Exemplo   cláusula reduction

https://msdn.microsoft.com/pt-br/library/88b1k8y5.aspx

Sintaxe

```
reduction(operation:var_list)
```

Reduction aplica-se para as seguintes diretivas:

- for
- paralelo
- seções

```
// omp_reduction.cpp
// compile with: /openmp
#include <stdio.h>
#include <omp.h>

#define NUM_THREADS 4
#define SUM_START   1
#define SUM_END     10
#define FUNC_RETS   {1, 1, 1, 1, 1}

int bRets[5] = FUNC_RETS;
int nSumCalc = ((SUM_START + SUM_END) * (SUM_END - SUM_START + 1)) /
2;

int func1( ) {return bRets[0];}
int func2( ) {return bRets[1];}
int func3( ) {return bRets[2];}
int func4( ) {return bRets[3];}
int func5( ) {return bRets[4];}

int main( )
{
    int nRet = 0,
        nCount = 0,
        nSum = 0,
        i,
        bSucceed = 1;

    omp_set_num_threads(NUM_THREADS);

    #pragma omp parallel reduction(+ : nCount)
    {
        nCount += 1;

        #pragma omp for reduction(+ : nSum)
        for (i = SUM_START ; i <= SUM_END ; ++i)
            nSum += i;

        #pragma omp sections reduction(&& : bSucceed)
        {
            #pragma omp section
            {
                bSucceed = bSucceed && func1( );
            }

            #pragma omp section
            {
                bSucceed = bSucceed && func2( );
            }

            #pragma omp section
            {
                bSucceed = bSucceed && func3( );
            }

            #pragma omp section
            {
                bSucceed = bSucceed && func4( );
            }

            #pragma omp section
            {
                bSucceed = bSucceed && func5( );
            }
        }
    }

    printf_s("The parallel section was executed %d times "
             "in parallel.\n", nCount);
    printf_s("The sum of the consecutive integers from "
             "%d to %d, is %d\n", 1, 10, nSum);

    if (bSucceed)
        printf_s("All of the the functions, func1 through "
                 "func5 succeeded!\n");
    else
        printf_s("One or more of the the functions, func1 "
                 "through func5 failed!\n");

    if (nCount != NUM_THREADS)
    {
        printf_s("ERROR: For %d threads, %d were counted!\n",
                 NUM_THREADS, nCount);
        nRet |= 0x1;
    }

    if (nSum != nSumCalc)
    {
        printf_s("ERROR: The sum of %d through %d should be %d, "
                "but %d was reported!\n",
                SUM_START, SUM_END, nSumCalc, nSum);
        nRet |= 0x10;
    }

    if (bSucceed != (bRets[0] && bRets[1] &&
                     bRets[2] && bRets[3] && bRets[4]))
    {
        printf_s("ERROR: The sum of %d through %d should be %d, "
                 "but %d was reported!\n",
                 SUM_START, SUM_END, nSumCalc, nSum);
        nRet |= 0x100;
    }
}
```

Output

```
The parallel section was executed 4 times in parallel.
The sum of the consecutive integers from 1 to 10, is 55
All of the the functions, func1 through func5 succeeded!
```