Programarea aplicaţiilor SCADA

Cuprins

Obiective

- ⇧ Prezentarea structurilor repetitive din cadrul limbajului de programare Cicode
- ⇧ Prezentarea instrucţiunii repetitive FOR şi realizarea de aplicaţii cu aceasta instrucţiune
- ⇧ Prezentarea instrucţiunii repetitive WHILE şi realizarea de aplicaţii cu aceasta instrucţiune
- ⇧ Utilizarea facilitaţii de scanare a paginilor grafice pentru a implementa structuri repetitive
- ⇧ Utilizarea instrucţiunilor repetitive în realizarea diverselor aplicaţii SCADA

Organizarea sarcinilor de lucru

- ⇧ Parcurgeţi cele trei capitole ale cursului.
- ⇧ In cadrul fiecărui capitol urmăriţi exemplele ilustrative şi încercaţi sa le realizaţi în medul de dezvoltare "Citect".
- ⇧ Fixaţi principalele idei ale cursului, prezentate în rezumat.
- ⇧ Completaţi testul de autoevaluare.
- ⇧ Timpul de lucru pentru parcurgerea testului de autoevaluare este de 15 minute.

1. Instrucţiunea FOR

In multe cazuri e nevoie sa se repete execuţia uneia sau mai multor instrucţiuni. Exista o serie de instrucţiuni repetitive cum ar fi: instrucţiunea **for**, instrucţiunea **while**, etc.

Instrucţiunea **for** Se foloseşte pentru a executa repetitiv o instrucţiune sau o secventa de instrucţiuni. De obicei implementează structura ciclica cu număr cunoscut de paşi.

Formatul instrucţiunii:
Instrucţiunea **for** are următorul format:

FOR Variabila=expresie1 TO expresie2 DO
instrucţiune(instrucţiuni;)
END
Unde expresie1 este valoarea de start a variabilei iar expresie2 este valoarea de stop a variabilei.

Aplicaţii care utilizează instrucţiunea FOR

Să realizăm o aplicaţie în care să utilizăm instrucţiunea repetitivă **for** pentru a aprinde secvenţial 10 LED-uri.
Se va realiza o pagina grafica **"led4"** asemănătoare cu imaginea de mai jos.

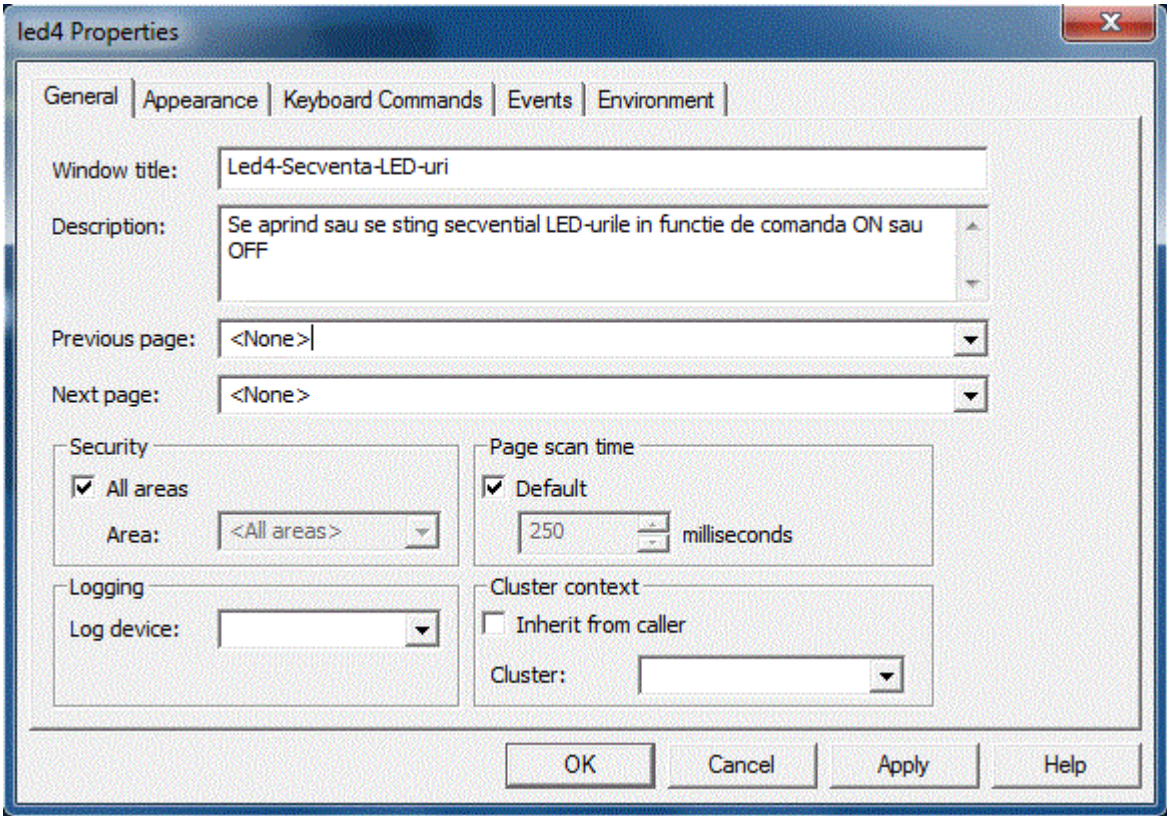
La apăsarea butonului "Start", trebuie sa se aprindă secvenţial cele 10 LED-uri, iar la comanda "Stop" se vor stinge secvenţial cele 10 LED-uri.
Vom scrie deci doua funcţii: **leduri_on()** şi **leduri_off()** care vor fi atribuite celor doua butoane prin intermediul facilitaţii "Button Properties".

```
FUNCTION leduri_on()
    INT i;
        FOR i=1 TO 10 DO
                ld[i]=1;
                Sleep(1);
        END
END

FUNCTION leduri_off()
    INT i;
        FOR i=1 TO 10 DO
                ld[i]=0;
                Sleep(1);
        END
END
```

După cum se observă, s-a folosit instrucţiunea **for** pentru a atribui tag-ului ld[i] valoarea 1 respectiv 0. După fiecare instrucţiune de atribuire, se invocă funcţia Sleep(1) care introduce o întârziere de cate o secunda.

Funcţia "Sleep" are ca parametru, numărul de secunde de întârziere. Pentru a obţine întârzieri mai mici de o secundă, vom aplica o altă metodă, şi anume vom scrie o funcţie care se apelează din pagina principală de fiecare data când se scanează pagina. Timpul după care se face o noua scanare poate fi stabilit pe pagina principală :



După cum se observă, timpul de scanare implicit este de 250 ms dar se poate stabili alt timp prin invalidarea opţiunii default.

După setarea corespunzătoare a intervalului de timp dintre două scanări, plasăm funcţia f(x) pe pagina principală **"led5"** .

Cu click dreapta pe f(x), completam Citect Object Properties->Command cu: **ecran_0( )**. Cu alte cuvinte de fiecare dată când se scanează pagina grafica, se lansează funcţia **ecran_0( )** cu următorul conţinut:

```
FUNCTION ecran_0()
  IF (ld_on=1) THEN
    ld_off=0;
       ld[i]=1;
       i=i+1;
       IF (i=11) THEN
          ld_on=0;
          i=0;
       END
  END
  IF (ld_off=1) THEN
    ld_on=0;
       ld[i]=0;
       i=i+1;
       IF (i=11) THEN
          ld_off=0;
          i=0;
       END
  END
END
```

Pe evenimentele click ale butoanelor "ON" respectiv "OFF" nu se mai lansează funcţiile **leduri_on()** respectiv **leduri_off()**. Sunt introduse însa doua variabile tag locale astfel: tag-urile corespunzătoare adică:

Tag-uri aferente				
Nume	**Tip**	**Domeniu**	**Um**	**Comentariu**
ld_on	DIGITAL	-	-	Comanda LED-uri on
ld_off	DIGITAL	-	-	Comanda LED-uri off

Funcţie de variabilele de sus, la fiecare scanare a paginii, LED-urile se vor aprinde secvenţial, respectiv se vor aprinde secvenţial. Deşi este o operaţie repetitivă, aceasta nu s-a mai implementat folosind instrucţiunea repetitiva **for**, repetitivitatea s-a realizat prin intermediul scanării implicite a paginii la intervalul stabilit anterior.

Sa presupunem acum ca dorim activarea LED-urilor într-o anumita ordine. Pentru acest lucru vom defini un vector de priorităţi de genul:

INT Prior[11]=0,1,5,7,4,2,9,3,6,8,10;

După cum se vede, s-au definit 11 elemente nu 10, pe motiv ca numerotarea LED-urilor începe de la 1 nu de la 0.

Conform modului de iniţializare al valorilor elementelor vectorului "Prior", ordinea de activare al LED-urilor este: 1,5,7,4,2,9,3,6,8,10

In următoarea aplicaţie **"led6"** , vom rescrie deci funcţiile: **leduri_on()** si **leduri_off()** ele devenind:

```
FUNCTION leduri_on()
    INT i;
        FOR i=1 TO 10 DO
        ld[Prior[i]]=1;
        Sleep(1);
        END
END
FUNCTION leduri_off()
    INT i;
        FOR i=1 TO 10 DO
        ld[Prior[11-i]]=0;
        Sleep(1);
        END
END
```

După cum se observă, nu se mai activează LED-ul i ci LED-ul Prior[i]. Dezactivarea se face în ordine inversă adică: nu se mai dezactivează LED-ul i ci LED-ul Prior[11-i].

In următoarea aplicaţie **"led7"** , se optează pentru folosirea intervalului de scanare al paginii principale, vom modifica funcţia: **ecran_0( )** astfel:

```
FUNCTION ecran_0()
  IF (ld_on=1) THEN
    ld_off=0;
       i=i+1;

       IF (i=11) THEN
          ld_on=0;
          i=0;
       END
       ld[Prior[i]]=1;
  END
  IF (ld_off=1) THEN
    ld_on=0;
       i=i+1;
       IF (i=11) THEN
          ld_off=0;
          i=0;
       END
       ld[Prior[11-i]]=0;
  END
END
```

2. Instrucţiunea WHILE

Instrucţiunea **while** se foloseşte pentru a executa repetitiv o instrucţiune sau o secventa de instrucţiuni atâta timp cat o expresie este adevărată.

Formatul instrucţiunii:

Instrucţiunea **while** are următorul format:

WHILE (expresie) DO
instrucţiune(instrucţiuni;)
END

Instrucţiunea se execută repetat atâta timp cât valoarea expresiei "expresie" este adevărata.

Testul are loc înaintea fiecărei execuţii a instrucţiunii. Modul de funcţionare al instrucţiunii este următorul:

- ⇧ Se testează expresia din paranteze. Dacă ea este adevărată (sau expresia din paranteze are o valoare diferita de zero)
- ⇧ Se execută corpul instrucţiunii while

- ⇧ Se reia testarea şi execuţia pană expresia devine falsă (sau valoarea expresiei din paranteze este zero)
- ⇧ Se continua execuţia cu instrucţiunea de după corpul instrucţiunii while, deci instrucţiunea while se termina.

Aplicaţii care utilizează instrucţiunea WHILE

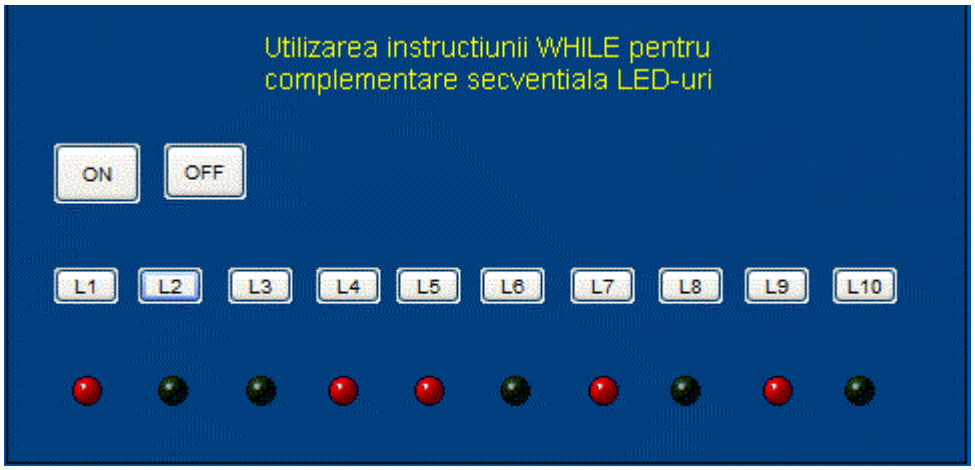
⇧

Pornind de la aplicaţia în care avem plasate cele 10 LED-uri, să realizăm o nouă aplicaţie în care pornind de la o anumită secventă de LED-uri utilizând butoanele L1-L10, la apăsarea butonului ON sa complementăm pe rând starea fiecărui LED pană la apăsarea butonului OFF.

Funcţiile: **leduri_on()** si **leduri_off()** lansate de butonul ON respectiv OFF fiind:

```
FUNCTION leduri_on()
    INT i=1;
    ld_off=0;
    ld_on=1;
       WHILE ld_on=1 DO
              Toggle(ld[i]);
              Sleep(1);
              i=i+1;
              IF i=11 THEN
                     i=1;
              END
       END
END
FUNCTION leduri_off()
    ld_off=1;
    ld_on=0;
END
```

Rulam aplicaţia şi obţinem **"led8"** :



Instrucţiunea FOR poate fi înlocuită cu o instrucţiune WHILE. O instrucţiune FOR de forma:

FOR Variabila=expresie1 TO expresie2 DO
instrucţiune(instrucţiuni;)
END

este echivalenta cu:

Variabila=expresie1;
WHILE (expresie2) DO
instrucţiune(instrucţiuni;)
expresie3;
END

Unde :
expresie1 constituie iniţializarea ciclului şi se execută o singura dată înaintea ciclului.
expresie2 specifică testul care controlează ciclul. El se execută înaintea fiecărei iteraţii. Dacă condiţia din test este adevărată atunci se execută corpul ciclului, după care se execută expresie3, care constă de cele mai multe ori în modificarea valorii variabilei de control al ciclului. Se revine apoi la reevaluarea condiţiei. Ciclul se termina când expresie2 devine falsă.

Spre exemplu, vom putea rescrie funcţiile: **leduri_on()** şi **leduri_off()** astfel:

```
FUNCTION leduri_on()
    INT i=1;
        WHILE i<=10 DO
                ld[Prior[i]]=1;
                Sleep(1);
                i=i+1;
        END
END
FUNCTION leduri_off()
    INT i=1;
        WHILE i<=10 DO
                ld[Prior[11-i]]=0;
```

```
                Sleep(1);
                i=i+1;
        END
END
```

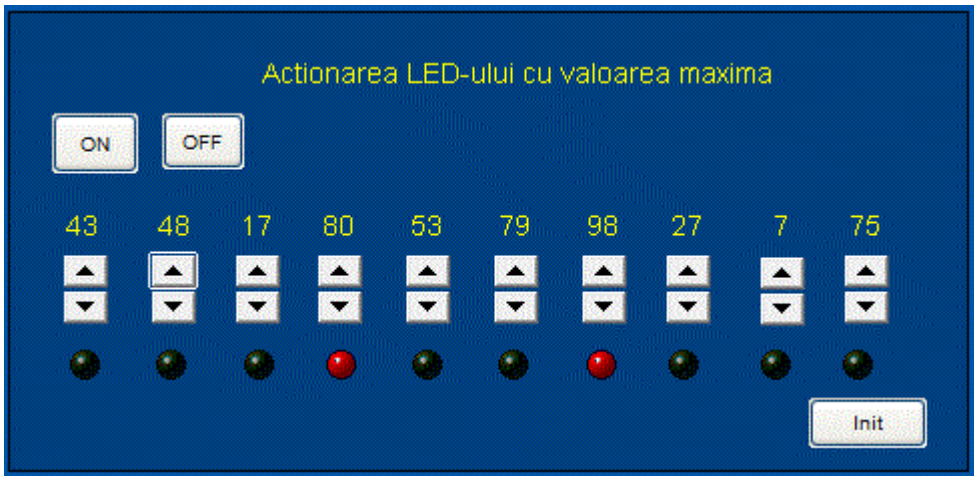
3. Diverse aplicaţii în care sunt utilizate instrucţiunile repetitive

Următoarele aplicaţii vor fi de o complexitate mai ridicată, înglobând diverse instrucţiuni descrise anterior.

Vom încerca sa implementam diverşi algoritmi. Pentru început vom realiza o aplicaţie în care avem parametri de valori diferite, şi în funcţie de valoarea acestora vom lua diferite decizii.

Determinarea valorilor maxime

Să presupunem că avem 10 valori diferite şi vrem sa semnalizam cu nişte LED-uri valoarea maxima. La fiecare apăsare a butonului, se va aprinde LED-ul care are valoarea corespunzătoare cea mai mare dintre valorile LED-urilor neaprinse încă. Valorile vor fi generate aleator, de asemenea vom plasa controale de tip Genie -> controls -> Ramp_UpDown_btn1 **"led9"** .



Avem nevoie de urmatoarele tag-uri:

Tag-uri aferente				
Nume	**Tip**	**Array Size**	**Um**	**Comentariu**
ld	DIGITAL	11	-	Ledurille ld[1]...ld[10]
val_l	DIGITAL	11	-	Valori prag pentru comanda led-urilor

La apăsarea butonului ON se lansează funcţia leduri9_on()

```
FUNCTION leduri9_on()
    INT i=1;
    INT j=1;
    INT mx=0;
        FOR i=1 TO 10 DO
               IF ((val_l[i]>mx) AND (ld[i]=0)) THEN
                      mx=val_l[i];
                      j=i;
               END
        END
        ld[j]=1;
END
```

La apăsarea butonului OFF se lansează funcţia leduri9_off(),

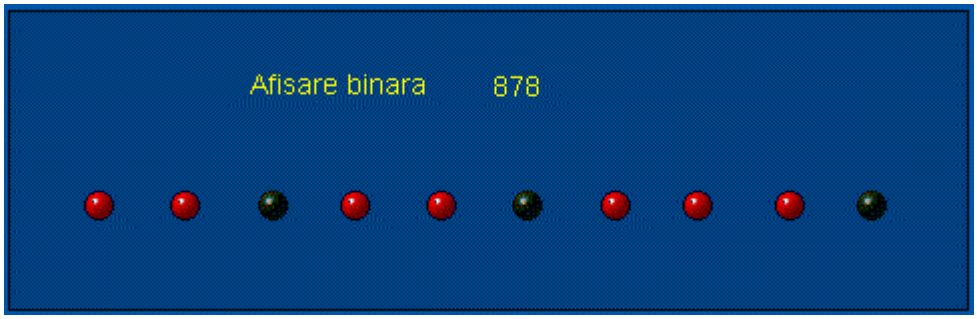
```
FUNCTION leduri9_off()
    INT i=1;
    INT j=1;
    INT mx=0;
        FOR i=1 TO 10 DO
               IF ((val_l[i]>mx) AND (ld[i]=1)) THEN
                      mx=val_l[i];
                      j=i;
                END
        END
        ld[j]=0;
END
```

La apăsarea butonului INIT se lansează funcţia init_9()

```
FUNCTION init_9()
        INT i=1;
        FOR i=1 TO 10 DO
               val_l[i]=Rand(100);
        END
END
```

Afişare binară

Pentru a afişa un număr întreg sub forma binară, se utilizează metoda împărţirilor repetate cu 2. Resturile împărţirilor reprezinta valorile biţilor în ordine inversă. Afişarea binară va fi făcută prin intermediul simbolurilor de tip LED. Vom crea deci pagina grafică **"led13"**



Numărul întreg i ce urmează a fi convertit este generat aleator utilizând funcţia **rand(1024)** . Maximul este fixat la 1024 deoarece avem 10 leduri deci numărul maxim este 2 la puterea 10=1024.

Avem nevoie de următoarele tag-uri:

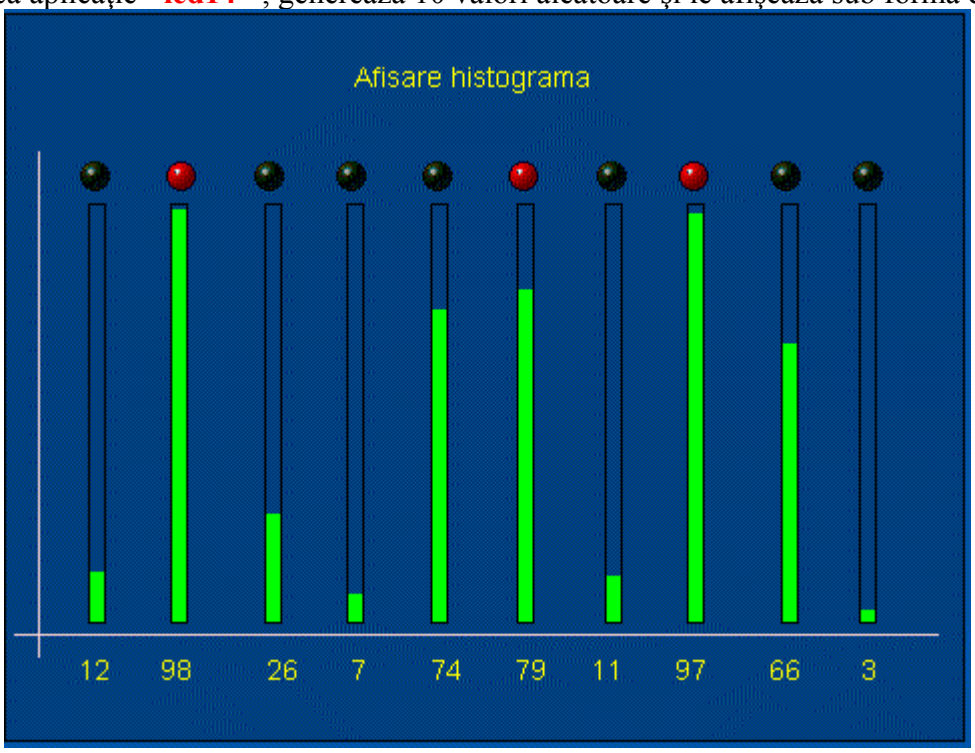
Tag-uri aferente				
Nume	**Tip**	**Array Size**	**Um**	**Comentariu**
ld	DIGITAL	11	-	Ledurille ld[1]...ld[10]
i	INTEGER	-	-	Contor

Funcţia care face conversia se intitulează **af_bin( ) şi** este plasata pe pagina grafică.

```
FUNCTION af_bin()
INT j
INT nr
       nr=Rand(1023);
       i=nr;
       FOR j=1 TO 10 DO
              ld[11-j]= nr MOD 2
              nr=nr/2;
       END
END
```

Afişare histogramă

Următoarea aplicaţie **"led14"** , generează 10 valori aleatoare şi le afişează sub forma de histograma.



Valorile generate sunt elementele unui vector, la fel şi valorile LED-urilor.

Avem nevoie deci de următoarele tag-uri:

Tag-uri aferente				
Nume	**Tip**	**Array Size**	**Um**	**Comentariu**
ld	DIGITAL	11	-	Ledurille ld[1]...ld[10]
val_l	DIGITAL	11	-	Valorile elem. histogramei

Funcţia care afişează histograma **histo( )** este plasată pe pagina grafică.

```
FUNCTION histo()
INT j=1;
i=i+1;
IF i>4 THEN
       i=0;
       FOR j=1 TO 10 DO
              val_l[j]=Rand(100);
```

```
                IF val_l[j]>77 THEN
                        ld[j]=1;
                ELSE
                        ld[j]=0;
                END
        END
END
END
```

Contorul i permite afişarea mai lentă, aceasta făcându-se din 5 în 5 scanări. LED-urile reţin poziţiile în care au fost depăşiri peste pragul de 77

Test de autoevaluare

⇧ -Marcaţi răspunsurile corecte la întrebările următoare.
⇧ -ATENTIE: pot exista unul, niciunul sau mai multe răspunsuri corecte la aceeaşi întrebare.
⇧ -Timp de lucru: 10 minute

1.-- Care este rolul instrucţiunii FOR ?

☐a. Execută repetitiv de un număr precizat de ori, o instrucţiune sau mai multe instrucţiuni
☐b. Execută repetitiv, o instrucţiune sau mai multe instrucţiuni
☐c. Execută repetitiv o instrucţiune sau mai multe instrucţiuni atâta timp cat este îndeplinită o anumita condiţie
☐d. Afişează repetitiv mai multe simboluri pe pagină grafică

2. *Instructiunea WHILE necesită:*

☐a. Cel puţin o expresie relaţională
☐b. O instrucţiune sau mai multe care urmează a fi repetate
☐c. Variabile de tip tag
☐d. Variabile de tip Array

3. *Pentru a seta elementele unui vector, avem nevoie de:*

☐a. 1 instrucţiune FOR
☐b. 1 instrucţiune WHILE
☐c. Mai multe instrucţiuni for
☐d. Mai multe instrucţiuni WHILE

4. *Pentru a gestiona mai multe LED-uri cea mai potrivita structura este:*

☐a. Symbol Set Multi-state
☐b. Tablou unidimensional
☐c. Symbol Set Array
☐d. Symbol light

5. *Conversia unui număr zecimal într-un număr binar presupune:*

☐a. Descompunerea numărului
☐b. Împărţiri repetate cu 2
☐c. Apelarea funcţiei af_binar()
☐d. Utilizarea unui vector de numere binare

Grila de evaluare: 1-a; **2-a, b 3-a, 4-b, 5-b, d**.

Rezumat

Limbajul de programare Cicode

Cicode este un limbaj de programare integrat în mediul de dezvoltare CitectSCADA destinat dezvoltării de aplicaţii SCADA, permiţând controlul software a elementelor utilizate pentru mimarea şi controlul proceselor.
Limbajul Cicode permite utilizarea structurilor repetitive cum ar fi:

⇧ Instrucţiuni FOR
⇧ Instrucţiuni WHILE
⇧ Facilitaţi de scanare care permit structuri repetitive

Instrucţiunea for

-Se foloseşte pentru a executa repetitiv o instrucţiune sau o secventa de instrucţiuni.
-Implementează structura ciclica cu număr cunoscut de paşi.
Formatul instrucţiunii:

```
FOR Variabila=expresie1 TO expresie2 DO
  instrucţiune(instrucţiuni;)
END
```

-Expresie1 este valoarea de start a variabilei iar expresie2 este valoarea de stop a variabilei.

Instrucţiunea while

-Se foloseşte pentru a executa repetitiv o instrucţiune sau o secventa de instrucţiuni atâta timp cat o expresie este adevărată.
Formatul instrucţiunii:

```
WHILE (expresie) DO
  instrucţiune(instrucţiuni;)
END
```

-Instrucţiunea se execută repetat atâta timp cat valoarea expresiei "expresie" este adevărată.
-Testul are loc înaintea fiecărei execuţii a instrucţiunii. Modul de funcţionare al instrucţiunii este următorul:

- Se testează expresia din paranteze. Dacă ea este adevărată (sau expresia din paranteze are o valoare diferita de zero)
- Se executa corpul instrucţiunii while
- Se reia testarea şi execuţia pana expresia devine falsă (sau valoarea expresiei din paranteze este zero)
- Se continua execuţia cu instrucţiunea de după corpul instrucţiunii while, deci instrucţiunea while se termină.

Rezultate aşteptate

După studierea acestui modul, ar trebui sa ştiţi:

- Să scrieţi aplicaţii utilizând limbajul Cicode
- Să utilizaţi instrucţiuni repetitive în cadrul aplicaţiilor SCADA
- Să realizaţi structuri repetitive bazate pe funcţia de scanare a paginii grafice
- Să realizaţi aplicaţii SCADA în care sa folosiţi elemente complexe de programare

Termeni esenţiali

Termen	Descriere
SCADA	Supervisory Control And Data Aquisition
Tag	Nume generic pentru elementele din procesul monitorizat codificate prin intermediul variabilelor
HMI	Human Machine Interface -Interfaţa dintre aplicaţie şi utilizator
Limbaj Cicode	Limbaj de programare inclus in mediul de dezvoltare Citect SCADA
Instrucţiuni repetitive	Instrucţiune care forţează repetarea unei instrucţiuni sau grup de instrucţiuni de un anumit număr de ori sau pana când o expresie relaţională este adevărată
Expresie relaţională	Expresie a cărui rezultat este o valoare logica
Instrucţiunea FOR	Instrucţiune care forţează repetarea unei instrucţiuni sau grup de instrucţiuni de un anumit număr de ori
Instrucţiunea WHILE	Instrucţiune care forţează repetarea unei instrucţiuni sau grup de instrucţiuni pana când o expresie relaţională este adevărata

Recomandări bibliografice

- ⇧ [1] Traian Turc, Elemente de programare C++ utile in ingineria electrica, Ed.Matrixrom, Bucuresti,2010
- ⇧ [2] Traian Turc, Programare avansata C++ pentru ingineria electrica, Ed.Matrixrom, Bucuresti,2010
- ⇧ [3] Traian Turc, Programarea in limbaje de asamblare, uz intern, Univ."Petru Maior" ,Tg.Mures,2009
- ⇧ [4] Traian Tur,Brevet de inventie nr:11863 "Sistem pentru automatizarea si monitorizarea proceselor industriale", OSIM, 2003
- ⇧ [5] Jeff Kent, C++ fara mistere,Ed.Rosetti Educational 2004 .
- ⇧ [6] Boldur Barbat - Informatica industriala - Programarea în timp real – Institutul Central pentru Conducere si informatica 1984
- ⇧ [7] Ioan Babuita – Conducerea automata a proceselor – Ed. Facla 1985
- ⇧ [8] Ghercioiu-National în struments - Orizonturi în instrumentatie 1995

Link-uri utile

- 1. http://www.free-scada.org/ - Free SCADA - 2009.
- 2. http://www.7t.dk/igss/default.asp - IGSS SCADA System - 2009
- 3. http://www.7t.dk/igss/default.asp?showid=374 - IGSS Online SCADA Training - 2009
- 4. http://www.7t.dk/free-scada-software/index.html- IGSS Free SCADA Software -2009
- 5. http://www.citect.com/ - CITECT SCADA -2009
- 6. http://www.citect.com/index.php?option=com_content&view=article&id=1457&Itemid=1314 - Download CITECT demo - 2009
- 7. http://www.indusoft.com/index.asp - INDUSOFT SCADA - 2009
- 8 http://www.gefanuc.com/products/2819 - Proficy HMI/SCADA - CIMPLICITY - 2009.
- 9. http://www.genlogic.com/ - Dynamic Graphics, Data Visualization, Human-Machine Interface (HMI) - 2010
- 10 http://www.genlogic.com/demos.html - On-Line Java and AJAX Demos - 2010
- 11 http://www.free-scada.org/ - - 2009
- 12 http://www.free-scada.org/ - - 2009

Test de evaluare

- -Marcati raspunsurile corecte la intrebarile urmatoare.
- -ATENTIE: pot exista unul, niciunul sau mai multe raspunsuri corecte la aceeasi intrebare.
- -Timp de lucru: 10 minute

1. *Care rolul instrucţiunii WHILE ?*

☐a. Afisează repetitiv mai multe simboluri pe pagina grafica atât timp cat este îndeplinită o anumita condiţie
☐b. Execută repetitiv o instrucţiune sau mai multe instrucţiuni atât timp cat este îndeplinită o anumita condiţie
☐c. Execută repetitiv de un număr precizat de ori, o instrucţiune sau mai multe instrucţiuni
☐d. Execută repetitiv o instrucţiune sau mai multe instrucţiuni

2. *O instrucţiune FOR necesită:*

☐a. Cel puţin o expresie relaţională
☐b. Cel puţin o instrucţiune care sa fie repetată
☐c. Cel puţin o variabila tag
☐d. Cel puţin o variabila Array

3. *Care este cea mai potrivita instrucţiune pentru a citi valorile unui vector*

☐a. Instrucţiunea WHILE
☐b. Instrucţiunea FOR
☐c. Instrucţiunea IF
☐d. Instrucţiunea CASE

4. *Sortarea mai multor valori presupune:*

☐a. Indexarea bazei de date
☐b. Căutarea maximului
☐c. Căutarea repetitiva a maximului
☐d. Afişarea în ordine crescătoare

5. *Pentru a afişa în pagină grafică elemente care îşi schimbă simbolul la anumite intervale, folosim:*

☐a. Funcţia Sleep()
☐b. Facilităţile de scanare ale paginii grafice
☐c. Bucle de întârziere
☐d. Timere

Grila de evaluare: 1-b; **2-b, 3-b, 4-, 5-a, b**.