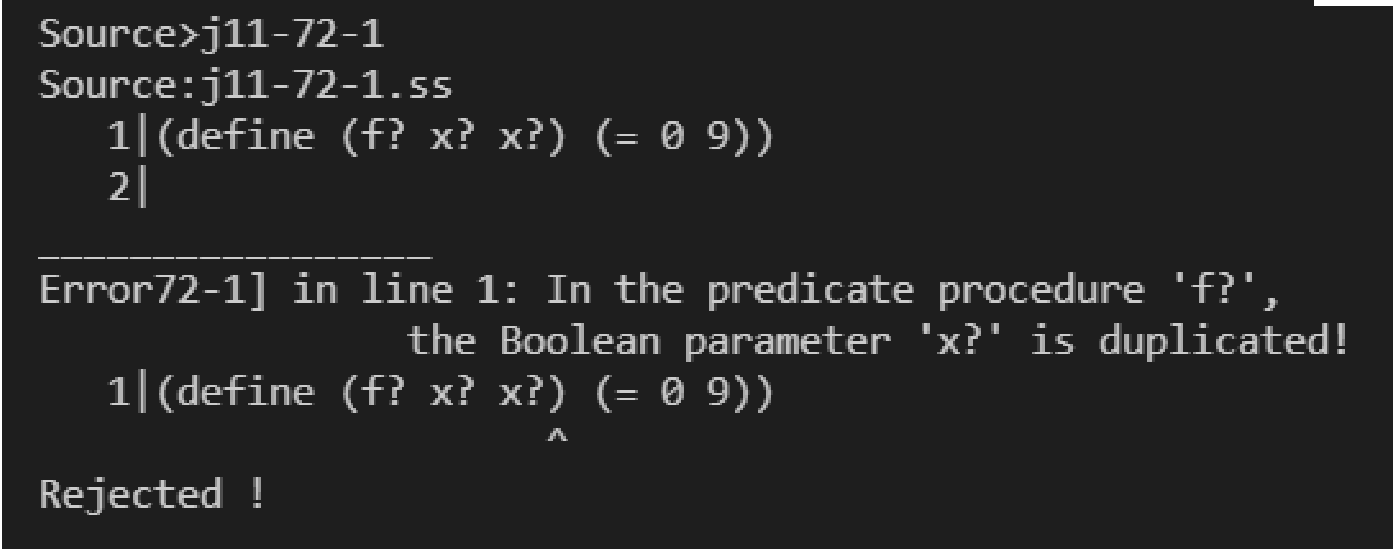
```
Source>j11-70-1
Source:j11-70-1.ss
   1|(define (g? x) (f? (= 9 8) 7))
   2|(define (f? y x?) (= 7 8))
   3|

_________________
Error[70-1] in line 2: The types of the call parameters and the definition
                of the predicate procedure 'f?' do not match!
                The predicate procedure has already been called in line 1!
                Previously, there was types = (bool numeric ), and now types = (numeric bool )!
   2|(define (f? y x?) (= 7 8))
                      ^
Rejected !
```



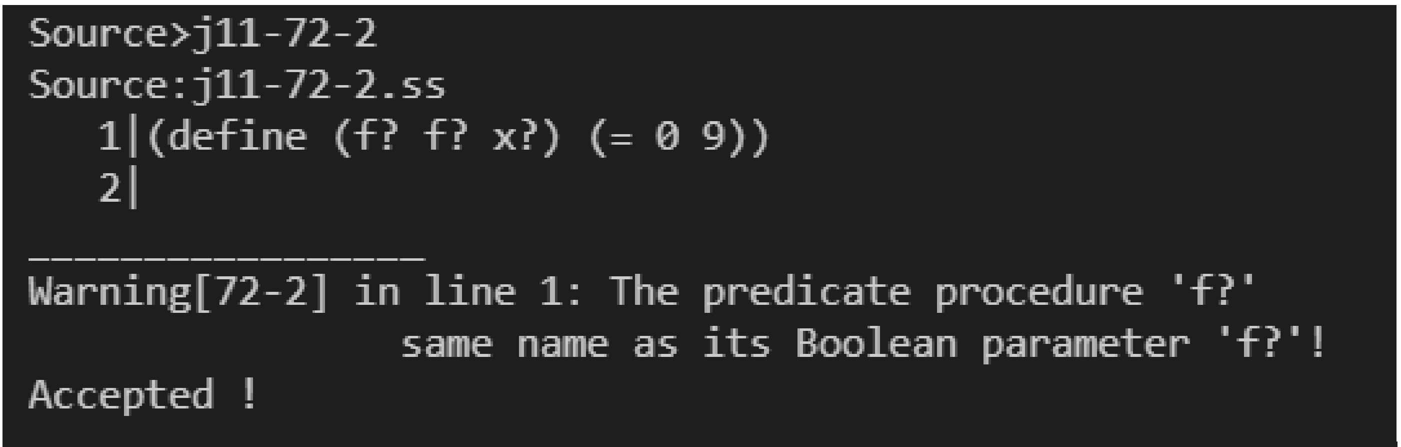
```
-------------------------
Source>j11-70-2
Source:j11-70-2.ss
   1|(define (g? x) (f? (= 9 8)))
   2|(define (f? y? x) (= 7 8))
   3|

-----------------
Error[70-2] in line 2: The arities of the call and the definition of the procedure 'f?'
                do not match!
                The predicate procedure has already been called in line 1!
                Previously, there was arity = 1, and now arity = 2!
   2|(define (f? y? x) (= 7 8))
                       ^
Rejected !
```



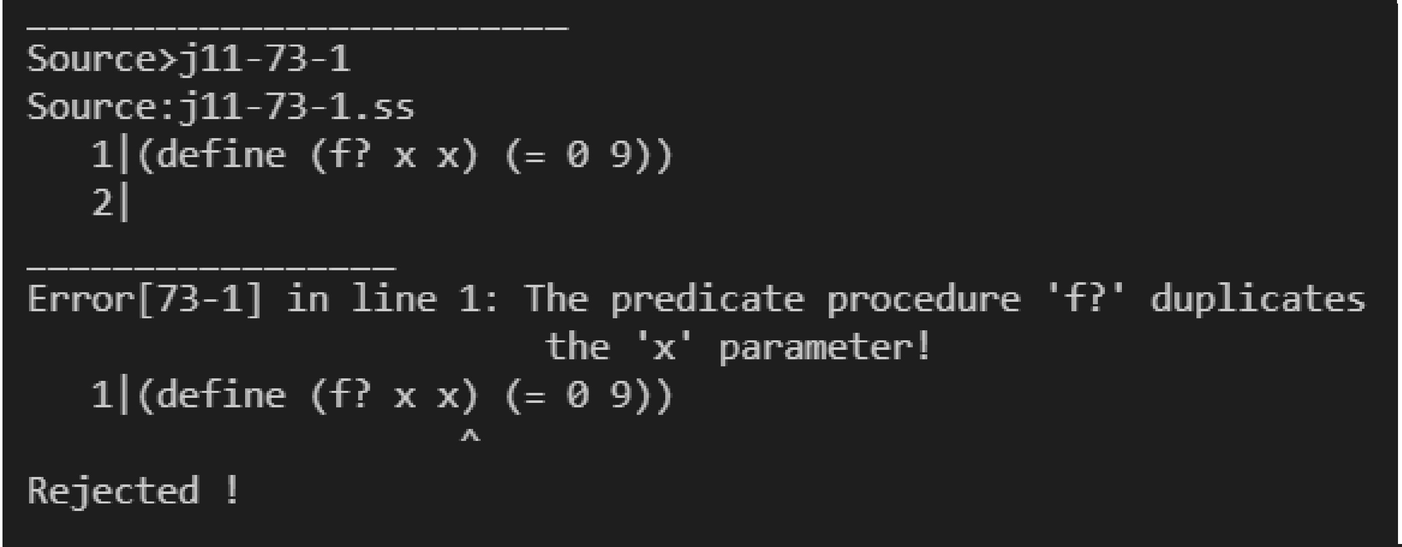
```
Source>j11-71-1
Source:j11-71-1.ss
   1|(define (f? x) (= 8 9))
   2|(define (f? u) (= 5 9))
   3|

-------------------
Error[71-1] in line 2: The predicate procedure 'f?'
                   has already been defined in 1 line!
   2|(define (f? u) (= 5 9))
                   ^
Rejected !
```



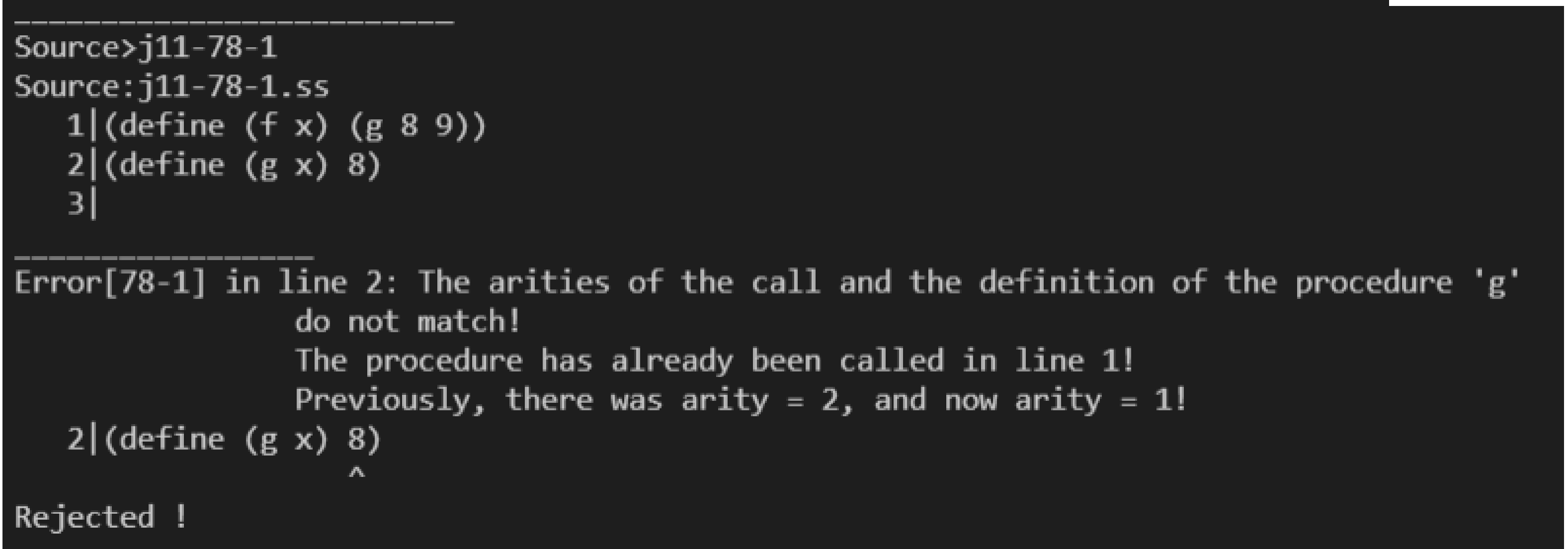
```
Source>j11-72-1
Source:j11-72-1.ss
   1|(define (f? x? x?) (= 0 9))
   2|

-------------------
Error72-1] in line 1: In the predicate procedure 'f?',
                   the Boolean parameter 'x?' is duplicated!
   1|(define (f? x? x?) (= 0 9))
                    ^
Rejected !
```



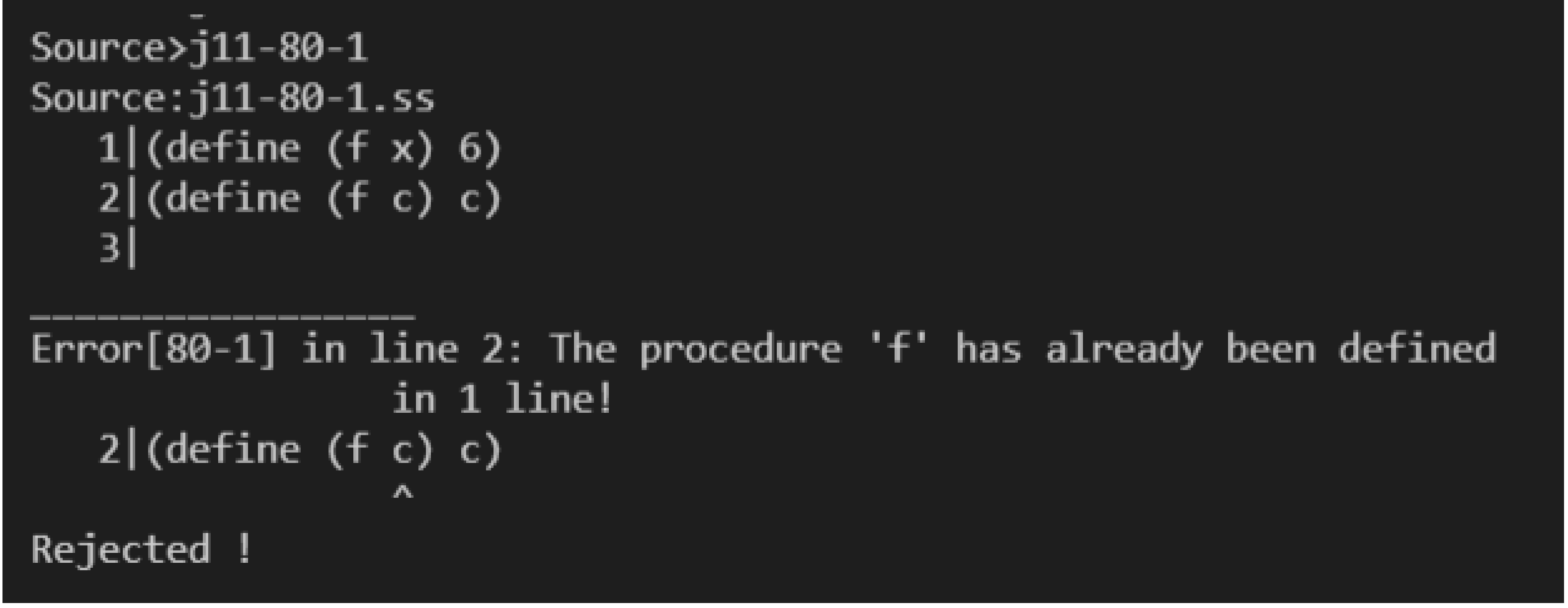
```
Source>j11-72-2
Source:j11-72-2.ss
   1|(define (f? f? x?) (= 0 9))
   2|

_________________
Warning[72-2] in line 1: The predicate procedure 'f?'
                same name as its Boolean parameter 'f?'!
Accepted !
```



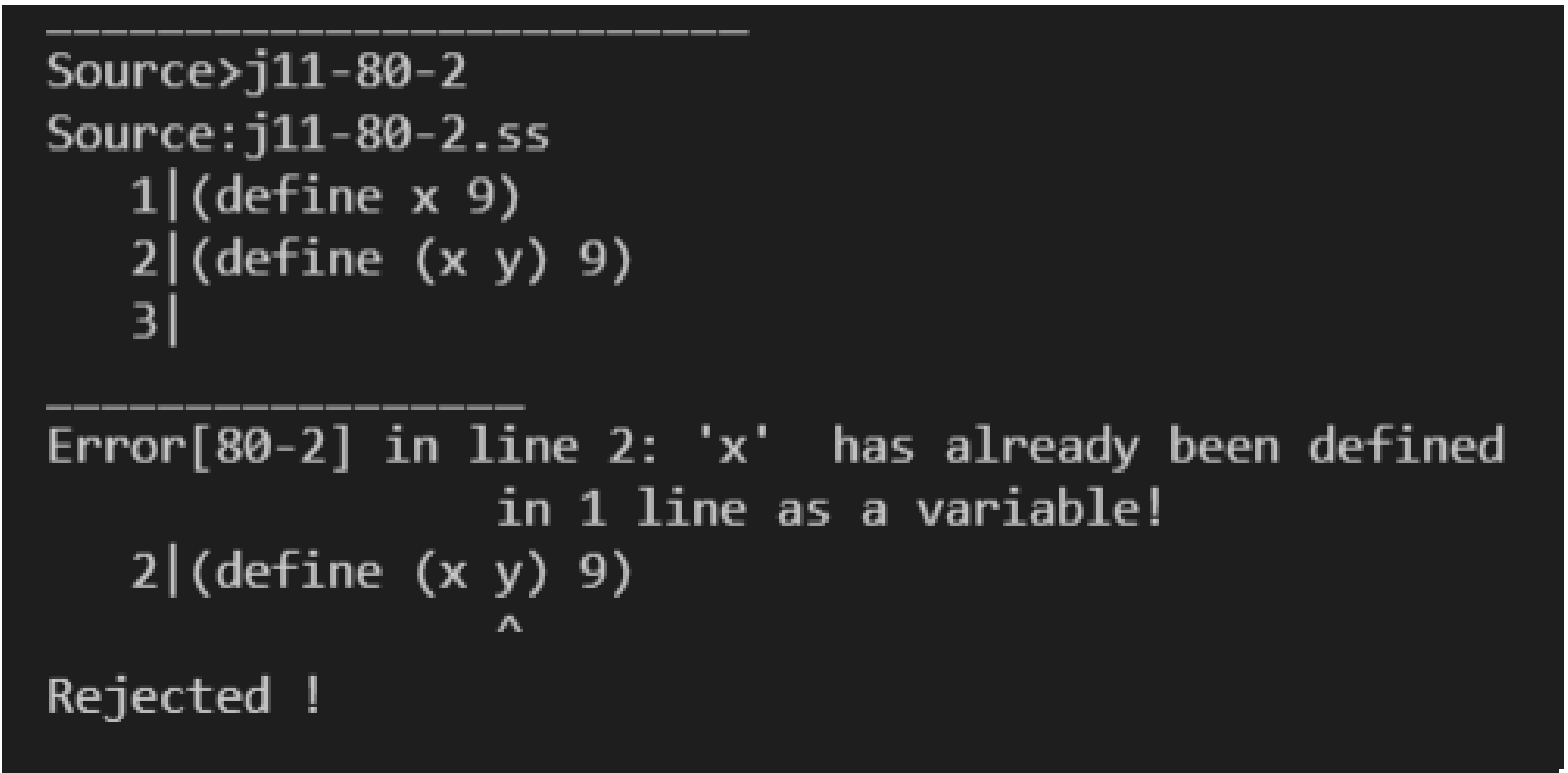
```
________________________
Source>j11-73-1
Source:j11-73-1.ss
   1|(define (f? x x) (= 0 9))
   2|

_________________
Error[73-1] in line 1: The predicate procedure 'f?' duplicates
                        the 'x' parameter!
   1|(define (f? x x) (= 0 9))
                    ^
Rejected !
```



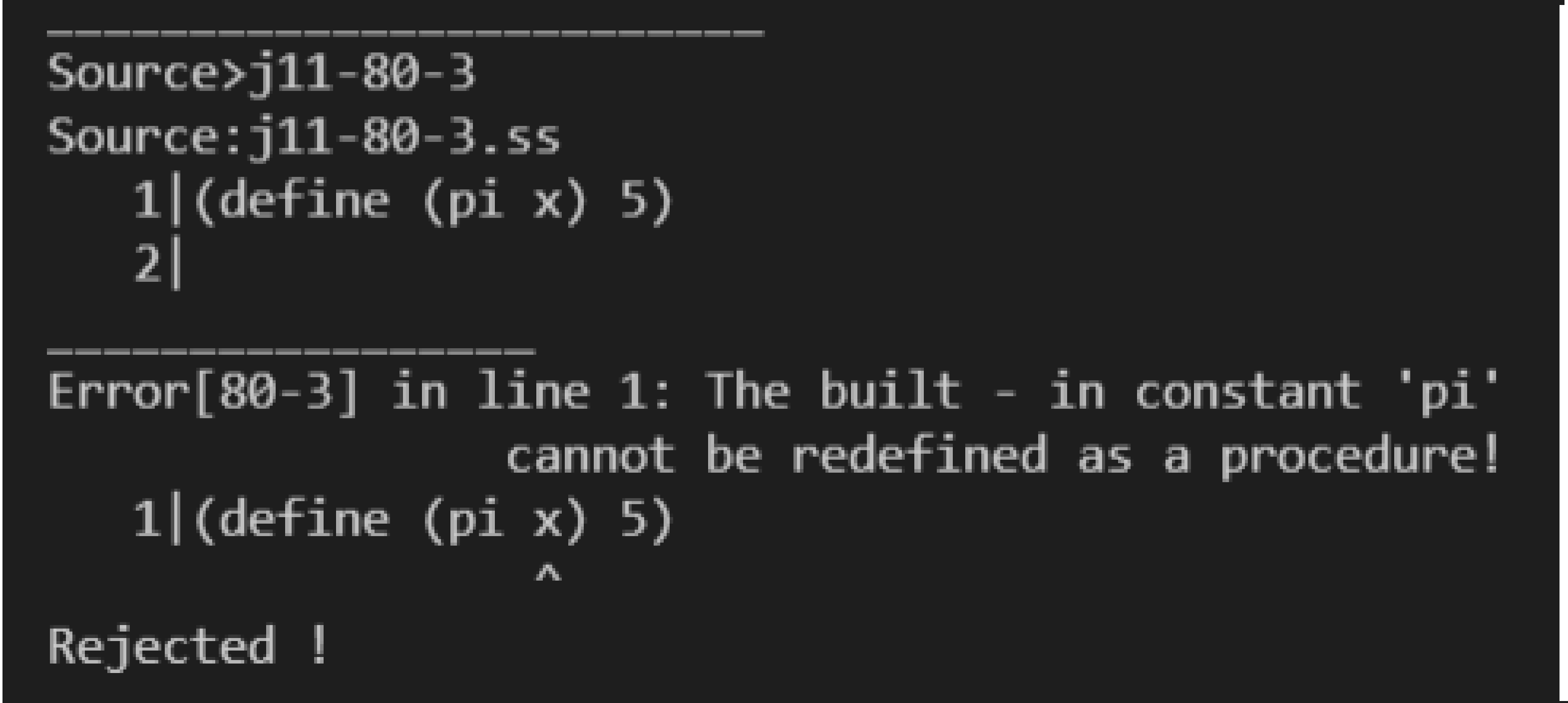
```
________________________
Source>j11-78-1
Source:j11-78-1.ss
   1|(define (f x) (g 8 9))
   2|(define (g x) 8)
   3|

_________________
Error[78-1] in line 2: The arities of the call and the definition of the procedure 'g'
                do not match!
                The procedure has already been called in line 1!
                Previously, there was arity = 2, and now arity = 1!
   2|(define (g x) 8)
                   ^
Rejected !
```



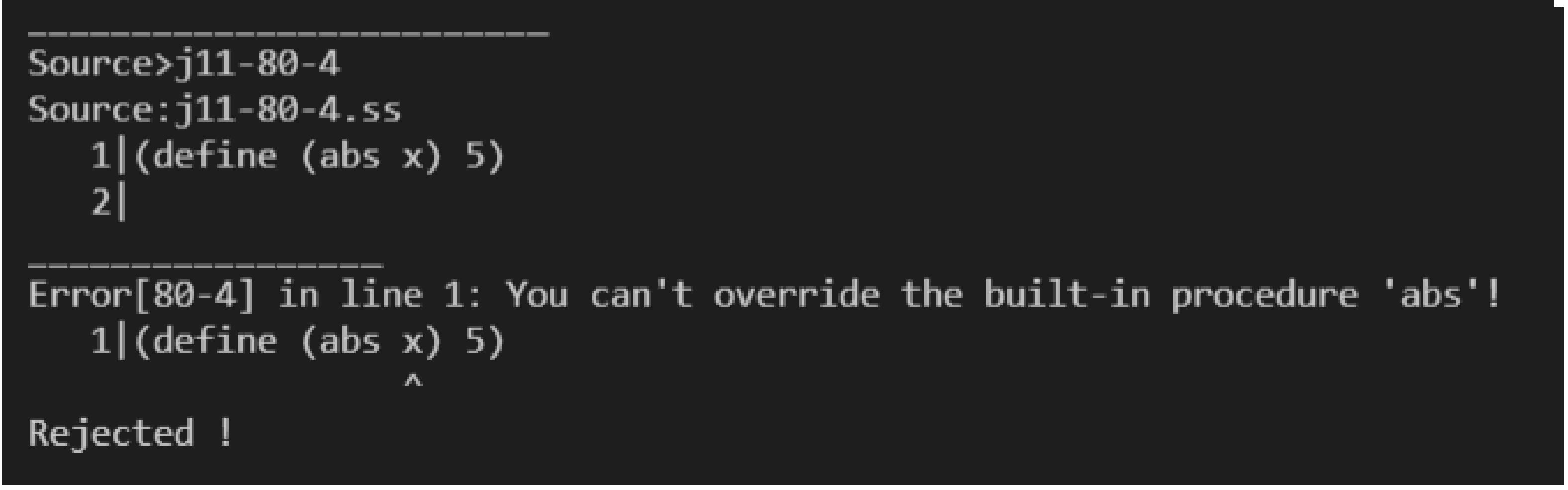
```
Source>j11-80-1
Source:j11-80-1.ss
   1|(define (f x) 6)
   2|(define (f c) c)
   3|

_________________
Error[80-1] in line 2: The procedure 'f' has already been defined
                in 1 line!
   2|(define (f c) c)
                 ^
Rejected !
```



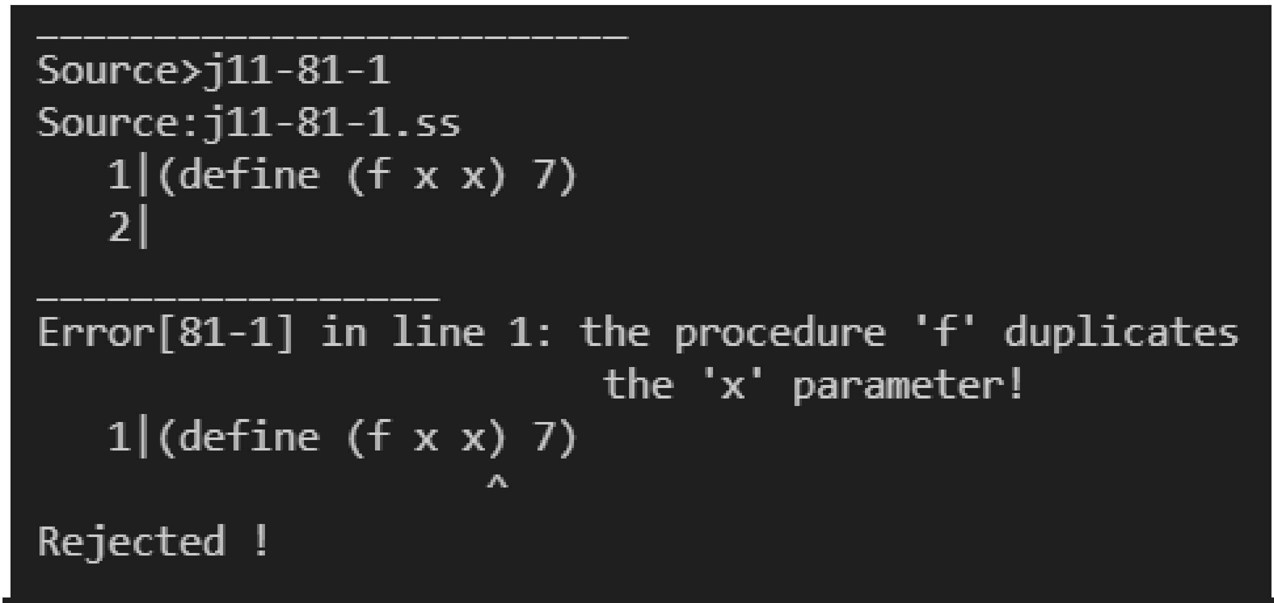
```
 -------------------------------
 Source>j11-80-2
 Source:j11-80-2.ss
    1|(define x 9)
    2|(define (x y) 9)
    3|

 --------------------
 Error[80-2] in line 2: 'x'  has already been defined
                 in 1 line as a variable!
    2|(define (x y) 9)
                 ^
 Rejected !
```



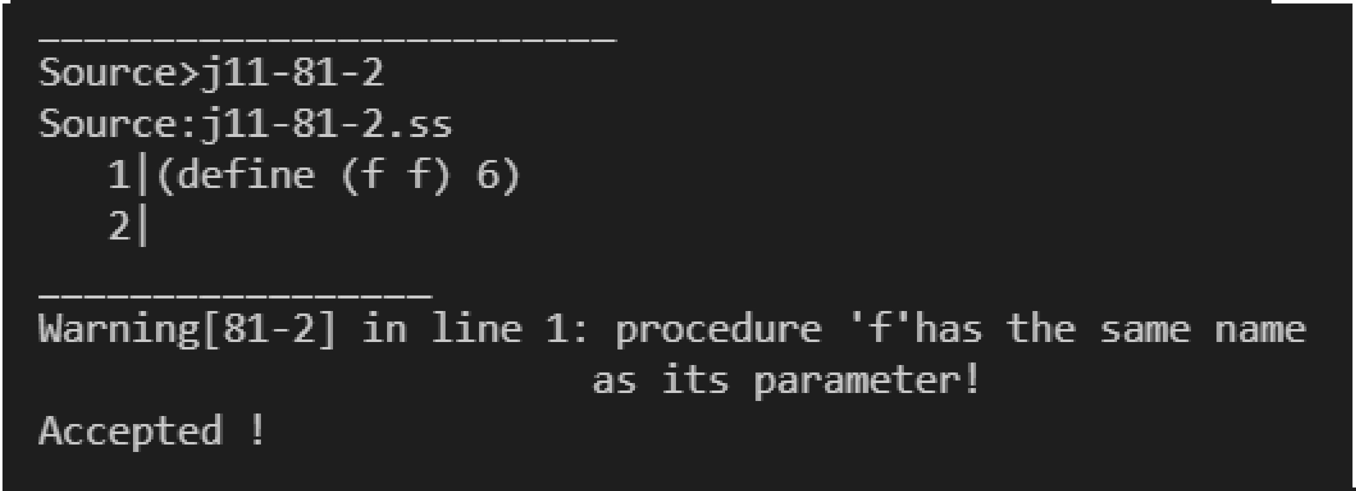
```
 -------------------------------
 Source>j11-80-3
 Source:j11-80-3.ss
    1|(define (pi x) 5)
    2|

 --------------------
 Error[80-3] in line 1: The built - in constant 'pi'
                 cannot be redefined as a procedure!
    1|(define (pi x) 5)
                 ^
 Rejected !
```



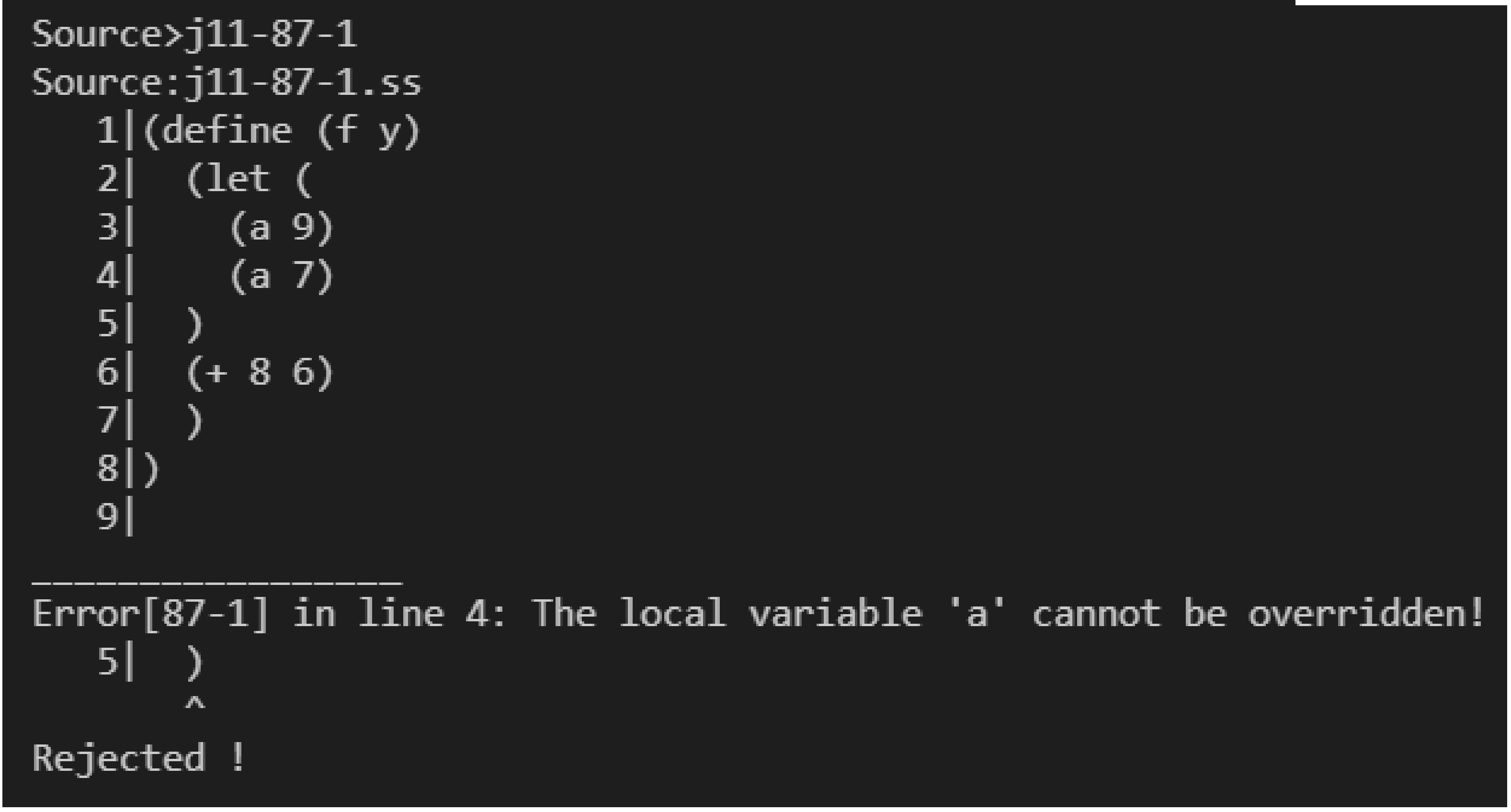
```
-----------------------------
Source>j11-80-4
Source:j11-80-4.ss
   1|(define (abs x) 5)
   2|

--------------------
Error[80-4] in line 1: You can't override the built-in procedure 'abs'!
   1|(define (abs x) 5)
                  ^
Rejected !
```



```
_________________________
Source>j11-81-1
Source:j11-81-1.ss
   1|(define (f x x) 7)
   2|

_________________
Error[81-1] in line 1: the procedure 'f' duplicates
                        the 'x' parameter!
   1|(define (f x x) 7)
                    ^
Rejected !
```



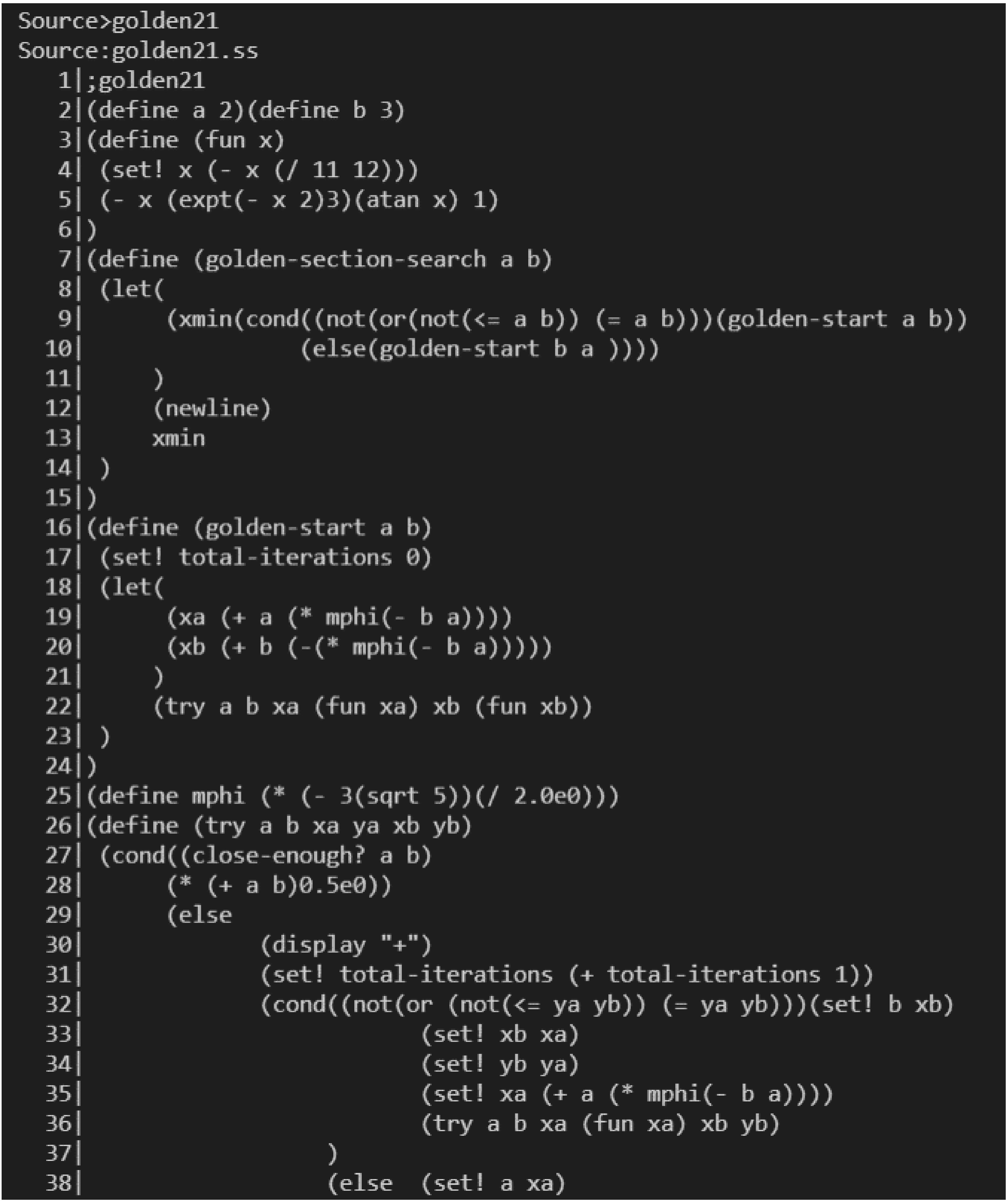
```
_________________________
Source>j11-81-2
Source:j11-81-2.ss
   1|(define (f f) 6)
   2|

_________________
Warning[81-2] in line 1: procedure 'f'has the same name
                         as its parameter!
Accepted !
```



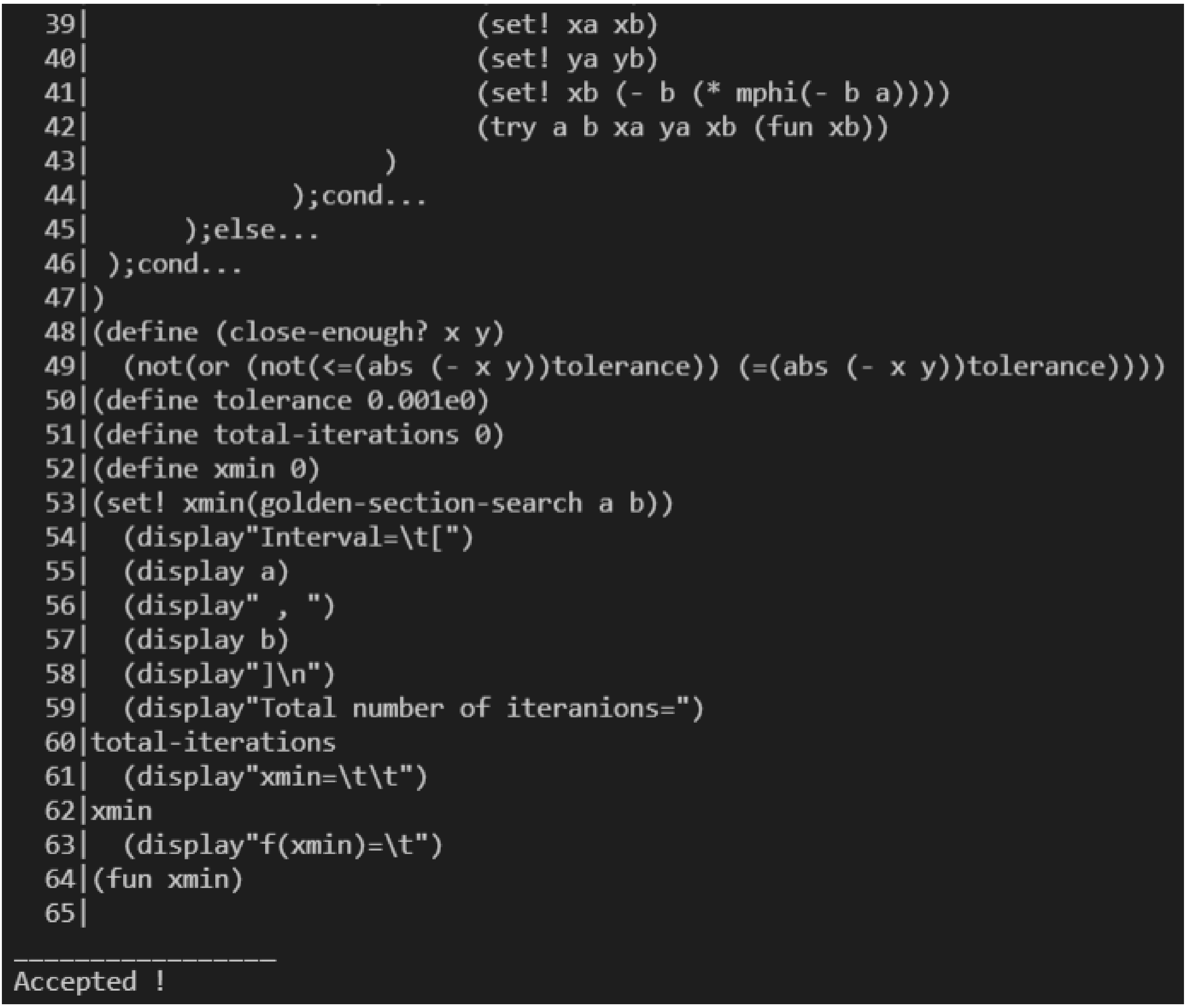
```
Source>j11-87-1
Source:j11-87-1.ss
   1|(define (f y)
   2|  (let (
   3|    (a 9)
   4|    (a 7)
   5|  )
   6|  (+ 8 6)
   7|  )
   8|)
   9|

_________________
Error[87-1] in line 4: The local variable 'a' cannot be overridden!
   5|  )
       ^
Rejected !
```

Полные скриншоты анализа своих вариантов программ golden21 и coin21
>

golden21

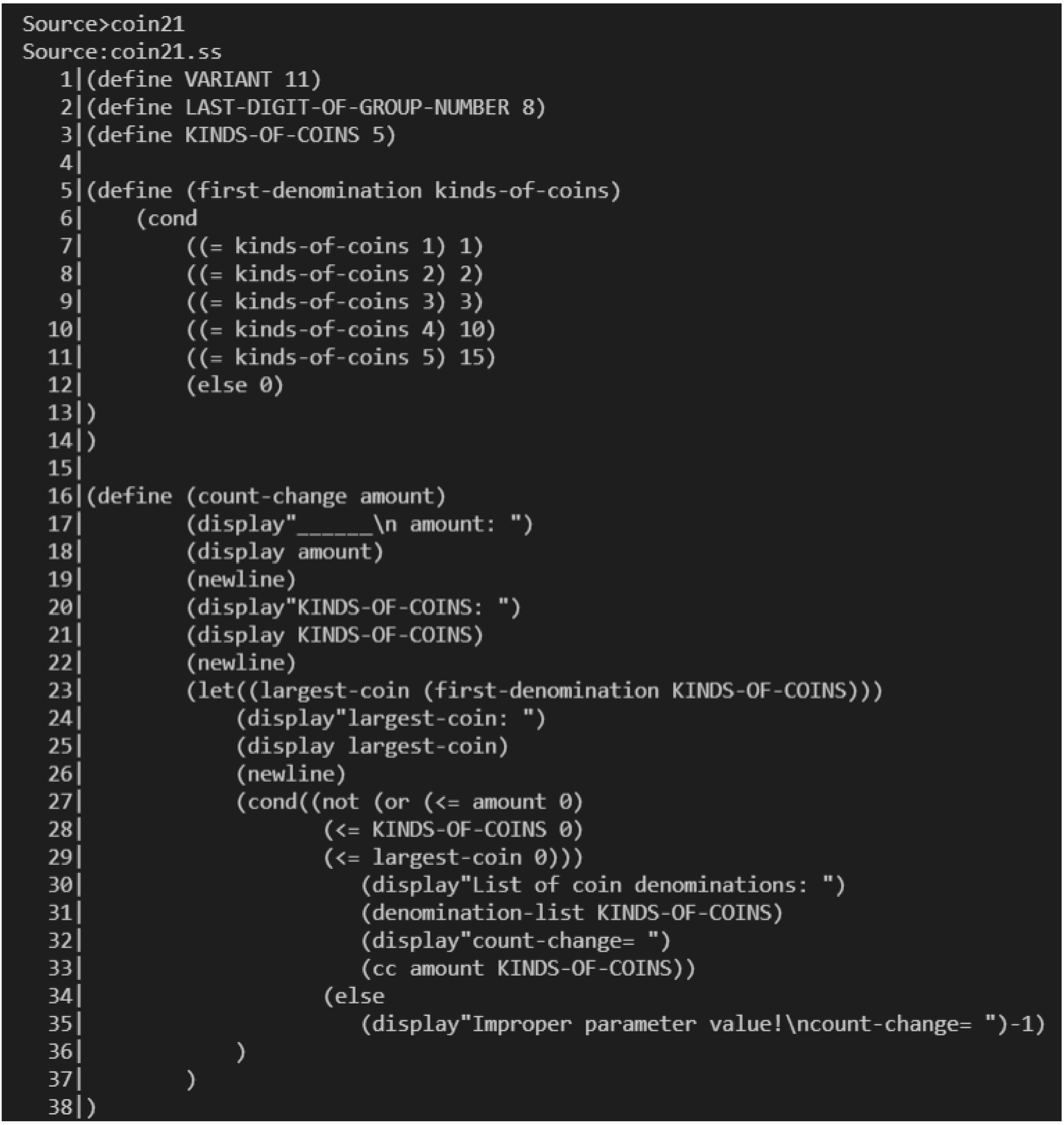


```
Source>golden21
Source:golden21.ss
;golden21
(define a 2)(define b 3)
(define (fun x)
 (set! x (- x (/ 11 12)))
 (- x (expt(- x 2)3)(atan x) 1)
)
(define (golden-section-search a b)
 (let(
      (xmin(cond((not(or(not(<= a b)) (= a b)))(golden-start a b))
                (else(golden-start b a ))))
     )
     (newline)
     xmin
 )
)
(define (golden-start a b)
 (set! total-iterations 0)
 (let(
      (xa (+ a (* mphi(- b a))))
      (xb (+ b (-(* mphi(- b a)))))
     )
     (try a b xa (fun xa) xb (fun xb))
 )
)
(define mphi (* (- 3(sqrt 5))(/ 2.0e0)))
(define (try a b xa ya xb yb)
 (cond((close-enough? a b)
      (* (+ a b)0.5e0))
      (else
             (display "+")
             (set! total-iterations (+ total-iterations 1))
             (cond((not(or (not(<= ya yb)) (= ya yb)))(set! b xb)
                         (set! xb xa)
                         (set! yb ya)
                         (set! xa (+ a (* mphi(- b a))))
                         (try a b xa (fun xa) xb yb)
                  )
                  (else  (set! a xa)
```

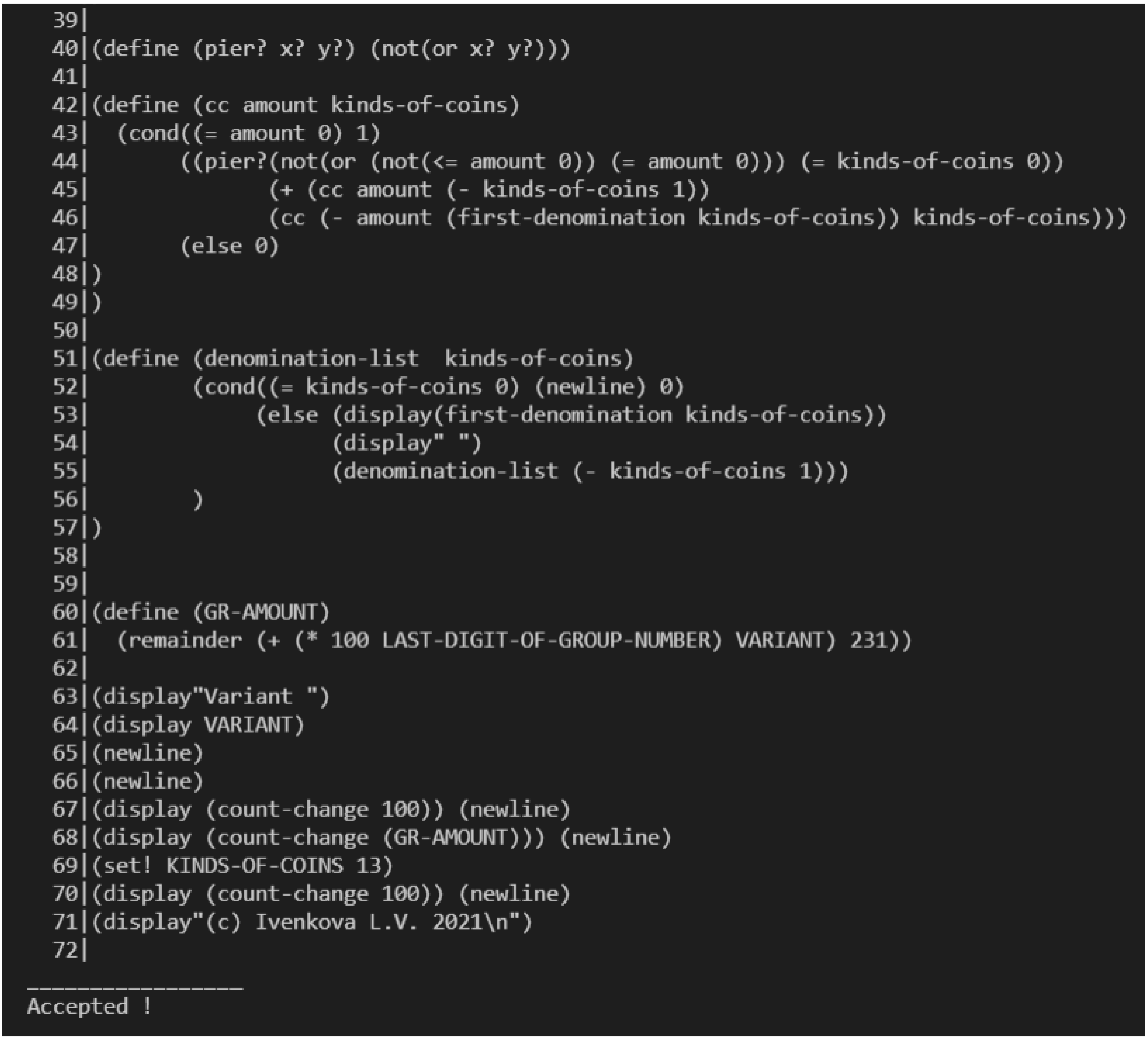


```
                         (set! xa xb)
                         (set! ya yb)
                         (set! xb (- b (* mphi(- b a))))
                         (try a b xa ya xb (fun xb))
                   )
             );cond...
      );else...
 );cond...
)
(define (close-enough? x y)
  (not(or (not(<=(abs (- x y))tolerance)) (=(abs (- x y))tolerance))))
(define tolerance 0.001e0)
(define total-iterations 0)
(define xmin 0)
(set! xmin(golden-section-search a b))
  (display"Interval=\t[")
  (display a)
  (display" , ")
  (display b)
  (display"]\n")
  (display"Total number of iteranions=")
total-iterations
  (display"xmin=\t\t")
xmin
  (display"f(xmin)=\t")
(fun xmin)

________________
Accepted !
```

coin21



```
Source>coin21
Source:coin21.ss
(define VARIANT 11)
(define LAST-DIGIT-OF-GROUP-NUMBER 8)
(define KINDS-OF-COINS 5)

(define (first-denomination kinds-of-coins)
    (cond
        ((= kinds-of-coins 1) 1)
        ((= kinds-of-coins 2) 2)
        ((= kinds-of-coins 3) 3)
        ((= kinds-of-coins 4) 10)
        ((= kinds-of-coins 5) 15)
        (else 0)
)
)

(define (count-change amount)
        (display"______\n amount: ")
        (display amount)
        (newline)
        (display"KINDS-OF-COINS: ")
        (display KINDS-OF-COINS)
        (newline)
        (let((largest-coin (first-denomination KINDS-OF-COINS)))
            (display"largest-coin: ")
            (display largest-coin)
            (newline)
            (cond((not (or (<= amount 0)
                   (<= KINDS-OF-COINS 0)
                   (<= largest-coin 0)))
                      (display"List of coin denominations: ")
                      (denomination-list KINDS-OF-COINS)
                      (display"count-change= ")
                      (cc amount KINDS-OF-COINS))
                   (else
                      (display"Improper parameter value!\ncount-change= ")-1)
            )
        )
)
```



```

(define (pier? x? y?) (not(or x? y?)))

(define (cc amount kinds-of-coins)
  (cond((= amount 0) 1)
       ((pier?(not(or (not(<= amount 0)) (= amount 0))) (= kinds-of-coins 0))
             (+ (cc amount (- kinds-of-coins 1))
             (cc (- amount (first-denomination kinds-of-coins)) kinds-of-coins)))
       (else 0)
)
)

(define (denomination-list  kinds-of-coins)
        (cond((= kinds-of-coins 0) (newline) 0)
             (else (display(first-denomination kinds-of-coins))
                   (display" ")
                   (denomination-list (- kinds-of-coins 1)))
        )
)


(define (GR-AMOUNT)
  (remainder (+ (* 100 LAST-DIGIT-OF-GROUP-NUMBER) VARIANT) 231))

(display"Variant ")
(display VARIANT)
(newline)
(newline)
(display (count-change 100)) (newline)
(display (count-change (GR-AMOUNT))) (newline)
(set! KINDS-OF-COINS 13)
(display (count-change 100)) (newline)
(display"(c) Ivenkova L.V. 2021\n")

_________________
Accepted !
```

Распечатка файла semantics.cpp.

```
>
/* $j11 */
#include "semantics.h"
#include <cmath>
using namespace std;

string TypesToString(int types, int count){
   std::string message = "(";
   for(int i = 0; i < count; i++) {
      if( (types%10) > 0)
         message += "bool ";
      else
         message += "numeric ";
      types /= 10;
   }
   message += ")";
   return message;
}

void tSM::init(){
   globals.clear();
   locals.clear();
   params.clear();
   scope = 0; // вне процедуры

   //константы:
   globals["e"] = tgName(VAR|DEFINED|BUILT);
   globals["pi"] = tgName(VAR|DEFINED|BUILT);

   // встроенные процедуры:
   globals["abs"] = tgName(PROC|DEFINED|BUILT,"", 1);
   globals["sqrt"] = tgName(PROC|DEFINED|BUILT,"", 1);
   globals["atan"] = tgName(PROC|DEFINED|BUILT,"", 1);
   globals["cos"] = tgName(PROC|DEFINED|BUILT,"", 1);
   globals["sin"] = tgName(PROC|DEFINED|BUILT,"", 1);
   globals["tan"] = tgName(PROC|DEFINED|BUILT,"", 1);
   globals["exp"] = tgName(PROC|DEFINED|BUILT,"", 1);
   globals["expt"] = tgName(PROC|DEFINED|BUILT,"", 2);
   globals["log"] = tgName(PROC|DEFINED|BUILT,"", 1);
   globals["quotient"] = tgName(PROC|DEFINED|BUILT,"",
2);
   globals["remainder"] =
tgName(PROC|DEFINED|BUILT,"", 2);
   return;
}
int tSM::p01(){ //       S -> PROG

   bool error=false;

   struct NameVarProc {
   std::string name;
   bool VarProc = 0; // Var == 0; Proc == 1;
   };

   std::map<std::string,NameVarProc> Arr;

   for(tGlobal::iterator it=globals.begin();
       it!=globals.end();
       ++it){
       // Просмотреть таблицу глобальных имен
       // и в сообщении об ошибках указать имена
       // ВСЕХ вызванных, но не определенных процедур,
       // а также использованных, но не определенных
```

```
      // глобальных переменных.
      //   it->first   имя
      //   it->second  учетная запись
      if(it->second.test(PROC|USED) && !it-
>second.test(DEFINED)) {
         Arr[it->second.line].name = it->first;
         Arr[it->second.line].VarProc = 1;
         error = true;
      }
      if(it->second.test(VAR|USED) && !it-
>second.test(DEFINED)) {
         Arr[it->second.line].name = it->first;
         error = true;
      }
   }//for...
   for(const auto& kv: Arr) {
      if(kv.second.VarProc == 0) {
         ferror_message += "Error[01-1] in line " + kv.first +
         ": The undefined global variable '" + kv.second.name
+ "' is used!\n";
         // Использована неопределённая глобальная
переменная 'z'!
         // The undefined global variable 'z' is used!
      }
      else{
         ferror_message += "Error[01-2] in line " + kv.first +
         ": The undefined procedure '" + kv.second.name + "'
is called!\n";
         // Вызвана неопределённая процедура 'f'!
         // The undefined procedure 'f' is called!
      }
   }
   if(error) return 1;
   return 0;
}
int tSM::p02(){ //    PROG -> CALCS
   return 0;}
int tSM::p03(){ //    PROG -> DEFS
   return 0;}
int tSM::p04(){ //    PROG -> DEFS CALCS
   return 0;}
int tSM::p05(){ //       E -> $id
   string name = S1->name;
   switch(scope){
      case 2:if(locals.count(name))break;
      case 1:if(params.count(name))break;
      default:tgName& ref = globals[name];
      if(ref.empty()){
         ref = tgName(VAR|USED, S1->line);
         break;
      }
      if(ref.test(VAR)){
         ref.set(USED);
         break;
      }
      if(ref.test(BUILT)){
         ferror_message+=
            "Error[05-1] in line "+ S1->line +": The built-in '"
            +name+
            "' procedure \n\t\t\t cannot be used as a
variable!\n";
         // Встроенную процедуру 'abs' нельзя
использовать в качестве переменной
         // The built-in 'abs' procedure cannot be used as a
variable
         return 1;
      }
      ferror_message+=
            "Error[05-2] in line "+ S1->line +":The name '"
            +name+
            "' cannot be used to refer to a variable;\n"+
            "\t\t\tit was previously declared as a procedure in
line "+ ref.line +" !\n";
      // Имя 'f' нельзя использовать для ссылки на
переменную, в строке 1 оно ранее объявлено как
процедура
      // The name 'f' cannot be used to refer to a variable; it
was previously declared as a procedure in line 1
      return 1;
   }//switch...
   return 0;
}
int tSM::p06(){ //       E -> $int
   return 0;}
int tSM::p07(){ //       E -> $dec
   return 0;}
int tSM::p08(){ //       E -> AREX
   return 0;}
int tSM::p09(){ //       E -> COND
   return 0;}
int tSM::p10(){ //       E -> CPROC
   string name = S1->name;
   S1->types = 1;
   switch(scope){
      case 2:
         if(locals.count(name)) {
            ferror_message+=
            "Error[10-1] in line "+ S1->line +": The procedure
call identifier '"
            +name+
            "' \n\t\tcannot be a local variable!\n";
            // Идентификатор вызова процедуры 'f' не
может быть локальной переменной!
            // The procedure call identifier 'f' cannot be a
local variable!
            return 1;
          }
       case 1:
          if(params.count(name)) {
            ferror_message+=
            "Error[10-2] in line "+ S1->line +": The procedure
call identifier '"
            +name+
            "' \n\t\tcannot be a parameter name!\n";
            // Идентификатор вызова процедуры 'f' не
может быть параметром!
            // The procedure call identifier 'f' cannot be a
parameter name!
            return 1;
          }
       default:tgName& ref = globals[name];
       if(ref.test(VAR)) {
          ferror_message+=
          "Error[10-3] in line "+ S1->line +": The procedure
call identifier '"
          +name+
          "' \n\t\tcannot be a variable!\n";
          // Идентификатор вызова процедуры 'f' не может
быть переменной!
          // The procedure call identifier 'f' cannot be a
variable!
          return 1;
       }
       if(ref.empty()){
          ref = tgName(PROC|USED, S1->line);
          ref.arity = S1->count;
       }
       if(ref.test(DEFINED) && !(ref.test(BUILT))){
          ref.set(PROC|USED);
          if(ref.arity != S1->count){
            ferror_message+=
            "Error[10-4] in line "+ S1->line +": The arities of
the call and the definition of the procedure '"
            +name+
            "'\n\t\tdo not match!\n\t\t"+
            "The procedure has already been declared in line
"+
            globals[name].line + "!\n\t\t"+
            "Previously, there was arity = " +
std::to_string(ref.arity) +
            ", and now arity = " + std::to_string(S1->count) +
"!\n";
            // Арности вызова и определения процедуры 'f'
не совпадают!
            // Процедура уже была объявлена в строке 1!
            // Ранее была арность = 2, а сейчас арность =
1.
            // The arities of the call and the definition of the
procedure 'f' do not match!
            // The procedure has already been declared in line
1!
            // Previously, there was arity = 2, and now arity =
1.
            return 1;
         }
         ref.arity = S1->count;
      }
      if(ref.test(DEFINED|BUILT)){
         ref.set(PROC|USED);
         if(ref.arity != S1->count){
            ferror_message+=
            "Error[10-5] in line "+ S1->line +": The arities of
the call and the definition of the procedure '"
            +name+
            "'\n\t\t do not match!\n\t\t"+
            "The built-in procedure '" + name +
            "' should have arity = " + std::to_string(ref.arity)
+
            ", not arity = " + std::to_string(S1->count) +
"!\n";
            // Арности вызова и определения процедуры 'f'
не совпадают!
            // Встроенная процедура 'abs' должна иметь
арность = 1, а не арность = 2!
            // The arities of the call and the definition of the
procedure 'f' do not match!
            // The built-in procedure 'abs' should have arity =
1, not arity = 2!
            return 1;
         }
         ref.arity = S1->count;
      }
```

```
    }//switch...
    return 0;
}
int tSM::p11(){ //    AREX -> HAREX E )
    return 0;}
int tSM::p12(){ //   HAREX -> ( AROP
    return 0;}
int tSM::p13(){ //   HAREX -> HAREX E
    return 0;}
int tSM::p14(){ //    AROP -> +
    return 0;}
int tSM::p15(){ //    AROP -> -
    return 0;}
int tSM::p16(){ //    AROP -> *
    return 0;}
int tSM::p17(){ //    AROP -> /
    return 0;}
int tSM::p18(){ //   CPROC -> HCPROC )
    return 0;
}
int tSM::p19(){ //  HCPROC -> ( $id
    S1->name = S2->name;
    return 0;
}
int tSM::p20(){ //  HCPROC -> HCPROC E
    S1->types += S2->types;
    S1->count++;
    return 0;
}
int tSM::p21(){ //    COND -> ( cond BRANCHES )
    return 0;}
int tSM::p22(){ //BRANCHES -> ELSE
    return 0;}
int tSM::p23(){ //BRANCHES -> CLAUS BRANCHES
    return 0;}
int tSM::p24(){ //   CLAUS -> ( BOOL CLAUSB )
    return 0;}
int tSM::p25(){ //  CLAUSB -> E
    return 0;}
int tSM::p26(){ //  CLAUSB -> INTER CLAUSB
    return 0;}
int tSM::p27(){ //    ELSE -> ( else ELSEB )
    return 0;}
int tSM::p28(){ //   ELSEB -> E
    return 0;}
int tSM::p29(){ //   ELSEB -> INTER ELSEB
    return 0;}
int tSM::p30(){ //     STR -> $str
```

```
   return 0;}
int tSM::p31(){ //     STR -> SIF
   return 0;}
int tSM::p32(){ //     SIF -> ( if BOOL STR STR )
   return 0;}
int tSM::p33(){ //    BOOL -> $bool
   S1->types = 1;
   return 0;
}
int tSM::p34(){ //    BOOL -> $idq
   S1->types = 1;
   string name = S1->name;
   switch(scope){
      case 2:if(locals.count(name))break;
      case 1:if(params.count(name))break;
      default:tgName& ref = globals[name];
      ferror_message+=
             "Error[34-1] in line "+ S1->line +":The name '"
             +name+
             "' cannot be used to refer to a variable,\n"+
             "\t\t\tit was previously declared as a predicate
procedure in line "+ ref.line +" !\n";
      // Имя 'f' нельзя использовать для ссылки на
переменную, оно ранее объявлено как процедура-
предикат в строке 1!
      // The name 'f' cannot be used to refer to a variable; it
was previously declared as a predicate procedure in line 1!
      return 1;
   }//switch...
   return 0;
}
int tSM::p35(){ //    BOOL -> REL
   S1->types = 1;
   return 0;}
int tSM::p36(){ //    BOOL -> OR
   S1->types = 1;
   return 0;
}
int tSM::p37(){ //    BOOL -> ( not BOOL )
   S1->types = 1;
   return 0;
}
int tSM::p38(){ //    BOOL -> CPRED
   string name = S1->name;
   switch(scope){
      case 1:
         if(params.count(name)) {
            ferror_message+=
            "Error[38-1] in line "+ S1->line +": The predicate
procedure call identifier '"
            +name+
            "' \n\t\tcannot be a parameter name!\n";
            // Идентификатор вызова процедуры-предиката
'f' не может быть параметром!
            // The predicate procedure call identifier 'f' cannot
be a parameter name!
            return 1;
      }
      default:tgName& ref = globals[name];
      if(ref.empty()){
         ref = tgName(PROC|USED, S1->line);
         ref.arity = S1->count;
         ref.types = S1->types;
      }
      if(ref.test(DEFINED)){
         ref.set(PROC|USED);
         /* При конфликтующих арностях я не смотрела на
"лишние" параметры
           и сравнивала только общее число типов
параметров
           (начиная с начала), и, если они не совпадали,
           то считала это несовпадением типов параметров.
           Numeric – численный тип параметров. */
         string types1, types2;
         if(ref.arity > S1->count){
            types1 = TypesToString(ref.types, S1->count);
            types2 = TypesToString(S1->types, S1->count);
         }
         else {
            types1 = TypesToString(ref.types, ref.arity);
            types2 = TypesToString(S1->types, ref.arity);
         }
         if(types1 != types2 || ref.arity != S1->count){
            if(types1 != types2){
               ferror_message+=
               "Error[38-2] in line "+ S1->line +": The types of
the call parameters and the definition \n\t\tof the predicate
procedure '"
               +name+
               "' do not match!\n\t\t"+
               "The predicate procedure has already been
declared in line "+
               ref.line + "!\n\t\t"+
               "Previously, there was types = " +
TypesToString(ref.types, ref.arity) +
                ", and now types = " + TypesToString(S1-
>types, S1->count) + "!\n";
                // Типы параметров вызова и определения
процедуры-предиката 'f?' не совпадают!
                // Процедура-предикат уже была объявлена
в строке 1!
                // Ранее были типы = (numeric bool ), а
сейчас типы = (bool bool )!
                // The types of the call parameters and the
definition of the predicate procedure 'f?' do not match!
                // The predicate procedure has already been
declared in line 1!
                // Previously, there were types = (numeric bool
), and now types = (bool bool )!
             }
             if(ref.arity != S1->count){
                ferror_message+=
                "Error[38-3] in line "+ S1->line +": The arities
of the call and the definition \n\t\tof the predicate
procedure '"
                +name+
                "' do not match!\n\t\t"+
                "The predicate procedure has already been
declared in line "+
                globals[name].line + "!\n\t\t"+
                "Previously, there was arity = " +
std::to_string(ref.arity) +
                ", and now arity = " + std::to_string(S1->count)
+ "!\n";
                // Арности вызова и определения процедуры-
предиката 'f?' не совпадают!
                // Процедура-предикат уже была объявлена
в строке 1!
                // Ранее была арность = 2, а сейчас арность =
1.
                // The arities of the call and the definition of
the predicate procedure 'f?' do not match!
                // The predicate procedure has already been
declared in line 1!
                // Previously, there was arity = 2, and now
arity = 1.
             }
             return 1;
      }
         ref.arity = S1->count;
         ref.types = S1->types;
      }
   }//switch...
```

```
    return 0;
}
int tSM::p39(){ //   CPRED -> HCPRED )
    return 0;}
int tSM::p40(){ //  HCPRED -> ( $idq
    S1->name = S2->name;
    return 0;
}
int tSM::p41(){ //  HCPRED -> HCPRED ARG
    if(S2->types != 0) S1->types += pow(10, S1->count);
    S1->count++;
    return 0;
}
int tSM::p42(){ //     ARG -> E
    return 0;}
int tSM::p43(){ //     ARG -> BOOL
    return 0;}
int tSM::p44(){ //     REL -> ( = E E )
    return 0;}
int tSM::p45(){ //     REL -> ( <= E E )
    return 0;}
int tSM::p46(){ //      OR -> HOR BOOL )
    return 0;}
int tSM::p47(){ //     HOR -> ( or
    return 0;}
int tSM::p48(){ //     HOR -> HOR BOOL
    return 0;}
int tSM::p49(){ //     SET -> HSET E )
    return 0;}
int tSM::p50(){ //    HSET -> ( set! $id
    S1->name = S3->name;
    return 0;
}
int tSM::p51(){ // DISPSET -> ( display E )
    return 0;}
int tSM::p52(){ // DISPSET -> ( display BOOL )
    return 0;}
int tSM::p53(){ // DISPSET -> ( display STR )
    return 0;}
int tSM::p54(){ // DISPSET -> ( newline )
    return 0;}
int tSM::p55(){ // DISPSET -> SET
    string name = S1->name;
    switch(scope){
        case 2: if(locals.count(name)) break;
        case 1: if(params.count(name)) break;
        default:tgName& ref = globals[name];
        if(ref.test(VAR|BUILT)) {
            ferror_message+=
            "Error[55-1] in line "+ S1->line +": The identifier in
the assignment\n\
             statement cannot be the name of a constant!\n";
            // Идентификатор в операторе присваивания не
может быть именем константы!
            // The identifier in the assignment statement cannot
be the name of a constant!
            return 1;
        }
        if(ref.test(PROC)) {
            ferror_message+=
            "Error[55-2] in line "+ S1->line +": The identifier in
the assignment\n\
             statement cannot be the name of the procedure!\n";
            // Идентификатор в операторе присваивания не
может быть именем процедуры!
            // The identifier in the assignment statement cannot
be the name of the procedure!
            return 1;
        }
    }//switch...
    return 0;
}
int tSM::p56(){ //   INTER -> DISPSET
    return 0;}
int tSM::p57(){ //   INTER -> E
    return 0;}
int tSM::p58(){ //   CALCS -> CALC
    return 0;}
int tSM::p59(){ //   CALCS -> CALCS CALC
    return 0;}
int tSM::p60(){ //    CALC -> E
    return 0;}
int tSM::p61(){ //    CALC -> BOOL
    return 0;}
int tSM::p62(){ //    CALC -> STR
    return 0;}
int tSM::p63(){ //    CALC -> DISPSET
    return 0;}
int tSM::p64(){ //    DEFS -> DEF
    return 0;}
int tSM::p65(){ //    DEFS -> DEFS DEF
    return 0;}
int tSM::p66(){ //     DEF -> PRED
    return 0;}
int tSM::p67(){ //     DEF -> VAR
    string name = S1->name;
```

```
   tgName& ref = globals[name];
   if(ref.test(DEFINED|VAR) && !(ref.test(BUILT))){
      ferror_message+=
      "Error[67-1] in line "+ S1->line +": The '"
      +S1->name+
      "' variable has already been defined \n\t\tin " +
ref.line + " line!\n";
      // Переменная 'x' уже была определена в 1 строке!
      // The 'x' variable has already been defined in 1 line!
      return 1;
   }
   if(ref.test(PROC) && !(ref.test(BUILT))){
      ferror_message+=
      "Error[67-2] in line "+ S1->line +": '"
      +S1->name+
      "'  has already been defined \n\t\tin " + ref.line + "
line as a procedure!\n";
      // 'x' уже был определен в 1 строке как процедура!
      // The 'x' has already been defined in 1 line as a
procedure!!
      return 1;
   }
   if(ref.test(DEFINED|PROC|BUILT)){
      ferror_message+=
      "Error[67-3] in line "+ S1->line +": The built - in
procedure '"
      +S1->name+
      "'  \n\t\tcannot be redefined as a variable!\n";
      // Встроенную процедуру 'abs' нельзя
переопределить как переменную!
      // The built - in procedure 'abs' cannot be redefined as
a variable!
      return 1;
   }
   if(ref.test(DEFINED|VAR|BUILT)){
      ferror_message+=
      "Error[67-4] in line "+ S1->line +": You can't override
the built-in constant '"
      +S1->name+ "'!\n";
      // Нельзя переопределять встроенную константу 'x'!
      // You can't override the built-in constant 'x'!
      return 1;
   }
   if(ref.empty()){
      ref = tgName(VAR|DEFINED, S1->line);
   }//if(ref.empty())...
   if(ref.test(USED)){
      ref.set(DEFINED);
```

```
    }
    return 0;
}
int tSM::p68(){ //     DEF -> PROC
    return 0;}
int tSM::p69(){ //    PRED -> HPRED BOOL )
    // точка анализа выходит из тела
    // процедуры
    params.clear();
    scope = 0;
    return 0;
}
int tSM::p70(){ //   HPRED -> PDPAR )
    string name = S1->name;
    tgName& ref = globals[name];
    if(ref.test(USED)){
        ref.set(DEFINED);
        /* При конфликтующих арностях я не смотрела на
"лишние" параметры
            и сравнивала только общее число типов
параметров
            (начиная с начала), и, если они не совпадали,
            то считала это несовпадением типов параметров.
            Numeric – численный тип параметров. */
        string types1, types2;
        if(ref.arity > S1->count){
            types1 = TypesToString(ref.types, S1->count);
            types2 = TypesToString(S1->types, S1->count);
        }
        else {
            types1 = TypesToString(ref.types, ref.arity);
            types2 = TypesToString(S1->types, ref.arity);
        }
        if(types1 != types2 || ref.arity != S1->count){
            if(types1 != types2){
                ferror_message+=
                "Error[70-1] in line "+ S1->line +": The types of
the call parameters and the definition\n\t\tof the predicate
procedure '"
                +name+
                "' do not match!\n\t\t"+
                "The predicate procedure has already been called
in line "+
                ref.line + "!\n\t\t"+
                "Previously, there was types = " +
TypesToString(ref.types, ref.arity) +
                ", and now types = " + TypesToString(S1->types,
S1->count) + "!\n";
```

```
            // Типы параметров вызова и определения
процедуры 'f' не совпадают!
            // Процедура-предикат уже была вызвана в
строке 1!
            // Ранее были типы = (numeric bool ), а сейчас
типы = (bool bool )!
            // The types of the call parameters and the
definition of the procedure 'f' do not match!
            // The predicate procedure has already been
called in line 1!
            // Previously, there were types = (numeric bool ),
and now types = (bool bool )!
         }
         if(ref.arity != S1->count){
            ferror_message+=
            "Error[70-2] in line "+ S1->line +": The arities of
the call and the definition of the procedure '"
            +name+
            "' \n\t\tdo not match!\n\t\t"+
            "The predicate procedure has already been called
in line "+
            globals[name].line + "!\n\t\t"+
            "Previously, there was arity = " +
std::to_string(ref.arity) +
            ", and now arity = " + std::to_string(S1->count) +
"!\n";
            // Арности вызова и определения процедуры 'f'
не совпадают!
            // Процедура-предикат уже была вызвана в
строке 1!
            // Ранее была арность = 2, а сейчас арность =
1.
            // The arities of the call and the definition of the
procedure 'f' do not match!
            // The predicate procedure has already been
called in line 1!
            // Previously, there was arity = 2, and now arity =
1.
         }
         return 1;
      }
   }
   ref.arity = S1->count;
   ref.types = S1->types;
   // точка анализа входит в тело
   // процедуры
   scope = 1;
   return 0;
```

```
}
int tSM::p71(){ //   PDPAR -> ( define ( $idq
   S1->name = S4->name;
   S1->count = 0;
   string name = S1->name;
   tgName& ref = globals[name];
   if(ref.test(DEFINED|PROC)){
      ferror_message+=
      "Error[71-1] in line "+ S1->line +": The predicate
procedure '"
      +S1->name+
      "' \n\t\thas already been defined in " + ref.line + "
line!\n";
      // Процедура-предикат 'f?' уже была определена в 1
строке!
      // The predicate procedure 'f?' has already been
defined in 1 line!
      return 1;
   }
   if(ref.empty()){
      ref = tgName(PROC|DEFINED, S1->line);
      ref.arity = S1->count;
      ref.types = S1->types;
   }//if(ref.empty())...
   return 0;
}
int tSM::p72(){ //   PDPAR -> PDPAR $idq
   if(params.count(S2->name)){
      ferror_message+=
      "Error[72-1] in line "+ S2->line +": In the predicate
procedure '"
      +S1->name+
      "', \n\t\tthe Boolean parameter '"
      +S2->name+"' is duplicated!\n";
      // В процедуре-предикате 'f' дублируется булевский
параметр 'x?'
      // In the predicate procedure 'f', the Boolean
parameter 'x' is duplicated!
      return 1;
   }

   if(S2->name==S1->name){
      ferror_message+=
      "Warning[72-2] in line "+ S2->line +": The predicate
procedure '"
      +S1->name+
      "' \n\t\tsame name as its Boolean parameter '"
      +S2->name+ "'!\n";
```

```
       // У процедуры-предиката 'f?' такое же имя, как у ее
булевского параметра!
       // The predicate procedure 'f?' same name as its
Boolean parameter 'x'!
    }
    params.insert(S2->name);
    S1->types += pow(10, S1->count);
    ++S1->count;
    return 0;
}
int tSM::p73(){ //   PDPAR -> PDPAR $id
    if(params.count(S2->name)){
       ferror_message+=
       "Error[73-1] in line "+ S2->line +": The predicate
procedure '"
       +S1->name+
       "' duplicates \n\t\t\tthe '"
       +S2->name+"' parameter!\n";
       // В процедуре-предикате 'f' дублируется параметр
'x?'
       // The 'f' procedure duplicates the 'x?' parameter
       return 1;
    }
    params.insert(S2->name);
    ++S1->count;
    return 0;
}
int tSM::p74(){ //     VAR -> VARDCL E )
    scope = 0;
    return 0;
}
int tSM::p75(){ //  VARDCL -> ( define $id
    S1->name = S3->name;
    return 0;
}
int tSM::p76(){ //    PROC -> HPROC BLOCK )
    params.clear();
    scope = 0;
    return 0;
}
int tSM::p77(){ //    PROC -> HPROC E )
    params.clear();
    scope = 0;
    return 0;
}
int tSM::p78(){ //   HPROC -> PCPAR )
    string name = S1->name;
    tgName& ref = globals[name];
```

```
   if(ref.test(USED)){
      ref.set(DEFINED);
      if(ref.arity != S1->count){
         ferror_message+=
         "Error[78-1] in line "+ S1->line +": The arities of the
call and the definition of the procedure '"
         +name+
         "' \n\t\tdo not match!\n\t\t"+
         "The procedure has already been called in line "+
         globals[name].line + "!\n\t\t"+
         "Previously, there was arity = " +
std::to_string(globals[name].arity) +
         ", and now arity = " + std::to_string(S1->count) +
"!\n";
         // Арности вызова и определения процедуры 'f' не
совпадают!
         // Процедура уже была вызвана в строке 1!
         // Ранее была арность = 2, а сейчас арность = 1.
         // The arities of the call and the definition of the
procedure 'f' do not match!
         // The procedure has already been called in line 1!
         // Previously, there was arity = 2, and now arity = 1.
         return 1;
      }
   }
   ref.arity = S1->count;
   // точка анализа входит в тело
   // процедуры
   scope = 1;
   return 0;
}
int tSM::p79(){ //   HPROC -> HPROC INTER
   return 0;}
int tSM::p80(){ //   PCPAR -> ( define ( $id
   S1->count = 0;
   S1->name = S4->name;
   string name = S4->name;
   tgName& ref = globals[name];
   if(ref.test(DEFINED|PROC) && !(ref.test(BUILT))){
      ferror_message+=
      "Error[80-1] in line "+ S1->line +": The procedure '"
      +S1->name+
      "' has already been defined \n\t\tin " + ref.line + "
line!\n";
      // Процедура 'f' уже была определена в 1 строке!
      // The procedure 'f' has already been defined in 1 line!
      return 1;
   }
   if(ref.test(VAR) && !(ref.test(BUILT))){
      ferror_message+=
      "Error[80-2] in line "+ S1->line +": '"
      +S1->name+
      "'  has already been defined \n\t\tin " + ref.line + "
line as a variable!\n";
      // 'x' уже был определен в 1 строке как переменная!
      // The 'x' has already been defined in 1 line as a
variable!!
      return 1;
   }
   if(ref.test(VAR|BUILT)){
      ferror_message+=
      "Error[80-3] in line "+ S1->line +": The built - in
constant '"
      +S1->name+
      "'  \n\t\tcannot be redefined as a procedure!\n";
      // Встроенную константу 'abs' нельзя
переопределить как процедуру!
      // The built - in constant 'abs' cannot be redefined as a
procedure!
      return 1;
   }
   if(ref.test(PROC|BUILT)){
      ferror_message+=
      "Error[80-4] in line "+ S1->line +": You can't override
the built-in procedure '"
      +S1->name+ "'!\n";
      // Нельзя переопределять встроенную процедуру 'f'!
      // You can't override the built-in procedure 'f'!
      return 1;
   }
   if(ref.empty()){
      ref = tgName(PROC|DEFINED, S1->line);
      ref.arity = S1->count;
   }//if(ref.empty())...
   return 0;
}
int tSM::p81(){ //   PCPAR -> PCPAR $id
   if(params.count(S2->name)){
      ferror_message+=
      "Error[81-1] in line "+ S2->line +": the procedure '"
      +S1->name+
      "' duplicates \n\t\t\tthe '"
      +S2->name+"' parameter!\n";
      //в процедуре 'f' дублируется параметр 'x'
      //the 'f' procedure duplicates the 'x' parameter
      return 1;
```

```
    }

    if(S2->name==S1->name){
       ferror_message+=
       "Warning[81-2] in line "+ S2->line +": procedure '"
       +S1->name+
       "' has the same name \n"
       "\t\t\tas its parameter!\n";
       //у процедуры 'f' такое же имя, как у ее параметра
       //procedure 'f' has the same name as its parameter
    }
    params.insert(S2->name);
    S1->types += S2->types;
    ++S1->count;
    return 0;
}
int tSM::p82(){ //   BLOCK -> HBLOCK E )
    S1->name = S2->name;
    S1->line = S2->line;
    S1->types = S2->types;
    scope = 0;
    return 0;
}
int tSM::p83(){ //  HBLOCK -> BLVAR )
    scope = 2;
    return 0;
}
int tSM::p84(){ //  HBLOCK -> HBLOCK INTER
    return 0;}
int tSM::p85(){ //   BLVAR -> ( let ( LOCDEF
    return 0;
}
int tSM::p86(){ //   BLVAR -> BLVAR LOCDEF
    return 0;
}
int tSM::p87(){ //  LOCDEF -> ( $id E )
    string name = S2->name;
    std::map<std::string,std::string> Ar;
    if(locals.count(name)) {
       ferror_message +=
          "Error[87-1] in line "+ S2->line +": The local
variable '"
          +name+
          "' cannot be overridden!\n";
       // Локальную переменную 'x' нельзя
переопределять!
       // The local variable 'x' cannot be overridden!
       return 1;
```

```
   }
   else
      locals.insert(name);
   return 0;
}
//______________________
int tSM::p88(){return 0;} int tSM::p89(){return 0;}
int tSM::p90(){return 0;} int tSM::p91(){return 0;}
int tSM::p92(){return 0;} int tSM::p93(){return 0;}
int tSM::p94(){return 0;} int tSM::p95(){return 0;}
int tSM::p96(){return 0;} int tSM::p97(){return 0;}
int tSM::p98(){return 0;} int tSM::p99(){return 0;}
int tSM::p100(){return 0;} int tSM::p101(){return 0;}
int tSM::p102(){return 0;} int tSM::p103(){return 0;}
int tSM::p104(){return 0;} int tSM::p105(){return 0;}
int tSM::p106(){return 0;} int tSM::p107(){return 0;}
int tSM::p108(){return 0;} int tSM::p109(){return 0;}
int tSM::p110(){return 0;}
```