EXPR_BAD_RANGE

In this section:

Overview

This error is generated whenever an expression uses a pointer that is outside its legal range. In many circumstances, these pointers are then turned into legal values before use (code generated by automated programming tools such as `lex` and `yacc`), so this error category is suppressed by default. If used with their illegal values, other Insure++ errors will be displayed which can be tracked to their source by re-enabling this error class.

Code	Description	Enabled	Reported	Platform
EXPR_BAD_RANGE	Expression exceeded range	✗	Runtime	Windows/Unix

Problem

In the following code, the a pointer initially points to a character string. It is subsequently incremented beyond the end of the string. When the resulting pointer is used to make an array reference, a range error is generated.

```
*
* File: exprange.cpp
*/
main()
{
        char *a = "test";
        char *b;

        a += 6;
        b = &a[1];
        return (0);
}
```

Diagnosis at Runtime

```
[exprange.c:10] **EXPR_BAD_RANGE**
>>                        b = &a[1];
        Expression exceeded range: &a[1]
        Index used: 1
        Pointer                 : 0x0000e226
        In block         : 0x0000e220 thru 0x0000e224 (5 bytes)
                                        a, declared at exprange.c, 6
        Stack trace where the error occurred:
                        main() exprange.c, 10
```

- Line 1: Source line at which the problem was detected.
- Line 3: Description of the problem and the expression that is in error.
- Line 5: Description of the memory block to which the out of range pointer used to point, including the location at which it is declared.
- Line 8: Stack trace showing the function call sequence leading to the error.

Repair

In most cases, this error is caused by incorrect logic in the code immediately prior to that at which the message is generated. Probably the simplest method of solution is to run the program under a debugger with a breakpoint at the indicated location.

If you cannot find the error by examining the values of other variables at this location, the program should be run again, stopped somewhere shortly before the indicated line, and single-stepped until the problem occurs.