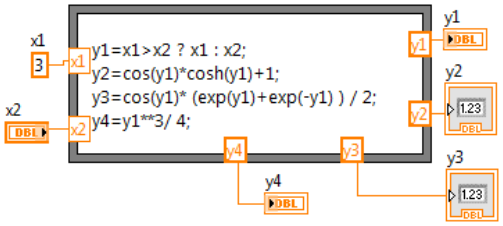
Laborator #6

Programare, an1, MTR+MEC, UTCluj, Prof.dr.ing. Iulian Lupea

1. Formula Node (Labview), utilizare cod limbajul C

1.1. Executaţi diagrama alăturată; observaţi operatorii din expresii (? : , **, /) şi funcţiile matematice apelate.

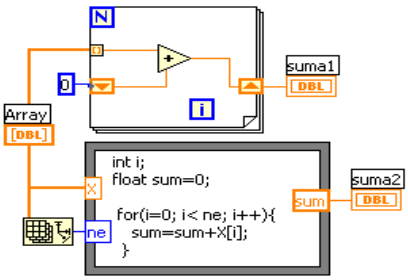


1.2. **Calculaţi suma valorilor numerice din Array1D/şir numeric**

Este folosit ciclul For (Labview) şi în paralel cod C scris în *Formula Node*

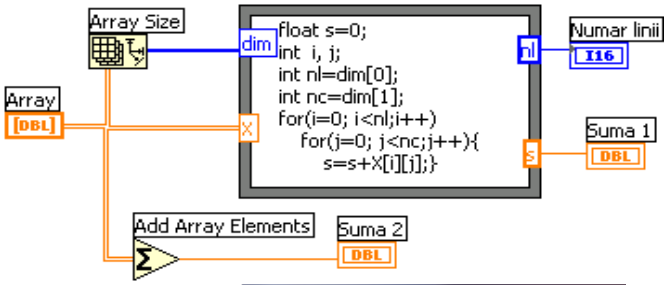
x şi ne sunt parametri de intrare în Formula node
sum este parametru de ieşire (calculat)

```
int i   // declaratie variabila i de tip întreg
float sum=0 //declaratie var sum de tip real şi iniţializare cu 0
```



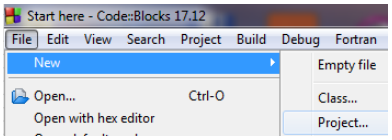
1.3. **Suma valori num. din Array2D/ matrice, cod C în Formula Node**

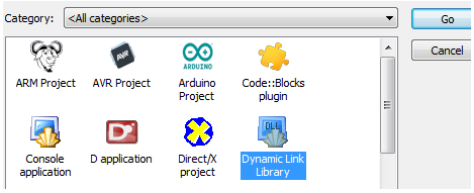
parametri de intrare: **dim** este tablou 1D-întregi,
X este tablou 2D-reale;
param. ieşire: **nl, s** sunt scalari.
Obs: nl=număr linii , nc=număr coloane



2. Apel C/C++ DLL din Labview

2.1. Creează proiect DLL din CodeBlocks IDE în C/C++
File/New/Project...; Din fereastră selectăm pictograma Dynamic Link Library;
Scriem titlul proiectului: SumaTablouDLL; selectăm Compiler: GNU GCC Compiler şi bifăm numai Create "Release"...





Fişierul cod sursă **main.cpp** după <u>adăugare cod pentru realizare sumă elemente pozitive mai mici decât 10 din tablou</u> va conţine:

```
#include "main.h"
// a sample exported function
float DLL_EXPORT  SumPoz(const LPCSTR sometext, float X[],  int ne )
{
   MessageBoxA(0, sometext, "DLL Message", MB_OK | MB_ICONINFORMATION);
      int i;  float sum=0;
      for (i=0; i<ne; i++) {
         if( X[i]>0 && X[i]<10)
               sum=sum+X[i];
      }
       return sum;
}
.....
```

Fişierul header **main.h** conţine <u>declaraţia funcţiei SumPoz( )</u>:

```
......
extern "C"
{  #endif
float DLL_EXPORT SumPoz(const LPCSTR sometext,  float X[],  int ne );
#ifdef __cplusplus} ...
```

2.2.Generare bibliotecă dinamică DLL de programe compilate în format executabil folosind Build Ctrl+F9 =>Output file is bin\Release\SumaTablouDLL.dll ; se corecteza erori dacă sunt.

2.3.Call Library Function Nod din Labview

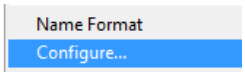
Funcţia Call Library Function Nod din *Connectivity/ Libraries& Executables\...* apelează cod extern din biblioteca DLL (Dynamic Link Library). Se poate expanda (asemănător cu Bundle) şi configura nodul 'Call Library Function Node' pentru a preciza biblioteca, funcţia, parametrii funcţiei, valoarea returnată.

Nodul conţine câte un terminal **pereche intrare-ieşire pentru fiecare parametru** al funcţiei **SumPoz**. Prin terminalul din stânga se trimite o valoare în funcţie iar prin cel din dreapta se citeşte val. param. după execuţia funcţiei.

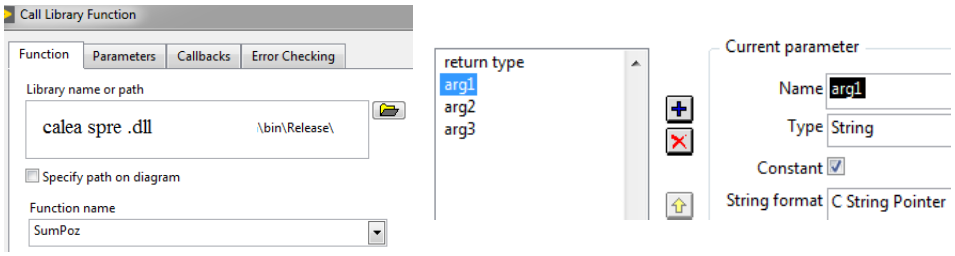
Return value este valoarea returnată a funcţiei; dacă tipul datei returnate de funcţie este **Void**, terminalul cel mai de sus-dreapta nu este folosit. Implicit 'calling convention' este C.

2.4. In funcţia din SumPoz.dll cei 3 parametri sunt şir de caractere/string, tablou/array numeric real (float) şi numeric întreg pe 32 biţi. Funcţia returnează suma valorilor din tablou cu valoare între 0 şi 10 (float 4 bytes). In general pentru tipul numeric se pot folosi sub-tipurile: întreg pe 8, 16, 32, 64-bit, cu semn sau fără semn şi tipurile real pe 4-byte (single-precision numbers) şi 8-byte (double-precision numbers).

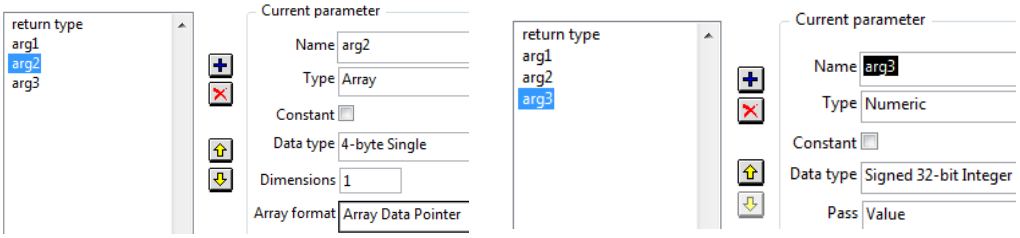
In figurile de mai jos este prezentat modul de configurare/ **Configure...** a parametrilor nodului 'Call Library Function Nod'.



In secţiunea Function se specifică calea ...bin\Release\ nume?.dll spre biblioteca .dll creată.



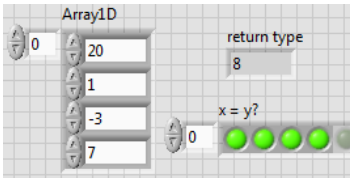
Cu butonul ⊞ se adaugă trei parametri: arg1, arg2, arg3 şi conform cu cele 3 figuri alăturate se specifică tipul fiecărui parametru.

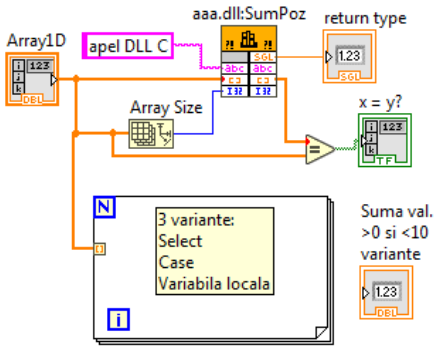


Funcţia SumPoz primeşte 3 param. actuali (cunoscuţi):

şirul 'apel DLL C', valori din tabloul de reale (control PF) şi nr. valori reale din tablou.

2.5.Tema. Creaţi 3 variante de cod Labview pentru problema precedentă (2.4.) pentru calcul sumă elemente cu valori pozitive şi mai mici decât 10. O variantă de program cu Select, a doua cu Case şi a treia folosiţi variabila locală.

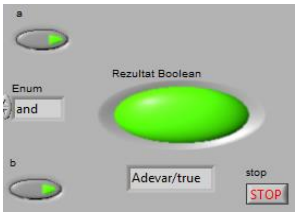


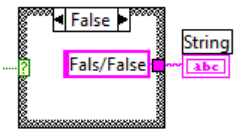


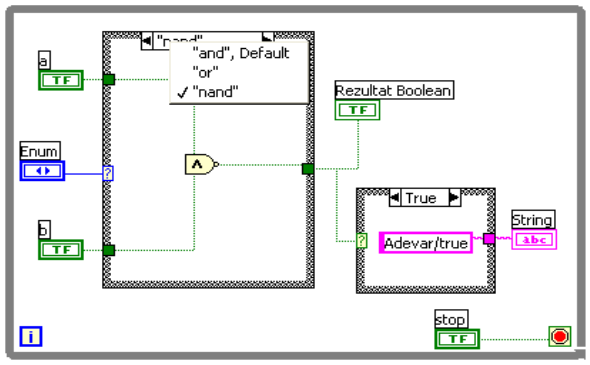
2.6. Tema Adăugaţi în funcţia C creată cu CodeBlocks (SumPozOrd) ordonarea tabloului real de intrare şi afişaţi în PF inclusiv şirul ordonat.

3. Modularizare program. Generare subVI ca funcţie apelabilă individual din diagrama altei aplicaţii (apel subVI)

3.1. Creaţi aplicaţa alăturată pentru evaluarea repetitivă a unor expresii logice cu operanzii logici (led) a şi b şi operatorii logici And, Or, Nand.



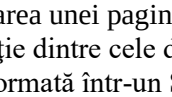


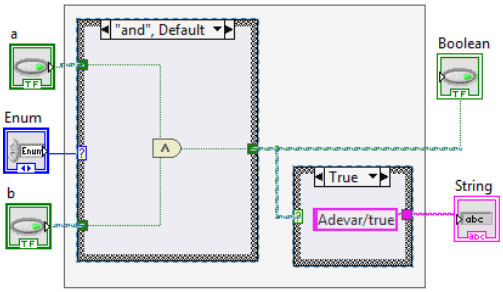


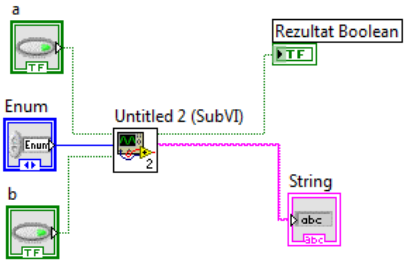
3.2. **Generare SubVI**.

a) o zona dreptunghiulara dintr-o diagrama (de preferinţă nu se includ în zona terminale control sau indicator) se selectează cu mouse,

b) cu comanda *Create SubVI din ...bara de meniu orizontală diagrama bloc / Edit / Create SubVI* se înlocuieste zona selectată cu o pictograma Untitled 2 (SubVI);

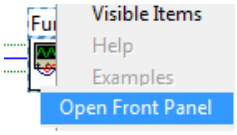






c) dublu click pe noul SubVI (sau Open Front Panel), se deschide şi observa PF şi diagrama si apoi se salveaza cu numele dorit pentru o viitoare utilizare ca program .VI independent. Se poate edita/personaliza pictograma asociată noului SubVI prin click dreapa pe colţul dreapta-sus a panoului frontal a aplicaţiei.

d) se poate include/apela noul SubVI intr-o aplicaţie mai generală cu comanda **Select a VI...** din diagrama aplicaţiei apelantă.



3.3. Se generează o aplicaţie care conţine un Tab Control cu 3 pagini după modelul din figura de mai jos. La selectarea unei pagini se selectează un caz al unei instrucţiuni CASE. In fiecare caz se va executa o altă aplicaţie dintre cele deja efectuate în laboratore precedente. Fiecare dintre cele trei aplicaţii este în prealabil transformată într-un SubVI cu controalele şi indicatoarele proprii prezente în pagina corespunzătoare a obiectului Tab Control.