oszcillátor), ezért a keletkezett rezgések nem hallhatók, (a megahertzes tartományba eső ultrahangok).

A rezgő kvarckristály által keltett ultrahangok intenzitása (a rezgések amplitúdója) függ a rezgő kristály méreteitől. A maximális intenzitás rezonancia esetén adódik, amikor a kristály saját rezgési frekvenciája megegyezik a váltakozó elektromos tér frekvenciájával.

Azt a jelenséget, melynek során egy kristály elektromos tér hatására, alakváltozást (mechanikai deformációt) szenved ***inverz piezoelektromos-hatásnak*** nevezzük. Az elnevezés nyilvánvalóan arra utal, hogy a jelenség a piezoelektromos-hatásnak a reverzibilis, fordított folyamata.

(folytatjuk)

Puskás Ferenc

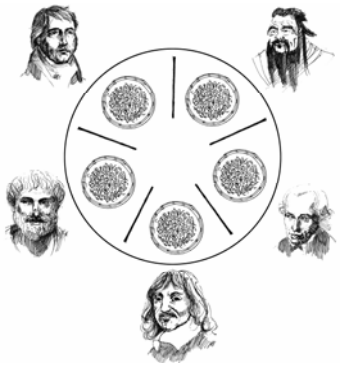
Érdekes informatika feladatok

XIX. rész

Az étkező filozófusok esete

A párhuzamos paradigma talán legnépszerűbb feladata az *étkező filozófusok (the dining philosophers)*, amelyet Edsger Wybe Dijkstra (1930–2002) javasolt 1971-ben a *holtpont* helyzetek és a párhuzamos szinkronizálás szemléltetésére.

Egy tibeti kolostorban öt filozófus él. Minden idejüket egy asztal körül töltik. Mindegyikük előtt egy tányér, amelyből soha nem fogy ki a rizs. A tányér mellett jobb és bal oldalon is egy-egy étkezőpálcika található.



A filozófusok gondolkodnak, majd amikor megéheznek felveszik a tányérjuk mellett lévő két pálcikát, esznek, majd visszateszik a pálcikákat és ismét gondolkodni kezdenek. Evés közben – mivel mindkét pálcika foglalt – a filozófus szomszédai nem ehetnek. Amikor egy filozófus befejezte az étkezést, leteszi a pálcikákat, így ezek elérhetővé válnak a szomszédai számára.

A nagy kérdés pedig az, hogy mit kell, hogy csináljanak a filozófusok, hogy ne veszszenek össze a pálcikákon. Ha mindegyikük felveszi például a jobb pálcikát és nem teszi le, mindegyik várakozni fog a szomszédjára és éhen halnak.

A *holtpont (deadlock)* akkor következhet be, amikor két (vagy több) folyamat egyidejűleg verseng erőforrásokért, és egymást kölcsönösen blokkolják. A két vagy több folyamat közül egyik sem tud továbblépni, mert mindkettőnek éppen arra az erőforrásra lenne szüksége, amit a másik lefoglalt.

A feladat szimulálásához elengedhetetlenül szükséges a szinkronizálás.

Valósítsuk meg az Étkező filozófusok szimulálását Visual C++-ban folyamatszálakat és kritikus szakaszokat használva!

Megoldás

Visual C++-ban több lehetőségünk van folyamatszálak létrehozására. Most a *Visual C++* 6.0 verzióját ismertetjük, és legegyszerűbb, ha MFC szálakat használunk.

Hozzunk létre egy egyszerű szöveges (Win 32 Console Application) projektet, majd végezzük el a szükséges beállításokat. A **Project | Settings...** dialógusablakban a **C/C++** fület választva a **Category** listán válasszuk a **Code Generation**-t. A **Use RunTime Library** beállítást válasszuk *MultiThread*-re. A **Debug** konfigurációnál a *Debug MultiThread*-et válasszuk. A *ClassWizard*-al létrehozott alkalmazások esetében helyesek a beállítások.

Folyamatszálat az *AfxBeginThread(SZÁLELJÁRÁS, NULL);* függvénnyel indíthatunk.

A *SZÁLELJÁRÁS* függvény fut párhuzamosan a többi szállal, ennek a fejléce: *UINT SZÁLELJÁRÁS(LPVOID lVoid)* – ezt a függvényt tehát meg kell hogy írjuk minden egyes szálhoz (öt filozófus van, öt szálunk, öt szálfüggvényünk lesz).

A szinkronizáláshoz kritikus szakaszokat használunk.

Kritikus szakasz típusú változó deklarálása:
CRITICAL_SECTION KritikusSzakaszVáltozóNeve;

Használata:
EnterCriticalSection(&KritikusSzakaszVáltozóNeve);

Kritikus szakasz törzse
LeaveCriticalSection(&KritikusSzakaszVáltozóNeve);

A program

A *stdafx.h* rendszer-headerállomány:

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//
#if
!defined(AFX_STDAFX_H__89F703E4_702D_4884_A657_0D4C815AC6B4__INCLUD
ED_)
#define
AFX_STDAFX_H__89F703E4_702D_4884_A657_0D4C815AC6B4__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

// A folyamatszalakhoz szukseges beallitasok:

#define VC_EXTRALEAN     // Exclude rarely-used stuff from Windows
headers

#include <afxwin.h>     // MFC core and standard components
#include <afxext.h>     // MFC extensions
#include <afxdisp.h>    // MFC Automation classes
#include  <afxdtctl.h>     // MFC support for Internet Explorer 4
Common Controls
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>     // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT
//{{AFX_INSERT_LOCATION}}
```

```
#endif                                                          //
!defined(AFX_STDAFX_H__89F703E4_702D_4884_A657_0D4C815AC6B4__INCLUD
ED_)
```

A program állománya:

```
#include "stdafx.h"
#include <conio.h>

// globalis valtozok

int palcika[6] = {0, 0, 0, 0, 0, 0};
CRITICAL_SECTION cs;
HANDLE hndEvents[6];

// A folyamatszalak eljarasai

UINT Filozofus1(LPVOID lVoid)
{
  while(1)
  {
    int nVarniKell = 0;
    //Belepek a kritikus szakaszba
    EnterCriticalSection(&cs);
    //Vannak palcikaim
    if(!palcika[1] && !palcika[5])
    {
      palcika[1]=palcika[5]=1;
      //Ehetek
      nVarniKell=0;
    }
    else //Nem tudok enni
      nVarniKell=1;
    //Kilepek a kritikus szakaszbol
    LeaveCriticalSection(&cs);
    if(nVarniKell)
      WaitForSingleObject(hndEvents[1], INFINITE);
    printf("Filozofus1 ESZIK!\n");
    HANDLE hndTmp = NULL;
    EnterCriticalSection(&cs);
    //Felszabaditom a palcikakat
    palcika[1]=palcika[5]=0;
    if(!palcika[1] && !palcika[2])
    {
      palcika[1]=palcika[2]=1;
      hndTmp = hndEvents[2];
    }
    else
    if(!palcika[4] && !palcika[5])
    {
      palcika[4]=palcika[5]=1;
      hndTmp = hndEvents[5];
    }
    LeaveCriticalSection(&cs);
    if(hndTmp != NULL)
      SetEvent(hndTmp);
    else
      printf("hiba 1\n");
    Sleep(10);
  }
    return 1;
}
```

```
UINT Filozofus2(LPVOID lVoid)
{
  while(1)
  {
    int nVarniKell = 0;
    EnterCriticalSection(&cs);
    if(!palcika[1] && !palcika[2])
    {
      palcika[1]=palcika[2]=1;
      nVarniKell=0;
    }
    else
      nVarniKell=1;
    LeaveCriticalSection(&cs);
    if(nVarniKell)
      WaitForSingleObject(hndEvents[2], INFINITE);
    printf("Filozofus2 ESZIK!\n");
    HANDLE hndTmp = NULL;
    EnterCriticalSection(&cs);
    palcika[1]=palcika[2]=0;
    if(!palcika[2] && !palcika[3])
    {
      palcika[2]=palcika[3]=1;
      hndTmp = hndEvents[3];
    }
    else
    if(!palcika[1] && !palcika[5])
    {
      palcika[1]=palcika[5]=1;
      hndTmp = hndEvents[1];
    }
    LeaveCriticalSection(&cs);
    if(hndTmp != NULL)
      SetEvent(hndTmp);
    else
      printf("hiba 2\n");
    Sleep(10);
  }
    return 1;
}

UINT Filozofus3(LPVOID lVoid)
{
  while(1)
  {
    int nVarniKell = 0;
    EnterCriticalSection(&cs);
    if(!palcika[2] && !palcika[3])
    {
      palcika[2]=palcika[3]=1;
      nVarniKell=0;
    }
    else
      nVarniKell=1;
    LeaveCriticalSection(&cs);
    if(nVarniKell)
      WaitForSingleObject(hndEvents[3], INFINITE);
    printf("Filozofus3 ESZIK!\n");
    HANDLE hndTmp = NULL;
    EnterCriticalSection(&cs);
    palcika[2]=palcika[3]=0;
    if(!palcika[3] && !palcika[4])
```

```
    {
      palcika[3]=palcika[4]=1;
      hndTmp = hndEvents[4];
    }
    else
    if(!palcika[1] && !palcika[2])
    {
      palcika[1]=palcika[2]=1;
      hndTmp = hndEvents[2];
    }
    LeaveCriticalSection(&cs);
    if(hndTmp != NULL)
      SetEvent(hndTmp);
    else
      printf("hiba 3\n");
    Sleep(10);
  }
    return 1;
}

UINT Filozofus4(LPVOID lVoid)
{
  while(1)
  {
    int nVarniKell = 0;
    EnterCriticalSection(&cs);
    if(!palcika[3] && !palcika[4])
    {
      palcika[3]=palcika[4]=1;
      nVarniKell=0;
    }
    else
      nVarniKell=1;
    LeaveCriticalSection(&cs);
    if(nVarniKell)
      WaitForSingleObject(hndEvents[4], INFINITE);
    printf("Filozofus4 ESZIK!\n");
    HANDLE hndTmp = NULL;
    EnterCriticalSection(&cs);
    palcika[3]=palcika[4]=0;
    if(!palcika[4] && !palcika[5])
    {
      palcika[4]=palcika[5]=1;
      hndTmp = hndEvents[5];
    }
    else
    if(!palcika[2] && !palcika[3])
    {
      palcika[2]=palcika[3]=1;
      hndTmp = hndEvents[3];
    }
    LeaveCriticalSection(&cs);
    if(hndTmp != NULL)
      SetEvent(hndTmp);
    else
      printf("hiba 4\n");
    Sleep(10);
  }
    return 1;
}

UINT Filozofus5(LPVOID lVoid)
```

```
{
  while(1)
  {
    int nVarniKell = 0;
    EnterCriticalSection(&cs);
     if(!palcika[4] && !palcika[5])
    {
      palcika[4]=palcika[5]=1;
      nVarniKell=0;
    }
    else
      nVarniKell=1;
    LeaveCriticalSection(&cs);
    if(nVarniKell)
      WaitForSingleObject(hndEvents[5], INFINITE);
    printf("Filozofus5 ESZIK!\n");
    HANDLE hndTmp = NULL;
    EnterCriticalSection(&cs);
    palcika[4]=palcika[5]=0;
    if(!palcika[1] && !palcika[5])
    {
      palcika[1]=palcika[5]=1;
      hndTmp = hndEvents[1];
    }
    else
    if(!palcika[3] && !palcika[4])
    {
      palcika[3]=palcika[4]=1;
      hndTmp = hndEvents[4];
    }
    LeaveCriticalSection(&cs);
    if(hndTmp != NULL)
      SetEvent(hndTmp);
    else
      printf("hiba 5\n");
    Sleep(10);
  }
    return 1;
}

// foprogram
int main(int argc, char* argv[])
{
    //Inicializalom a kritikus szakaszt es az esemenyeket
    InitializeCriticalSection(&cs);
    for(int i=1;i<=5;i++)
    {
      hndEvents[i] = CreateEvent(NULL, 0, 0, NULL);
    }
    //Inditom a folyamatszalakat
    AfxBeginThread(Filozofus1, NULL);
    AfxBeginThread(Filozofus2, NULL);
    AfxBeginThread(Filozofus3, NULL);
    AfxBeginThread(Filozofus4, NULL);
    AfxBeginThread(Filozofus5, NULL);
    //Billentyuleutesre varakozom
    while(!kbhit());
    return 0;
}
```