

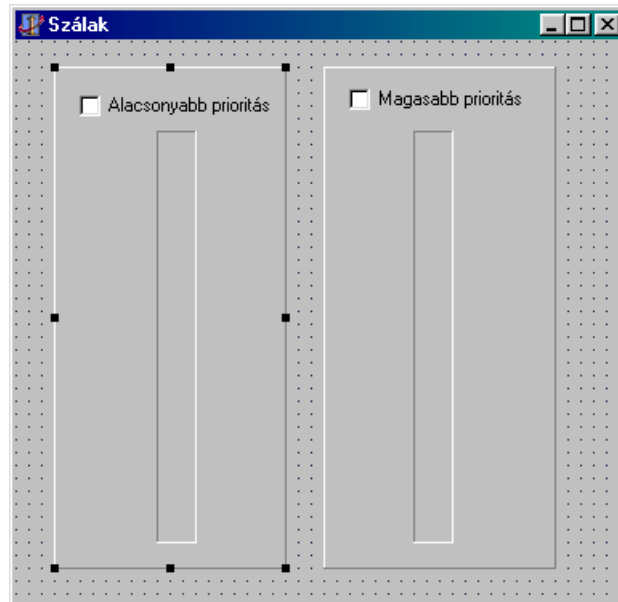
8.4 Többszálú alkalmazások készítése

1. Egyszerű többszálú alkalmazás [Szalak](#)
2. Prímszámok előállítása programszámban [Primszalp](#)
3. Grafika készítése programszálakban [SzalGrafika](#)



Készítsünk programot, amely két azonos számlálást végző programszálat tartalmaz! A szálak legyenek indíthatók és leállíthatók, illetve különböző prioritásúak! (*Szálak*)

Helyezzünk egy formra két **Panelt**, a két panelre egy-egy **ProgressBar** vezérlőt a szálak folyamatainak kijelzésére, és két jelölőnégyzetet a szál indítására és felfüggesztésére.



Készítsünk a programban egy új szálát a „*New Thread Object*” választással. Legyen az új szál *TSzal*, az elkészített modul pedig nevezzük át *Szalu*-ra. Duplázzuk meg a modulban a szálobjektumokat, legyen a második neve *TSzali*. Mindkét osztálydefiníció az **Execute** metódus mellett tartalmazzon egy *Friss* metódust az állapotkijelzés frissítésére! Mindkét **Execute** metódus számlálást végez az *i1* és az *i2* változóknban.

```
uses
  Classes;

var
  i1:integer=0; // Az első számláló
  i2:integer=0; // Az második számláló
```

A szálakat azonos módon deklaráljuk:

```
type // Az egyik szál objektum
  TSzal = class(TThread)
    procedure Friss; //Kijelzés a ProgressBar1-n
  private
    { Private declarations }
  protected
    procedure Execute; override; // A szál kódja
  end;
```

Az implementation részben gondoskodunk arról, hogy elérjük a formot.

```
implementation
  // Hogy elérjük a formot
  uses unit1;
```

A szálak hasonló metódusokkal rendelkeznek, így csak az egyik szálét mutatjuk be:

```
// TSzal
procedure TSzal.Friss;
begin
    Form1.ProgressBar1.Position:=il div 10;
end;

procedure TSzal.Execute;
begin
    while not terminated do
        begin
            inc(il);    // Számlálás
            Synchronize(Friss);    // A kijelzés szinkron frissítése
            if il=1000 then il:=0;
        end;
    { Place thread code here }
end;
```

A főablak moduljába beépítjük a *szalu* modult és deklaráljuk a *szal* és *szali* szálakat

```
uses szalu;
var    szal:TSzal;
        szali:TSzali;
```

A form létrehozásakor a szálak különböző prioritással jönnek létre.

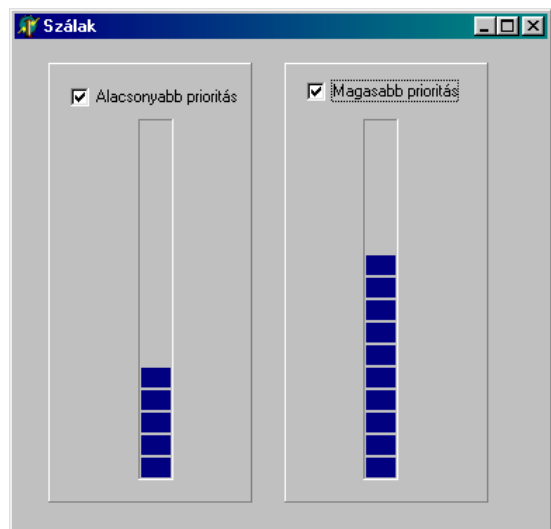
```
procedure TForm1.FormCreate(Sender: TObject);
begin
    // Szálak létrehozása és a prioritások beállítása
    Szal:=TSzal.Create(true);
    Szal.Priority:=tpLower;
    Szali:=TSzali.Create(true);
    Szali.Priority:=tpHigher;
end;
```

Lebontáskor megszüntetjük a szálakat is.

```
procedure TForm1.FormDestroy(Sender: TObject);
begin
    // Szálak lebontása
    Szal.Terminate;
    Szal.Free;
    Szali.Terminate;
    Szali.Free;
end;
```

A jelölőnégyzetek állapotától függ, hogy fut-e a szál.

```
procedure TForm1.CheckBox1Click(Sender:
                                TObject);
begin
    // A jelölőnégyzet
    // felfüggeszti/indítja a szálát
    if CheckBox1.Checked then
        Szal.Resume
    else
        Szal.Suspend;
end;
```





Készítsünk alkalmazást, amely prímszámokat állít elő önálló szinkron és aszinkron számban, és a generált számokat listába tölti! (*Primszalp*)

Először tervezzük meg az alkalmazás felületét! Legyen beállítható egy alsó határ és egy felső határ, amelyek között a prímszámokat keressük (*edAlsoHatar* és *edFelsőHatar*)! A *Szinkronizáció* című (*Radiogroup1*) választógomb-csoportban szabályozzuk, hogy szinkron, vagy aszinkron módon fusson a szám.

A *btnPrimClick* indítja a számgenerálást. A számok az *lstPrimek* listába kerülnek, melynek tartalma a *btmListaTorles* gombbal törölhető.

Az tesztelő program főablakának modulja:

```
unit primForm;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  PrimSzal, StdCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    btnPrim: TButton;
    lstPrimek: TListBox;
    edFelsőHatar: TEdit;
    Label1: TLabel;
    RadioGroup1: TRadioGroup;
    Label2: TLabel;
    edAlsoHatar: TEdit;
    btmListaTorles: TButton;
    procedure btnPrimClick(Sender: TObject);
    procedure btmListaTorlesClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  protected
    SajatSzal : TPrimSzal;
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  StartPosX,
  StartPosY: Integer;

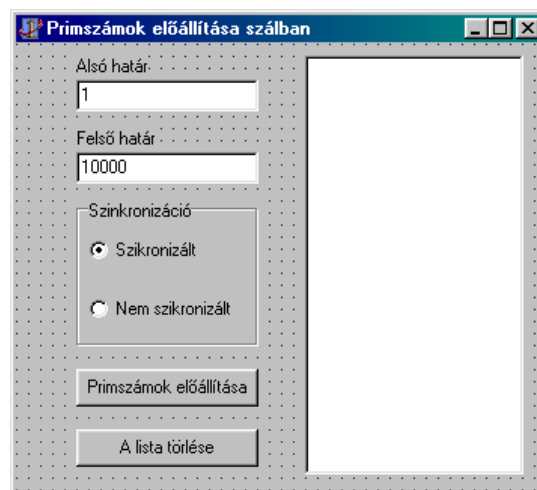
implementation

{$R *.DFM}

procedure TForm1.btnPrimClick(Sender: TObject);
begin
  SajatSzal := TPrimSzal.Create(False);
end;

procedure TForm1.btmListaTorlesClick(Sender: TObject);
begin
  lstPrimek.Clear;
end;

end.
```



Miután az alkalmazás felületét megterveztük, és mielőtt elindítanánk a programot készítsük el a szálát (*File/New/ThreadObject*), amelyre a *SajatSzal* hivatkozik! Legyen a szál neve *TPimSzal*! A prímszám-generálás választógomb-csoporttól függően, szinkron vagy aszinkron, azaz a *Shynchronize* metódussal, vagy a nélkül indítjuk a szálát. A *Primszamok* metódus a megadott határok között a lokális *Prim* függvény hívásával generálja a számokat és beírja a listába azokat. Amíg a szál fut, az alkalmazás ablakának címsorában a „A feltöltés folyamatban ...” szöveget látjuk, amikor végez a prímszámgeneráló metódus az „Az alkalmazás főablaka” szöveget írja a form címsorába.

```
unit PrimSzal;

interface
uses Classes, StdCtrls;

type
  TPrimSzal = class(TThread)
  private
    { Private declarations }
  protected
    procedure Execute; override;
    procedure Primszamok;
    procedure Kilepeskor(Sender: TObject);
  end;

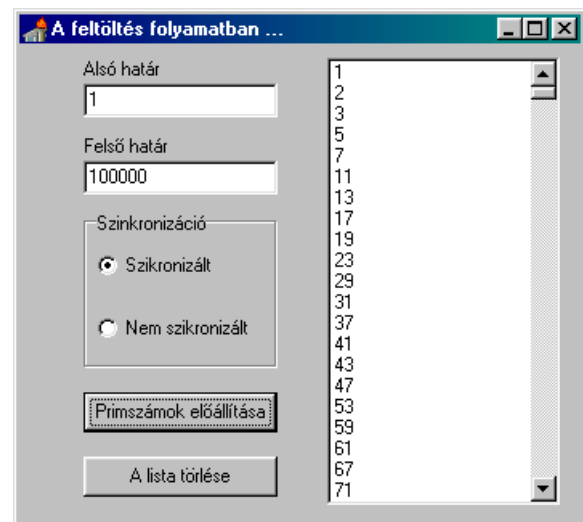
implementation
uses PrimForm, SysUtils;

procedure TPrimSzal.Execute;
begin
  Self.OnTerminate:=Kilepeskor;
  case form1.RadioGroup1.ItemIndex of
    0 : Synchronize(Primszamok);
    1 : Primszamok;
  end;
end;

procedure TPrimSzal.Primszamok;

  function Prim(p: Int64): boolean;
  var m: int64; i : longint;
  begin
    m:=round(sqrt(1.0*p));
    result:=true;
    for i:=2 to m do
      if p mod i = 0 then
        begin
          result:=false; break;
        end;
    end;
  end;

var
  khatar, vhatar : p: int64;
begin
  Form1.Caption := 'A feltöltés folyamatban ...';
  try
    khatar:=strtoint64(Form1.edAlsoHatar.text);
    vhatar:=strtoint64(Form1.edFalsoHatar.text);
  except
    khatar:=1;
    vhatar:=100;
  end;
  p:=khatar;
  while p<=vhatar do
    begin
      if Prim(p) then
        Form1.lstPrimek.Items.Add(Format('%d', [p]));
        inc(p);
      end;
    end;
end;
```



```
procedure TPrimSzal.Kilepeskor(Sender: TObject);  
begin  
    Beep;  
    Form1.Caption := 'Az alkalmazás főablaka';  
end;  
  
end.
```



Készítsünk többszálú alkalmazást, ahol szálak téglalapot jelenítenek meg véletlen pozícióban és színnel! A szálobjektumok tárolásához használjunk objektumlistát! (*SzalGrafika*)

A rajzoláshoz önálló szálát készítünk, amelyet azonban most nem külön modulban, hanem a form moduljában helyezünk el. A szál létrehozásakor paraméterként megkapja a form azonosítóját (*AForm*), amelyen dolgoznia kell, és a színt (*AColor*), amellyel rajzolni fog. A szál, futás közben ciklusban véletlen módon előállított téglalapokat rajzol a formra, a megadott színű vonallal. Ahhoz, hogy a közösen használt formon a különböző szálak ne keverjék össze az adatokat, a rajzolás megkezdésekor lezárjuk a formot a többi szál elől (*Lock*), majd a végén felszabadítjuk (*UnLock*) azt. A szál deklarációja:

```
TRajzoloSzal = class(TThread)
private
    FColor: TColor;
    FForm: TForm;
public
    constructor Create(AForm: TForm; AColor: TColor);
    procedure Execute; override;
end;
```

A metódusok megvalósítása:

```
constructor TRajzoloSzal.Create(AForm: TForm; AColor: TColor);
begin
    FColor := AColor;
    FForm := AForm;
    inherited Create(False);
end;

procedure TRajzoloSzal.Execute;

    function VeletlenTeglalap:Trect;
    var
        MaxX, MaxY: Integer;
    begin
        MaxX := FForm.ClientWidth;
        MaxY := FForm.ClientHeight;
        result.left := Random(MaxX-100);
        result.top := Random(MaxY-100);
        result.right := result.left+100;
        result.bottom := result.top+100;
    end;

var
    R : TRect;
begin
    // Megállás után megszűnik a szál
    FreeOnTerminate := True;
    // A szál legfeljebb az alkalmazás megállásáig fut
    while not (Terminated or Application.Terminated) do
    begin
        R:=VeletlenTeglalap;
        with FForm.Canvas do
        begin
            Lock; // a canvas lezárása
            // a következő utasításokat egy időben csak egy szál hajthatja végre
            Pen.Color := FColor;
            Brush.Style:=bsClear;
            with r do Rectangle(left, top, right, bottom);
            Unlock; // a canvas megnyitása
        end;
    end;
end;
```

Maga az alkalmazás menüvezérelten működik. A szálak szinkronizálásához szükséges a *SyncObj* modul. A szálak adatait a *SzalLista* objektumlistában tároljuk.

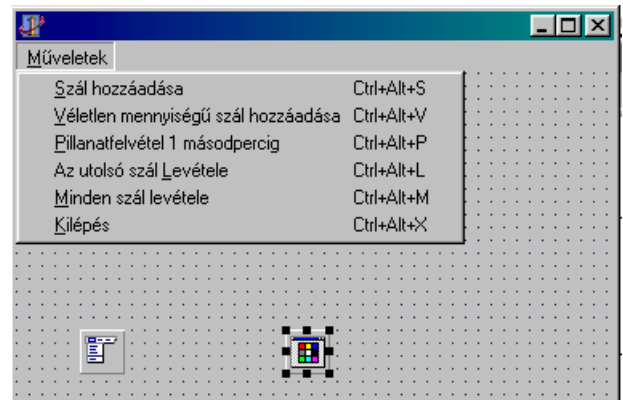
```
unit SzalGrafU;

interface
uses    Windows, Messages, SysUtils, Classes,
        Graphics, Controls, Forms, Dialogs,
        Menus, Contnrs, Math, syncobjs;

type
    TSzalakForm = class(TForm)
        MainMenu: TMainMenu;
        Muveletek: TMenuItem;
        SzalHozzaadasa: TMenuItem;
        SzalLevetele: TMenuItem;
        ColorDialog1: TColorDialog;
        VeletlenSzal: TMenuItem;
        MindenLevetele: TMenuItem;
        Kilepes: TMenuItem;
        PillanatFelvetel: TMenuItem;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure SzalHozzaadasaClick(Sender: TObject);
    procedure SzalLeveteleClick(Sender: TObject);
    procedure VeletlenSzalClick(Sender: TObject);
    procedure MindenLeveteleClick(Sender: TObject);
    procedure KilepesClick(Sender: TObject);
    procedure Kijelzes(db: Integer);
    procedure PillanatFelvetelClick(Sender: TObject);
    private
        SzalLista: TObjectList;
    public
        { Public declarations }
    end;

var
    SzalakForm: TSzalakForm;

resourcestring
    Fejlec='Többszálú grafikus alkalmazás';
```



A funkciók megvalósítása az **implementation** részben történik. A form létrehozásakor létrehozuk a szállistát is, amit a lebontáskor szintén meg kell szüntetni.

```
procedure TSzalakForm.FormCreate(Sender: TObject);
begin
    // Objektumlista a szálobjektumok tárolására
    SzalLista := TObjectList.Create;
    // A lista nem törli a levett szálakat
    SzalLista.OwnsObjects:=false;
    Kijelzes(0);
end;

procedure TSzalakForm.FormDestroy(Sender: TObject);
begin
    // Minden szál levétele a listából
    MindenLeveteleClick(nil);
    // Az objektumlista lebontása
    SzalLista.Free;
end;
```


A form fejlécébe a *Kijelzes* metódussal írunk.

```
procedure TSzalakForm.Kijelzes(db:Integer);
begin
    caption:=fejlec+format(' - a szálak száma: %03d.',[db]);
end;
```

A menüpontok metódusai (a menüpontok elnevezése utal az elvégzendő feladatra):

```
procedure TSzalakForm.SzalHozzaadasaClick(Sender: TObject);
begin
    // Új szál létrehozása a bekért szín átadásával és hozzáadása a listához
    if ColorDialog1.Execute then
    begin
        with Szallista do
        begin
            Add(TRajzoloSzal.Create(Self, ColorDialog1.Color));
            TRajzoloSzal(Szallista[Szallista.count-1]).Priority := tpLowest;
            Kijelzes(Szallista.count);
        end;
    end;
end;
```

```
procedure TSzalakForm.SzalLeveteleClick(Sender: TObject);
begin
    if Szallista.Count>0 then
    begin
        // az utolsó szál kivétele a listából és a szál törlése
        TRajzoloSzal(Szallista[Szallista.Count - 1]).Terminate;
        Szallista.Delete(Szallista.Count - 1);
        Kijelzes(Szallista.Count);
        if Szallista.Count=0 then Invalidate;
    end;
end;
```

```
procedure TSzalakForm.VeletlenSzalClick(Sender: TObject);
var
    i: Integer;
begin
    // Véletlen számú szál felvétele a listába
    // véletlen színbeállításokkal
    for i := 1 to 2+random(10) do
        Szallista.Add(TRajzoloSzal.Create(Self,
            rgb(Random(256),Random(256),Random(256))));
    Kijelzes(Szallista.count);
end;
```

```
procedure TSzalakForm.MindenLeveteleClick(Sender: TObject);
var
    i: Integer;
begin
    Cursor := crHourGlass;
    try
        for i := Szallista.Count - 1 downto 0 do
        begin
            TRajzoloSzal(Szallista[i]).Terminate;
            TRajzoloSzal(Szallista[i]).WaitFor;
        end;
        Szallista.Clear;
        Kijelzes(Szallista.count);
        Repaint;
    finally
        Cursor:= crDefault;
    end;
end;
```

```

procedure TSzalakForm.KilepesClick(Sender: TObject);
begin
    Close;
end;

```

Pillanatfelvételnként megállítjuk egy másodpercre a szálak futását.

```

procedure TSzalakForm.PillanatFelvetelClick(Sender: TObject);
var
    i: Integer;
begin
    for i:=0 to SzalLista.Count-1 do
        TRajzoloSzal(SzalLista[i]).Suspend;
    Sleep(1000);
    for i:=SzalLista.Count-1 downto 0 do
        TRajzoloSzal(SzalLista[i]).Resume;
end;

```

A modul inicializáló részében inicializáljuk a véletlenszám-generátort.

```

initialization
    Randomize;
end.

```

Az alkalmazás futás közbeni ablaka:

