Vector Pascal ISO Specification report

21st August 2006

Introduction

Vector Pascal is an extension of the Pascal programming language which introduces operations on arrays. The compiler uses the appropriate vector instructions of the target system, in order to speed up the execution time. Vector Pascal support several CPU such as the P3, P4 and the Sony Emotion Engine. However Vector Pascal compiler doesn't seems to pass every test of the ISO test suite. The purpose of this document is to present each failures with a short description in order to correct the the compiler.

This document also draws a parallele between Vector Pascal Compiler and others compiler such as Turbo Pascal, GnuPascal, Free Pascal and Irie Pascal. The main comparison is between Vector Pascal and Turbo Pascal version 7. Turbo Pascal is a major compiler in the Pascal world. The goal of Vector Pascal is, at least, to pass the tests which Turbo Pascal pass.

The first part of this document list the tests which fails with Vector Pascal but not Turbo Pascal. The second part of this document presents the other tests.

Part I
Overview

This part quickly describes the most important errors or failures of the Vector Pascal Compiler, and tries to summarizes

1 Number of failures

Execution of the ISO test suite shows that Vector Pascal fails on 65 tests at compilation time and 34 at execution time. But, we can remove from this list failures related to reserved words or specific behaviors of Vector Pascal. Here are the list of these tests : *conf005*, *conf088*, *conf154*, *conf158* (these are the failures because of a reserved word). *conf037*, *conf169* (these are failures because of the definition of true and false).

Removing these tests ends up with 93 errors in total.

2 Most frequent failure

This section presents the most frequent causes of failure. Correcting them might reduce the number of failure on the ISO test suite.

2.1 File pointer

Like Turbo Pascal, Vector Pascal doesn't support the pointer notation for *file* or *text* variable type. This failure shows that it it impossible to use the ^postfix operator on file or text type.

Failures related to this problem are with tests :

conf067, *conf068*, *conf069*, *conf070*, *conf071*, *conf072*, *conf073*, *conf074*, *conf075*, *conf076*, *conf078*, *conf090*, *conf091*, *conf111*, *conf118*, *conf121*, *conf202*, *conf203*.

2.2 Reset function

The use of this function produces an error if they are used with other file related function (such as eof, write, etc ...).

Failures related to this problem are with tests :

conf143, *conf164*, *conf192*, *conf193*, *conf194*, *conf196*, *conf199*, *conf205*, *conf213*, *conf219*, *conf220*, *conf221*.

2.3 String type and packed array

Vector Pascal doesn't recognize a packed array of char with the smaller bound equals to 1 as a string type. So, it is impossible to use relational operators between packed arrays or between a packed array and a string.

Failures related to this problem are with tests :

conf106, *conf160*, *conf162*, *conf163*, *conf200*, *conf201*, *conf204*, *conf206*,

3 Most important failures

3.1 Div and mod operators

This problem is related to test *conf152*. This test shows that the div and mod operator don't complies with the required specification.

3.2 Sets operation

This problem is related to test *conf159*. This test shows issue with the use of set. For example, values are implicitly an automatically added to a set with some set operations.

3.3 Code generation issue

This problem is related to test *conf180*. This test shows a problem with the code generation of a simple for loop with a single statment in the loop.

3.4 relational operators and floating-point variables

Using a relational operator with a floating point variable seems to be the cause a serveral failures. This might be the cause of failure for tests : *conf032* and *conf135*.

Part II
Important Errors

This part of this document presents the tests which fails with Vector Pascal but not with Turbo Pascal.

4 conf032

Description This program exhibits all legal productions for a constant in a constant-definition.

Conformance level 0

This program fails at exection time. It seems that Vector Pascal conforms with the iso specification on this point. But the program still fails due to this expression :

```
(abs(pi+minuspi) < 0.001)
```

which returns false, and this is an error.

In this test, *pi* and *minuspi* are two constants defined in that way :

```
pi = 3.1415926;
minuspi = - pi;
```

During compilation time, Vector Pascal compiler reports this error :

```
class Op reports error in real eval of ABS
```

Although the whole expression doesn't work correctly, this one works :

```
abs(pi+minuspi) = 0
```

Using a temporary variable works as well :

```
zero := abs(pi+minuspi)
if (zero < 0.001) then write('PASS')
```

will write *PASS* on the console

5 conf037

Description This test checks that the required constant identifiers, true and false, have been correctly enumerated.

Conformance level 0

This program fails at execution time, because the ISO specification requires that the constant *false* must be equal to 0, and the constant *true* must be equal to 1.

Vector Pascal defines *false* as 0 and *true* as -1.

6 conf065

Description This test checks that packing applies to one level only using a set in a record.

Conformance level 0

This program fails at compilation time on line 48 :

```
if vudigit <> recpu.x + [d] then
```

In this test, *vudigit* is a variable of type *set of '0'..'9'*, and *d* is a variable.

```
bad set expression java.lang.ArrayIndexOutOfBoundsException: 1
```

Other compilers

- fpc fails on this test on line 21.
- gpc fails on this test on line 47.
- ipc fails on this test on line 47.

7 conf082

Description This test checks that string types with the same number of components are compatible.

Conformance level 0

This test fails on compilation time, on line 26 :

```
if string1 <> string2 then
```

In this test, *string1* and *string2* are packed arrays of char of the same size. The compiler porduces this error :

```
a variable has a rank greater than it is permited by its context.
context demands 0 variabe has rank 1.
```

The ISO specification requires that a packed array of char with its smallest range equal to 1 and its largest range greater than 1 shall be a *string type* [1].

Relational operators should work with compatible string type.

As this expression works correctly :

```
if 'ABCD' <> 'EFGH' then ...
```

It seems that Vector Pascal Compiler doesn't reconize packed arrays of char as string type.

8 conf083

Description This test checks the type-compatibility rules for sets.

Conformance level 0

This test fails on compilation time, on line 34 :

```
cset1:=[]; cset2:=['A','Z']; cset3:=['0'..'9'];
```

In this test, *cset2* has a type *set of 'A'..'Z'*, and it is the assignment of cset2 which causes the error. The compilator's output is :

```
 assignment invalid.
bad set expression java.lang.NegativeArraySizeException
```

Comment In that test, the assignment of cset3 produces an error as well. The problem might come from the definition of these variable types. If the type of these two variables changes into a set of char, there won't be any problem anymore, and the test will pass.

9 conf088

Description This test contains examples of legal type and variable declarations.

Conformance level 0

[1] see 6.4.3.2 - Array-Types

This test fails on compilation time, on line 39 :

```
name, firstname : charsequence;
```

This test fails because the *name* is a reserved word of Vector Pascal. If this variable is renamed, the compiler successfuly pass this test.

10 conf093

Description This program tests the implementation of forward directives, recursive activation, and multilevel referencing of a var parameter in procedures.

Conformance level 0

This test fails on compilation time, on line 39 :

```
one(c);
```

In this test, *one* is a procedure which had been forward declared. It seems that the error is due to the forward declaration, because if the forward declaration is removed, then the compiler successufuly pass this test.

11 conf095

Description This program tests that forward declaration and recursion in functions is permitted.

Conformance level 0

This test fails on compilation time, on line 43 :

```
c:=one(c);
```

Same error as test *conf093*.

12 conf101

Description This program checks that set, record and array parameters are permitted.

Conformance level 0

This test fails on compilation time on line 53 and produces this output :

```
java.lang.Exception: code generation failed line 53.
```

Here is the code of the method on line 53 :

```
procedure testone(set1,set2,set3,set4,set5,set6 : sett;
                  rec1,rec2,rec3,rec4,rec5 : rekord;
                  urray1,urray2,urray3,urray4 : urray);
begin
   if (set1 + set2 + set3 + set4 + set5 + set6 = [1])
   and (rec1.a + rec2.a + rec3.a + rec4.a + rec5.a = 5)
   and ((not urray1[true]) and (not urray2[true])
   and (not urray3[true]) and (not urray4[true]))
   then
      counter:=1
end;
```

And here are the definition of the types :

```
type
   sett     = set of 0..20;
   rekord   = record
               a : integer
              end;
   urray    = array[boolean] of boolean;
```

the problem might come from the set part of this function. If the set part of the expression is commented, then the compilation doesn't fail anymore.

Comment This test takes a very long time to compile, maybe because of the sets.

13 conf105

Description This test checks that the scope of parameter identifiers is separate from that of the function result.

Conformance level 0

This test fails on compilation time, on line 22 :

```
function f( t: integer): t;
```

The output message of the compiler is :

```
(ref int32)mem(+(^ebp),    8)) is not a constant
```

In this test t is the parameter of the f function, but it is also a type defined in this way :

```
t = 0..10;
```

the problem comes from the scope of the parameter. If it is renamed, then the compiler successfuly pass this test.

Other compilers

- ipc fails on the same line and for the same reason.

14 conf106

Description This test checks the scope of procedures combined with nesting.

Conformance level 0

This test fails on compilation time, on line 23 :

```
procedure trace( name: char);
```

This test fails because *name* is a reseverd word of Vector Pascal. If this variable is renamed into something different, then the test fails on line 74 :

```
if fail or (tra <> 'zzzzydyzyxdyzzydyzyxxdcdbcd') then
```

Here, the variable *tra* is a packed array of char. This test fail because *tra* is a *string* and Vector Pascal doesn't reconize packed array of char as a string type (see test *conf082*).

15 conf116

Description This program tests that predefined standard procedures may be redefined with no conflict.

Conformance level 0

This test fails on compilation time, on line 20 :

```
procedure write(var a : integer);
```

This test fails because *write* is a reserved word in Vector Pascal.

The ISO specification requires that a standard function may be redefined. This seems not possible with the *write* function.

16 conf117

Description This test checks that a predefined function can be redefined.

Conformance level 0

This test fails on compilation time, on line 20 :

```
function abs(y:integer): integer;
```

This test fails because *abs* is a reserved word in Vector Pascal.

The ISO specification requires that a standard function may be redefined. This seems not possible with the *abs* function.

17 conf120

Description This test writes a fixed number of characters to a file of char, reads back the same number of characters, and then checks that eof is true.

Conformance level 0

This test fails on compilation time, on line 20 :

```
min,max,ch1,ch2 : char;
```

This test fails because the two variable names *min* and *max* are reserved keywords in Vector Pascal. If these two variables are renamed, the test still fails on line 58 :

```
for ch1 := min to max do write(f,ch1);
```

The compiler produces this message :

```
java.lang.Exception: code generaion failed line 58.
```

In this test *ch1*, *min* and *max* are char, and *f* is a file of char. This test takes a long time to compile, and finaly fails by producing the error message above.

18 conf129

Description This test checks that the first parameter of dispose can be a function-designator.

Conformance level 0

This test fails on compilation time, on line 35 :

```
dispose(alterptr(ptr1));
```

In this test, *ptr1* is a pointer, and the function *alterptr* returns a pointer. The ISO specification requires that a function call might be the first parameter of dispose (as long as this function returns a pointer).

Vector Pascal doesn't seems to allow this. The compiler produces this message :

```
Processing built in procedure
disposeilcg.Pascal.TypeIdError: alterptr.
```

Other compilers

- ipc fails on the same line and for the same reason.

19 conf130

Description This test checks that new can be called with a component of packed structure as its parameter.

Conformance level 0

This test fails on execution time just because Vector Pascal defines *true* as -1 and *false* as 0. Except this point, the compiler pass this test successfuly.

20 conf135

Description This program tests the implementation of the arithmetic functions sin, cos, exp, ln, sqrt, and arctan.

Conformance level 0

This test fails on execution time. This test fails on these functions :

- sin
- cos
- ln
- sqrt

This cause of this failure is similar to test *conf032*. Here is a part of this test :

```
if ((-0.001<sin(pi)) and (sin(pi)<0.001)) and
   ((0.70<sin(pi/4)) and (sin(pi/4)<0.71)) then
```

In this test, the *sin* function seems to work correctly because Vector Pascal is able to pass this test with the use of intermediate variables. For example :

```
r1 :=sin(pi);
r2 :=sin(pi/4);
if ((-0.001<r1) and (r1<0.001)) and
   ((0.70<r2) and (r2<0.71)) then
```

21 conf137

Description This program checks that the implementation of the ord function.

Conformance level 0

This test fails on execution time because of this line :

```
if (ord(false)=0) and (ord(true)=1) then
```

This test fails because Vector Pascal defines *true* as -1. Except this fact, Vector Pascal pass this test successfuly.

22 conf139

Description This program tests that the required ordinal functions succ and pred.

Conformance level 0

This test fails on compilation time, on line 31 :

```
if succ(false) and not pred(true) then
```

Here is the output of the compiler :

```
Error (pred = 149) found when symbol (identifier) expected
```

This is just a parenthesis problem wich can be solved in that way :

```
if succ(false) and not (pred(true)) then
```

The problem might come from the *not* keyword which expects an identifier.
When the parenthesis are used, the compiler pass this test successfuly.

23 conf144

Description This program tests the precedence of operators.

Conformance level 0

This test fails on execution time. This test is composed of several parts. Each part test a kind of operator (such as addition, multiplication, relational, etc ...).

This test fails on the boolean and relational parts of this test. Mainly because of the definition of *true* and *false*[2].

24 conf145

Description This test checks that the member designator x..y, where x > y, denotes no members.

Conformance level 0

[2] see conf037

This test fails on compilation time, on line 23 :

```
if ([x..y]=[]) and ([127..0]=[]) then
```

In this test, x = 2 and y = 1. The first part of the expression (*[x..y]=[]*) produces this error :

```
ArrayIndexOutOfBoundsException: 1
```

And the second part of the expression (*[127..0]=[]*) produces this error:

```
NegativeArraySizeException
```

Whereas the first error seems to be a bug of Vector Pascal, the second message shows that Vector Pascal doesn't allow to define a set with the lower bound greater than the higher bound.

Other compilers

- gpc fails for the same reason.

25 conf146

Description This test examines valid productions of set-constructors.

Conformance level 0

This test fails on compilation time, on line 36 :

```
if [1,2,n,succ(n),1]=[1..3] then
```

produces the same error as test *conf145*.

26 conf147

Description This test checks that the set-constructor can denote both packed and unpacked set types in the appropriate contexts.

Conformance level 0

This test fails on compilation time, on line 41 :

```
(cnormal=['C'..'N']-['I'..'M']) and (cpakked=['C'..'N']-['I'..'M']) ...
```

In this test *cnormal* is a set of char, and *cpakked* is packed set of char. This test tries to use the same constructor on packed and unpacked set. This test also use different kind of set (char, enum and integer). Sets of char a the only ones which produce this error :

```
processing dyadic expression SET OF 64..71 - SET OF 71..79
java.lag.NegativeArraySizeException and no user defined operator
matches context after rank reduction.

Cannot find operator - implemented by procedure (
record data:array[0..1]of ieee32 end,
record data:array[0..1]of ieee32 end->
    record data:array[0..1]of ieee32 end)
to match context (
record data:array[0..1]of ieee32 end,
record data:array[0..1]of ieee32 end
)
type mismatch: SET OF 64..79 incompatible with record
data:array[0..1]of ieee32
can't find cast operator to match context for
SET OF 64..79
```

The same expression works in an assignment statment :

```
cnormal:=['C'..'N']-['I'..'M']
```

Other compilers

- fpc fails on the same line.
- gpc fails on the same line.

27 conf148

Description This test checks that the set-constructor can denote all values allowed by the canonic set-type to which it belongs.

Conformance level 0

This test fails on compilation time, on line 52 :

```
    max: integer;
```

This test fails because the variable name*max* is a reserved keyword of Vector Pascal.

If this variable is renamed, then the test fails on compilation time, on line 125 :

```
for Cv:=MinChar to MaxChar do Cray[Cv]:=false;
```

The compiler produces this message :

```
java.lang.Exception: code generation failed line 125
```

The problem is similar to test *conf120*.

28 conf149

Description This test checks that the set-constructor can denote all values allowed by the canonic set-type to which it belongs.

Conformance level 0

This test fails on compilation time, on line 32 :

```
Sset:=[LoIn..HiIn];
```

In this test the variable *Sset* is a set of LoIn..HiIn. *LoIn* and *HiIn* are constants with value 0 and 127. The compiler produces this error message :

```
Value beyond range of set type : 16 derived from 16
produces offset 2 but bitmap is of length 2 bytes.
```

The problem may to come from the size of the bitmap. The highest possible value of *HiIn* is 15, even if *LoIn* is greater than 0.

Comment The compiler successuly pass this test with *HiIn* lower than 16.

29 conf150

Description This test checks that the set-constructor can denote all values allowed by the canonic set-type to which it belongs.

Conformance level 0

This test fails on compilation time, on line 41 :

```
max: integer;
```

This test fails because the name *max* is a reserved keyword in Vector Pascal. Once renamed, the test fails on compilation time, on line 67. The error seems to be the same as the test *conf120*.

30 conf152

Description This program checks that div and mod are implemented by the rule specified by the Pascal Standard.

Conformance level 0

This test fails on execution time

The Pascal standard specifies the *div* operator in that way :

```
abs(i) - abs(j) < abs((i div j) *j) <= abs(i)
```

In fact, it seems that Vector Pascal has strange behaviour with the div operator. Here is an example of code :

```
r : real;

r:= (2 div 3);
writeln((2 div 3));   { 1) prints 0}
writeln(r);           { 2) prints 1.000}

r:= (2 div 3)*3;
writeln((2 div 3)*3); { 3) prints 2}
writeln(r);           { 4) prints 3.000}
```

As you can see, only the first expression prints the right result. One of the errors can also comes from an implicit cast with the use of the function *writeln*. If the expression *2 div 3* is used with a relational operator as in :

```
if ((2 div 3)*3) = 3 Then writeln('3');
```

will print 3, wich shows that the *writeln* (as in expression 1 and 3) function should do something more than printing the result of an expression.

It is also interesting to note that if the fractonal part of the result is smaller or equal than 0.5, then everything seems to work correctly (except the third statment, which still prints 2).

31 conf154

Description This program checks that maxint satisfies the conditions laid down in the Pascal Standard.

Conformance level 0

This test fails on compilation time, on line 20 :

```
i, max : integer;
```

This test fails because the variables name *max* is a keyword in Vector Pascal. If this variable is properly renamed then the compiler successfuly pass this test.

32 conf158

Description This test checks the set operators, with all patterns possible.

Conformance level 0

This test fails on compilation time, on line 74 :

```
procedure Perm(k:Range; var sp:esetType; var vp:erayType);
```

This test fails because the function name *perm* is a keyword in Vector Pascal. If this method is properly renamed, then the compiler successfuly pass this test.

33 conf159

Description This test checks the set operators, with random values.

Conformance level 0

This test fails on execution time.

Although this test is quite simple (it test union, product and difference between two sets), it show many problems of the Vector Pascal compiler.

Fistly, if we have this definition :

```
range = 0..10;
setType = set of range;
jj : setType;
```

Then, the compiler allows to write this code :

```
k: range;
for k:=0 to 25 do
    if k in jj then write(k);
```

And more surprisely, the result of an execution will be :

```
11 12 19
```

It is important to notice that k has a type *range*. If k were an integer, the following error doesn't occur.

This shows that even if the larger bound of the set is ten, it is possible to access some data above this boundary, and even shows some values outside the boundary are "inside" the set. This probleme occures with packed and unpacked set. Unfortunately, the behaviour of tests of set membership with values outside of the boundary is a little more complicated. If we take the previous example, it is possible to check the membership to values up to 126. If we try to check the membership to values up to 127, then the program fails at execution time with a *General Protection Fault*. If we put a value greater than 127, the for loop works correctly and shows *jj* empty.

Secondly, suppose we have this definition :

```
range = 0..10;
setType = set of range;
s1,s2,rr : setType;

...
s1:=[3];
```

```
s2:=[2];
rr := s1+s2;

k: integer;
for k:=0 to 10 do
    if k in rr then write(k);
```

Then the output of this program will be :

```
2 3 10
```

We can see that the value 10 has been added to the resulting set during the union operation. This problem seems to occures only when variables of same type are declared on the same line. And the bug occurs only with certain value in the set (in our case, the problem comes from the value 2). In this example, *rr* get an additional 10 because *rr* is defined on the same line, and after *s2*. If we put 2 into *s1*, then *s2* will get an additional 10, because *s2* is defined after *s1*. Other values can produces this bug.

```
s1:=[0]; {s2 get 8}
S1:=[1]; {s2 get 9}
s1:=[2]; {s2 get 10}
```

It is also possible to produces this bug with other kind of definition (one variable per line for example).

Finally, this test uses three sets of 127 value, and runs a 100 times the set operations on them. The executable produced by Vector Pascal takes several minutes to execute this test whereas the program compiled bu Turbo Pascal take less than 1 secondes. This shows that something is going wrong with set operations.

34 conf162

Description This test checks the use of relational operations on long strings.

Conformance level 0

This test fails on compilation time, on line 25 :

```
if s1 <> s2 then
```

In this test, *s1* and *s2* are packed array [1..37] of char. So they should be considered as string type. The same error occures on test *conf082*.

35 conf169

Description This test checks a nested if-statement whose syntax is apparently ambiguous.

Conformance level 0

This test doesn't print anything on the command line because of line 23 :

```
for b:=false to true do
```

As Vector Pascal defines *false* as 0 and *true* as -1, the progam doesn't execute the for statment. When the for statment is rewritten in the correct way, Vector Pascal pass this test successfuly.

36 conf171

Description This test checks that a processor will handle a case-statement where the case-constants are not close together.

Conformance level 0

This test fails on compilation time, the compiler produces a stack overflow error. Here is a comment included in the test source file :

> Most processors issue a jump table for a case, regardless of its structure. It is easy to optimise case-statements to generate conditional statements if this is more compact. Processors which employ simple strategies for implementation of case-statements may fail this test.

And here is the code of the test :

```
var
   i,j:integer;
begin
   i:=-1000;
   for j:=1 to 2 do
      case i of
      -1000: i:=-i;
      1000: writeln(' PASS...6.8.3.5-2 (CONF171)')
      end
end.
```

If the case-constants are closer than this test (10 and -10 for example), then the compiler successfuly pass this test.

37 conf177

Description This program checks that assignment follows the evaluation of both expressions in a for-statement.

Conformance level 0

This test fails on compilation time, on line 23 :

```
for i:= (i+1) to (i+10) do
```

This test fails because Vector Pascal doesn't allow to alter a variable both in the loop and in the range definition of the loop.

38 conf178

Description This test checks that extreme values may be used in a for-statement.

Conformance level 0

This test count 4 times from 0 to maxint, so it take some times to get the results of this test.

39 conf188

Description This test checks that the selection of a variable in the record-variable-list is performed before the component statement is executed.

Conformance level 0

This test fails on execution time, because of this part of the test :

```
k:=1; {(0)}
with a[k] do
begin
   j:=1; {(1)}
   k:=2; {(2)}
   i:=2  {(3)}
end;
```

In this test, a is an array of two records (which contains two integers i and j). On the line 0, we choose the first record of a. On line 1, *a[1].j* gets 1. On line we change the value of k, which also change the record in the array. So on line 3, *a[2].i* gets 2, and this is not conform with the specification. It requires that the selection of the variable in the *with* statment is performed before the execution of the componment statment.

40 conf189

Description This test checks that the selection of a variable in the record-variable-list is performed before the component statement is executed.

Conformance level 0

This test fails on execution time. This test is quite equivalent to *conf188*.

41 conf200

Description This test checks that the default value for the field width of a char-type is one.

Conformance level 0

This test fails on compilation time, on line 35 :

```
if (whatwasread=whatitshouldbe) then
```

In this test, *whatitshouldbe* is a string, and *whatwasread* is a packed array of char. This test fails because Vector Pascal doesn't reconize a packed array of char as a string type. See *conf082*.

42 conf201

Description This test checks the implementation of integer output.

Conformance level 0

This test fails on compilation time, on line 28 :

```
if (b='  0  1 -1 10 99100-1001111') then
```

In this test, *b* is a packed array of char. See test *conf082*.

43 conf204

Description This program checks that the fixed point representation of real numbers is correctly written to textfiles.

Conformance level 0

This test fails on compilation time, on line 44 :

```
if (b=' 0.0   1.0  -1.0123.4  0.01.0-1.0123.4  ') and pass then
```

In this test, *b* is a packed array of char. See test *conf082*.

44 conf206

Description This program checks the implementation of procedure writeln.

Conformance level 0

This test fails on compilation time, on line 32 :

```
if (a=b) then
```

In this test, *a* and *b* are two packed set of char. See test *conf082*.

45 conf214

Description The case-index of the case statement is an expression and hence no subrange check can be performed upon the value.

Conformance level 0

This test fails on execution time This test is quite simple, so here is its source code :

```
var i,j : 99..100;

begin
  i := 100;
  j := 100;
  case i+j of
    200: writeln(' PASS...6.8.3.5-23 (CONF214)');
  end;
end.
```

The problem comes from the range of i and j. Vector Pascal check the results of the expression $i+j$, were the specification requires that no subrange check should be performed.

46 conf215

Description This program checks that a directive is not an identifier.

Conformance level 0

This test fails on compilation time, on line 16 :

```
direction = (backward,forward);
```

This test fails because *forward* is a reserved keyword.

Part III
Other Errors

This part of this document presents the other failures

47 conf003

Description This test contains every valid pair of adjacent lexical units.

Conformance level 0

This test fails on compilation time, on line 192 :

```
if ('A' <= 'Z') or ([i1..3] <= stv) then
```

The compiler produces this output :

```
bad set expression java.lang.ArrayIndexOutOfBoundsException:1
```

Is the same error as test *conf083*.

Other compilers

- tp7 fails on a forward declaration line 95.
- fp fails on line 40, with a syntax error.
- gpc fails on line 298 on a syntax error.

48 conf005

Description This program contains identifiers which some processors may disallow because they have extended the word-symbol list.

Conformance level 0

This test fails on compilation time, on line 33 :

```
new:     (dispose,string,oct,hex,origin,address,absolute,
```

Other compilers

- tp7
- vp
- ipc

In this line, string is a reserved key-word for Vector Pascal. Other string literals are tested in this program, and here is a list of those which cause problem :

```
external, unit, interface, implementation, otherwise, uses, exit
```

Here is the complete description of the test :

> Clause 5.1 of the Standard requires processors to be capable of accepting all features of the language specified in clause 6, which includes the domain of allowed identifiers. Clearly, as this is an infinite domain, a test can only inspect a few identifiers for conformance. This test has been constructed to contain a collection of identifiers that have been disallowed by various implementations, or which are thought to be under threat of such action. Definition 3.2 defines 'extension' as permitting additional reserved words; however the wording of clause 5.1 makes it clear that the processor must be able to accept all identifiers in its Standard-conforming mode; an extended mode may be needed to enable extensions that add new reserved words.

Comment

- Turbo Pascal fails on this test on line 33
- Free Pascal fails on line 33
- Irie Compiler fails on line 76

49 conf019

Description This program checks that the two equivalent forms of comment delimiters are implemented correctly.

Conformance level 0

This program fails at execution time, because two kinds of comment are used here. Moreover, It is specified that a comment which begins with { or *(** could end either with } or ***)**. It is exactly what it is been tested here. Vector Pascal forbids to mix these two kinds of comments.

Comment

- Turbo Pascal fails on this test at execution time.
- Free Pascal fails on this test at execution time.
- GnuPascal fais on this test at execution time.

50 conf024

Description This is the minimal-program.

Conformance level 0

This test fails on execution time

51 conf054

Description This test checks that arrays, whose component-types are file-types, behave as required.

Conformance level 0

This test fails on compilation time, on line 29 :

```
f^ := element;
```

In this test, *f* is a file. Vector Pascal doesn't support the pointer like notation for file variables.

Comment

- TurboPascal fails this test on the same line.
- FreePascal fails this test on the same line.
- Irie Interpretor fails on this test at execution time.

52 conf067

Description This program tests if an end-of-line marker is inserted at the end of the line, if not explicitly done in the program.

Conformance level 0

This test fails because Vector Pascal doesn't support dereferencing of text or file variable (as test *conf054*). the failure is on line 29 :

```
if file1^='A' then get(file1) else state:=wrong;
```

Other compilers

- tp7 fails for the same reason.

53 conf068

Description This test checks that file-types can appear in a record declaration.

Conformance level 0

This test fails on compilation time, on line 41 :

```
r.out.outint^:=ord(c);
```

In this test, outint is a type file. As Vector Pascal doesn't support dereferencing of file variable, it produces the same error as tests *conf067* and *conf054*.

54 conf069

Description This test checks that non-text local files are processed correctly.

Conformance level 0

This test fails on compilation time, on line 57 :

```
f^:=a; put(f); a:=not(a)
```

In this test, f is a file of boolean, so Vector Pascal produces the same errors as tests *conf068*, *conf067* and *conf054*.

55 conf070

Description This test checks that non-text local files are processed correctly.

Conformance level 0

This test fails on compilation time, on line 57 :

```
f^:=a; put(f); next(a)
```

In this test, *f* is a file. Vector Pascal produces the same errors as test *conf054*.

56 conf071

Description This test checks that non-text local files are processed correctly.

Conformance level 0

This test fails on compilation time, on line 50 :

```
f^:=temp^.p; put(f); temp:=temp^.p
```

In this test, *f* is a file. Vector Pascal produces the same errors as test *conf054*.

57 conf072

Description This test checks that packed non-text local files are processed correctly.

Conformance level 0

This test fails on compilation time, on line 57 :

```
f^:=a; put(f); a:=not(a)
```

In this test, *f* is a file. Vector Pascal produces the same errors as test *conf054*.

58 conf073

Description This test checks that packed non-text local files are processed correctly.

Conformance level 0

This test fails on compilation time, on line 57 :

```
f^:=a; put(f); next(a)
```

In this test, *f* is a file. Vector Pascal produces the same errors as test *conf054*.

59 conf074

Description This test checks that packed non-text local files are processed correctly.

Conformance level 0

This test fails on compilation time, on line 50 :

```
f^:=temp^.p; put(f); temp:=temp^.p
```

In this test, *f* is a file. Vector Pascal produces the same errors as test *conf054*.

60 conf075

Description This test checks that two local files with similar identifiers are correctly distinguished.

Conformance level 0

This test fails on compilation time, on line 53 :

```
g^:=random; put(g);
```

In this test, *g* is a file. Vector Pascal produces the same errors as test *conf054*.

61 conf076

Description This test contains six simultaneous open files.

Conformance level 0

This test fails on compilation time, on line 44 :

```
0: begin a0^:=a; put(a0) end;
```

In this test, *a0* is a file. Vector Pascal produces the same errors as test *conf054*.

62 conf078

Description This test checks that the domain-type of a pointer-type can be a file-type.

Conformance level 0

This test fails on compilation time, on line 28 :

```
write(tofile,fromfile^);
```

In this test, *fromfile* is a text. Vector Pascal produces the same errors as test *conf054*

63 conf090

Description This program tests that file buffers may be referenced.

Conformance level 0

This test fails on compilation time, on line 31 :

```
fyle^.urray[1]:='O';
```

In this test *fyle* is a file. Vector Pascal produces the same error as test *conf054*.

Other compilers

- tp7 fails for the same reason

64 conf091

Description This test checks that a value may be assigned to a buffer-variable while mode=inspection.

Conformance level 0

This test fails on compilation time, on line 20 :

```
string : packed array[1..10] of char;
```

This test fails because *string* is a reserved word of Vector Pascal. When properly renamed, this test fails on line 29 :

```
f^ := 'A';
```

f is a text. Vector Pascal doesn't seems to reconize this syntax. see test *conf054*

Other compilers

- tp7 fails for the same reason

65 conf103

Description This test checks the scope conformance of procedural parameters.

Conformance level 0

This test fails on compilation time, on line 41 :

```
conforms(alsoconforms)
```

Here, *conforms* is a function which takes another function as parameter. On this line, the test try to give *alsoconforms* (a previously defined function) to *conforms*. Vector Pascal doesn't seem to accept such syntax, and produces this error :

```
in statment ( expected
```

Other compilers

- tp7 fails at the definition of the *conforms* function, line 32.
- fpc also fails line 32.

66 conf111

Description This test checks that non-text file parameters are accepted.

Conformance level 0

This test fails on compilation time, on line 79 :

```
        if a.p <> gˆ.p then
```

In this test, *g* is a file. Vector Pascal doesn't support the pointer like notation for file variables. See test *conf054*.

67 conf112

Description This program tests that procedures may be passed to other procedures and functions as parameters.

Conformance level 0

This test fails on compilation time, on line 40 :

```
b(a);
```

Here , b is a procedure which takes another procedure as a parameter (a in this case). The compiler produces this error :

```
compilation failed
in statment internal compiler error
processing procesdure call
java.lang.ArithmeticException: / by zero
```

As test *conf103* fails, it seems normal that this one fails as well.

Other compilers

- tp7 fails on line 24.

68 conf113

Description This program tests the environment of procedural parameters.

Conformance level 0

This test fails on compilation time, on line 51 :

```
p(f,r)
```

In this test p, f and r are procedures. The compiler produces the same error as test *conf103*.

Other compilers

- tp7 fails on line 21 (definition of the procedure p)

69 conf114

Description This test contains all the four possible combinations of procedural and functional parameters whose parameters are themselves procedures and functions.

Conformance level 0

This test fails on compilation time, on line 30 :

```
p11(pglobal)
```

In this test, *p11* and *pglobal* are procedures. The compiler produces the same error as test *conf112*.

Other compilers

- tp7 fails on line 28.

70 conf115

Description This program tests that functions may be passed to procedures and functions as parameters.

Conformance level 0

This test fails on compilation time, on line 35 :

```
b(a);
```

This test is similar to test *conf112*.

Other compilers

- tp7 fails on line 25.

71 conf118

Description This program tests that the first element of the file fyle is assigned to the buffer variable fylewˆhen the procedure reset is used with the file.

Conformance level 0

This test fails on compilation time, on line 26 :

```
if fyleˆ='A' then
```

In this test, *fyle* is a text. Vector Pascal doesn't support the pointer like notation for text variables. See test *conf054*.

Other compilers

- tp7 fails for the same reason.

72 conf119

Description This program checks that a rewrite on the file sets eof to be true.

Conformance level 0

This test fails on execution time. This test just executes the *rewrite* function on a text variable, and check if *eof* is set to be true.

Vector Pascal doesn't set *eof* to true when this function is called, that is why this test fails.

73 conf121

Description This test checks the behaviour of the procedures 'get' and 'put' in the same program.

Conformance level 0

This test fails on compilation time, on line 34 :

```
f^.country := 'holland   ';
```

In this test, *f* is a file. This test fails for the same reason as test *conf054*.

Other compilers

- tp7 fails for the same reason

74 conf131

Description This program tests that pack and unpack are implemented in this processor.

Conformance level 0

This test fails on compilation time, on line 30 :

```
unpack(pacone,unone,5);
```

This test fails because the function *unpack* is not defined in Vector Pascal. This function is described in part 6.6.5.4 of the Pascal ISO Specification.

Other compilers

- tp7 fails for the same reason.

75 conf132

Description This test contains calls of pack and unpack in which the index-types of the two arrays do not possess the same type.

Conformance level 0

This test fails on compilation time, on line 28 :

```
unpack(pac1,unpac,one);
```

This test fails for the same reason as test *conf131*.

Other compilers

- tp7 fails for the same reason.

76 conf141

Description This program tests the functions eof and eoln.

Conformance level 0

This test fails on execution time, and even doesn't finish correctly. This test fails and produces this error :

```
Pascal file sub-system error : 2.
```

The line which produces this error is line 31 :

```
reset(fyle);
```

Here, *fyle* is a text. If this line is commented, the program terminates correctly with this message :

```
FAIL...6.6.6.5-1, EOF FUNCTION(1) (CONF141)

FAIL...6.6.6.5-1, EOLN FUNCTION(2) (CONF141)
```

However, the use of the *reset* function alone doesn't crash the program. It is the call of either *eof* or *eoln* after a call of *reset* which produces a crash.

77 conf143

Description This test checks that the eof function continues to yield true after the end of the file has been reached.

Conformance level 0

This test fails on execution time, and even doesn't finish correctly. The program exits with this message :

```
binary file output error
```

This error seems to come from the type of the file used in this test. The file used is a file of integer, because it fails when it tries to write integer into the file. If the type is changed into a file of char, then we get the same error as test *conf141*.

78 conf153

Description This program checks that constant and variable operands for div and mod produce the same result, and that negative operands, where permitted, are accepted.

Conformance level 0

This test fails on execution time. As this test uses the operators *div* and *mod*, the errors a similar as test *conf152*.

Other compilers

- tpc also fails this test.
- fpc fails this test.

79 conf160

Description This program tests the use of relational operators on strings.

Conformance level 0

This test fails on compilation time, on line 20 :

```
string=packed array[1..7] of char;
```

This test fails because *string* is a reserverd word iun Vector Pascal. If properly renamed, this test fails at compilation time. See test *conf082*.

Other compilers

- tp7

80 conf163

Description This test checks that relational-operators applied to strings behave correctly.

Conformance level 0

This test fails on compilation time, on line 44 :

```
max: integer;
```

This test fails because *max* is a reserved key word in Vector Pascal. When properly renamed, this test fails on line 71 :

```
low := rec1.s255 < rec2.s255
```

In this test, *rec1.s255* and *rec2.s255* are packed array[1..255] of char. The compiler produces the same error as test *conf082*.

Other compilers

- tp7 fails on line 85, for the same reason.

81 conf164

Description This program checks that function calls can appear in different positions.

Conformance level 0

This test fails on execution time, and even doesn't finish correctly. The output message is :

```
Pascal file sub-system error: 2
```

This crqsh occures when lines 82-83 is executed :

```
reset(x[1]);
read(x[sideeffect(1)],ch, ch, ch);
```

This produces the same error as test *conf141*.

Other compilers

- tp7 at compilation time on line 78.

82 conf165

Description This program checks that function calls can appear in different positions with pack and unpack.

Conformance level 0

This test fails on compilation time, on line 51 :

```
pack(a[sideeffect(1)], sideeffect(1),
```

This program fails for the same reason as test *conf131*.

Other compilers

- tp7 fails for the same reason.

83 conf166

Description This test checks that a label in a recursive procedure can be the destination of a goto-statement.

Conformance level 0

This test fails on execution time. The test shows errors with the use of goto-statment with recursive procedure. Experimentation on this test shows that the control flow is not correct.

Here is an example taken from the source code of test *conf166* :

```
procedure p;
label
    111;
var
    s:integer;

    procedure q;
    begin
        goto 111;
    end;
begin
s:=0;
q;
111: writeln(s);
end;


begin
p;
end;
```

The result of this program will be *684160* and *0*. Where the result should be *0*. This output shows two problems.

The first one is a problem of the control flow, because the *writeln* instruction is executed two times. As the label is ouside of the *q* procedure, this *q* procedure should terminate a soon as the control flow leaves its scope(after the execution of the goto statment).

The second one is a problem of definition of *s*, because the program prints 684160 as the value of *a* whereas it is equal to 0. This problem might be a side effect of the first problem.

Other compilers

- tp7 fails on line 40, because Turbo Pascal doesn't support goto and label in different blocks.

84 conf180

Description This program checks the order of evaluation of the limit expressions in a for-statement.

Conformance level 0

This test fails on execution time. Here is a simplified part of the source :

```
var
   i,j:integer;

begin
   j:=0;
   for i:=1 to 12 do
      j:=j+1; {(1)}
   if (j=12) then
      writeln(' PASS...6.8.3.9-4 (CONF180)')
   else
      writeln(' FAIL...6.8.3.9-4 (CONF180)')
end.
```

This test fails, because the value of j is 13. However strange results appears if we put line (1) between begin and end statment. In this way, the final value of j will be 1. One way to pass the test is to add an write instruction in the for loop.

```
for i:=1 to 12 do
begin
    writeln('foo');
    j:=j+1; {(1)}
end;
```

85 conf192

Description This program attempts to perform recursive I/O using a different file for the second I/O action.

Conformance level 0

This test fails on execution time. It produces the same error as test *conf141*.

Other compilers

- tpc fails at execution time, and doesn't even finish correctly.

86 conf193

Description This program attempts to perform recursive I/O using the same file for the second I/O action.

Conformance level 0

This test fails on execution time. It produces the same error as test *conf141*.

Other compilers

- tpc fails at execution time, and doesn't even finish correctly.

87 conf194

Description This test checks that a single read statement with many variables is equivalent to many read statements containing one variable each.

Conformance level 0

This test fails on execution time. It produces the same error as test *conf141*.

Other compilers

- tpc fails at execution time, and doesn't even finish correctly.

88 conf195

Description This test checks that a read of a character variable is equivalent to correctly positioning the buffer variable.

Conformance level 0

This test fails on compilation time, on line 28 :

```
a1:=f^; get(f);
```

This test fails on execution time. It produces the same error as test *conf143*.

Other compilers

- tpc fails for the same reason.

89 conf196

Description This test checks that integers and reals are read correctly from a file.

Conformance level 0

This test fails on execution time. It produces the same error as test *conf141*.

Other compilers

- tpc fails at execution time, and doesn't even finish correctly.

90 conf197

Description This test checks that readln is correctly implemented.

Conformance level 0

This test fails on compilation time, on line 37 :

```
while not eoln(f) do get(f);
```

This test fails on execution time. It produces the same error as test *conf143*.

Other compilers

- tpc fails for the same reason.

91 conf198

Description This test checks that a write that does not specify the file always writes on the default file at the program level, not any local variable with the same identifier.

Conformance level 0

This test fails on execution time. It also doesn't finish correctly, and produces this error message :

```
Unassigned file handle 129064.
```

Other compilers

- tpc fails on execution time.

92 conf199

Description This test checks that a write procedure with many parameters is equivalent to many write procedures with one parameter each.

Conformance level 0

This test fails on execution time. It produces the same error as test *conf141*.

Other compilers

- tpc fails at execution time, and doesn't even finish correctly.

93 conf202

Description This test checks that the literal constant 0.0 is correctly written to a textfile (its exact representation can be derived directly from the Standard).

Conformance level 0

This test fails on compilation time, on line 31 :

```
while f^ <> '+'  do
```

This test fails because Vector Pascal doesn't support pointer notation for fils or text type. See test *conf054*.

Other compilers

- tp7 fails for the same reason

94 conf203

Description This program checks that the floating-point representation of real numbers is correctly written to textfiles.

Conformance level 0

This test fails on compilation time, on line 31 :

```
    if f^ <> ' ' then
```

This test fails because Vector Pascal doesn't support pointer notation for fils or text type. See test *conf054*.

Other compilers

- tp7 fails for the same reason

95 conf205

Description This test checks that strings are correctly written onto a textfile.

Conformance level 0

This test fails on execution time. It produces the same error as test *conf141*.

Other compilers

- tpc fails at execution time, and doesn't even finish correctly.

96 conf207

Description This program checks that the procedure page is implemented.

Conformance level 0

This test fails on compilation time, on line 25 :

```
page(output);
```

This test fails because the page function is not implmented in Vector Pascal.

Other compilers

- tp7 fails for the same reason.

97 conf212

Description A single test that checks external file binding for binary files.

Conformance level 0

This test fails on execution time. It produces the same error as test *conf143*.

Other compilers

- tpc fails at execution time, and doesn't even finish correctly.

98 conf213

Description A single test that checks external file binding for text files.

Conformance level 0

This test fails on execution time. It produces the same error as test *conf141*.

Other compilers

- tpc fails at execution time, and doesn't even finish correctly.

99 conf219

Description The revision of ISO 7185 in 1990 amended the definition of unsigned-real to allow values to be read from textfiles with digit-sequences which exceed the local value of maxint.

Conformance level 0

This test fails on execution time. It produces the same error as test *conf141*.

Other compilers

- tpc fails at execution time, and doesn't even finish correctly.

100 conf220

Description The revision of ISO 7185 in 1990 clarified the definition of textfile input. Multiple parameters are evaluated in textual order as though by separate statements.

Conformance level 0

This test fails on execution time. It produces the same error as test *conf141*.

Other compilers

- tpc fails at execution time, and doesn't even finish correctly.

101 conf221

Description The revision of ISO 7185 in 1990 clarified the definition of textfile input. The file parameter is evaluated once at the beginning of a read statement.

Conformance level 0

This test fails on execution time. It produces the same error as test *conf141*.

Other compilers

- tpc fails at execution time, and does't even finish correctly.