Base Language - Bug #7759

improve CASE statement for integer type

08/29/2023 06:54 AM - Constantin Asofiei

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#1 - 08/29/2023 06:56 AM - Constantin Asofiei

From an email discussion:
Constantin Asofiei wrote:

> Quick question, I don't recall if this is how it is supposed to work or not.
>
> A "CASE i" like this:

```
def var i as int.
case i:
   when 1 then do: message "1". end.
   when 2 then do: message "2". end.
   when 3 then do: message "3". end.
end.
```

> gets converted always using string:

```
         switch (StringHelper.safeTrimTrailing((i).toStringMessage()))
         {
            case "1":
               {
                  message("1");
               }
               break;
            case "2":
               {
                  message("2");
               }
               break;
            case "3":
               {
                  message("3");
               }
               break;
         }
```

> Is this expected?

From Greg Shah:

> No, that isn't something I expect.

From Constantin Asofiei

This was added/forced back in trunk rev 10639, for #2414
The reason is to handle all data types plus the unknown value.

But considering at least 'integer' type, if all CASE expressions are integer literals, maybe we can save the switch expression result, check if is not unknown, and emit the Java integer switch. For all other cases (unknown literal usage, complex expressions, other data types), we can fallback to the string switch.

This may be an improvement for some situations.

From Greg Shah:

Go ahead and create a task for this. I don't know that we want to work it right now. I agree we could do something to bypass when the expression is unknown but it will be fragile. Any future hand editing of the code could easily break the logic since it would be a subtle thing.

From Ovidiu Maxiniuc:

The switch for integer types (also including logical and dates) might work faster if we replace "StringHelper.safeTrimTrailing((i).toStringMessage())" with something specialized for each case, but keeping the String as the compare type.

To switch to specialized data types and accommodate the unknown/null case, the converted switch must be expressly guarded like this (for an int64 4GL data type in case statement):

```
Long exp = <converted-case-expression>.toJavaLongType();
if (exp == null) {
    // investigate which is the case branch to be executed and duplicate it here
} else {
    switch (exp) {
        case l1: ...
        case l2: ...
        case l3: ...
        default: ...
    }
}
```

Important note: while thinking of this issue I realized that the current implementation is a bit broken: the safeTrimTrailing() does trim all the white characters, but we learnt some time ago that only the simple SPACE is actually trimmed, the TAB, CR and LF are not actually trimmed in string comparing operations. The safeTrimTrailing() is used in multiple places, but I think all of them work incorrectly from this POV. IMO, the method should trim only the SPACE characters (0x20) at the end in all those cases.

#2 - 08/29/2023 06:59 AM - Constantin Asofiei

What I had in mind (for integer type, as Java switch AFAIK works with 32 bit values, not 64, at least for Java 8), is to have something like this, when the CASE uses integer type expression and only constants at the WHEN clause (without unknown):

```
integer switchVal = ... evaluate the CASE expression ...
if (!switchVal.isUknown()
{
   switch (switchVal.intValue())
   {
      case 1:
      ...
   }
}
```

#3 - 08/29/2023 05:01 PM - Ovidiu Maxiniuc

Constantin,

I was not aware of the 64 bit limitation of Java switch statement. Probably we will need to use the current implementation for 64 bit integers. However, remember that the result of an expression with integer operand is an int64 value so this will be chosen in more cases. Think of

```
define variable i as integer init 2000000000.
case i*i:
   // cases to handle the value
end.
```

In this case, the expression cannot be stored on 32 bit and a trim will cause incorrect results.

I know that the latest versions add new, improved support in this area, maybe that would help up get a better conversion with the downside of imposing a higher minimum Java version for generated code.

Otherwise, I think our ideas are pretty much the same. The if branches are just reversed. In my suggestion, l1, l2.. are (generic) constants. Yet, I think it would be better to store the evaluated expression (switchVal) as a native Java wrapper.