**Содержание**

[Задание 1 3](#_Toc59445658)

[Задание 2 7](#_Toc59445659)

[Задание 3 9](#_Toc59445660)

[Задание 4 11](#_Toc59445661)

[Задание 5 14](#_Toc59445662)

[Задание 6 16](#_Toc59445663)

[Список литературы 18](#_Toc59445664)

# Задание 1

22 Охарактеризуйте назначение системных диалогов.

В приложениях часто приходится выполнять стандартные действия: открывать и сохранять файлы, задавать атрибуты шрифтов, выбирать цвета палитры, производить контекстный поиск и замену и т.п.

Разработчики Delphi позаботились о том, чтобы включить в библиотеку простые для использования компоненты, реализующие соответствующие диалоговые окна. Они размещены на странице Dialogs. В таблице 8.1 приведен перечень этих диалогов.

Таблица 1. Системные диалоги и их фрагменты

| **Пиктограмма** | **Компонент** | **Страница** | **Описание** |
| --- | --- | --- | --- |
| http://beluch.ru/progr/100comp/gif/2~87.gif | **OpenDialog** «Открыть файл» | Dialogs | Предназначен для создания окна диалога «Открыть файл». |
| http://beluch.ru/progr/100comp/gif/2~88.gif | **SaveDialog** «Сохранить файл» | Dialogs | Предназначен для создания окна диалога «Сохранить файл как». |
| http://beluch.ru/progr/100comp/gif/2~89.gif | **OpenPictureDialog** «Открыть рисунок» | Dialogs | Предназначен для создания окна диалога «Открыть рисунок», открывающего графический файл. Начиная с Delphi 3. |
| http://beluch.ru/progr/100comp/gif/2~90.gif | **SavePictureDialog** «Сохранить рисунок» | Dialogs | Предназначен для создания окна диалога «Сохранить рисунок» — сохранение изображения в графическом файле. Начиная с Delphi 3. |
| http://beluch.ru/progr/100comp/gif/2~91.gif | **FontDialog** «Шрифты» | Dialogs | Предназначен для создания окна диалога «Шрифты» — выбор атрибутов шрифта. |
| http://beluch.ru/progr/100comp/gif/2~92.gif | **ColorDialog** «Цвет» | Dialogs | Предназначен для создания окна диалога «Цвет» — выбор цвета. |
| http://beluch.ru/progr/100comp/gif/2~93.gif | **PrintDialog** «Печать» | Dialogs | Предназначен для создания окна диалога «Печать». |
| http://beluch.ru/progr/100comp/gif/2~94.gif | **PrinterSetupDialog** «Установка принтера» | Dialogs | Предназначен для создания окна диалога «Установка принтера». |
| http://beluch.ru/progr/100comp/gif/2~95.gif | **FindDialog** «Найти» | Dialogs | Предназначен для создания окна диалога «Найти» — контекстный поиск в тексте. |
| http://beluch.ru/progr/100comp/gif/2~96.gif | **ReplaceDialog** «Заменить» | Dialogs | Предназначен для создания окна диалога «Заменить» — контекстная замена фрагментов текста. |
| http://beluch.ru/progr/100comp/gif/2~a3.gif | **FileListBox** (список файлов) | Win3.1 | Отображает список всех файлов каталога. |
| http://beluch.ru/progr/100comp/gif/2~a4.gif | **DirectoryListBox** (структура каталогов) | Win3.1 | Отображает структуру каталогов диска. |
| http://beluch.ru/progr/100comp/gif/2~a5.gif | **DriveComboBox** (список дисков) | Win3.1 | Выпадающий список доступных дисков. |
| http://beluch.ru/progr/100comp/gif/2~a6.gif | **FilterComboBox** (список фильтров) | Win3.1 | Выпадающий список фильтров для поиска файлов. |
| http://beluch.ru/progr/100comp/gif/2~b5.gif | **DirectoryOutline** (дерево каталогов) | Samples | Пример компонента, используемого для отображения структуры каталогов выбранного диска. |

Последние четыре компонента в таблице 1 являются не законченными диалогами, а их фрагментами, позволяющими строить свои собственные диалоговые окна.

Все диалоги являются невизуальными компонентами, так что место их размещения на форме не имеет значения. При обращении к этим компонентам вызываются стандартные диалоги, вид которых зависит от версии Windows и настройки системы. Так что при запуске одного и того же приложения на компьютерах с разными системами диалоги будут выглядеть по-разному. Например, при русифицированной версии Windows все их надписи будут русскими, а при англоязычной версии надписи будут на английском языке.

Основной метод, которым производится обращение к любому диалогу, — **Execute**. Эта функция открывает диалоговое окно и, если пользователь произвел в нем какой-то выбор, то функция возвращает **true**. При этом в свойствах компонента-диалога запоминается выбор пользователя, который можно прочитать и использовать в дальнейших операциях. Если же пользователь в диалоге нажал кнопку Отмена или клавишу Esc, то функция **Execute** возвращает **false**.

# Задание 2

60 Опишите технологию OLE.

OLE (Object Linking and Embedding – Связывание и Внедрение Объектов) – это технология, разработанная компанией Microsoft , реализующая механизм , дающий возможность вставить в приложение документ, подготовленный в другом приложении. OLE в свою очередь разработана на основе технологии COM (Component Object Model). COM –это спецификация, созданная для описания структуры COM-объектов. COM-объекты могут использоваться в любых языках программирования, вне зависимости от того, какая программная среда применялась при их создании.

Приложение, в которое можно вставить данные из другого приложения, называется клиентом OLE, а приложение-поставщик данных –сервером OLE. Клиент может обратиться к доступному серверу OLE для выполнения такой операции, которую он сам выполнить не может.

Например, если нам доступен OLE- Сервер Microsoft Word, предоставляющий услугу отобразить документ в формате \*.doc, то можно вызвать этот сервер из своего приложения и отобразить документ Word его средствами, без необходимости реализовать эту функцию самим.

Механизм OLE может действовать двумя способами: -Внедрение (embedding). Внедрённый документ становится частью того документа, в который он вставляется.- Связывание (linking). Связанный объект в приложении представляет собой не сам документ, а только ссылку на него.

Компонент OLE ContainerОсновным компонентом для работы с OLE является OLEContainer (находится на вкладке System) . Этот компонент предоставляет приложению возможность связывать и внедрять объекты, подготовленные сервером OLE.

Когда пользователь активирует объект, помещённый в контейнер OLE, управление переходит к приложению-серверу OLE, функциональность которого становится доступна из приложения, содержащего контейнер.Контейнер OLE позволяет вставить данные из любого доступного OLE-сервера: текстовый документ Word или WordPad, таблицу Excel, точечный рисунок Paint, звук WAV и т.п.. Набор доступных серверов зависит от установленного на конкретном компьютере программного обеспечения.

Обычный способ работы с компонентом OleContainer состоит в том, что в ответ на требование пользователя открыть объект (выраженное, например, нажатием кнопки) приложение вызывает метод TOleContainer.InsertObjectDialo.

Этот Метод открывает диалоговое окно, содержащее список типов встраиваемых объектов, поддерживаемых системой в данный момент. Очевидно, что этот список зависит от набора доступных OLE - серверов. Начиная с версии 5 в Delphi на палитре компонентов появилась вкладка Servers, на которой находится более трёх десятков компонентов, предназначенных для встраивания документов, подготовленных конкретными серверами OLE – приложениями Microsoft Office.

Эти компоненты облегчают разработчику задачу управления офисными приложениями из своей программы по сравнению с написанием программного кода.

# Задание 3

70 Дайте определение указателя. Опишите базовые операции с указателями в языке Паскаль.

Указатель – это переменная, которая содержит адрес другой переменной (байта памяти). Имеется два вида указателей: указатель на объект некоторого типа (типизированный) и указатель, не связанный с типом.

Для объявления типизированных указателей используется значок ^, который помещается перед соответствующим типом:

*Type T = ^T1;*

*Var A :T;*

*где: T – имя типа;*

*T 1 - базовый тип (любой в т.ч. указатель);*

*^ - указатель.*

Примеры:

*Var*

*a:byte; {выделение памяти для переменной где хранится ее значение}*

*a;^byte; {выделение памяти для переменной где хранится ее адрес}*

*p1:^integer;*

*p2, p3:^real;*

Для объявления переменных не связывая их, с каким либо типом данных можно использовать указатель без типа (pointer).

*Var*

*p:pointer;*

где: pointer - не типизированный указатель, который занимает в памяти 4 байт (2-байта сегмент, 2байта смещение.).

Для указателей допустимы операции сравнения и присваивания.

Присваивание. Указателю можно присвоить содержимое другого указателя того же самого типа или константу NIL – пустой, или адрес объекта с помощью функции ADDR или оператора @.

Процедуры и функции для работы с указателями и адресами в Паскале:

Функции:

* ADDR(X) – результат POINTER, в котором содержится адрес аргумента. (X –имя любой переменной, процедуры или функции).
* OFS(X):WORD – возвращает значение смещения адреса объекта X.
* SEG(X):WORD – возвращает значение сегмента адреса объекта X.
* CSEG(X):WORD – возвращает текущее значение регистра Cs.
* DSEG(X):WORD – возвращает текущее значение регистра Ds.
* SSEG(X):WORD – возвращает текущее значение регистра Ss.
* SPRT(X):WORD - возвращает текущее значение регистра Sp.
* PRT(SEG,OFS) – преобразует отдельно заданные значение сегмента и смещения к типу указателя.
* MAXAVAIL:LONGINT - возвращает размер наибольшего непрерывного участка кучи.
* MEMXAVAIL:LONGINT - возвращает размер общего свободного пространства кучи.

Процедуры:

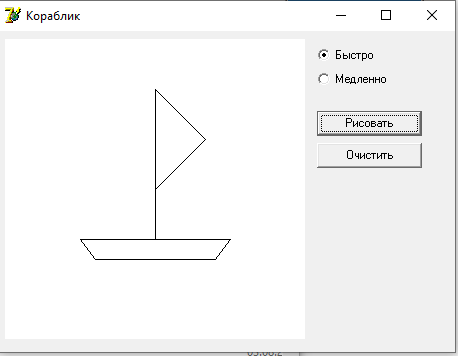
* DISPOSE(TP:POINTER) – уничтожает динамическую переменную и возвращает в кучу фрагмент динамической памяти, который был зарезервирован указателем.
* NEW(TP:POINTER) – резервирует фрагмент кучи для размещения переменной.
* GETMEM(P:POINTER; ZIZE:WORD) –выделяет из кучи блок заданного размера и адрес его начала присваивает указателю.
* FREEMEM(P:POINTER; ZIZE:WORD) – освобождает блок заданного размера..
* MARK(P:POINTER) – запоминает текущую вершину кучи (адрес начала свободного участка).
* RELEASE(P:POINTER) – освобождает участок кучи от адреса с P до конца.

# Задание 4

113 На канве компонента Image при нажатии кнопки «Рисовать» построите изображение кораблика. Использовать рисование по точкам. При помощи компонентов RadoButton задавать способ вывода изображения – мгновенно, замедленно. Для замедления использовать компонент Timer.

Первым делом построим форму. Для этого перенесем на нее 2 радиокнопки, 2 обычные кнопки и изображение.

Форма приложения



После создания интерфейса переходим к написанию кода. Для отображения кораблика, необходимо по точкам вычитать местоположение линий. Поскольку отсчет введется с левого верхнего угла, а размер изображения 300 на 300, получилось, что рассчитать нужно 9 точек, чтоб рисовать не отрывая пера от листа.

Для рисования используем обычный массив на 8 элементов (девятая точка, это точка начала рисования). Для мгновенного рисования просто проходим все элементы.

Для рисования медленно по линиям, используем компонент таймер. Отсчет у таймера – каждые 1000мс. Для правильного отображения создаем флаг isDrawed, на который будем ориентироваться. При выборе медленного рисования, активируем его. Событие каждую секунду проверяет, активно ли оно. И если активно, рисует следующую часть корабля. Если нет – ничего не рисует.

Код программы

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, ExtCtrls;

type

TForm1 = class(TForm)

Image1: TImage;

RadioButton1: TRadioButton;

RadioButton2: TRadioButton;

Button1: TButton;

Button2: TButton;

Timer1: TTimer;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Timer1Timer(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

iAll:integer;

isDrawed:bool;

arr:array[1..8] of array[1..2] of integer;

implementation

{$R \*.dfm}

//paint

procedure TForm1.Button1Click(Sender: TObject);

var

i:integer;

begin

//ship

arr[1][1]:=200; arr[1][2]:=100;

arr[2][1]:=150; arr[2][2]:=50;

arr[3][1]:=150; arr[3][2]:=200;

arr[4][1]:=75; arr[4][2]:=200;

arr[5][1]:=90; arr[5][2]:=220;

arr[6][1]:=210; arr[6][2]:=220;

arr[7][1]:=225; arr[7][2]:=200;

arr[8][1]:=150; arr[8][2]:=200;

// start point

Image1.Canvas.MoveTo(150,150);

//fast drawing

IF RadioButton1.Checked then

for i:=1 to 8 do

begin

Image1.Canvas.LineTo(arr[i][1],arr[i][2]);

end

ELSE

//slow drawing

begin

//

iAll:=1;

isDrawed:=true;

end;

end;

// clear

procedure TForm1.Button2Click(Sender: TObject);

begin

Image1.Canvas.Rectangle(0,0,500,500);

end;

//slow drawing

procedure TForm1.Timer1Timer(Sender: TObject);

begin

// если сказано рисовать

if isDrawed then

//рисуем линию

begin

Image1.Canvas.LineTo(arr[iAll][1],arr[iAll][2]);

iAll:=iAll+1;

if iAll>8 then

begin

iAll:=1;

isDrawed:=false;

end;

end;

end;

end.

# Задание 5

131 Введите строку и букву. Вывести, сколько раз буква встречается в строке. Добавить меню дублирующее кнопки.

Для решения задачи необходимо создать форму и разместить на ней два компонента Edit, три Button, три Label, один MainMenu. В инспекторе объектов надо изменить их свойства:

У компонентов Label в свойстве Caption задать необходимую подпись.

У компонентов Edit в свойстве Text удалить все.

У компонентов Button в свойстве Caption задать необходимую подпись – Найти, Очистить, Выход.

Для написания только одной буквы в второе текстовое поле, в его свойствах была выбрана максимальный размер поля 1 символ.

Двойным щелчком по компоненту Button1 был вызван редактор кода, в нем написан следующий код для решения задачи:

//find

procedure TForm1.Button1Click(Sender: TObject);

var s,s2:string;

i,n:integer;

begin

s:=Edit2.Text;//Запоминаем символ

s2:=Edit1.Text; //Запоминаем текст

if(Length(s2)<1) or (Length(s)<1) then Exit;

n:=0;//Обнуляем счетчик символов

for i:=1 to Length(s2) do//перебираем номера символов в строке

if s[1]=s2[i] then n:=n+1; //Если текущий символ равен искомому то увеличиваем счетчик

Label3.Caption:='Символ ' + s[1]+ ' встречается '+inttostr(n)+ ' раз';//вывод результата

end;

Двойным щелчком по компоненту Button2 был вызван редактор кода, в нем написан следующий код для очистки:

//clear

procedure TForm1.Button2Click(Sender: TObject);

begin

Edit2.Text:='';

Edit1.Text:='';

Label3.Caption:='';

end;

//exit

procedure TForm1.Button3Click(Sender: TObject);

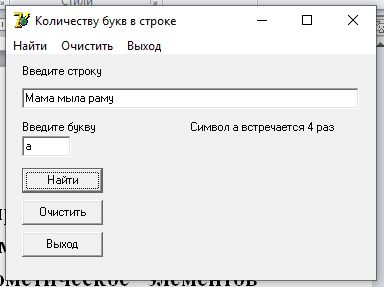
begin

form1.Close;

end;

Двойным щелчком по компоненту MainMemu1 был вызван редактор меню, в нем выделен первый пункт и в инспекторе объектов на закладке Events для события onClick выбран Button1Click. Аналогично для остальных.

Результат выполнения

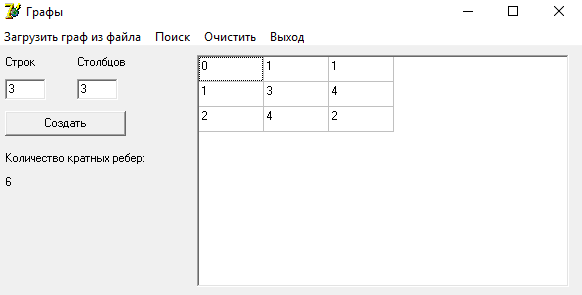


# Задание 6

156 Посчитайте количество кратных ребер в графе, заданном матрицей смежности вершин.

В приложении Delphi 7 создаем новый проект Application. Из раздела компонентов переносим нужные нам в нужном количестве, а именно: Button, Edit, Label, MainMenu, OpenDialog, String grid. Расставляем их.

Получилась следующая форма



Далее переходим к написанию кода. Для написания событий щелчков мыши по кнопкам, два раза нажимаем на нужную кнопку и пишем код.

Для чтения матрицы смежности из файла, текстовый файл должен следовать следующим правилам:

1) Каждое число должно идти с новой строки

2) Первая строка файла является количеством строк матрицы

3) Вторая строка файла является количеством столбцов матрицы

4) Все последующие строки должны представлять значения матрицы слева-направо сверху-вниз без пробелов, каждое с новой строки.

Пример кода, отвечающий за чтение из фала, отражен ниже.

procedure TForm1.N1Click(Sender: TObject);

var

f:textfile;

temp, x, y: integer;

tempstr: string;

begin

If OpenDialog1.Execute then

begin

assignfile(f, OpenDialog1.FileName);

reset(f);

readln(f, temp);

stringgrid1.colcount := temp;

readln(f, temp);

stringgrid1.rowcount := temp;

for X := 0 to stringgrid1.colcount - 1 do

for y := 0 to stringgrid1.rowcount - 1 do

begin

readln(F, tempstr);

stringgrid1.cells[x, y] := tempstr;

end;

closefile(f);

end;

end;

Пример кода, отвечающего за поиск кратных ребер графа представлен ниже

procedure TForm1.N7Click(Sender: TObject);

var

countOrientReb,i,j,tempFirstIntValue,tempSecondIntValue:Integer;

ArrayLengthRow, ArrayLengthCol: Integer;

begin

countOrientReb:=0;

getArrayFromTable();

ArrayLengthRow := Length(arrMatrix);

ArrayLengthCol := Length(arrMatrix[0]);

for i:=0 to ArrayLengthRow-1 do

for j:=0 to ArrayLengthCol-1 do

begin

tempFirstIntValue:=strToInt(stringgrid1.cells[i, j]);

tempSecondIntValue:=strToInt(stringgrid1.cells[j,i]);

if (i<>j) and

(tempFirstIntValue>0) and

(tempSecondIntValue>0) then

countOrientReb:=countOrientReb+1;

end;

Label4.Caption:=IntToStr(countOrientReb);

end;

# Список литературы

1. 100 Компонентов Delphi [Электронный ресурс]. – Режим доступа: http://beluch.ru/progr/100comp/8\_1.htm. – Дата доступа: 21.12.2020.
2. Средства технологии OLE в Delphi [Электронный ресурс]. – Режим доступа: https://studylib.ru/doc/3766812/sredstva-tehnologii-ole-v-delphi. – Дата доступа: 21.12.2020.
3. Указатели в Паскале [Электронный ресурс]. – Режим доступа: https://life-prog.ru/view\_algoritmleng.php?id=122. – Дата доступа: 21.12.2020.
4. Графика в Delphi [Электронный ресурс]. – Режим доступа: http://www.delphi-manual.ru/drawing.php. – Дата доступа: 21.12.2020.
5. Массивы [Электронный ресурс]. – Режим доступа: https://www.bestprog.net/en/2016/09/29/arrays/. – Дата доступа: 21.12.2020.
6. Компонент StringGrid [Электронный ресурс]. – Режим доступа: http://www.delphi-manual.ru/stringgrid.php?com=yes. – Дата доступа: 21.12.2020.
7. Очистка таблиц [Электронный ресурс]. – Режим доступа: http://www.stringgrid-delphi.ru/cleartable.php. – Дата доступа: 21.12.2020.
8. Работа с функцией Mod [Электронный ресурс]. – Режим доступа: https://delphisources.ru/pages/faq/faq\_delphi\_basics/Mod.php.html. – Дата доступа: 21.12.2020.