ДОДАТКОВІ МАТЕРІАЛИ ДО УРОКІВ

Урок № 7. Програмне забезпечення комп'ютера

Сумісність програмного забезпечення

Проблема сумісності програмного забезпечення зумовлена перш за все розмаїттям операційних систем. Повністю розв'язати її дуже складно. Проте в деяких випадках її вдається подолати.

- Багато популярних програм випускаються одночасно для різних операційних систем. Наприклад, редактор векторної графіки Inkscape можна завантажити з офіційного сайту у варіантах для Windows, macOS і Linux.
- Існують програми-емулятори, які дозволяють на комп'ютері з однією операційною системою запускати програми, розроблені для іншої ОС. Наприклад, встановивши під ОС Linux програмний пакунок Wine, користувач отримує змогу запускати багато програм (але не всі!), розроблених для ОС Windows.
- Програми для підтримки віртуальних машин дозволяють на комп'ютері з однією ОС (наприклад, Windows) створити віртуальний комп'ютер і встановити на ньому іншу ОС (наприклад, Linux). Після цього на віртуальному комп'ютері можна встановлювати і запускати всі програми, розроблені для відповідної ОС. Прикладом програми для підтримки віртуальних машин є VirtualBox від компанії Oracle.

Урок № 9. Списки в текстовому документі

Формати файлів текстових документів

Поява багатьох форматів файлів для однотипних даних має декілька причин:

— програми подібного призначення (в нашому випадку — для роботи з текстом) розробляють різні колективи, конкуруючи між собою. Вдало вибраний формат даних надає власнику деякі переваги в розробці програмного засобу;

— залежно від задачі користувачам потрібні програми різного рівня складності, а отже, й різні формати файлів:

а) для чернеткових записів — простий в опануванні текстовий редактор і, відповідно, формат, хоч і без можливості форматування та додавання ілюстрацій, зате з невеликим розміром кінцевого файла;

б) для офісних застосувань — текстовий процесор, повноцінна робота з яким вимагає тривалого навчання, а файли відповідного формату мають значний розмір;

в) для видавничого дизайну — у програмах зручно виконувати специфічні для цього виду діяльності операції, що спричинило появу нових форматів файлів.

Проблема обміну даними між різними програмами спричинила появу додаткових форматів.

Цікавий факт (із Вікіпедії). Хоча розширення .doc використане в багатьох різних версіях Word, насправді мова йде про чотири різні формати файлів:

— Word for DOS;

— Word for Windows 1 та 2; Word for Mac 4 та 5;

— Word 6 та Word 95; Word 6 for Mac;

— Word 97, 2000, 2002, 2003 та 2007; Word 98, 2001, X та 2004 for Mac.

Урок № 11. Опрацювання текстового документа, що містить різні об'єкти

Робота з символами

Кодова таблиця KOI8-U, окрім літер, цифр, розділових знаків тощо, містить добірку псевдографічних символів для побудови таблиць або простих малюнків:

Псевдографіка	KOI-8

Одним із недоліків була необхідність використання моноширинних шрифтів, у яких кожен символ має на екрані однакову ширину. Фрагмент на попередньому малюнку набрано моноширинним шрифтом Courier New. Якщо змінити шрифт на пропорційний, наприклад Times New Roman, то таблиця «руйнується»:

Псевдографіка	KOI-8	

Пізніше було розроблено шрифти, в яких замість звичних символів містяться малюнки. Наприклад, разом із системами родини Windows встановлюються шрифти Webdings і Windings. Слово «Інформатика», набране цими шрифтами, матиме такий вигляд:

	Webdings
✧⬋⇳⬊⇨⬈→⇩➔⬇→	Windings

З появою системи кодування Unicode графічні можливості шрифтів знову зросли. До стандартної таблиці включено дуже багато різноманітних символів. Наприклад, символи з шістнадцятковими кодами від 2654 до 265F зображають шахові фігури.

За посиланнями на сторінці http://www.unicode.org/charts/ доступні таблиці зі всіма символами Unicode. Ця система постійно розвивається. Наприклад, у версії 9.0, яка вийшла 2016 року, додано 7500 символів, після чого загальна їх кількість склала 128 172.

Вірш Т. Г. Шевченка «За сонцем хмаронька пливе...»

За сонцем хмаронька пливе,
Червоні поли розстилає
І сонце спатоньки зове
У синє море: покриває
Рожевою пеленою,
Мов мати дитину.
Очам любо. Годиночку,
Малую годину
Ніби серце одпочине,
З Богом заговорить…

А туман, неначе ворог,
Закриває море
І хмароньку рожевую,
І тьму за собою
Розстилає туман сивий,
І тьмою німою
Оповиє тобі душу,
Й не знаєш, де дітись,
І ждеш його, того світу,
Мов матері діти.

Урок № 12. Оформлення документів. Структура складного текстового документа

Стильове оформлення документа

У різних текстових процесорах (навіть у різних версіях одного текстового процесора) можливості роботи зі стилями можуть відрізнятись:

Текстовий процесор	Підтримувані стилі
Microsoft Word 2003	знак, абзац, таблиця
Microsoft Word 2007	знак, абзац, таблиця, список
Libre Office Writer 5.2	знак, абзац, рамка, сторінка, список

Урок № 16. Програмне забезпечення для опрацювання об'єктів мультимедіа

Програми для опрацювання відеоданих

Існує безкоштовна версія цієї програми, яку можна звантажити із сайта розробників: https://www.lwks.com/.

Ще один вільний відеоредактор із широкими можливостями вбудовано в редактор тривимірної графіки Blender. Зокрема, в ньому є можливість додавати до реального відео змодельовані у програмі спецефекти.

Урок № 37. Практична робота № 11. Створення об'єктно-орієнтованої програми, що відображає вікно повідомлення

Додаткове завдання

Запрограмувати виведення діалогового вікна MessageBox із заданим заголовком і повідомленням.

1. Діалог програми з користувачем можна організувати за допомогою методу MessageBox (рис. 1), який дозволяє викликати діалогове вікно із заданим заголовком і повідомленням.

Метод має такий формат:

MessageBox(Handle, Текст, Заголовок, Тип_вікна).

Handle — це параметр, що указує на вікно, з якого викликано вікно повідомлення. Залишимо його без змін.

Текст — текст повідомлення.

Заголовок — рядкове значення, яке задає назву вікна.

Тип_вікна — числовий вираз, який визначає кількість і типи кнопок або малюнок у вікні.

ДЕЯКІ КОНСТАНТИ ДІАЛОГУ MESSAGEBOX

ЗНАК	ЗНАЧЕННЯ	ОПИС
0	MB_OK	Кнопка OK
4	MB_YESNO	Кнопки Yes і No
	MB_ICONERROR	Знак помилки
	MB_ICONQUESTION	Знак питання
	MB_ICONWARNING	Знак оклику
	MB_ICONINFORMATION	Знак інформації

Рис. 1.

Кнопки можна задавати як константою (mb_OkCancel), так і номерними знаками (0, 1, 2 …).

Функція MessageBox повертає значення, що відповідає тій кнопці, яку було натиснуто в діалоговому вікні (рис. 2).

КОНСТАНТИ, ЯКІ ПОВЕРТАЄ ФУНКЦІЯ MESSAGEBOX

КНОПКА	ЗНАЧЕННЯ
Кнопка Yes	IDYES
Кнопка No	IDNO

Рис. 2.

Інструкція для учнів

Додайте на форму кнопку Button2 і змініть значення властивості Caption кнопки на Додаткове завдання. Створіть обробник події для кнопки і запрограмуйте виведення повідомлення:

MessageBox(Handle, 'Ти робиш успіхи!', 'Вітання', MB_OK+ MB_ICONWARNING);

Додайте до програмного коду оператор, який реалізує відображення вікна, зображеного на рис. 3.

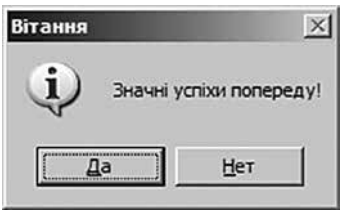


Рис. 3

2. Додайте до коду обробника події кнопки Button2 оператор:
   If MessageBox(Handle, '2*2 = 5?',
   'Приклад', 4+MB_ICONQUESTION) = IDYES Then ShowMessage('Ти помиляєшся!')
   Else ShowMessage('Ти маєш рацію.');

   Проаналізуйте дію кнопки.

Урок № 41. Практична робота № 13. Складання та виконання лінійних алгоритмів опрацювання величин у навчальному середовищі програмування

Додаткове завдання

Створити проект для переведення значення відстані, заданої в кілометрах, у милі.

Теоретичні відомості.

Миля (від лат. *mille passuum* — тисяча подвійних кроків римських солдатів) — одиниця вимірювання відстані, введена в Стародавньому Римі. Розрізняють сухопутну і морську милю:

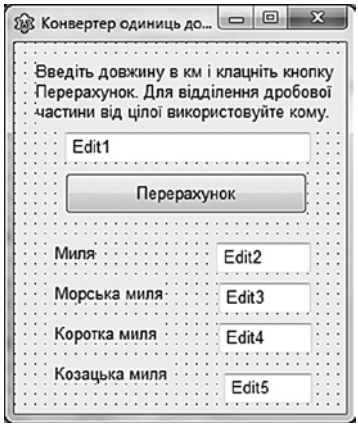
1 сухопутна миля = 1,609 км;

1 морська миля = 1,852 км.

В Україні до метричної системи були відомі короткі та довгі милі. Коротка (турецька) миля = 1,67 км; довга (козацька) = 8,35 км.

Інструкція для учнів

1. Розробіть інтерфейс проекту згідно з рисунком:



2. Створіть процедуру обробки події OnClick для кнопки Перерахунок. Запишіть код процедури:

```
var km, mil, mmil, tmil, kmil: real;
begin
    km := StrToFloat(Edit1.Text);
    mil := km/1.609;
    Edit2.Text := FloatToStr(mil);
end;
```

3. Додайте до програмного коду оператори для обчислення значень змінних mmil (морська миля), tmil (коротка миля), kmil (козацька миля) і виведення цих значень до текстових полів Edit3, Edit4, Edit5.

   Обчислення значення змінної mmil (морська миля):

```
mmil := km/1.852;
Edit3.Text := FloatToStr(mmil);
```

4. Виконайте проект і збережіть його в папці Mile.

Урок № 44. Величини логічного типу

Слово вчителя

Інформатика тримається на трьох основних китах: логіці, алгоритмах і програмах. Згадаймо етапи розвитку логіки як науки. Основи формальної логіки, науки про закони і форми людського мислення, були закладені давньогрецьким філософом Аристотелем (384–322 рр. до н. е.). Г. В. Лейбніц (1646–1716) вказав шляхи для переведення логіки «із словесного царства, повного невизначеностей, до царства математики, де відношення між об'єктами або висловлюваннями визначаються абсолютно точно». Засновником математичної логіки (булевої алгебри) є Дж. Буль (1815–1864). У XX ст. вчені об'єднали створений ним математичний апарат із двійковою системою числення, заклавши тим самим основи для розробки комп'ютера.

На цьому уроці ми розглянемо логічний тип даних, який має велике значення в програмуванні.

Урок № 45. Величини символьного типу

Символьний тип даних

Ми вводимо з клавіатури символи: буква «А», цифра «1», пробіл тощо. Комп'ютер же може оперувати тільки цифрами 0 або 1. Для того щоб обробляти символи, придумали спеціальну систему для «перекладу» інформації на зрозумілу комп'ютеру мову. Так з'явилися кодові сторінки — таблиці, що зіставляють з кожним значенням байта деякий символ.

На початку розвитку комп'ютерів (1963) була розроблена кодова сторінка ASCII (Американський кодовий стандарт для обміну інформацією). Перша ASCII-таблиця містила 255 символів. Кожному символу відповідав власний 8-бітний номер у таблиці. Якщо ми вводили англійську букву «А», то в комп'ютер потрапляв номер цього символу в таблиці — 65, або у двійковому вигляді — 100 0001. Таким чином, символи можна було порівнювати між собою. Англійське «В» знаходилося під номером 66, а, отже, воно було більше, ніж «А». Ми вводимо символи, що автоматично перетворюються в цифри, з якими вже оперує комп'ютер.

У 1991 р. був запропонований стандарт Unicode — універсальна система кодування символів, що представляє знаки практично всіх мов: символи кирилиці, китайські ієрогліфи, знаки математичних формул, музичні знаки тощо. В Unicode для кодування використовується 2 байти (16 біт), тобто таблиця містить 216 = 65 535 символів. У Lazarus використовується формат UTF-8, у якому символи мають нефіксований розмір. Символи з номером меншим, ніж 128, займають 1 байт, а символи з номером від 128 і більше можуть займати від 2 до 4 байтів. Символи кирилиці займають, наприклад, по 2 байти. Тож ланцюжок символів у 5 байтів у UTF-8 не завжди означає рядок із п'яти символів.

Урок № 48. Практична робота № 14. Налагодження готової програми.

Додаткове завдання

Створити проект Калькулятор.

Теоретичні відомості

При виконанні програми може виникнути виняткова ситуація, в результаті якої генерується помилка і виконання програми переривається. Для контролю виняткових ситуацій програміст повинен підготувати як основний варіант фрагмента коду, де можлива виняткова ситуація, так і його варіант, в якому виняткова ситуація неможлива. Синтаксис оператора контролю виняткових ситуацій:

```
try
«небезпечна» команда;
except
альтернативний варіант фрагмента;
end;
```

Контроль виняткових ситуацій спрацьовує при запуску exe — файла програми.

Інструкція для учнів

1. Розробіть інтерфейс проекту згідно з рис. 1.

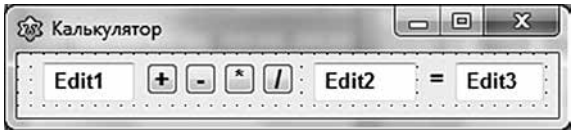


Рис. 1.

2. Опишіть змінні для збереження операндів і результату обчислення як глобальні:

```
var Form1: TForm1;
a, b, c: real;
```

3. Створіть обробник події OnClick для кнопки +.
   Запишіть код процедури:

```
a := StrToInt(Edit1.Text);
b := StrToInt(Edit2.Text);
c := a + b;
Edit3.Text := IntToStr(c);
```

4. Створіть процедури-обробники для командних кнопок – і *, вказуючи операцію, що відповідає заголовку кнопки.
5. Створіть процедуру-обробник для кнопки /.

```
var c:real; // опис локальної змінної c для збереження частки
begin
    a := StrToInt(Edit1.Text);
    b := StrToInt(Edit2.Text);
    try
        c := a/b;
        Edit3.Text := FloatToStr(c);
    Except
        ShowMessage('Дільник дорівнює 0!');
    end;
end;
```

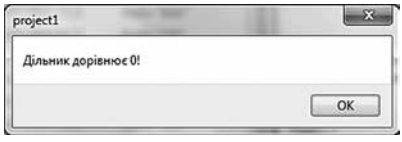


Рис. 2

6. Збережіть проект у папці Калькулятор. Скомпілюйте проект, виконавши команди Виконати → Компілювати. Запустіть на виконання файл project1.exe. Перевірте дію кнопок.

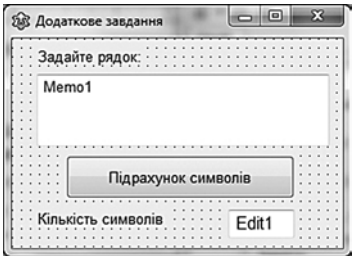
Урок № 57. Практична робота № 15. Складання та виконання алгоритмів із повтореннями та розгалуженнями для опрацювання величин

Додаткове завдання

Написати програму, яка буде підраховувати кількість символів C у введеному рядку. Символ C задається з клавіатури.

Інструкція для учнів

1. Розробіть інтерфейс проекту згідно з рисунком.



   Для введення рядка використовується компонент TMemo (вкладка Standard), призначений для роботи з багаторядковим текстом.
2. Створіть процедуру обробки події кнопки Підрахунок символів. Додайте до розділу uses модулі LCLType, LCLProc для роботи із символами кирилиці. Опишіть змінні:
   var s, sym: string; i, k: integer; c: TUtf8Char;

3. Запишіть оператор для задання значення змінної S:
   S := Memo1.Lines.Text;

4. Запишіть оператори для підрахунку кількості входжень символу c до рядка S:
   k := 0;
   for i := 1 to Utf8Length(S) do
   if Utf8Copy(S, i, 1) = c then k := k + 1;
   Edit2.Text := IntToStr(k); // виведення результатів підрахунку
   Label2.Caption := Label2.Caption +’’+ c;

5. Збережіть проект у папці Підрахунок символів.

Урок № 59. Налаштування властивостей графічних примітивів

Диктант із взаємоперевіркою

1. Опишіть систему координат для роботи з графікою.
2. Запишіть оператори для заливання елемента Image1 білим кольором.
3. Запишіть оператор для побудови трикутника з вершинами в точках (100,100), (150,100), (80,70).
4. Запишіть оператор для замальовування червоним кольором точки елемента Image1 з координатами (100,100).
5. Запишіть оператор для малювання лінії чорного кольору від позиції з координатами (20,20) до точки (100,20).
6. Запишіть оператор для малювання квадрата зі стороною 100 пікселів.

Відповіді:

2) Image1.Canvas.Brush.Color := clWhite;
   Image1.Canvas.FillRect (Image1.ClientRect);
3) Image1.Canvas.Polyline ([Point (100,100), Point(150,100), Point (80,70), Point (100,100)]);
   Image1.Canvas.Pixels [100,100] := clRed;
   Image1.Canvas.MoveTo (20,20); Image1.Canvas.LineTo (100,20);
6) Image1.Canvas.Rectangle (200,200,300,300);

Урок № 60. Створення програм із графічним відображенням даних

Робота в парах

Який малюнок буде створено в результаті виконання програмного коду? Чи можна організувати цикл для створення малюнку?

```
procedure TForm1.Button1Click(Sender: TObject);
var x, y, n: integer;
begin
    x := 100; y := 100;
    Canvas.Pen.Color := clRed;
    Canvas.Brush.Color := clRed;
    Canvas.Ellipse (x, y, 2 * x, 2 * y);
    x := 2 * x; n := 10;
    Canvas.Pen.Color := clgreen;
    Canvas.Brush.Color := clGreen;
    Canvas.Ellipse (x, y + n, x + 100 –2 * n, 2 * y – n);
    x := x +100 – 2 * n; n := 20;
    Canvas.Pen.Color := clBlue;
    Canvas.Brush.Color := clBlue;
    Canvas.Ellipse (x, y + n, x + 100 – 2 * n, 2 * y – n);
    x := x + 100 – 2 * n; n := 30;
    Canvas.Pen.Color := clMaroon;
    Canvas.Brush.Color := clYellow;
    Canvas.Ellipse (x, y + n, x + 100 – 2 * n, 2 * y – n);
end;
```

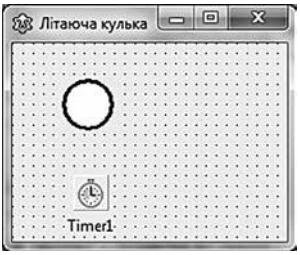
Урок № 64. Практична робота № 16. Створення програми з графічними компонентами

Додаткове завдання

Створити проект «Літаюча кулька», в якому кулька рухається по формі, відбиваючись від відповідних меж форми.

Інструкція для учнів

1. Додайте на форму компонент Shape (сторінка Additional) (див. рисунок).



Властивості Shape елемента Shape1 задайте значення stCircle. Додайте на форму компонент Timer (сторінка System).

Властивості Interval таймера задайте значення 20.

2. Для початку нехай кулька літає вліво–вправо, відбиваючись від відповідних меж форми.

Опишіть глобальні змінні:

```
var x, hx: integer;
```

У процедурі TForm1.FormCreate задайте початкове положення кульки:

```
x := Shape1.Left;
```

і крок руху:

```
hx := 10;
```

Створіть процедуру TForm1.Timer1Timer і запишіть її код:

```
x := x+hx; // зміна координати x
{перевіряємо границі і змінюємо напрям руху}
    If (x >= Form1.Width – Shape1.Width) or (x <= 0) then hx := –hx;
    Shape1.Left := x; // кулька займає нове положення
```

Додайте рух кульки по діагоналі. Для цього потрібно додати опис глобальної змінної hy; до процедури FormCreate додати оператори:

```
y := Shape1.Top;
hy := 10;
```

До процедури TForm1.Timer1Timer потрібно додати оператори (координата y визначається як Shape1.Top):

```
y := y+hy; // зміна координати y
{перевіряємо границі і змінюємо напрям руху}
    If (y >= Form1.Height – Shape1.Height) or (y <= 0) then hy := –hy;
    Shape1.Top := y; // кулька займає нове положення
```