Практична робота.

Тема: Робота з елементами стеку.

Мета : Ознайомитися з алгоритмами роботи з елементами стеку.

Теоретичні відомості

Стеком називається структура даних, що організована за принципом «***останній прийшов – першим пішов***».

Оскільки організація одновимірного масиву аналогічна лінійній структурі пам'яті комп'ютера, то логічним буде представлення стека саме у вигляді одновимірного масиву (див. малюнок)

a1	a2	a3	…	an

А от організація обробки елементів цього масиву буде такою, що відповідає означенню структури даних. Для простішого розуміння поняття «стек» проведемо аналогію з стопкою книг.

Додаючи нову книжку в стопку, ми кладемо її зверху, тобто операція запису нового елемента відбувається в кінець масиву додаванням нового елемента. А для того щоб узяти книжки із середини стосу, нам треба всі верхні книжки по одній зняти у зворотному порядку щодо їх складання. Таким чином, операція читання зі стека відбувається також з кінця масиву, але в зворотному порядку.

Обробляючи елементи масиву, ми вводимо поняття поточного порядкового номеру елемента масиву ***i***. В даному випадку нас цікавитиме лише останній елемент: ми його або зчитуємо(↑), або після цього записуємо новий елемент (↓)

Індекс останнього елемента стека називається ***вершиною***. Це означає, що для обробки елементів стека на достатньо знати значення лише однієї величини і вершини стека.

Перш за все треба оговорити критичні ситуації, які можуть виникнути при читанні та записі інформації. А саме: при читанні може виникнути ситуація, коли вже ***читати нема чого***, тобто **стек порожній**, а при записі – **стек переповнений**, тобто ***досягнуто останнього елемента масиву***, оскільки описуючи масив у програмі, ми повинні вказати межі зміни його індексу. Відповідні повідомлення мають з'явитися на екрані під час тестування програми.

Логічним є твердження, що на початку роботи програми з обробки елементів стека вершина стека повинна мати значення **0**, оскільки в початковому стані дозволена лише операція запису в стек.

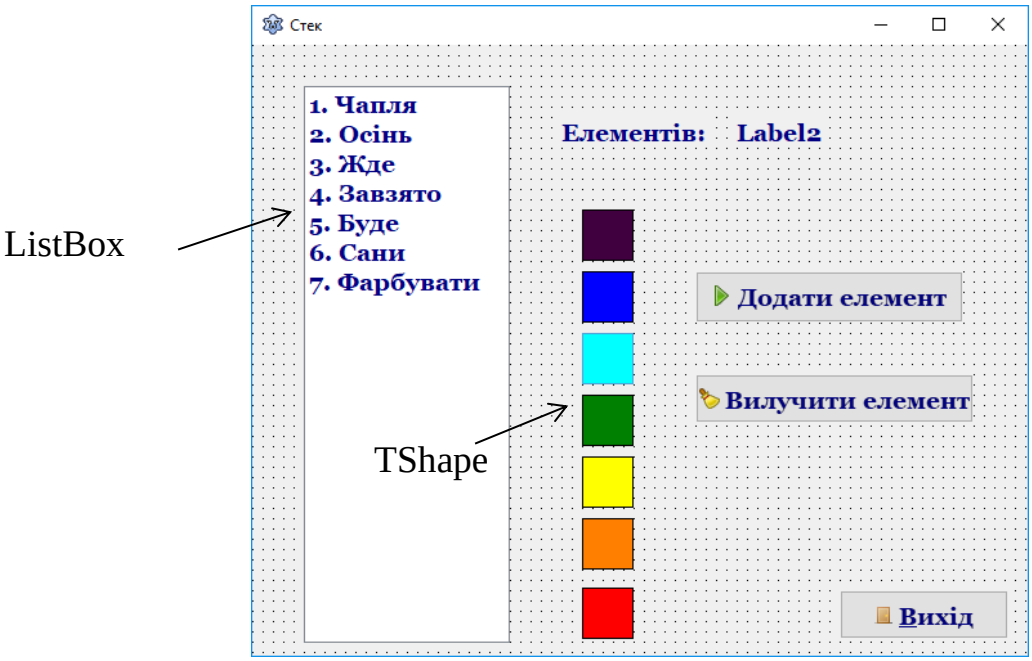
При виконанні програми, що реалізує роботу зі стеком, нам цікаво спостерігати за вмістом стека під час виконання операцій читання та запису (додавання елементів до стеку).

Корисною також є інформація про вміст усього масиву, відведеного для організації стека, під час обробки елементів стека.

Практичне завдання

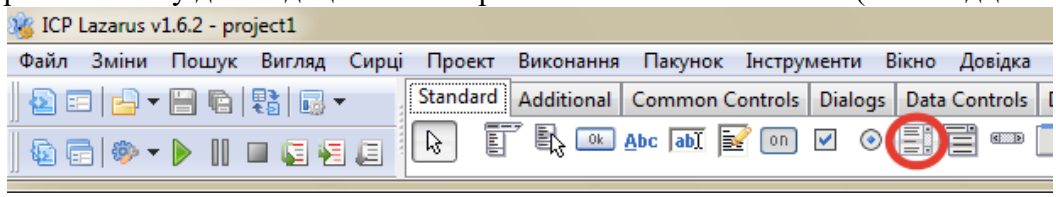
Принцип роботи програми наступний: натиснувши кнопку **ДОДАТИ** повинен додатися елемент до стеку, натиснувши кнопку **ВИЛУЧИТИ**, елемент стека має бути вилучений. Про переповнення та пустоту стека має з'явитися повідомлення на екрані.

Макет

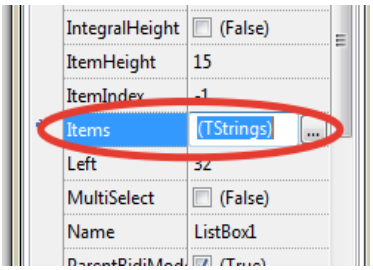


За бажанням можна використати інші варіанти для списку!

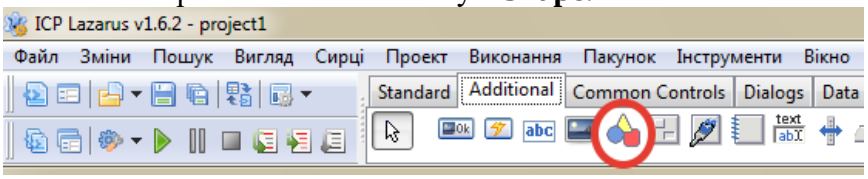
1) Для створення масиву даних доцільно використати компонет **ListBox** (на вкладці Standard).



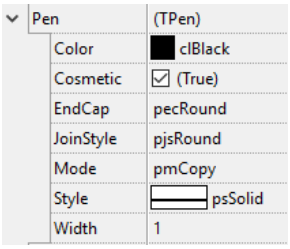
Компонент **ListBox** являє собою прямокутну область, заповнену списком однорядкових текстових елементів. Доступ до рядків надається властивістю **Items** цього компонента. Кожний елемент слід набрати з нового рядка, перехід на новий рядок здійснюється через ENTER.



2) Для візуалізації процесів додавання і вилучення елементів доцільним є використання компоненту **TShape**.



За допомогою компонента **TShape** (Фігура), можна під час розробки намалювати просту геометричну фігуру певної форми. У відповідності зі значеннями властивості Shape можна вивести на екран **коло, еліпс, прямокутник** (з прямими або округленими вершинами) або квадрат. Властивості **Brush** і **Pen** задають вид заповнення і межі відповідно.



Під час роботи з компонентом Shape, зверніть увагу на послідовність додавання їх на форму і відповідність кольорів. Принцип роботи стека «останній прийшов – першим пішов».

Програмний код

```
uses…
```

// Опис глобальний констант

```
const
   NO_EXIST   = 'Глибина стеку обмежена';
   DEEP_STACK = 'Стек переповнений ';
   N_MAX = 7; {кількість елементів стеку}
   RANGE : array [1..N_MAX] of string = ('Чапля',
                                         'Осінь',
                                         'Жде',
                                         'Завзято',
                                         'Буде',
                                         'Сани',
                                         'Фарбувати');
```

// Опис глобальних типів

```
type
    p_my_stack = ^t_my_stack;
   t_my_stack = record
      next : p_my_stack;
      s    : string;
   end;

 { TForm1 }
  TForm1 = class(TForm)
    BitBtn1: TBitBtn;
                                                    ….
```

// Опис глобальних змінних

```
var
  Form1: TForm1;
   p_head   : p_my_stack;
    p       : p_my_stack;
    n       : 0..N_MAX;
…
procedure TForm1.FormCreate(Sender: TObject);
begin
 n := 0;
   p_head:= nil;
   Shape1.Visible := False;
   Shape2.Visible := False;
   Shape3.Visible := False;
   Shape4.Visible := False;
   Shape5.Visible := False;
   Shape6.Visible := False;
   Shape7.Visible := False;
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
   if p_head <> nil then
      p := p_head;
end;
```

Кнопка «Вихід»

```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  close
end;
```

Кнопка «Додати елемент»

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
  if n < N_MAX then begin
       Inc(n);
       Label2.Caption := IntToStr(n);
       New(p);
       p^.next := p_head;  // новий елемент вказує на «колишню» вершину стека
       p^.s := RANGE[n];
       p_head := p; // вершиною стеку стає новий елемент
       ListBox1.Items.Add(IntTOStr(n) + '. ' + p^.s);
       case n of
          1 : begin
                Shape1.Visible := True;
              end;
          2 : begin
                Shape2.Visible := True;
              end;
          3 : begin
                Shape3.Visible := True;
              end;
          4 : begin
                Shape4.Visible := True;
              end;
          5 : begin
                Shape5.Visible := True;
              end;
          6 : begin
                Shape6.Visible := True;
              end;
          7 : begin
                Shape7.Visible := True;
              end;
       end;
    end
    else ShowMessage(DEEP_STACK + IntToStr(N_MAX));
end;
```

Кнопка «Видалити елемент»

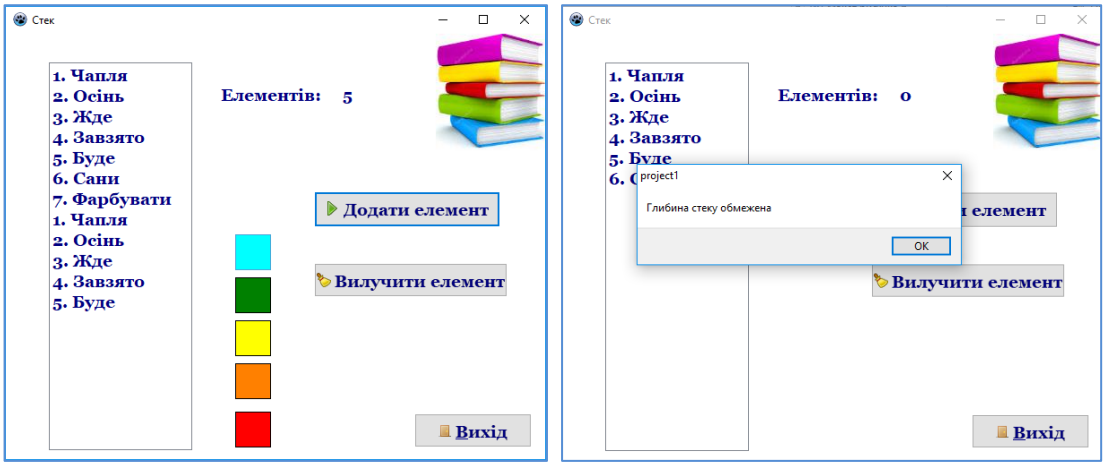
```
procedure TForm1.BitBtn3Click(Sender: TObject);
begin
      ListBox1.Items.Delete(n+n_Max-1);
      case n of
        1 : begin
              Shape1.Visible := False;
            end;
        2 : begin
              Shape2.Visible := False;
            end;
        3 : begin
              Shape3.Visible := False;
```

```
            end;
        4 : begin
                Shape4.Visible := False;
            end;
        5 : begin
                Shape5.Visible := False;
            end;
        6 : begin
                Shape6.Visible := False;
            end;
        7 : begin
                Shape7.Visible := False;
            end;
    end;
        if p_head <> nil then begin
        Dec(n);
        p := p_head;
        p_head := p_head^.next;
    end
    else ShowMessage(NO_EXIST);
    Label2.Caption := IntToStr(n);
end;
```

Завдання

1. Протестуйте програму за такою схемою:
   - ❖ Заповнити повністю стек;
   - ❖ Спорожнити стек;
   - ❖ Заповнити стек до половини;
   - ❖ Додати один елемент;
   - ❖ Видалити два елементи.

Варіанти тестування:



2. Результати тестування представте у вигляді звіту. Звіт має містити скріншоти, які демонструють роботу програми по кожному із завдань. ***Завдань є 5 тому і скріншотів має бути 5***. Кожний скрін має бути підписаний, відповідно до умови завдання. Надішліть звіт учителю.
3. Дайте відповіді на контрольна запитання в зошиті.

Контрольні запитання

1. Дайте означення структури даних «стек».

2. Що називається вершиною стека?

3. Зобразіть схематично роботу з елементами стека.

4. У якому випадку під час тестування може з’явитися повідомлення 'Глибина стеку обмежена' або 'Стек переповнений '?