

mgr inż. Stanisław Ubermanowicz

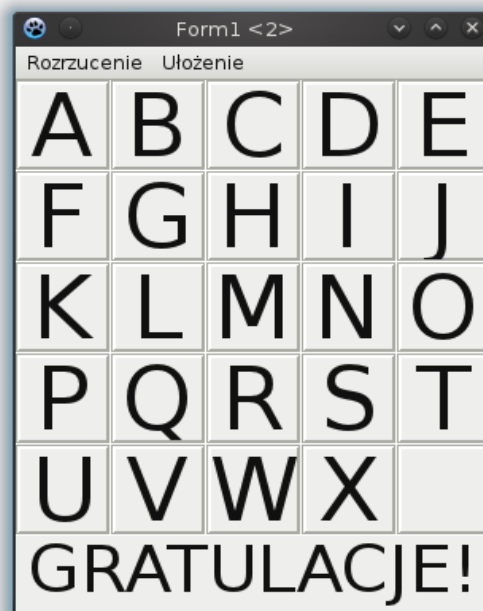
Piotr Fiorek

### Układanka alfabetyczna (puzzle przesuwane) a001

Zaprojektuj grę z alfabetycznym porządkowaniem 24 liter (od A do X) poprzez kliknięcia i przemieszczanie pól literowych stykających bokiem do miejsca pustego. Wymagana jest jedynie symulacja przemieszczania, bez przesuwania obiektów. Gra musi mieć *Menu* z trzema poziomami „Rozrzucenia” (łatwe, średnie i trudne) oraz możliwość automatycznego „Uporządkowania” liter.



Wygląd okna w projekcie



Wygląd podczas działania

#### Projektowanie obiektów przykładowej implementacji

Na formularzu należy umieścić napis TLabel, który będzie się pojawiał w momencie ułożeniu puzzli we właściwej kolejności. Dodatkowo należy umieścić obiekt reprezentujący pasek menu TMainMenu oraz skonfigurować go po podwójnym kliknięciu. Należy dodać dwa menu rozwijane ‘Rozrzucenie’ i ‘Ułożenie’, zawierające pola: ‘Rozrzucenie’ – ‘Łatwo’, ‘Średnio’ i ‘Trudno’ oraz ‘Ułożenie’ – ‘Ułóż’. Doda to kolejne elementy do okna „Object Inspector”. Należy wybrać element reprezentujący wpis ‘Średnio’ w menu ‘Rozrzucenie’ i ustawić jego atrybut Tag na wartość 1, a ten sam atrybut obiektu ‘Trudno’ na 2. W tej

implementacji nie trzeba zmieniać nazw. mogą pozostać domyślne – ‘Label1’ dla obiektu typu TLabel, będącego napisem końcowym.

**UWAGA:** Do zapisywania implementacji nie wolno używać tych samych nazw dla pliku z kodem (\*.pas) oraz dla pliku projektu (\*.lpi’).

```
Kod źródłowy implementacji                                { Część elementów kodu generowana jest automatycznie }
unit puzzle;
{$mode objfpc}{$H+}
interface
uses                                     { Specyfikacja użytych bibliotek i formantów }
  Classes, SysUtils, FileUtil, LResources, Forms, Controls, Graphics, Dialogs,
  Menus, StdCtrls, ExtCtrls;
{ Należy dopisać ExtCtrls aby móc używać obiektu TPanel }
type
  { TForm1 }
  TForm1 = class(TForm)                { Zmienne odpowiadające obiektom na formularzu }
    Label1: TLabel;
    MainMenu1: TMainMenu;              { Opcje menu }
    MenuItem1: TMenuItem;
    MenuItem2: TMenuItem;
    MenuItem3: TMenuItem;
    MenuItem4: TMenuItem;
    MenuItem5: TMenuItem;
    MenuItem6: TMenuItem;
  { Deklaracja procedur obsługi układanki }
    procedure FormCreate(Sender: TObject); { wywoływana przy tworzeniu okna }
    procedure ClickAction(Sender: TObject); { obsługa kliknięcia na obiekt }
    procedure MoveAction(nadawca: Integer); { procedura zamieniająca elementy }
    procedure SprawdzKoniec();             { procedura sprawdzająca kolejność ułożenia }
    procedure Rozrzuc(Sender: TObject);    { procedura rozrzucająca elementy }
    procedure Uloz(Sender: TObject);       { procedura układająca elementy w kolejności }
  private
    { private declarations }
    kafelki: array[0..4, 0..4] of Tpanel; { Tablica dwuwymiarowa – matryca
    reprezentująca układ kafelków puzzli }
    nadawca: Integer;                    { Zmienna używana do identyfikacji klikniętego kafelka }
  public
    { public declarations }
  end;

var
  Form1: TForm1;
  r, k, i: Integer;                     { Zmienne używane podczas operacji na tablicy:
  r oznacza rzęd, k kolumnę, i jest używane jako licznik iteracji }
implementation
{ TForm1 }
{ Procedura wywoływana jednorazowo przy tworzeniu okna }
```

```

procedure TForm1.FormCreate(Sender: TObject);
begin
    Label1.Visible:= False;           { Ukryj napis o wygranej }
    for r:= 0 to 4 do                 { Przemieszczaj się po rzędach w tablicy }
    begin
        for k:= 0 to 4 do             { przemieszczaj się po kolumnach w tablicy }
        begin
            kafelki[r, k]:= TPanel.Create(self);    { Stwórz wszystkie elementy }
            kafelki[r, k].Parent:= self;           { przypisz je do okna }
            kafelki[r, k].Height:= 60;              { określ ich wysokość }
            kafelki[r, k].Width:= 60;               { określ szerokość }
            kafelki[r, k].Caption:= Char(r*5+k+65); { Ustaw napis na kafelku z kodu ASCII }
            if (r*5+k)=24 then kafelki[r, k].Caption:= ''; { ustaw napis na pusty }
            kafelki[r, k].Font.Size:= 34;           { Ustaw wielkość czcionki }
            kafelki[r, k].BorderWidth:= 5;          { Ustaw rozmiar ramki kafelka }
            kafelki[r, k].BorderStyle:= bsSingle;   { oraz styl ramki kafelka }
            kafelki[r, k].Top:= r*60;               { Ustaw obiekt w odpowiednim miejscu w oknie }
            kafelki[r, k].Left:= k*60;              { Ustaw obiekt w odpowiednim miejscu w oknie }
            kafelki[r, k].Tag:= r*5+k;              { Przypisz obiektowi odpowiedni numer porządkowy }
            kafelki[r, k].OnClick:= @ClickAction;    { Przypisz funkcję obsługi kliknięcia }
        end;
    end;
end;

procedure TForm1.ClickAction(Sender: TObject); { Procedura obsługująca kliknięcie }
begin
    { Przekształć typ ogólny Sender na obiekt TPanel i pobierz do zmiennej wartość numeru porządkowego }
    nadawca:= (Sender as TPanel).Tag;
    MoveAction(nadawca); { Przekaż pobrany numer porządkowy do procedury MoveAction }
    SprawdzKoniec();     { Sprawdź czy elementy zostały ułożone w prawidłowej kolejności }
end;

{ Procedura przemieszczania elementów }
procedure TForm1.MoveAction(nadawca: Integer);
begin
    for r:= 0 to 4 do { Przejdź przez wszystkie rzędy }
    begin
        for k:=0 to 4 do { oraz wszystkie kolumny }
        begin
            if kafelki[r, k].Tag=nadawca then { Poszukaj element kliknięty }
            begin { Analizuj zawartość i zakres macierzy }
                if (r > 0) and (kafelki[r-1, k].Caption = '') then
                begin { Zmień zawartości macierzy i napisów na kafelkach }
                    kafelki[r-1, k].Tag:= kafelki[r, k].Tag;
                    kafelki[r-1, k].Caption:= kafelki[r, k].Caption;
                    kafelki[r, k].Tag:= 24;
                    kafelki[r, k].Caption:= '';
                end
            else if (r < 4) and (kafelki[r+1, k].Caption = '') then
            begin
                kafelki[r+1, k].Tag:= kafelki[r, k].Tag;
                kafelki[r+1, k].Caption:= kafelki[r, k].Caption;
                kafelki[r, k].Tag:= 24;
            end
        end
    end
end

```

```

    kafelki[r, k].Caption:= '';
end
else if (k > 0) and (kafelki[r, k-1].Caption = '') then
begin
    kafelki[r, k-1].Tag:= kafelki[r, k].Tag;
    kafelki[r, k-1].Caption:= kafelki[r, k].Caption;
    kafelki[r, k].Tag:= 24;
    kafelki[r, k].Caption:= '';
end
else if (k < 4) and (kafelki[r, k+1].Caption = '') then
begin
    kafelki[r, k+1].Tag:= kafelki[r, k].Tag;
    kafelki[r, k+1].Caption:= kafelki[r, k].Caption;
    kafelki[r, k].Tag:= 24;
    kafelki[r, k].Caption:= '';
end;
Exit();                                     { Po przemieszczeniu napisów wyjdź z funkcji }
end;
end;
end;
end;
end;
    { Procedura sprawdzająca czy wszystkie elementy tablicy są ułożone we właściwej kolejności }
procedure TForm1.SprawdzKoniec();
begin
    for k:= 0 to 23 do                       { Przeszukaj całą tablicę z wyjątkiem ostatniego pola }
    begin
        i:=k+65;                             { Wylicz kod litery którą powinno mieć to pole }
        if Char(i)<>kafelki[k div 5, k mod 5].Caption then { Sprawdź kolejność liter }
        begin                                { Jeśli kolejność jest inna niż alfabetyczna, to... }
            Label1.Visible:= False;          { ukryj napis o wygranej }
            Exit();                          { wyjdź z funkcji }
        end;
    end;
    Label1.Visible:= True;                   { Jeśli kolejność się zgadza, to pokaż napis „Gratulacje!” }
end;

procedure TForm1.Rozrzuc(Sender: TObject); { Procedura rozrzucająca elementy }
var
    ilosc: Integer;                          { Zmienna lokalna dla tej procedury }
begin
    Randomize();                             { Ustaw wartość początkową generatora liczb losowych }
    Label1.Visible:= False;                  { Ukryj napis }
    nadawca:= (Sender as TMenuItem).Tag;    { Pobierz wartość numeru porządkowego }
    if nadawca=0 then ilosc:= 200            { zależnie od jego wartości ustal liczbę przemieszczeń }
    else if nadawca=1 then ilosc:= 1000
    else if nadawca=2 then ilosc:= 5000;

    for i:= 0 to ilosc do { określona ilość razy }
        MoveAction(Random(25)); { przemieść losowy element }
    end;

    { Procedura układająca wszystkie elementy w prawidłowej kolejności }

```

```

procedure TForm1.Uloz(Sender: TObject);
begin
    for r:= 0 to 4 do                                { Przejdź przez wszystkie rzędy }
    begin
        for k:= 0 to 4 do                            { i przez wszystkie kolumny }
        begin
            kafelki[r, k].Caption:= Char(r*5+k+65); { Przypisz kolejne litery }
            if (r*5+k)=24 then kafelki[r, k].Caption:= ''; { Jeśli to ostatni kafelek, to ustaw pole
napisu jako puste }
            kafelki[r, k].Tag:= r*5+k;                { ustaw kolejne numery porządkowe }
        end;
    end;
end;

initialization
    {$I puzzle.lrs}

end.

```