Практическая работа № 9, МАТРИЦА

Постановка задачи

Создайте программу, которая в зависимости от величин N (количество строк) и M (количества столбцов) создает матрицу размером NxM. Программа предоставляет возможность заполнить матрицу с помощью случайных чисел или ввести значения вручную. Кроме этого можно подсчитать сумму элементов матрицы, определить максимальный и минимальный элементы матрицы.

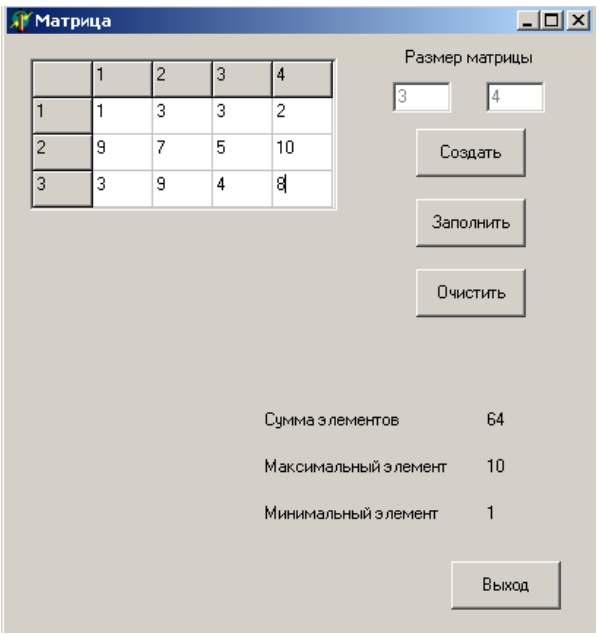


Рис.1

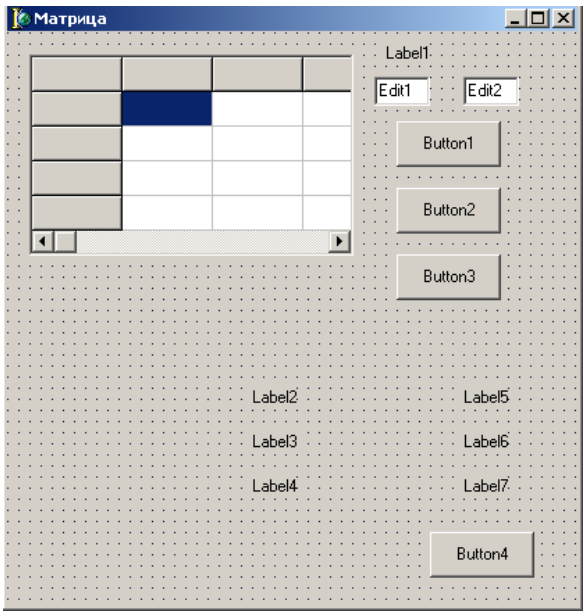


Рис.2

Новым в этой работе являются:

- компонент таблица строк **StringGrid** вкладки палитры компонентов **Additional**.

План разработки программы

1. Откройте новый проект.

2. Разместите в форме объекты в соответствии с рис.2.

3. Установите свойства компонент на вкладке **Object Inspector**:

Выделенный объект	Имя свойства	Значение
Label1	Caption	Размер матрицы
Edit1	Text	Удалить название объекта
Edit2	Text	Удалить название объекта
Button1	Caption	Создайте
Button2	Caption	Заполнить
Button3	Caption	Удалить название объекта
Label2	Caption	Сумма элементов
Label3	Caption	Максимальный элемент

Label4	Caption	Минимальный элемент
Label5	Caption	Удалить название объекта
Label6	Caption	Удалить название объекта
Label7	Caption	Удалить название объекта
Button4	Caption	Выход
StringGrid1	Name	MATR

4. Сохраните код программы и проект под именами, например, **Unit_M.pas** и **Pr_M.dpr**.

Немного теории

Компонент **StringGrid** (вкладка палитры компонентов **Additional**) предназначен для создания таблицы.

Свойству компонента **Cells** соответствует двухмерный массив ячеек, каждая из которых может содержать произвольный текст. Содержание ячейки массива, находящегося на пересечении столбца с номером **Col** и строки с номером **Row** определяется элементом **StringGrid 1.Cells [Col, Row]**. Это содержимое можно редактировать.

Двумерный массив состоит из двух частей: фиксированной и рабочей.

Фиксированная часть служит для показа заголовков столбцов (строк) и для ручного управления их размерами. Обычно фиксированная часть занимает крайний левый столбец и самую верхнюю строку таблицы. С помощью свойств **FixedCols** и **FixedRows** можно задать другое количество фиксированных столбцов и строк. Если свойства **FixedCols** и **FixedRows** имеют значение 0, то таблица не содержит фиксированной зоны.

Рабочая часть – это остальная часть таблицы. Она может содержать произвольное количество столбцов и строк, которое можно изменять в ходе выполнения программы. Если рабочая часть не умещается целиком в пределах окна компонента, то автоматически появляются полосы прокрутки. При прокрутке рабочей области фиксированная область не исчезает.

В таблице приведен перечень часто используемых свойств компонента **StringGrid**.

ColCount	Количество столбцов таблицы
RowCount	Количество строк таблицы
FixedCols	Количество столбцов фиксированной зоны
FixedRows	Количество строк фиксированной зоны
DefaultRowHeight	Высота строки таблицы
Height	Высота всей таблицы
DefaultColWidth	Ширина столбца таблицы
Width	Ширина всей таблицы
Options.goEditing	Признак редактирования содержимого ячеек таблицы. True – редактирование разрешено, False – запрещено
GridLineWidth	Ширина линий, ограничивающих ячейки таблицы
Font	Шрифт, используемый для отображения содержимого ячеек таблицы
Options .goEditing	Признак допустимости редактирования содержимого ячеек таблицы. True – редактирование разрешено, False – запрещено
Options .goTabs	Разрешает (True) или запрещает (False) использование клавиши «Tab» для перемещения курсора в следующую ячейку таблицы

5. Разместите в блоке реализации после слова **implementation** описание переменных:

```
Var
    N, M: Integer;    //N-количество строк, M-количество столбцов
    MATR_MAX,         //Значение максимального элемента таблицы
    MATR_MIN,         //Значение минимального элемента таблицы
    MATR_SUM: Integer;//Значение суммы элементов таблицы
```

6. Основная задача проекта – создать таблицу, размер которой будет определен после заполнения полей **Edit1** и **Edit2**.

Создадим следующие процедуры обработки событий:

Выделенный объект	Имя событие	Текст процедуры

Button4 «Выход»	OnClick	`procedure TForm1.Button4Click(Sender: TObject);`
		`begin`
		`Close;`
		`end;`

Form1	OnCreat	`procedure TForm1.FormCreate(Sender: TObject);`
		`MATR.Visible:=False;`
		`Button2.Enabled:=False;`
		`Button3.Enabled:=False;`
		`end;`
		Комментарий
		При создании формы установим компонент **StringGrid** невидимым (новое имя этого компонента **MATR**), т.к. в начале неизвестно сколько же строк и столбцов в таблице. Кроме этого до определения размера таблицы установим недоступными кнопки «Заполнить» и «Очистить».

Button1 «Создать таблицу»	OnClick	`procedure TForm1.Button1Click(Sender: TObject);`
		`Var I,J:Byte;`
		`begin`
		`If (Edit1.Text<>'') and (Edit2.Text<>'') Then`
		`begin`
		`Edit1.Enabled:=False;`
		`Edit2.Enabled:=False;`
		`Button1.Enabled:=False;`
		`Button2.Enabled:=True;`
		`Button3.Enabled:=True;`
		`N:=StrToInt(Edit1.Text);`
		`M:=StrToInt(Edit2.Text);`
		`MATR.DefaultColWidth:= 40;`
		`MATR.RowCount:=N+1;`
		`MATR.ColCount:=M+1;`
		`MATR.Height:=(MATR.DefaultRowHeight+2)*(N+1);`
		`MATR.Width:= (MATR.DefaultColWidth +2)*(M+1);`
		`MATR.Visible:=True;`
		`For I:=1 to N do`
		`MATR.Cells[0,I]:=IntToStr(I);`
		`For J:=1 to M do`
		`MATR.Cells[J,0]:=IntToStr(j);`

		end; end; **Комментарий** Данная процедура будет выполняться только при условии, что введены размеры матрицы, т.е **Edit1.Text** и **Edit2.Text** не являются «пустым символом». В начале устанавливаются недоступными компоненты ввода, определяющие размер таблицы, и кнопка «Создать», а недоступные ранее кнопки «Заполнить» и «Очистить» становятся доступными. После этого переходим к формированию таблицы. Для этого устанавливаем значения переменных N и M, определяющих количество строк и столбцов таблицы – переводим из текстовой информации значения полей **Edit1** и **Edit2** в числовые значения. После этого программно задаем свойства компонента **StringGrid** (**MATR**) и устанавливаем его видимым. В конце заполняем фиксированную часть таблицы значениями номеров строк и столбцов.

Button2 «Заполнить таблицу»	OnClick	procedure TForm1.Button2Click(Sender: TObject);
		Var I,J:Byte;
		begin
		Randomize;
		For I:=1 to N do
		For J:=1 to M do
		MATR.Cells[J,I]:=IntToStr(Random(N*M));
		
		MATR_MAX:= StrToInt(MATR.Cells[1,1]);
		MATR_MIN:= StrToInt(MATR.Cells[1,1]);
		MATR_SUM:=0;
		For I:=1 to N do
		For J:=1 to M do
		begin
		MATR_SUM:=MATR_SUM+StrToInt(MATR.Cells[J,I]);
		IF StrToInt(MATR.Cells[J,I])>MATR_MAX Then
		MATR_MAX:=StrToInt(MATR.Cells[J,I]);
		IF StrToInt(MATR.Cells[J,I])<MATR_MIN Then
		MATR_MIN:=StrToInt(MATR.Cells[J,I]);
		end;
		Label5.Caption:=IntToStr(MATR_SUM);
		Label6.Caption:=IntToStr(MATR_MAX);
		Label7.Caption:=IntToStr(MATR_MIN);
		end;
		Комментарий
		В начале процедуры заполняются ячейки таблицы (MATR.Cells[J,I]) случайными числами в диапазоне от 0 до (N*M-1). Дальше определяются максимальное и минимальное числа и сумма элементов таблицы. В конце процедуры полученные значения выводятся на экран. Для этого используются компоненты **Label5**, **Label6**, **Label7**.
		Для заполнения ячеек таблицы как положительными, так и отрицательными

		числами, например, в диапазоне от -10 до 10 можно записать: `MATR.Cells[J,I]:=IntToStr(Random(11))-10;`

Button3 «Очистить таблицу»		`procedure TForm1.Button3Click(Sender: TObject);`
		`Var I,J:Byte;`
		`begin`
		`For I:=1 to N do`
		`For J:=1 to M do`
		`MATR.Cells[J,I]:= '';`
		`Label5.Caption:= '';`
		`Label6.Caption:= '';`
		`Label7.Caption:= '';`
		`Edit1.Text:= '';`
		`Edit2.Text:= '';`
		
		`Matr.Visible:=False;`
		`Button2.Enabled:=False;`
		`Button3.Enabled:=False;`
		`Button1.Enabled:=True;;`
		`Edit1.Enabled:=True;`
		`Edit2.Enabled:=True;`
		`end;`
		Комментарий
		Все ячейки таблицы заполняется пустой строкой, а затем очищаются компоненты **Label5**, **Label6**, **Label7**, **Edit1.Text**, **Edit2.Text**. Компонент **StringGrid** (**MATR**) устанавливается невидимым, кнопки «Заполнить» и «Очистить» становятся недоступными, кнопка «Создать» и компоненты **Edit1** и **Edit2** – доступными.

7. Программа готова. Но если в поля ввода «Размерность матрицы» мы по ошибке вместо числа введем букву или какой-либо другой символ, то программа будет прервана. Для того чтобы это избежать создадим процедуры обработки события **KeyPress** для компонента **Edit1**.

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key:Char);
begin
case Key of
  '0'..'9': ;
  #8:       ;
  #13: Edit2.SetFocus;
  else Key := Chr(0);
end;
```

Комментарий

В зависимости от нажатой клавиши программа выполнит следующие действия:

- нажата любая цифровая клавиша или клавиша «Back Space» (код клавиши #8) – программа ничего не изменит,
- нажата клавиша «Enter» (код клавиши #13) – курсор перейдет в окно редактора ввода **Edit2**,
- во всех остальных случаях – введенный символ будет удален (присвоение значения пустого символа Chr(0)).

Для компонента **Edit2** самостоятельно создайте процедуру, обрабатывающую ввод. Она будет отличаться тем, что при нажатии клавиши «Enter», курсор должен перейти на кнопку «Создать таблицу» (**Button1**).

8. Созданная программа не позволяет редактировать элементы таблицы. Внесем изменения в свойства компонента **StringGrid** (**MATR**). Для этого на вкладке **Object Inspector** свойству **Options.goEditing** установим значение **True**. Процедура, обрабатывающая нажатие клавиши в поле компонента **StringGrid** (**MATR**) будет выглядеть так:

```
procedure TForm1.MATRKeyPress(Sender: TObject; var Key: Char);
Var I,J:Byte;
begin
 case Key of
  '0'..'9':  ;
  #8:        ;
  #13: if MATR.Col<MATR.ColCount-1 then MATR.Col:=MATR.Col+1;
  else Key:=Chr(0);
 end;
MATR_MAX:= StrToInt(MATR.Cells[1,1]);
MATR_MIN:= StrToInt(MATR.Cells[1,1]);
MATR_SUM:=0;
For I:=1 to N do
 For J:=1 to M do
  begin
  MATR_SUM:=MATR_SUM+StrToInt(MATR.Cells[J,I]);
  IF StrToInt(MATR.Cells[J,I])>MATR_MAX Then
MATR_MAX:=StrToInt(MATR.Cells[J,I]);
  IF StrToInt(MATR.Cells[J,I])<MATR_MIN Then
MATR_MIN:=StrToInt(MATR.Cells[J,I]);
  end;
Label5.Caption:=IntToStr(MATR_SUM);
Label6.Caption:=IntToStr(MATR_MAX);
Label7.Caption:=IntToStr(MATR_MIN);

end;
```

Комментарий

Обработка нажатия допустимой клавиши рассмотрена в п.7. Отличие заключается в действии, которое произойдет, если нажата клавиша «Enter» - переход к следующей ячейке в данной строке, если эта ячейка не последняя в строке.

Дальше в процедуре идет подсчет максимального, минимального элемента и суммы элементов в таблице, после внесения изменений.

9. Сохраните проект окончательно, запустите и протестируйте его.

Задание для самостоятельного выполнения

1. Дополните программу, вставив блок определения суммы по каждому столбцу матрицы.

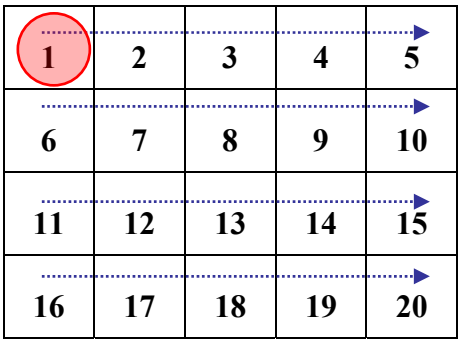
Подсказка. Необходима еще одна таблица (компонент **StringGrid**), в которой будут находиться подсчитанные суммы по столбцам. Формировать эту таблицу нужно в момент формирования основной таблицы.

Продолжение работы

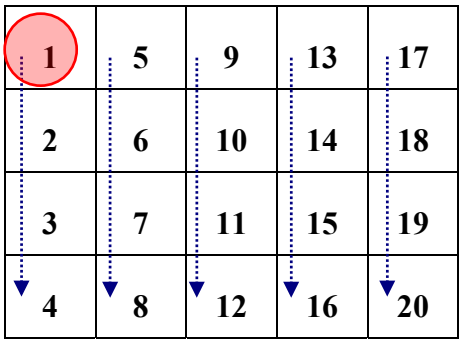
Доработать программу так, чтобы можно было бы заполнять матрицу различными способами:

на оценку «4»

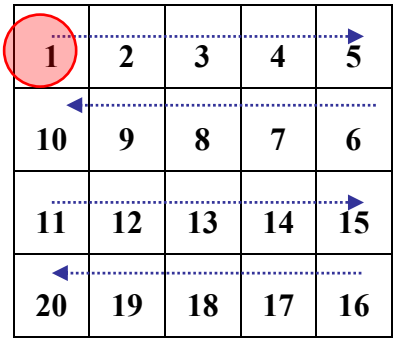
1 способ



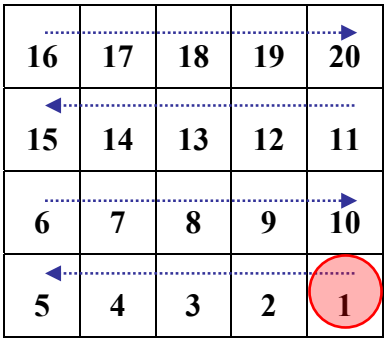
2 способ



3 способ

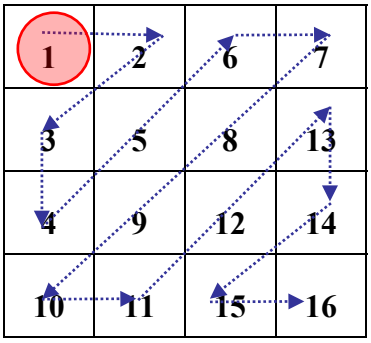


4 способ

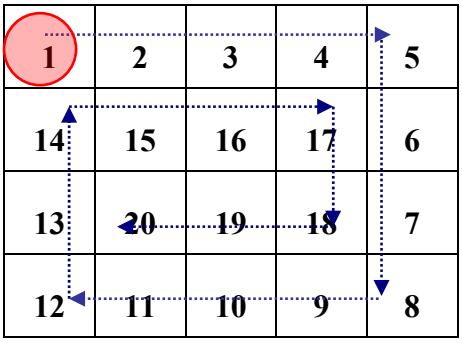


на оценку «5»

5 способ



6 способ



Подсказка.

Удалите кнопку *Заполнить таблицу*, добавив отдельные процедуры для заполнения каждым способом.

Вначале необходимо внести изменения в раздел объявления:

```
Const N=15; M=15;    //N-количество строк, M-количество столбцов
Type T_Mas=Array[1..N, 1..M] Of Integer; //Тип массив
Var
     Mas  :T_Mas;
     MATR_MAX,         //Значение максимального элемента таблицы
     MATR_MIN,         //Значение минимального элемента таблицы
     MATR_SUM: Integer;//Значение суммы элементов таблицы
```

Пример процедуры для заполнения с помощью случайных чилел может выглядеть так:

```
Procedure Z0 (Var MAS: T_MAS);
Var I,J:Byte;
begin
 Randomize;
 For I:=1 to N do
  For J:=1 to M do
   MAS[J,I]:=Random(N*M);

End.
```

Созданные вами процедуры должны располагаться в тексте программы сразу же после раздела объявления переменных.

Для выбора способа заполнения можно воспользоваться компонентой **ListBox.**

Для заполнения выделите объект **ListBox1**, найдите свойство **Items**, щелкните на кнопке с тремя точками, расположенной справа от него. В появившемся окне встроенного редактора **String List Editor** введите названия способов заполнения, каждый на новой строке. Например:

```
Случайными числами
По горизонтали
По вертикали
Змейкой слева
Змейкой справа
Зигзагом
Спиралью
```

Комментарий

а) Свойство **Items** содержит элементы списка.
б) Список может быть создан при создании формы или во время работы программы.
в) Свойство **ItemIndex** определяет номер элемента, выбранного из списка. Первый элемент имеет номер 0. Если не выбран ни один из элементов, то значение свойства **ItemIndex** равно – «-1».

Сохраните набранный текст в файле под именем ZAPOLN.txt. Для этого нажмите правую кнопку мыши и выберите режим **Save**. Для выхода из встроенного редактора щелкните на кнопке **OK**.

Комментарий

Просмотреть содержимое созданного текстового файла ZAPOLN.txt, можно с помощью любого текстового редактора, а также внести изменения в тестовый файл, не используя встроенный редактор Delphi.

Выполните следующие действия:

Выделенный объект	Вкладка окна **Object Inspector**	Имя свойства/ Имя события	Значение/Действие

ListBox1	Events	OnKeyPress	`if key=#13 then`
			`Case Listbox1.ItemIndex of`
			`0: Z0(MAS);`
			`1: Z1(MAS);`
			`2: Z2(MAS);`
			`3: Z3(MAS);`
			`4: Z4(MAS);`
			`5: Z5(MAS);`
			`6: Z6(MAS);`
			`end;`
			`For I:=1 to N do`
			`For J:=1 to M do`
			`MATR.Cells[J,I]:=IntToStr(MAS[I,J]);`

Задание для самостоятельного выполнения

1. Придумайте свой способ заполнения матрицы.