Оригинал: http://refoteka.ru/r-194991.html

Обработка одномерных массивов в среде программирования Lazarus

Содержание

Введение

Тема работы «Обработка одномерных массивов в среде программирования Lazarus»
Цель работы: получение практических навыков создания оконных приложений для обработки одномерных массивов в среде Lazarus.
Lazarus — свободная среда разработки программного обеспечения для компилятора Free Pascal Compiler. Интегрированная среда разработки предоставляет возможность кроссплатформенной разработки приложений в Delphi-подобном окружении.
На данный момент является единственным инструментом, позволяющим достаточно несложно переносить Delphi-программы с графическим интерфейсом в различные операционные системы: Linux, FreeBSD, Mac OS X, Microsoft Windows.

1 Теоретические сведения

В среде Lazarus для ввода массивов не предусмотрены никакие специальные компоненты, поэтому можно использовать компоненты Edit, Memo или любые другие, предназначенные для ввода данных.
Можно вводить элементы в окне по одному. В этом случае при однократном нажатии кнопки ввода в обработчике событий должны выполниться следующие операторы:

```
i:=i+1;
a[i]:=StrToInt(Edit1.Text);
```

Это означает, что нажатие кнопки приводит к добавлению одного элемента из окна Edit в массив. При выполнении последующих действий по обработке массива значение переменной i определит количество элементов массива. Начальное значение i=0 можно задать в разделе описания типизированных констант или в методе OnCreate для формы (см. ниже) и сбрасывать его каждый раз при вводе нового массива. В массив запишется столько элементов, сколько раз будет нажата кнопка ввода. Если не сбрасывать значение i, то даже после обработки массива можно продолжить ввод элементов в массив.
На рисунке 1 представлена форма для ввода элементов массива.

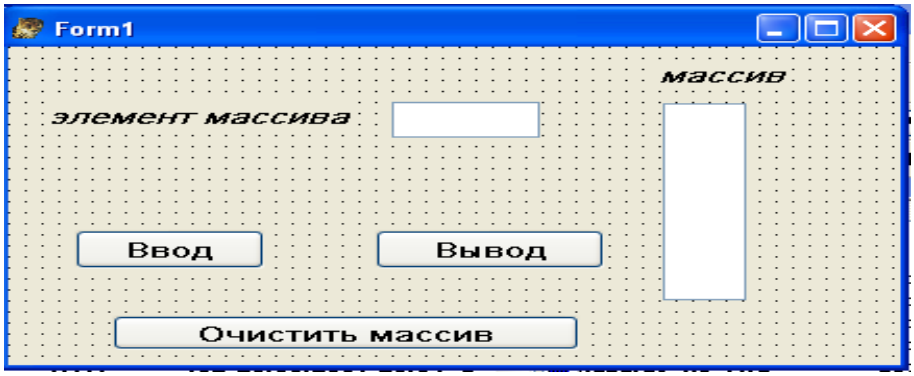

Рисунок 1- Форма программы для ввода и вывода массива

С кнопкой Button1 связан метод **procedure Button1Click - (Ввод).** Метод **edit1.SetFocus** устанавливает фокусна строке ввода Edit1.

```
type
{ TForm1 }
TForm1 = class(TForm)
Button1: TButton;
Button2: TButton;
Button3: TButton;
Edit1: TEdit;
Label1: TLabel;
Label2: TLabel;
ListBox1: TListBox;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
private
{ private declarations }
public
{ public declarations }
end;
var
Form1: TForm1;
a:array[1..20]of integer;
j,i,n:integer;
implementation
{ TForm1 }
procedure TForm1.FormCreate(Sender: TObject);
begin
i:=0;
listBox1.Clear;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
edit1.SetFocus;
i:=i+1;
```

```
a[i]:=strToInt (edit1.text);
edit1.Clear;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
for j:=1 to i do
listBox1.Items.Add(IntToStr(a[j]));
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
i:=0;
listBox1.Clear;
end;
initialization
{$I unit2.lrs}
end.
```

На рисунке 2 представлен результат работы программы.

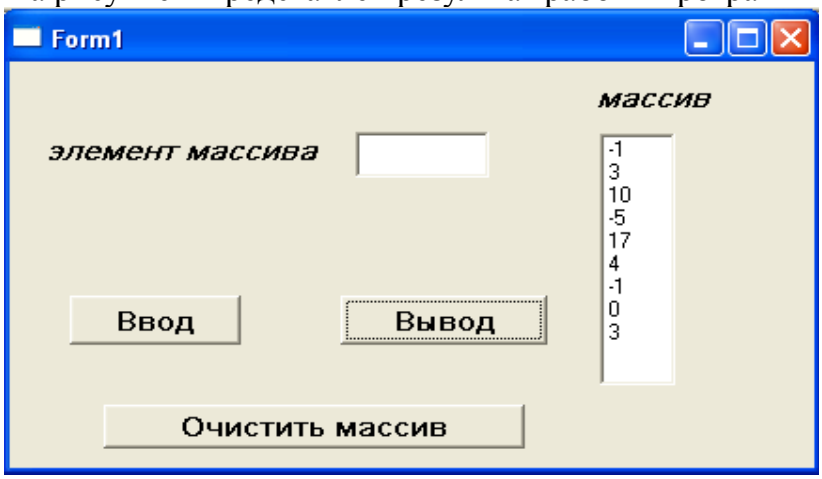

Рисунок 2- Результат работы программы ввода и вывода массива

В компоненте **Edit** можно ввести сразу все элементы массива, разделяя их пробелами. Нужно помнить, что количество пробелов-разделителей может быть любым.
Цикл для пропуска пробелов между словами :

```
while (st[i]=' ') and (i<=length(st)) do
inc(i);
```

Слова можно пропустить аналогичным циклом:

```
while (st[i]<>' ') and (i<=length(st)) do
inc(i);
```

Эти два цикла должны быть включены во внешний цикл, который закончится тогда, когда закончится строка.
Пример выделения всех слов строки st:

```
i:=1;
```

```
while i<=length(st) do
begin
while (st[i]=' ') and (i<=length(st)) do
inc(i);
k:=i;
while (st[i]<>' ') and (i<=length(st)) do
inc(i);
sl:=copy(st,k,i-k);
{обработка выделенного слова sl}
end;
```

В отличие от строки ввода, текстовый редактор **Memo** может содержать любое количество строк. Его свойство**text** представляет собой строку, состоящую из находящихся в поле **Memo** строк, разделенных последовательностью символов с кодами 13 и 10 (конец строки и переход на новую строку). Эти символы добавляются в поле Memo при нажатии клавиши Enter. Для выделения подстроки, содержащей элемент массива, нужно найти символ с кодом 13 (#13), скопировать в новую подстроку, а затем удалить ее вместе с кодами 13 и 10 и продолжить поиск конца строки.
В приведенном ниже фрагменте программы вначале в строку st записывается содержимое всего окна memo1. Затем отыскивается позиция конца первой строки (n) и эта строка копируется в st1 и удаляется из st. На рисунке 3 приведен результат работы программы.

```
procedure TForm1.Button1Click(Sender: TObject);
var
st,st1:string;
n:integer;
begin
st:=memo1.Text;
n:=pos(#13,st);
st1:=copy(st,1,n-1);
label1.Caption:='Первый элемент = '+st1;
delete(st,1,n+1);
ShowMessage('остальные элементы:'+#13+st);
end;
```

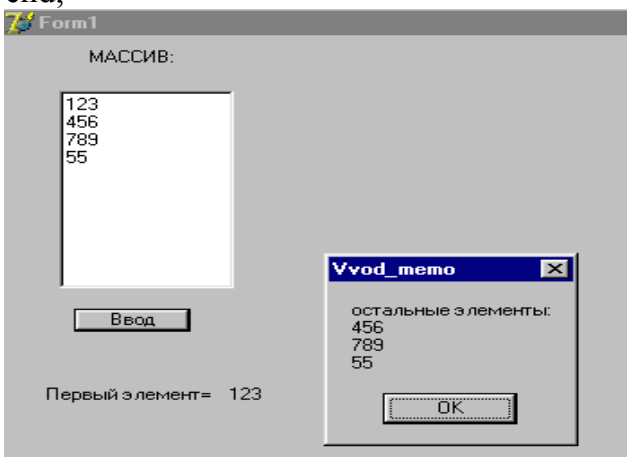

Рисунок 3 - Результат обработки memo.text

Приведенный выше пример не учитывает того обстоятельства, что любая подстрока может содержать любое количество пробелов. Их нужно удалить перед преобразованием строки в число.
Функция GetLine возвращает подстроку с номером n из строки.

```
Function GetLine(s:string;n:integer):string;
var
p:integer;
begin
{удалить пробелы в начале строки}
While (pos(' ',s)=1)and (length(s)>0) do
delete(s,1,1);
if n>1 then
repeat
p:=pos(#13,s);
if p<>0 then
begin
s:=copy(s,p+2,length(s)-p);
While (pos(' ',s)=1)and (length(s)>0) do
delete(s,1,1);
n:=n-1;
end
until(n=1) or (p=0);
if n>1 then result :=''
else
begin
p:=pos(#13,s);
if p<>0 then result:=copy(s,1,p-1)
else result:=s;
end;
end;
procedure TForm1.Button1Click(Sender: TObject);
Var a:array [1..10]of string[20];
i:integer;
st:string;
begin
For i:=1 to k do
a[i]:=GetLine(Memo1.Text,i);
st:='';
For i:=1 to k do
st:=st+a[i]+#13;
ShowMessage(st);
end;
```

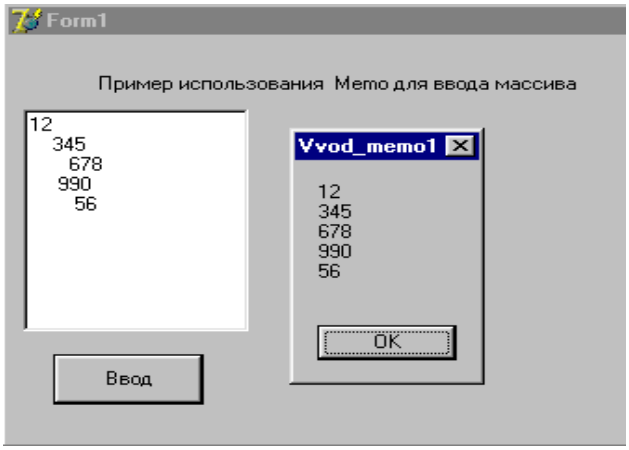

Рисунок 4 - Результат работы программы ввода массива

Для отображения на экране списка значений в Lazarus предназначен компонент **ListBox**. Этот компонент имеет в своем составе объект-список, содержащий набор строк с ассоциированными с ними произвольными объектами. Для того, чтобы отобразить последовательно на экране данные, которые вводятся в строку ввода, можно выполнить оператор ListBox1.Items.Add(Edit1.Text). Если нужно вывести в ListBox элементы массива, то используют оператор ListBox1.Items.Add(IntToStr(a[i])).

Комбинированная строка ввода (поле со списком) **ComboBox** объединяет в себе возможности строки ввода Edit и списка ListBox. Добавить элемент в список можно вызовом ComboBox1. Items.Add(<строка>). Свойство DroppedDown логического типа указывает, отображается ли раскрывающийся список в данный момент. Свойство доступно только на этапе выполнения программы.

Например, ComboBox1. DroppedDown:=true.

Установку начальных значений можно задать в обработчике события **OnCreate** для формы. Это событие возникает в момент создания формы. Размещение операторов в этом обработчике во многих случаях эквивалентно размещению операторов в секции инициализации модуля. Заготовка обработчика этого события создается при выполнении двойного щелчка по форме Form1. Методы **ListBox1.Clear, ComboBox1.Clear, Memo1.Clear и Edit1.Clear** позволяют очистить соответствующие компоненты.

2 Практическая часть

Выполнить обработку одномерного массива по индивидуальному заданию.
Предусмотреть 2 варианта ввода массива:
в строке Edit по одному элементу;
в редакторе Memo или в строке Edit ввести весь массив.
Вывод массива выполнить с помощью компонент ListBox и ComboBox. Выводить нужно как исходный массив, так и результирующий.

2.1 Задание

Даны целые числа X(n). Переписать в новый массив все числа, лежащие в диапазоне [-3,7].

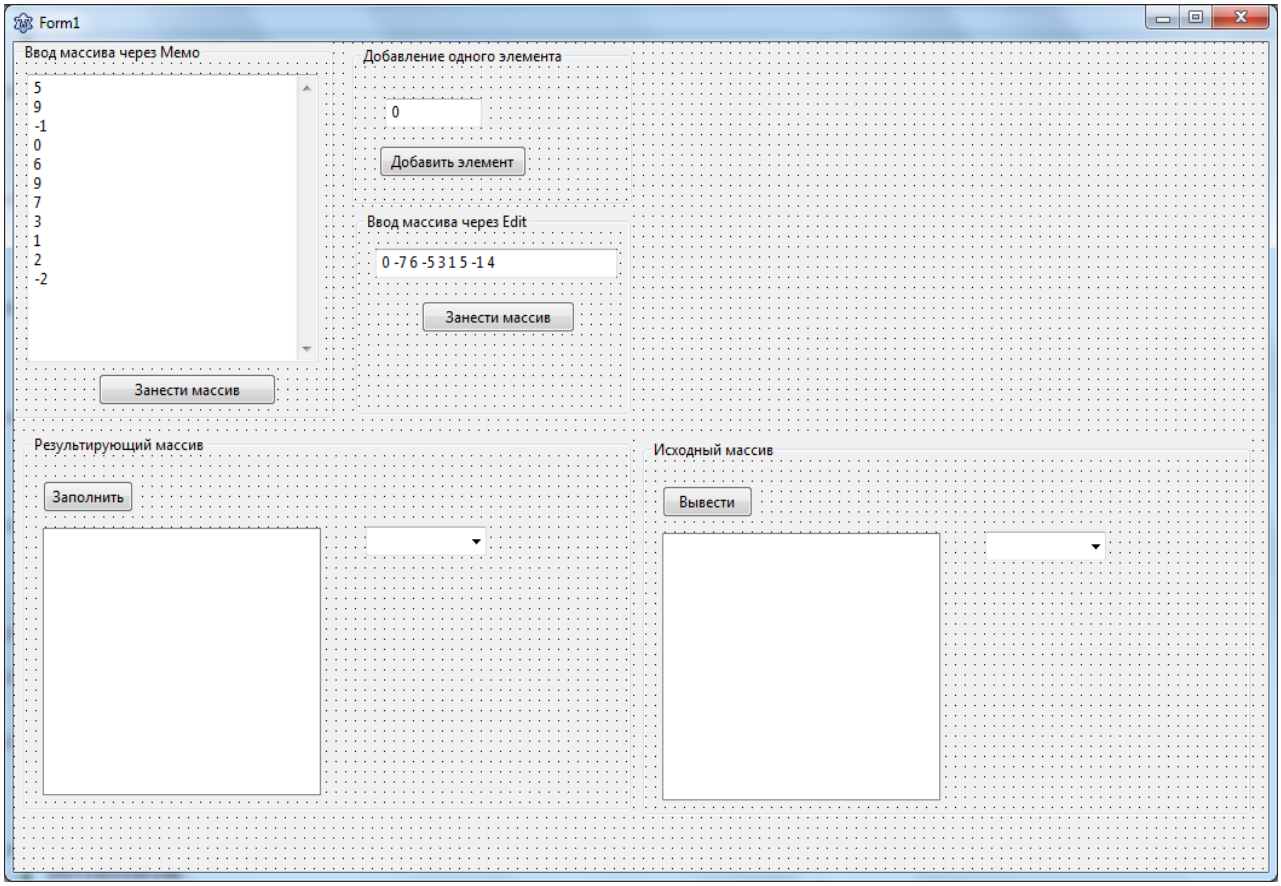

Рисунок 5 – Экранная форма

2.2 Листинг программы

```
unit Unit1;

{$mode objfpc}{$H+}

interface

uses
Classes, SysUtils, FileUtil, LResources, Forms, Controls, Graphics, Dialogs,
StdCtrls;

type

{ TForm1 }

TForm1 = class(TForm)
Button1: TButton;
Button2: TButton;
Button3: TButton;
Button4: TButton;
Button5: TButton;
ComboBox1: TComboBox;
ComboBox2: TComboBox;
```

```
Edit1: TEdit;
Edit2: TEdit;
GroupBox1: TGroupBox;
GroupBox2: TGroupBox;
GroupBox3: TGroupBox;
GroupBox4: TGroupBox;
GroupBox5: TGroupBox;
ListBox1: TListBox;
ListBox2: TListBox;
Memo1: TMemo;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
private
{ private declarations }
public
{ public declarations }
end;

var
Form1: TForm1;
X, Y: array [1..100] of integer;
n, m: integer;

implementation

{ TForm1 }

procedure TForm1.FormCreate(Sender: TObject);
begin
n:= 0;
end;

procedure TForm1.Button1Click(Sender: TObject);
var i: integer;
begin
// обнуление массива
n:= 0;
for i:= 1 to 100 do X[i]:= 0;
// заполенние массива X из строк мемо
n:= Memo1.Lines.Count; // установить длину массива
for i:= 1 to n do
X[i]:= StrToInt(Memo1.Lines[i-1]); // занесения значения элемента из строки мемо
end;
```

```
procedure TForm1.Button2Click(Sender: TObject);
begin
inc(n); // увеличиваем счетчик элементов
X[n]:= StrToInt(Edit1.Text); // занесение значения из эдит1
end;

procedure TForm1.Button3Click(Sender: TObject);
var
s: string;
i: integer;
begin
// обнуление массива
n:= 0;
for i:= 1 to 100 do X[i]:= 0;
// выделение элементов между пробелами в эдит2
s:= Edit2.Text+' ';
i:= pos(' ', s);
while i<>0 do
begin
inc(n);
X[n]:= StrToInt(copy(s, 1, i-1)); // занесение значения из эдит1
delete(s, 1, i); // удаление выделенного элемента из строки
i:= pos(' ', s);// поиск следующего элемента
end;
end;

procedure TForm1.Button4Click(Sender: TObject);
var i: integer;
begin
// обнуление результата
m:= 0;
ListBox1.Clear;
ComboBox1.Clear;
// перебор исходного массива
for i:= 1 to n do
// сравнение значения элемента с диапазоном
if (X[i]>=-3)and(X[i]<=7) then
begin
// занесение нового элемента в результирующий массив Y
inc(m);
Y[m]:= X[i];
// вывод элемента массива Y на экранные компоненты
ListBox1.Items.Add(IntToStr(Y[m]));
ComboBox1.Items.Add(IntToStr(Y[m]));
end;
end;

procedure TForm1.Button5Click(Sender: TObject);
```

```
var i: integer;
begin
ListBox2.Clear;
ComboBox2.Clear;
// перебор исходного массива
for i:= 1 to n do
begin
ListBox2.Items.Add(IntToStr(X[i]));
ComboBox2.Items.Add(IntToStr(X[i]));
end;
end;

initialization
{$I unit1.lrs}

end.
```

2.3 Экранные формы

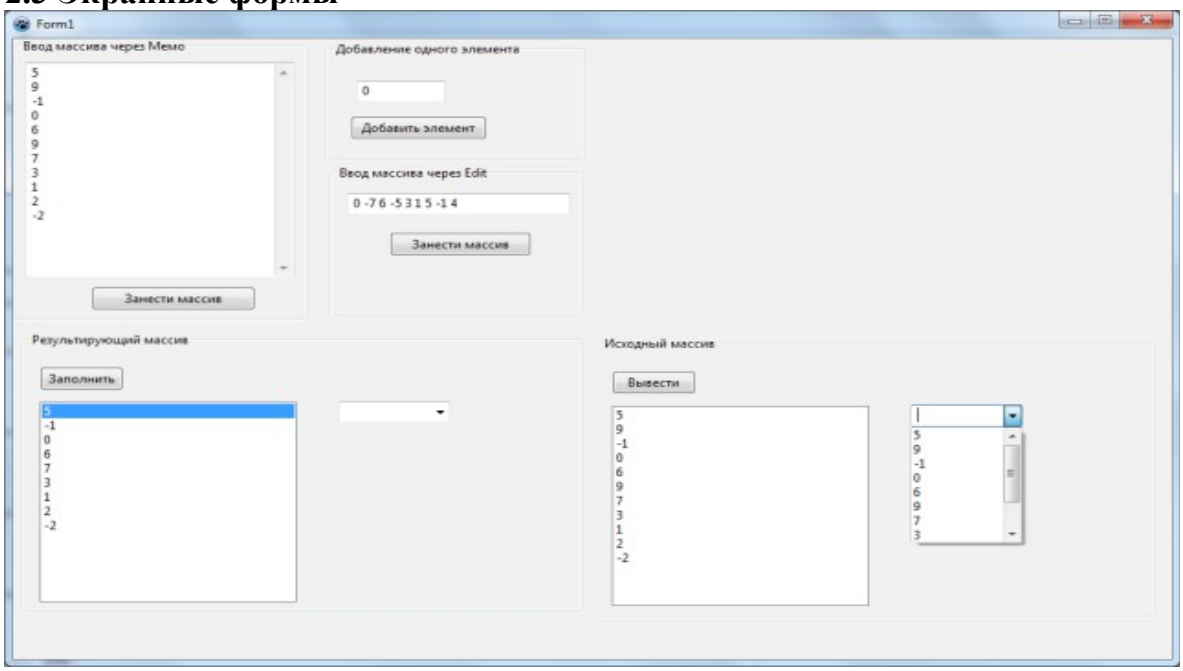

Рисунок 6 - Занесение массива через Memo1

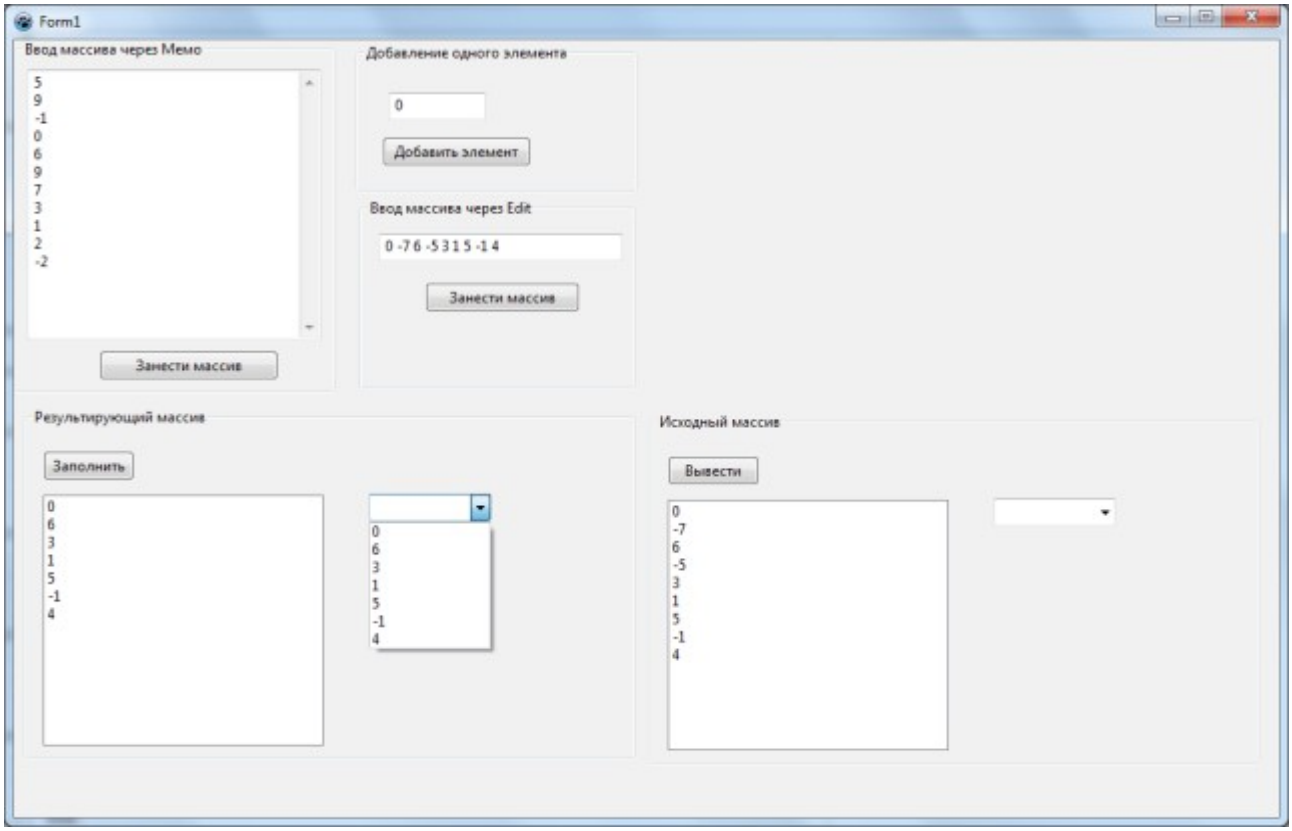

Рисунок 7 - Занесение массива через Edit1

Выводы

Недостатки Lazarus:
- Нет полной совместимости с Delphi (хотя в отличие от Delphi предоставляет возможность создавать кроссплатформенные приложения).
- При стандартных настройках скомпилированный файл имеет очень большой размер, так как включает отладочную информацию. На самом деле недостатком не является, поскольку это легко исправляется. Достаточно указать компилятору дополнительный ключ -Xg (Использовать внешний файл отладочных символов) и -Xs (Вырезать символы из исполнимого файла); Однако это может быть абсолютно не очевидно начинающим программистам, хотя эти настройки доступны и в графическом интерфейсе (Проект -> параметры компилятора -> связывание) и, даже в этом случае, исполняемый файл (по крайней мере, под Windows) заметно превосходит размером сгенерированный Delphi, что в значительной степени компенсируется упаковщиками. Для создания небольших приложений альтернативой лазарусу может служить MSE.
- Отсутствие полноценной документации. Но документация по самому компилятору доступна онлайн, либо в PDF/HTML документах, а документация по Lazarus доступна в виде Wiki — учебников, которые могут редактировать сами пользователи.
- Нет полноценной поддержки COM (реализована только поддержка методов), что, впрочем, вполне естественно, поскольку сфера интересов разработчиков Lazarus лежит в области кроссплатформенного программирования, а не в области взаимодействия с Windows-приложениями.

- Отладчик не позволяет просматривать значения свойств объектов во время отладки, только переменных и полей объектов.

Литература

Е.Р. Алексеев, О.В. Чеснокова, Т.В. Кучер Free Pascal и Lazarus: Учебник по программированию - ALT Linux; ДМК-пресс, 2010 – 442 с.