Ввод массива

Для того чтобы работать с массивом, его надо с начала получить от пользователя. Попробуем осуществить ввод одномерного массива всего в одно поле редактирования. Каждый элемент массива будет отделяться от предыдущего разделителем. После щелчка на кнопке программа выделит из строки, содержащей массив, первую подстроку, затем выделит вторую подстроку и т.д.

Давайте сейчас этим и займемся. Поставьте на Вашу форму кнопку **Button** и поле редактирования **Edit.**

Заголовок нашей функции будет такой:

function GetSubStr(st:string; expl:string; n:integer):string;

где

st - строка, содержащая массив expl - строка разделитель n - номер подстроки

function TForm1.GetSubStr(st:string; expl:string ;n:integer):string;

Var p,i:integer;

Begin

for i:= 1 to n-1 do

begin

p:=pos(expl,st);

st:=copy(st,p+1,Length(st)-p);

while (pos(expl,st)=1) and (length(st)>0) do

delete(st,1,1);

end;

p:=pos(expl,st);

if p<>0 then result:=copy(st,1,p-1)

else result:=st;

End;

Для получения n-ой подстроки (элемента массива) из полученной в качестве аргумента строки функция сначала удаляет предшествующую ей n-1 подстроку (цикл for), затем находит разделитель, который обозначает конец нужной подстроки, выделяет подстроку и возвращает ее в качестве значения функции (через свое имя).

Не забудьте добавить заголовок нашей функции в раздел **Public**модуля программы.

Теперь для проверки работы нашей функции напишем обработчик события **OnClick,**для нашей кнопки. В начале мы с помощью нашей функции получим массив из Edit1, а потом выведем его:

procedure TForm1.Button1Click(Sender: TObject);

var i:integer;

a:array[1..10] of string[10];

st:string;

begin

for i:=1 to 10 do

a[i]:=GetSubStr(Edit1.text,' ',i);//используем пробел в

качестве разделителя

for i:=1 to 10 do

st:=st+IntTostr(i)+' '+a[i]+#13;

ShowMessage(st);

end;

Поиск минимального (максимального) элемента массива

Будем искать минимальный элемент в целочисленном массиве. Для этого немного изменим обработчик события **OnClick**для кнопки:

procedure TForm1.Button1Click(Sender: TObject);

var i:integer;//номер элемента, сравниваемого с минимальным

a:array[1..10] of integer;

min:integer;//номер минимального элемента

begin

//Введем массив

for i:=1 to 10 do

//Преобразуем полученные подстроки в числа

a[i]:=StrToInt(GetSubStr(Edit1.text,' ',i));//используем пробел в качестве разделителя

//Найдем минимальный элемент

min:=1; //пусть номер минимального элемента = 1

for i:= 2 to 10 do // начнем искать со следующего

if a[i] < a[min] then min:=i;

Form1.caption:=IntToStr(a[min]); // выводим в заголовок

формы минимальный элемент

end;

В этом примере **a[min]** минимальный элемент массива, а **min -**номер минимального элемента. Алгоритм очень простой: сравниваем каждый следующий элемент с минимальным, если он меньше минимального, то запоминаем его номер в переменной **min**, и продолжаем сравнивать уже с ним.

Чтобы найти максимальный элемент, нужно изменить всего одну строку:

>>>

if a[i] < a[min] then min:=i;

Надо заменить на:

if a[i] > a[min] then min:=i;

Только теперь **a[min]** - максимальный элемент, а **min** - номер максимального элемента.

Поиск заданного элемента в массиве

Поступим методом простого перебора. Для этого будем перебирать все элементы массива, пока не встретим искомый элемент, или пока не дойдем до конца массива.

Элемент, совпадение с которым нам надо найти будем хранить в текстовом поле **Edit2**. Обработчик события OnClick нашей кнопки будет иметь такой вид:

procedure TForm1.Button1Click(Sender: TObject);

var i:integer;

a:array[1..10] of integer;

n:integer;//образец

found:boolean;

begin

//Введем массив

for i:=1 to 10 do

//Преобразуем полученные подстроки в числа

a[i]:=StrToInt(GetSubStr(Edit1.text,' ',i));//используем пробел в качестве разделителя

n:=StrToInt(Edit2.text);

found:=false;

i:=1;

REPEAT

if a[i] = n then found:=true

else i:=i+1;

UNTIL (i > 10) or (found = true);

if found then showmessage('Совпадение с элементом номер '+IntToStr(i));

end;

Сортировка массива

Вот мы и дошли до самого интересного - до сортировки массива. Рассмотрим алгоритм т.н. прямого выбора. Смысл его заключается в следующем:

Просматривая массив от первого элемента, найдем минимальный элемент и поместим его на место первого элемента, а первый элемент - на место минимального.

Затем будем просматривать массив, начиная со второго элемента, и далее поступим, как поступили перед этим шагом.

Алгоритм ясен, теперь приступим к написанию кода. Все тот же обработчик события OnClick принимает теперь такой вид:

procedure TForm1.Button1Click(Sender: TObject);

var i,k,j,min:integer;

buf:integer; // буфер для обмена

a:array[1..10] of integer;

st:string;

begin

//введем массив

for i:=1 to 10 do

a[i]:=StrToInt(GetSubStr(Edit1.text,' ',i));//используем пробел в качестве разделителя

for i:=1 to 10 - 1 do // кол-во элементов минус один

begin

//поищем минимальный элемент

min:=i;

for j:=i+1 to 10 do

if a[j] < a[min] then min:=j;

//поменяем местами

buf:=a[i];

a[i]:=a[min];

a[min]:=buf;

end;

for k:=1 to 10 do

Form1.Caption:=Form1.caption + ' '+ IntToStr(a[k]);

end;

**Массив**— это структура данных, представляющая собой набор переменных одинакового типа, имеющих общее имя. Массивы удобно использовать для хранения однородной по своей природе информации, например, таблиц и списков.

Объявление массива

Массив, как и любая переменная программы, перед использованием должен быть объявлен в разделе объявления переменных. В общем виде инструкция объявления массива выгладит следующим образом:

**Имя:** array  **[нижний\_индекс..верхний\_индекс] of тип**

где:

* имя — имя массива;
* array — зарезервированное слово языка [Delphi](http://delphicomponent.ru/" \o "Delphi" \t "_blank), обозначающее, что объявляемое имя является именем массива;
* нижний\_индекс и верхний\_индекс — целые константы, определяющие диапазон изменения индекса элементов массива и, неявно, количество элементов (размер) массива;
* тип — тип элементов массива.

Примеры объявления массивов:

temper:array[1..31] of real;

koef: array[0..2] of integer;

name: array[1..30] of string[25];

При объявлении массива удобно использовать именованные константы. Именованная константа объявляется в разделе объявления констант, который обычно располагают перед разделом объявления переменных. Начинается раздел объявления констант словом const. В инструкции объявления именованной константы указывают имя константы и ее значение, которое отделяется от имени символом "равно". Например, чтобы объявить именованную константу нв, значение которой равно 10, в раздел const надо записать инструкцию: нВ=10. После объявления именованной константы ее можно использовать в программе как обычную числовую или символьную константу. Ниже в качестве примера приведено объявление массива названий команд-участниц чемпионата по футболу, в котором используются именованные константы.

const

ВТ = 15; // число команд

SN = 25; // предельная длина названия команды

var

team: array[1..BT] of string[SN];

Для того чтобы в программе использовать элемент массива, надо указать имя массива и номер элемента (индекс), заключив индекс в квадратные скобки. В качестве индекса можно использовать константу или выражение целого типа.

Если массив не является локальным, т. е. объявлен не в процедуре обработки события, а в разделе переменных модуля, то одновременно с объявлением массива можно выполнить его инициализацию, т. е. присвоить начальные значения элементам массива. Инструкция объявления массива с одновременной его инициализацией в обшем виде выглядит так:

**Имя: array[нижний\_индекс. .верхний\_индекс] of тип = (список) ;**

где список — разделенные запятыми значения элементов массива.

Например:

a: array[lO] of integer = (0,0,0,0,0,0,0,0,0,0);

Team: array[1..5] of String[10]=('Зенит','Динамо','Спартак','Ротор','ЦСКА');

Обратите внимание, что количество элементов списка инициализации должно соответствовать размерности массива. Если это будет не так, то компилятор выведет сообщения об ошибке: Number of elements differs from declaration (количество элементов не соответствует указанному в объявлении).

При попытке инициализировать локальный массив компилятор выводит сообщение об ошибке: Cannot initialize local variables (локальная переменная не может быть инициализирована). Локальный массив можно инициализировать только во время работы программы.

Типичными операциями при работе с массивами являются:

* вывод массива;
* ввод массива;
* поиск максимального или минимального элемента массива;
* поиск заданного элемента массива;
* сортировка массива.

Вывод массива

Под выводом массива понимается вывод на экран монитора (в диалоговое окно) значений элементов массива.

Если в программе необходимо вывести значения всех элементов массива, то для этого удобно использовать инструкцию for, при этом переменная счетчик инструкции for может быть использована в качестве индекса элемента массива.

В качестве примера на рис. 5.1 приведено диалоговое окно приложения, которое демонстрирует инициализацию и процесс вывода значений элементов массива в поле метки. Программа выводит пронумерованный список футбольных команд. Следует обратить внимание, что для того чтобы список команд выглядел действительно как список, свойству Label1.AutoSize нужно присвоить значение False (присвойте свойству Label1.Autosize значение True и посмотрите, как будет работать программа). Текст программы приведен в листинге.

Листинг. Инициализация и вывод массива

01.unit outar\_;

02.interface

03.Windows, Messages, SysOtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;

04.type

05.TForml = class(TForm)

06.Buttonl: TButton;

07.Label1: TLabel;

08.procedure ButtonlClick(Sender: TObject);

09.private

10.{ Private declarations }

11.public

12.{Public declarations}

13.end;

14.var

15.Form1: TForm1;

16.implementation

17.{$R \*,dfm}

18.const

19.NT = 5;

20.var

21.team: array[1..NT] of string[10] =('Зенит1,'Динамо1,'Ротор1,'Спартак','ЦСКА') ;

22.procedure TForml.ButtonlClick(Sender: TObject);

23.var

24.st:string; // список команд

25.i:integer; // индекс, номер элемента массива

26.begin

27.// формирование списка для отображения в форме

28.for i:=l to NT do st := st + IntToStr(i)+ ' '+ team +#13;

29.// вывод  списка

30.Label1.Caption := st;

31.end;

32.end.

Ввод массива

Под вводом массива понимается процесс получения от пользователя (или из файла) во время работы программы значений элементов массива,

"Лобовое" решение задачи ввода элементов массива — для каждого элемента массива создать поле ввода. Однако если требуется ввести достаточно большой массив, то такое решение неприемлемо. Представьте форму, например, с десятью полями редактирования!

Очевидно, что последовательность чисел удобно вводить в строку таблицы, где каждое число находится в отдельной ячейке. Ниже рассматриваются два варианта организации ввода массива с использованием [компонентов](http://delphicomponent.ru/) StringGrid И Memo.

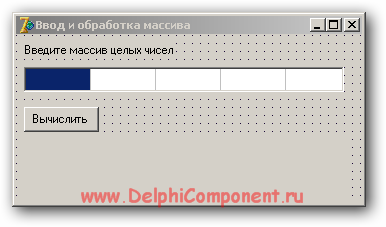
Использование [компонента](http://delphicomponent.ru/) StringGrid

Для ввода массива удобно использовать компонент StringGrid. Значок [компонента](http://delphicomponent.ru/) StringGrid находится на вкладке **Additiona**l (рис. 5.2).

Массивы в Delphi

Компонент strinqGrid представляет собой таблицу, ячейки которой содержат строки символов.

В качестве примера использования [компонента](http://delphicomponent.ru/) stringGrid для ввода массива рассмотрим программу, которая вычисляет среднее арифметическое значение элементов массива. Диалоговое окно программы приведено на рис. 5.3. Компонент stringGrid используется для ввода массива, компоненты Label1 и Label2 — для вывода пояснительного текста и результата расчета, Buttoni — для запуска процесса расчета.



Добавляется компонент stringGrid в форму точно так же, как и другие компоненты. После добавления [компонента](http://delphicomponent.ru/) к форме нужно выполнить его настройку в соответствии с табл. 5.2. Значения свойств Height и width следует при помощи мыши установить такими, чтобы размер [компонента](http://delphicomponent.ru/) был равен размеру строки.

Текст программы приведен в листинге.

Значения свойств компонента **StringGrid1**

|  |  |
| --- | --- |
| **Свойство** | **Значение** |
| ColCount | 5 |
| FixedCols | 0 |
| RowCount | 1 |
| DefaultRowHeight | 24 |
| Height | 24 |
| DefaultColWidth | 64 |
| Width | 328 |
| Options.goEditing | true |
| Options.AlwaysShowEditing | true |
| Options.goTabs | true |

 Листинг. Ввод и обработка массива целых чисел

01.unit getar\_;

02.interface

03.uses

04.Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,

05.Forms, Dialogs, Grids, StdCtrls;

06.type

07.TForml = class(TForm);

08.Labell: TLabel;

09.StringGridl; TStringGrid;

10.Buttonl: TButton;

11.Label2: TLabel;

12.procedure Buttoт1Click(Sender: TObject];

13.private

14.{Private declarations }

15.public

16.{ Public declarations }

17.end;

18.var

19.Forml: TForml;

20.implementation

21.{SR \*.dfm}

22.procedure TForml.Button1Click(Sender: TObject);

23.var

24.a : array[1..5] of integer; // массив

25.suntrn: integer; // сумма элементов

26.sr: real; // среднее арифметическое

27.i: integer/ // индекс

28.begin

29.// ввод массива

30.// считаем, что если ячейка пустая, то соответствующий

31.// ей элемент массива равен нулю

32.for i:= 1 to 5 do

33.if Length(StringGrid1.Cells[i-l,0]] 0

34.then a := StrToInt(StringGrid1.Cells[i-1,0])

35.else a := 0;

36.// обработка массива

37.summ := 0;

38.for i :=1 to 5 do

39.summ := summ + a;

40.sr : = summ / 5;

41.// вывод результата

42.Label2.Caption :='Сумма элементов: ' + IntToStr(summ) + #13+'Среднее арифметическое: ' + FloatToStr(sr);

43.end; end.

После пробных запусков программы возникает желание внести изменения в процесс ввода массива. Так, было бы неплохо, чтобы курсор автоматически переходил в следующую ячейку таблицы, например, в результате нажатия клавиши . Сделать это можно при помощи процедуры обработки события onkeypress. На эту же процедуру можно возложить задачу фильтрации вводимых в ячейку таблицы данных. В нашем случае надо разрешить ввод в ячейку только цифр.

Текст процедуры обработки события onkeypress приведен в листинге 5.3. Следует обратить внимание на свойство Col, которое во время работы программы содержит номер колонки таблицы, в которой находится курсор. Это свойство можно также использовать для перемещения курсора в нужную ячейку таблицы. Однако нужно учитывать, что колонки таблицы, впрочем, как и строки, нумеруются с нуля.

Листинг 5.3. Процедура обработки события onkeypress

procedure TForml.StringGridlKeyPress(Sender: TObject; var Key; Char);

begin

case Key of

#8, ' 0 ' . . '9' : ; // цифры и клавиша<backspsce></backspsce>

#13: // клавиша

if StringGridl.Col < StringGridl.ColCount - 1

then StringGridl.Col := StringGridl.Col + 1;

else key := Chr(O); // остальные символы запрещены

end;

end;

Если нужно ввести массив дробных чисел (a: array[1. .5] of real), то процедура обработки события onkeypress несколько усложнится, т. к. помимо цифр допустимыми символами являются символ-разделитель (запятая или точка — зависит от настройки Windows) и минус. С целью обеспечения некоторой дружественности программы по отношению к пользователю можно применить трюк: подменить вводимый пользователем неверный разделитель верным. Определить, какой символ-разделитель допустим в текущей настройке Windows, можно, обратившись к глобальной переменной Decimal separator. В листинге 5.4 приведен текст модуля приложения ввода и обработки массива дробных чисел. Процедура обработки события onkeypress обеспечивает ввод в ячейку таблицы только допустимых при записи дробного числа символов.

Листинг 5.4. Ввод и обработка массива дробных чисел

01.unit getar\_l;

02.interface

03.uses

04.Windows, Messages, SysUtils, Variants, Classes, Graphics, controls,

05.Forms, Dialogs, Grids, StdCtrls;

06.type

07.TForml - class(TFora)

08.Labell: TLabel;

09.StringGridl: TStringGrid;

10.Buttonl: TButton;

11.Label2: TLabel;

12.procedure ButtonlClick(Sender: TObject) ;

13.procedure StringGridlKeyPress(Sender: TObject; var Key: Char);

14.private

15.{Private declarations}

16.public

17.{ Public declarations }

18.end;

19.var

20.Forml: TForml;

21.implementation

22.{$R \*.dfm}

23.procedure TForml.ButtonlClick(Sender: TObject);

24.var

25.a : array[1..5] of real; // массив

26.summ: real,- // сумма элементов

27.sr: real; // среднее арифметическое

28.i: integer; // индекс

29.begin

30.// ввод массива

31.// считаем, что если ячейка пустая, то соответствующий

32.// ей элемент массива равен нулю

33.for i:= I to 5 do

34.if LengtMStringGridl.Cells [i-1,0]) 0

35.than a[i) := StrToFloat[StringGridl.Cells[i-1,0]]

36.else a := 0;

37.// обработка массива

38.summ := 0;

39.for i :=1 to 5 do

40.summ := summ + a;

41.sr := summ / 5;

42.// вывод результата

43.Label2.Caption :=

44.'Сумма элементов: ' + FloatToStr(summ) + #13+

45.'Среднее арифметическое: ' + FloatToStr(sr);

46.end;

47.// Функция обеспечивает ввод в ячейку только допустимых символов

48.procedure TForml.StringGridlKeyPress(Sender: TObject; var Key: Char);

49.begin

50.case Key of

51.# 8 , ' 0 ' . , ' 9 ' : ; // цифры и <backspace></backspace>

52.#13: // клавиша<enter></enter>

53.if StringGridl.Col < StringGridl. ColCount - 1

54.then StringGridl.Col := StringGridl.Col + 1;

55.' . ' , ' , ' : // разделитель целой и дробной частей числа

56.begin

57.if Key DecimalSeparator then

58.Key := Decimal Separator; // заменим разделитель

59.// на допустимый

60.if Pos (StringGridl . Cells [StringGridl . Col, 0] ,

61.DecimalSeparator) о О

62.then Key := Chr(O); // запрет ввода второго

63.// разделителя

64.end;

65.'- ' : // минус можно ввести только' первым символом,

66.// т. е. когда ячейка пустая

67.if Length(StringGridl.Cells[5tringGridl.Col,0] ) о О

68.then Key := Chr(O) ;

69.else // остальные символы запрещены

70.key := Chr[0) ;

71.end; end; end.