Міністерство освіти і науки України
Чернівецький національний університет
імені Юрія Федьковича

ПРИКЛАДНЕ ПРОГРАМУВАННЯ: ВІД ТЕОРІЇ ДО ПРАКТИКИ

Навчальний посібник

Укладачі ***М.П. Горський***
А.Л. Негрич
О.В. Олар

Чернівці
Чернівецький національний університет
імені Юрія Федьковича
2021

УДК 655.03(075.8)
П 759

Рекомендовано Вченою радою
Чернівецького національного університету імені Юрія Федьковича
(протокол № 11 від 25 жовтня 2021р.)

Рецензенти:

Вихор Л.М. доктор фіз.-мат. наук, головний науковий співробітник Інституту термоелектрики НАН та МОН України;

Киричок Т.Ю. доктор техн. наук, професор, завідувачка кафедри технології поліграфічного виробництва, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

П 759 **Прикладне** програмування : від теорії до практики : навч. посібник / укл. М.П. Горський, А.Л. Негрич, О.В. Олар. – Чернівці : Чернівец. нац. ун-т ім. Ю. Федьковича, 2021. – 120 с.

У методичному посібнику наведено методичні рекомендації з предметів «Інформатика» та «Прикладне програмування». Зміст запропонованих занять відображає основи програмування на мовах Delphi та PHP.

Для студентів, які навчаються за спеціальностями 122 Комп'ютерні науки; 186 Видавництво та поліграфія.

УДК 655.03(075.8)

ЗМІСТ

1. Створення програми обрахунку математичного виразу на Delphi

1. Мета роботи: вивчення елементів TButton, TEdit та TLabel середовища Delphi.

2. Завдання до роботи:

1. Ознайомитися з середовищем Delphi та палітрою компонентів.
2. Створити форму, що містить компоненти TButton, TEdit та TLabel.
3. Створити програму, що обраховує значення виразу.
4. У звіті навести:
   1) Зображення, що ілюструють роботу програми.
   2) Код програми.

3. Обладнання та матеріали: Персональний комп'ютер, середовище Delphi.

4. Теоретична частина

4.1. Основні інструменти Delphi

Після завантаження Delphi на екрані відкриваються чотири вікна: головне вікно, вікно проектувальника форм, вікно редактора коду, вікно інспектора об'єктів.

4.1.1. Головне вікно

Головне вікно складається з трьох частин: меню, панелі інструментів і палітри компонентів (рис.1). Delphi містить чотири стандартні панелі інструментів: View, Standard, Debug, Custom. Середовище дозволяє додавати чи видаляти кнопки в панелі інструментів за допомогою команди Customize з локального меню чи панелей інструментів.

У палітрі компонентів відображаються компоненти, за допомогою яких користувач створює свої додатки. Піктограми стандартних компонентів Delphi поділені на групи, кожна з яких розташована на окремій закладці.

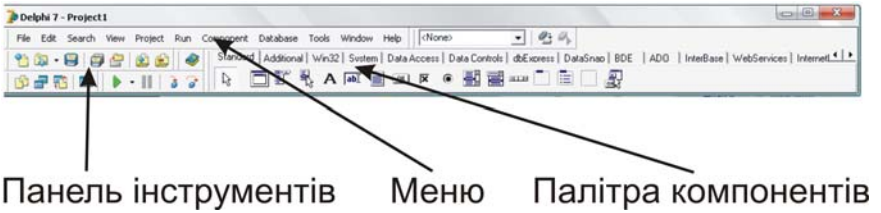


Рис.1. Головне вікно Delphi

4.1.2. Проектувальник форм Form Designer

У вікні проектувальника форм відображається форма як візуальний об'єкт (рис.2). Тут ви визначаєте, як буде виглядати ваш додаток із погляду користувача. Ви вибираєте компоненти з палітри компонентів і перетягуєте їх на форму, використовуючи мишку для точного розташування і визначення розмірів компонента. Ви можете керувати зовнішнім виглядом і поведінкою компонента за допомогою Object Inspector і Code Editor. Це, власне і є візуальне програмування.

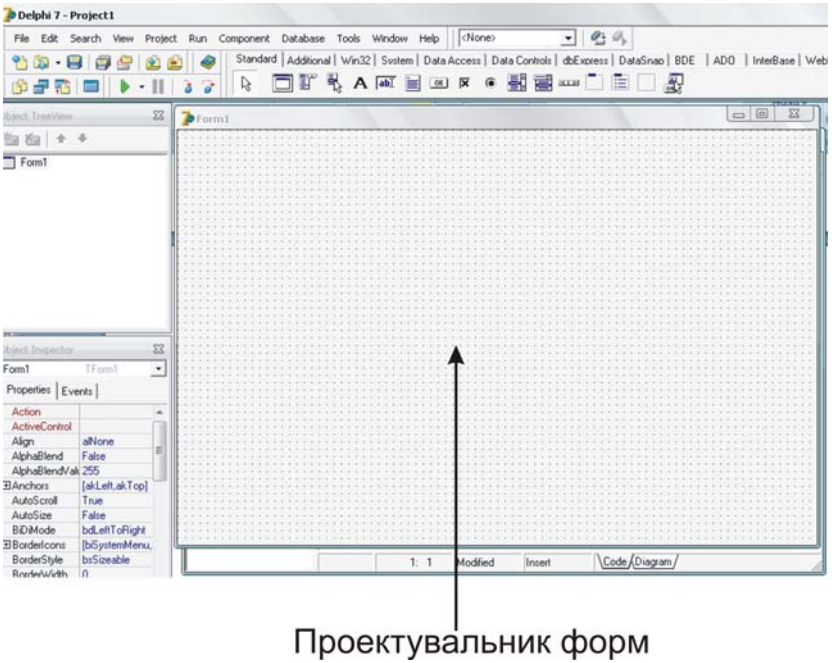


Рис.2. Проектувальник форм.

4.1.3. Інспектор об'єктів Object Inspector

З його допомогою ви можете змінювати властивості компонентів форми і визначати події, на які повинна реагувати форма чи її компоненти. Вікно інспектора об'єктів має дві закладки: Properties – властивості та Events – події (рис.3).

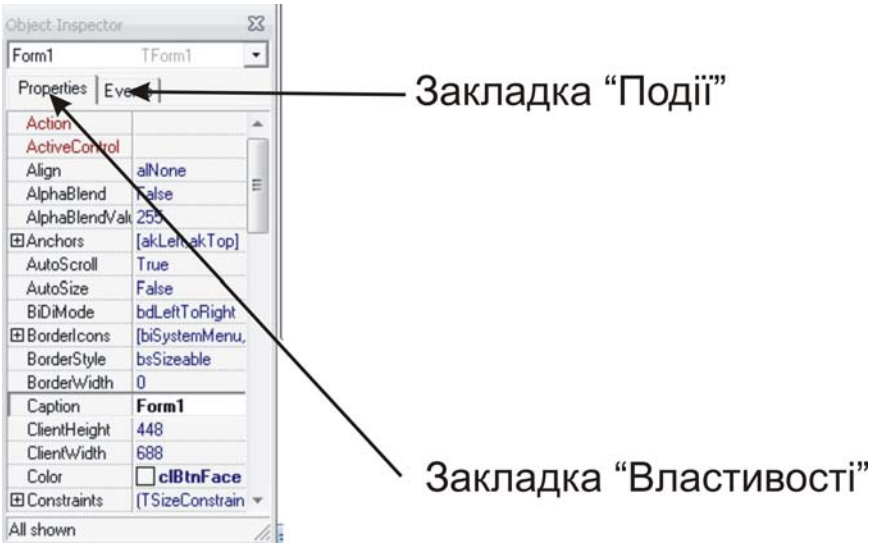


Рис.3. Інспектор об'єктів

4.1.4. Редактор коду Code Editor

У вікні редактора коду ви можете вводити розроблений вами код (програму) чи редагувати згенерований Delphi-код для компонентів розробленої форми (рис.4). Code Editor використовує технологію вкладок, кожна з яких відповідає своєму модулю чи файлу.

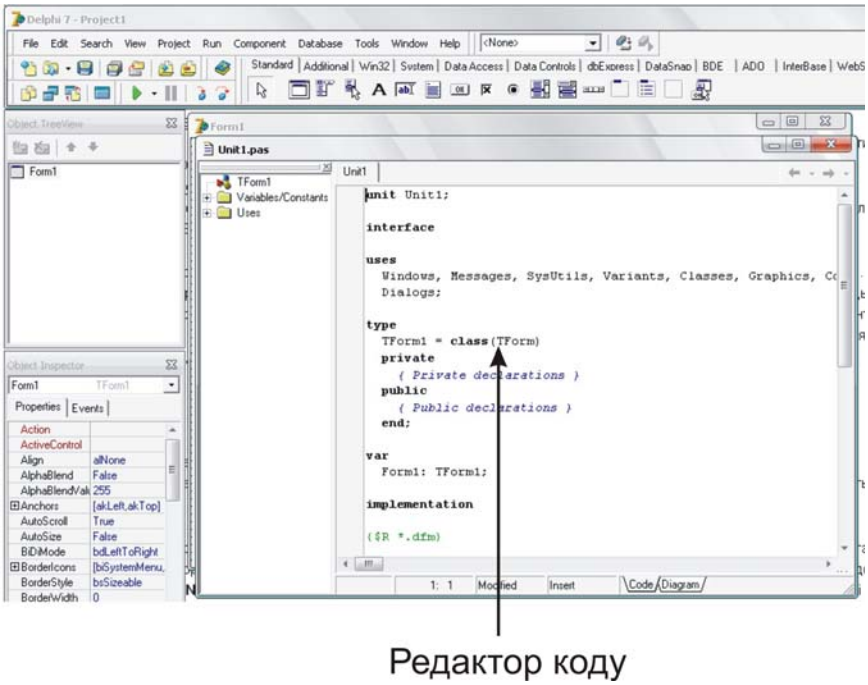


Рис.4. Редактор коду

4.1.5. Обробка подій

Події – це властивості процедурного типу, призначені для створення користувальницької реакції на ті чи інші вхідні впливи.

Усі події в Delphi прийнято іменувати з початком ”On”. Клацнувши в Інспекторі об'єктів на закладці Events у поле будь-якої події, ви одержите в програмі заготовку методу – оброблювача цієї події. При цьому його ім'я буде складатися з імені поточного компонента та імені події (без “On”), а відноситися він буде до поточної форми. Наприклад, якщо на формі розміщена кнопка (компонент TButton з ім'ям Button1), то заготовка оброблювача події OnClick (клацання) буде мати вигляд:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 end.
```

Заготовку оброблювача події OnClick можна також одержати двічі клацнувши по розміщеній на формі кнопці.

5. Порядок виконання роботи

Розробка нового додатка починається зі створення проекту. Для цього в меню File виберіть команду New Application.

Delphi створює проект, що містить три файли: Файл проекту *.dpr, Файл форми *.dfm, Файл модуля *.pas.

При цьому в проектувальнику форм (Form Designer) Ви побачите нову форму, а в редакторі коду (Code Editor) – заготовку вихідного тексту модуля, що асоційований зі створеною формою.

Файл проекту являє собою текст, схожий на файл Pascal. Переглянути файл проекту можна виконавши команду меню Project|View Sourse або Ctrl-F12 і вибрати зі списку ім'я файла проекту.

```
Program Project1;
Uses
  Forms,
    Unit ‘in Unit1.pas’{Form1};
    {$R*.RES}
begin
Application.Initialize;
Application.CreateForm(TForm1,Form1);
Application.Run;
end.
```

У секції Uses перераховані всі модулі, що входять до складу проекту. У блоці begin ... end після ініціалізації об'єкта додатка Application здійснюється створення форм, що входять до складу проекту. Метод Run здійснює запуск додатків.

При додаванні модулів і форм у додаток вони будуть з'являтися в секції Uses файлу проекту.

Автоматично згенерований код має такий вигляд :

```
unit Unit1;
interface
uses
```

```
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs;
    type
     TForm1 = class(TForm)
     private
       { Private declarations }
     public
       { Public declarations }
     end;
    var
     Form1: TForm1;
    implementation
    {$R *.DFM}
    end.
```

Текстове зображення форми можна одержати за допомогою команди View As Text контекстного меню форми.

Збережіть проект у вашому робочому каталозі, виконавши команду меню File|Save Project As.

Виберіть з палітри компонентів (сторінка Standard) компонент Edit і розмістіть його на формі.

Виберіть з палітри компонентів (сторінка Standard) компонент Label і розмістіть його на формі.

З палітри компонентів (сторінка Standard) додайте кнопку TButton у форму, виконавши подвійне клацання на піктограмі кнопки. Delphi автоматично центрує кнопку на формі.

Виконайте подвійне клацання на кнопці, що Ви додали у форму, для того щоб відкрити Code Editor. Саме тут уводяться рядки коду, що наказують необхідні дії програмі, коли хто-небудь клацне на кнопці в запущеній програмі. Знаходячись у Code Editor, у тілі процедури TForm1.Button1Click наберіть між зарезервованими словами begin і end-команди, необхідні для зчитування значення з Edit (перевірці чи є це значення допустимим), для обрахунку результату виразу та виводу результату в Label.

6. Індивідуальні завдання

Номер завдання	Вираз для обчислення
1	$\frac{\sin(x)}{(x-3)}+x$
2	$\frac{\sqrt{x}}{(x+5)}$
3	$\sqrt{x}+x^2$
4	$\cos(x)^2+\sqrt{x}$
5	$\cos(x)\sin(x)+\sqrt{x}$
6	$\cos(\lvert x\rvert)+\sqrt{x}$
7	$x\sqrt{x+5}+x$
8	$\sqrt{x+\lvert\cos(x)\rvert}$
9	$x^4+5x+\frac{8}{x}$
10	$\cos(x^2)+\frac{\sin(x)}{x}$
11	$e^x+\sqrt{x}\frac{x+5}{x-5}$
12	$\sin(\sqrt{x})+\frac{x}{x+2}$
13	$\lvert\sin(x)+\cos(x)+\sqrt{x}\rvert$
14	$\frac{x^5+x^4}{x}$
15	$\frac{\sqrt{x+5}}{10}+x^2$
16	$\lvert\sin(x)\rvert+e^x+\sqrt{x}$

Запитання для контролю

1. Де знаходяться вкладки «Властивості» та «Події», яке їх призначення?
2. Де знаходяться компоненти та «Проектувальник форм», яке їх призначення?
3. Організувати обрахунок за іншою формулою за допомогою додаткової кнопки.
4. Як створити процедуру яка виконується при натисканні кнопки?
5. Де можна оголошувати змінні і які типи змінних потрібні для виконання програми?

Список рекомендованої літератури

1. А. Хомоненко, В. Гофман Delphi 7 (2-е издание), БХВ-Петербург · 2010.
2. Н. Культин. Основы программирования в Delphi, БХВ-Петербург · 2009.
3. А. Хомоненко, В. Гофман. Самоучитель Delphi, БХВ-Петербург · 2008.
4. Абрамян М.Э. Delphi 7. Карманный справочник с примерами, Кудиц-Образ · 2006.
5. Л. Климова Delphi 7. Основы программирования. Решение типовых задач. Самоучитель, Кудиц-Образ · 2004.
6. Ю.А. Шпак Delphi 7 на примерах, BHV – Санкт-Петербург, 1997. – 176 с.
7. Галисеев Г. В. Компоненты в Delphi 7. Профессиональная работа, МЭСИ,1998.
8. Кэнту М. Delphi 7. Для профессионалов, Питер • 2004.

2. Створення програми, що здійснює обробку та сортування елементів випадково заданого масиву

1. Мета роботи: вивчення елемента TMemo та правил побудови циклу for середовища Delphi.

2. Завдання до роботи:

1. Вивчити засоби трасування програми в середовищі Delphi.
2. Створити форму, що містить необхідну кількість компонентів TButton, TEdit , TLabel та TMemo.
3. Створити програму, що генерує задану кількість елементів масиву в заданому діапазоні значень та здійснює їх подальшу обробку відповідно до індивідуального завдання.
4. У звіті навести:
    1) Зображення що ілюструють роботу програми,
    2) Код програми.
    3)

3. Обладнання та матеріали: Персональний комп'ютер, середовище Delphi.

4. Теоретична частина

4.1. Засоби трасування середовища Delphi

Помилки, які можуть бути в програмі, прийнято ділити на три групи:

- синтаксичні;
- помилки часу виконання;
- алгоритмічні.

Синтаксичні помилки, їх також називають помилками часу компіляції (помилка під час компіляції), найбільш легко виправляються. Їх виявляє компілятор, а програмісту залишається тільки внести зміни в текст програми і виконати повторну компіляцію.

Помилки часу виконання, в Delphi вони називаються винятками (Exception), теж, як правило, легко виправляються. Вони зазвичай проявляються вже при перших запусках програми і під час тестування.

При виникненні помилки в програмі, запущеною з Delphi, середовище розробки перериває роботу програми, про що свідчить розміщене в дужках слово «Зупинено» в заголовку головного вікна Delphi, і на екрані з'являється діалогове вікно, яке містить повідомлення про помилку та інформацію про тип (клас) помилки. На рис.1 наведено приклад повідомлення про помилку, що виникає при спробі відкрити неіснуючий файл.

Після виникнення помилки програміст може або перервати виконання програми, для цього треба з меню Виконати вибрати команду Program Reset, або продовжити її виконання, наприклад, по кроках (для цього з меню Виконати треба вибрати команду Step), спостерігаючи результат виконання кожної інструкції.

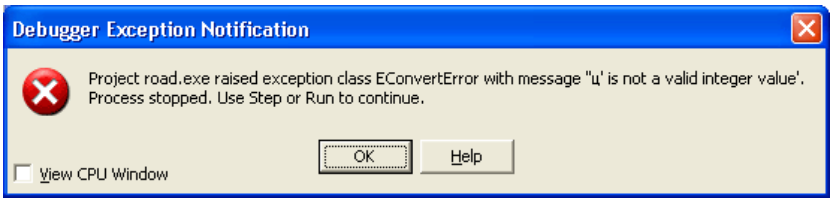


Рис. 1. Повідомлення про помилку при запуску програми з Delphi

Якщо програма запущена з Windows, то при виникненні помилки на екрані також з'являється повідомлення про помилку, але тип помилки (виключення) в повідомленні не вказується (рис. 2). Після клацання на кнопці OK програма, в якій виявилася помилка, продовжує (якщо зможе) роботу.

Рис. 2. Повідомлення про помилку при запуску програми з Windows

Алгоритмічні помилки виправляються інакше. Компіляція програми, в якій є алгоритмічна помилка, завершується успішно. При пробних запусках програма веде себе нормально, однак при аналізі

результату з'ясовується, що він неправильний. Для того, щоб усунути алгоритмічну помилку, «доводиться аналізувати алгоритм вручну», «прокручувати» його виконання.

Інтегроване середовище розробки Delphi надає програмістові потужний засіб пошуку та усунення помилок у програмі - наладник. Наладник дозволяє виконувати трасування програми, спостерігати значення змінних, контролювати виведені програмою дані.

У випадку неправильної роботи програми необхідно бачити реальний порядок виконання інструкцій. Це можна зробити, виконавши трасування програми. Трасування - це процес виконання програми по кроках (крок за кроком), інструкція за інструкцією програми. Під час трасування програміст дає команду: виконати чергову інструкцію.

Delphi забезпечує два режими трасування: без заходу в процедуру (Step Over) і з заходом у процедуру (Trace). Режим трасування без заходу в процедуру виконує трасування тільки головної процедури, при цьому трасування підпрограм не виконується, вся підпрограма виконується за один крок. У режимі трасування із заходом у процедуру виконується трасування всієї програми, тобто по кроках виконується не тільки головна програма, але і всі підпрограми.

Щоб почати трасування, необхідно з меню Виконати вибрати команду Step Over або Trace. У результаті у вікні редактора коду буде виділена перша програма інструкції. Щоб виконати виділену інструкцію, необхідно з меню «Виконати» вибрати команду Step Over натиснути клавішу <F8> або натиснути клавішу <F7>. Після виконання інструкції буде виділена наступна. Таким чином, вибираючи потрібну команду з меню Виконати, можна виконати трасування програми.

Активізувати та виконати трасування можна за допомогою функціональної клавіатури -- Команді за крок відповідає клавіша <F8>, а команді Трасування клавіша <F7>.

У будь-який момент часу можна завершити трасування й продовжити виконання програми в реальному темпі. Для цього треба з меню Виконати вибрати команду "Виконати".

При необхідності виконати трасування частини програми варто встановити курсор на інструкцію програми, з якою треба почати трасування, і з меню Виконати вибрати команду Виконати, натиснувши клавішу <F4>. Потім, натискаючи клавішу <F7> або клавішу <F8>, виконати трасування потрібного фрагмента програми.

Під час трасування можна спостерігати не тільки за порядком виконання інструкцій програми, але і за значенням змінних. Про те, як це зробити, розповідається в одному з наступних розділів.

Під час налагодження, зокрема при виконанні програми по кроках, досить часто буває корисно знати, чому дорівнює значення тієї чи іншої змінної. Наладник дозволяє спостерігати значення змінних програми.

Для того, щоб під час виконання програми по кроках мати можливість контролювати значення змінної, потрібно додати ім'я цієї змінної до списку спостережуваних елементів (Watch List). Для цього треба з меню Виконати вибрати команду Add Watch (Додати спостережуваний елемент) і в полі Вираз діалогового вікна Watch Properties (рис. 3) ввести ім'я змінної.

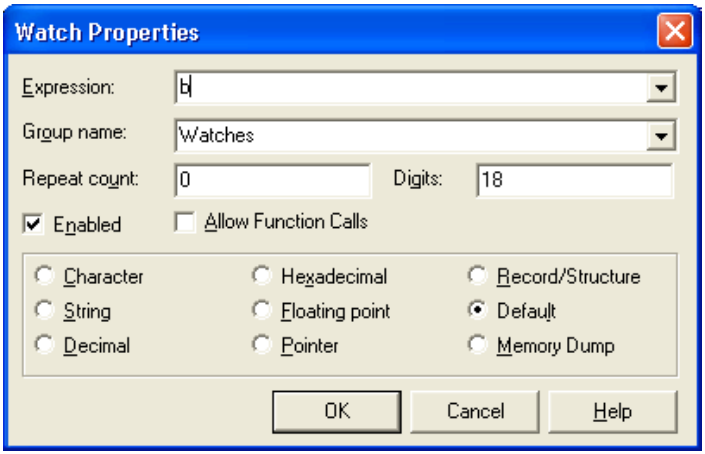


Рис. 3. Додавання імені змінної в список Watch List

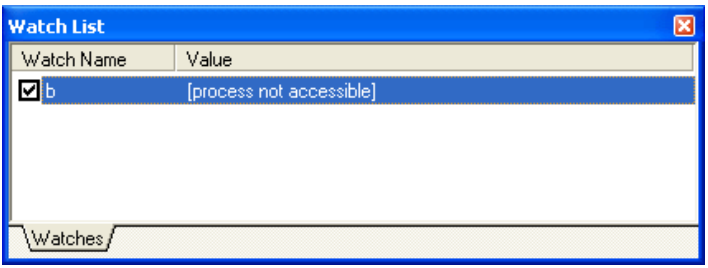


Рис. 4. Результат додавання імені змінної в список Watch List

У результаті до списку Watch List, вміст якого відображається в діалоговому вікні Watch List (рис. 4), буде доданий новий елемент. Оскільки змінні програми існують (і, отже, доступні) лише під час виконання програми, то після імені змінної виводиться повідомлення: процес недоступний (процес недоступний).

Як приклад на рис. 5 наведено вікна редактора коду і вікна Watch List під час покрокового виконання програми сортування масиву.

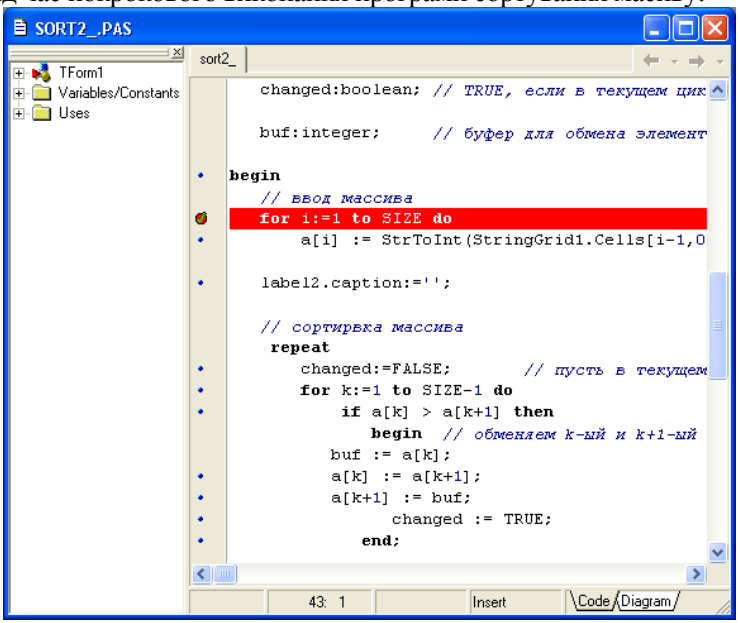


Рис. 5. Контроль значень змінних під час покрокового виконання програми

У вікні редактора коду стрілкою позначена інструкція, яка буде виконана на наступному кроці виконання програми (при натисканні клавіші <F8> або при виборі команди Step Over меню Виконати з), у діалоговому вікні Watch List виведені значення змінних.

Існує ще один спосіб, що дозволяє перевірити значення змінної, не додаючи її ім'я до списку Watch List. Полягає він у наступному. Після того, як програма досягне точки зупину, у результаті чого відкриється вікно редактора коду, потрібно встановити курсор миші на імені змінної, значення якої треба перевірити. У вікні редактора коду з'явиться вікно підказки, в якому буде виведено значення змінної (рис. 6).

Щоб завершити процес покрокового виконання програми, потрібно з меню Run вибрати команду Program Reset.

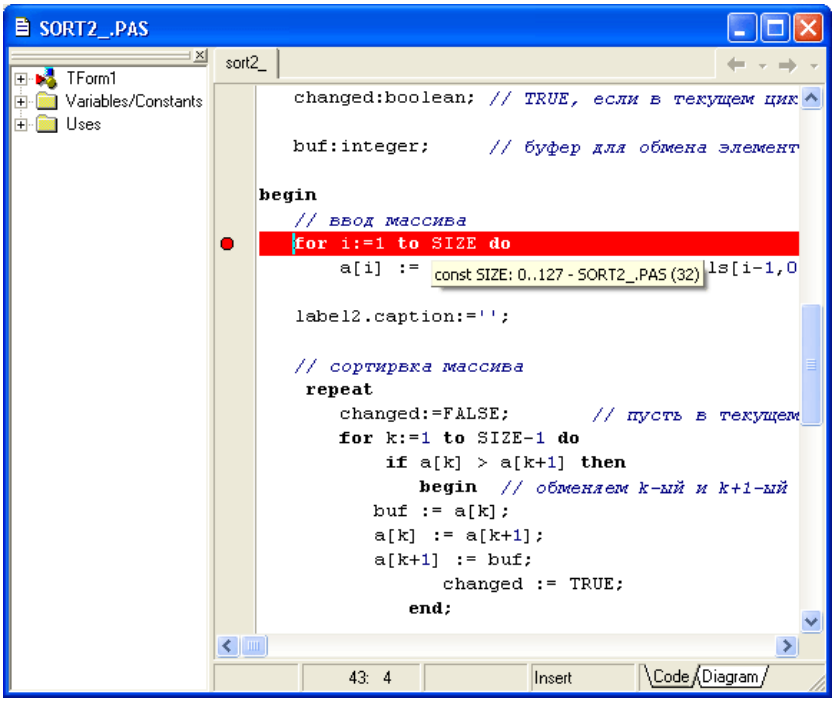


Рис. 6. Контроль значення змінної без додавання імені до списку Watch List

4.2. Елементи роботи з компонентою TMemo.

Компонента TMemo знаходиться на палітрі компонентів Standart, який являє собою поле для відображення і редагування неформатованого тексту. Текст можна завантажити з файла, набрати на клавіатурі, вставити з буфера обміну. Тобто він має багатодоступні властивості для тексту редагування, що сприяє його застосуванню в найпростіших текстових редакторах (рис.7).

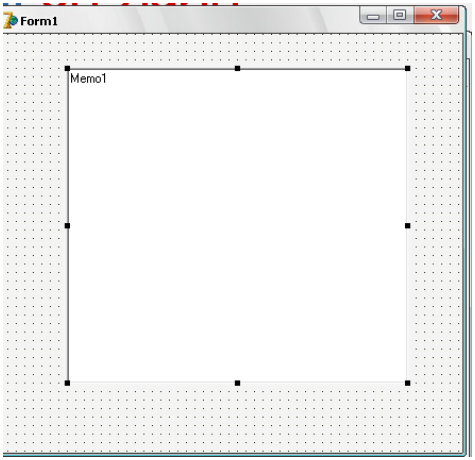


Рис.7. Зовнішній вигляд компоненти TMemo

Текст у компоненті доступний через властивість Lines містить масив рядків, що знаходяться в компоненті. Можна отримати доступ як до окремого рядка, так і до всього тексту. Цю властивість можна редагувати і під час роботи програми, і під час розробки в інспекторі об'єктів.

Кількість рядків у компоненті можна дізнатися через властивість Memo1.Lines.Count, а доступ до окремого рядка через функцію Strings. Наприклад:

ShowMessage (Memo1.Lines.Strings [0]); / / Показує у віконці перший рядок тексту.

Рядки - це масив рядків, що починаються з нульового знака, що закінчуються Count-1. Про це слід пам'ятати при написанні подібних процедур доступу, інакше відбудеться вихід за доступну масиву межу, що викличе помилку в програмі.

Можна для прикладу організувати цикл почергового відображення рядків із компоненти Memo1. Для реакції на натискання на кнопку Button1 це виглядає наступним чином:

Процедура TForm1.Button1Click (Sender: TObject);
Var I: Integer; / / оголошення цілочисельний змінної
почати
якщо Memo1.Lines.Count <> 0, то / / перевірка ненульової кількості рядків
для I: = 0 до Memo1.Lines.Count-1 робити / / задається цикл, що дорівнює кількості рядків
ShowMessage (Memo1.Lines.Strings [I]); / / виведення рядків до повідомлення
Кінець.
Очистити рядки можна однією процедурою Memo1.Lines.Clear.
Додати рядок -
Memo1.Lines.Add ("рядок");
Вставити рядок у задане місце -
Memo1.Lines.Insert (0, 'рядок');
Останній приклад вставляє текстовий рядок у перший рядок (перший рядок -1 = 0).
Видалити рядок -
Memo1.Lines.Delete (0); / / видаляє перший рядок.

Під час застосування процедури видалення рядків Видалити пам'ятайте, що спочатку потрібно перевіряти компоненту TMemo на наявність таких взагалі. Перевірка через функцію Count, яка повертає їх цілочисельне значення.

4.3. Створення випадкових значень у заданому діапазоні.

Для генерації випадкових значень використовується функція Random(Range).

Вона повертає значення 0 <= X < Range. Наприклад, Random(5) поверне значення 0 <= X < 5.

Якщо Range не задано, то повертається значення в діапазоні 0 <= X < 1.

Ініціалізація генератора випадкових чисел відбувається за допомогою процедури Randomize;

Тобто створення масиву випадкових значень в інтервалі 0 <= X < 1 буде виглядати так:

```
var
  I: Integer;
  A:Array [1..50]  of real;
begin
  Randomize;
  for I := 1 to 50 do begin
    A[I]:=Random;
  end;
end.
```

5. Порядок виконання роботи

Створити форму, яка повинна містити компоненти TMemo для відображення вхідного та вихідного масивів. Також форма повинна містити компоненти TEdit, в які користувач зможе вводити кількість елементів випадкового масиву та діапазон значень. На формі повинен бути компонент TLablel, якщо програма повинна виводити додаткові значення відповідно до індивідуального завдання.

Робота програми повинна починатися при натисканні на кнопку.

Алгоритм сортування масиву за зростанням методом прямого вибору може бути відображений так:

1. Переглядаючи масив від першого елемента, знайти мінімальний елемент і помістити його на місце першого елемента, а перший - на місце мінімального.

2. Переглядаючи масив від другого елемента, знайти мінімальний елемент і помістити його на місце другого елемента, а другий - на місце мінімального.

3. І так далі до передостаннього елемента.

6. Індивідуальні завдання

Номер завдання	Опис завдання
1	Відсортувати масив в порядку зростання значень.
2	Відфільтрувати всі елементи масиву, значення яких знаходяться в інтервалі 10 <= X < 15 або 20 <= X < 30. Зайти добуток усіх елементів масиву.
3	Відсортувати масив в порядку спадання значень.

4	Відфільтрувати всі елементи масиву, значення яких знаходяться в інтервалі 5 <= X < 15. Зайти суму всіх елементів масиву.
5	Відфільтрувати всі елементи масиву, квадрат яких < 10. Знайти максимальний елемент масиву.
6	Обрахувати нормований масив шляхом віднімання від кожного елемента мінімального значення масиву та ділення на максимальне значення масиву.
7	Відфільтрувати всі елементи масиву, cos яких <0. Знайти мінімальний елемент масиву.
8	Відфільтрувати всі елементи масиву, значення яких знаходяться в інтервалі 7 <= X < 10 та відняти від кожного елемента масиву мінімальне значення.
9	Відфільтрувати всі елементи масиву, значення яких знаходяться в інтервалі 3 <= X < 6 або 1 <= X < 2. Зайти добуток відфільтрованих елементів масиву.
10	Відфільтрувати всі елементи масиву, sin яких <0. Знайти максимальний елемент масиву.
11	Відсортувати масив у порядку зростання значень.
12	Відфільтрувати всі елементи масиву, значення яких знаходяться в інтервалі 7 <= X < 10 та відняти від кожного елемента масиву максимальне значення.
13	Обрахувати нормований масив шляхом кожного елемента ділення на максимальне значення масиву.
14	Відфільтрувати всі елементи масиву, значення яких знаходяться в інтервалі 1 <= X < 7 та відняти від кожного елемента масиву максимальне значення.
15	Відсортувати масив у порядку спадання значень.
16	Відфільтрувати всі елементи масиву квадрат, яких < 25. Обрахувати квадрат кожного з відфільтрованих значень.

Запитання для контролю

1. Які види помилок існують?
2. Як виконати трасування програми, передивитися значення змінних під час її виконання?
3. Побудувати цикл, який виконує сумування чисел від 1 до 100.
4. Пояснити, що таке масив і як виконати операції зчитування/запису його елементів.
5. Як здійснюється пошук максимального та мінімального значення серед елементів масиву?
6. Пояснити, як працює алгоритм сортування елементів масиву в порядку зростання або спадання.

Список рекомендованої літератури

9. А. Хомоненко, В. Гофман Delphi 7 (2-е издание), БХВ-Петербург · 2010.
10. Н. Культин. Основы программирования в Delphi, БХВ-Петербург · 2009.
11. А. Хомоненко, В. Гофман. Самоучитель Delphi, БХВ-Петербург · 2008.
12. Абрамян М.Э. Delphi 7. Карманный справочник с примерами, Кудиц-Образ · 2006.
13. Л. Климова Delphi 7. Основы программирования. Решение типовых задач. Самоучитель, Кудиц-Образ · 2004.
14. Ю.А. Шпак Delphi 7 на примерах, BHV – Санкт-Петербург, 1997. – 176 с.
15. Галисеев Г. В. Компоненты в Delphi 7. Профессиональная работа, МЭСИ,1998.
16. Кэнту М. Delphi 7. Для профессионалов, Питер • 2004.

3. Створення програми, що змінює властивості основної форми та компонентів під час виконання програми за вибором користувача

1. Мета роботи: вивчення компонентів TCheckBox, TRadioButton, TGroupBox, TRadioGroup та TColorDialog, TFontDialog діалогів.

2. Завдання до роботи:

1. Створити форму, що містить необхідну кількість компонентів TGroupBox, TCheckBox, TRadioButton, TButton, TEdit , TLabel та TMemo.
2. Створити програму, що містить компонент TMemo, який змінює свої розміри відповідно до зміни розмірів вікна
3. Передбачити можливість зміни властивостей основного вікна BorderIcons та BorderStyle за допомогою компонентів TCheckBox, TRadioButton, TGroupBox.
4. Відповідно до індивідуального завдання використати діалоги TColorDialog, TFontDialog для зміни кольору та шрифту заданих компонентів.
5. У звіті навести:
   1) Зображення, що ілюструють роботу програми.
   2) Код програми.

3. Обладнання та матеріали: Персональний комп'ютер, середовище Delphi.

4. Теоретична частина

4.1. Діалоги TColorDialog та TFontDialog

Діалоги в палітрі компонентів знаходяться на вкладці Dialogs.

TColorDialog - діалог вибору кольору. Колір за замовчуванням змінений і після виклику діалогу зберігається у властивості Color. Виклик діалогу виконується функцією Execute, яка повертає True або False в залежності від того, чи був обраний колір. Наприклад, зміна кольору об'єкта при натисканні кнопки буде виглядати так:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
```

```
  if ColorDialog1.Execute then
    begin
Shape1.Color := ColorDialog1.Color;
End;
End.
```

TFontDialog викликає стандартний діалог вибору шрифту. Властивість Font містить як початковий (обраний за замовчуванням) шрифт, так і змінений, після виклику діалогу. Аналогічно діалог викликається функцією Execute:

```
FontDialog1.Font := Edit1.Font;
if FontDialog.Execute then
begin
Edit.Font := FontDialog.Font;
End.
```

Тут показаний приклад зміни шрифту в компоненті Edit. Причому під час виклику діалогу FontDialog у ньому початково буде встановлений шрифт, заданий у компоненті Edit.

Властивість Device указує типи відображуваних шрифтів. fdBoth- усі, fdPrinter - принтерні, fdScreen - екранні

MaxFontSize MinFomtSize установлює межі мінімального і максимального розмірів шрифту. Цифра 0 - розмір обмежується тільки можливостями даного шрифту.

4.2. Компонента TCheckBox

Компонента **TCheckBox** є незалежним перемикачем для задання значення типу True/False або Так/Ні/Не_знаю. Таке значення, відображене у властивості State, компоненти, доступній як для читання, так і для записування. У вікні може бути кілька компонент TCheckBox. Стан кожного такого перемикача не залежить від стану решти перемикачів, тому їх називають незалежними.

Головні властивості компоненти:

Alignment - визначає розташування тексту з лівого чи правого боку компоненти;

AllowGrayed - дозволяє чи забороняє використовувати стан cbGrayed (Не_знаю); якщо AllowGrayed=True, то перемикач має три стани і внаслідок натискань рухається по циклу cbGrayed -cbChecked -cbUnchecked;

Checked - містить задане значення типу True/False; значення cbUnchecked і cbGrayed позначають False;

State - містить стан компоненти: cbUnchecked (ні), cbChecked (так), cbGrayed (не знаю).

Наприклад, код, що буде вмикати/вимикати компоненту TMemo на формі відповідно до стану компонента TCheckBox:

```
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  Memo1.Enabled := CheckBox1.Checked;
End.
```

4.3. Компонента TRadioButton

Компоненти **TRadioButton**, на відміну від TCheckBox, є залежними перемикачами (кнопками), що дають змогу вибрати одне з декількох значень. Вони об'єднані в групу, яка й визначає один з можливих виборів. Особливістю цих компонент є механізм їх перемикання. У разі натискання на одну з кнопок інша, раніше натиснута, вивільняється, тобто ввімкненою (вибраною) може бути лише одна. Якщо немає інших спеціальних компонент, то всі перемикачі належать одній групі - формі. За допомогою компонент-контейнерів можна створити декілька груп, однак самі групи між собою незалежні. Всередині кожної групи можна вибирати одне значення.

Головні властивості компоненти такі:

Checked - визначає, чи натиснута кнопка; у разі зміни цієї властивості генерується подія OnClick;

Alignment - визначає розташування тексту з лівого чи правого боку кнопки.

Компонента **TRadioGroup** є спеціальним контейнером для розташування залежних перемикачів

TRadioButton. Ці перемикачі можна розміщувати в декількох стовпцях.

Головні властивості компоненти:

Columns - кількість стовпців перемикачів;

ItemIndex - номер вибраного (натисненого) перемикача; номер обчислюють від 0; якщо він дорівнює -1, то жодна кнопка не натиснута;

Items - набір рядків із заголовками кнопок; додавання чи вилучення кнопок виконують шляхом додавання чи вилучення рядків списку Items.

4.4. Подія OnResize

Зміну розмірів компонентів при зміні розміру форми зручно робити при виконанні події OnResize. Дана подія викликається при кожній зміні розмірів форми. Для створення процедури що викликається даною подією, треба скористатися Object Inspector (рис.1.), виконавши подвійне клацання на події OnResize:

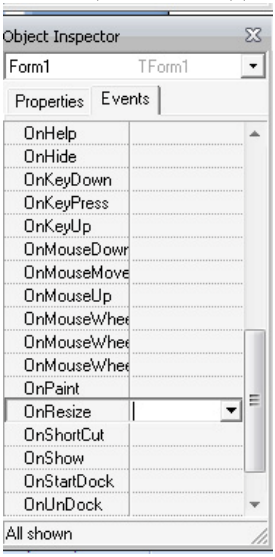


Рис.1. Створення процедури, що викликається при події OnResize

Створена процедура повинна виглядати так:

```
procedure TForm1.FormResize(Sender: TObject);
begin
end.
```

До даної процедури треба записати всі дії, що зв’язані зі зміною розмірів форми.

5. Порядок виконання роботи

Створити форму, що містить компоненту TMemo. Компонент повинен займати всю форму по висоті та половину форми по ширині. При зміні розмірів форми розмір компоненту TMemo повинен змінювати свої розміри відповідно.

На формі створити необхідну кількість компонентів TCheckBox та TRadioButton. Дані компоненти повинні забезпечувати повну зміну наступних властивостей форми: BorderIcons та BorderStyle. Приблизний зовнішній вигляд даних компонентів наведено на рис.2.

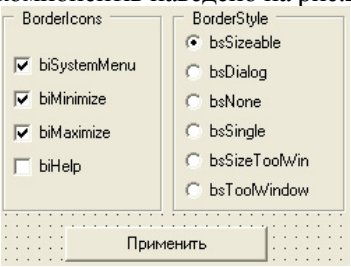


Рис.2. Компоненти TCheckBox та TRadioButton

Наприклад, зміна BorderIcons за допомогою першого компонента TCheckBox буде виглядати так:

```
 if CheckBox1.Checked then
begin
fMain.BorderIcons := fMain.BorderIcons + [biSystemMenu]
end
  else
begin
fMain.BorderIcons := fMain.BorderIcons - [biSystemMenu];
end.
```

Для зміни властивості BorderStyle відповідно до встановленого TRadioGroup зручно використати оператор case:

```
  case RadioGroup1.ItemIndex of
    0 : fMain.BorderStyle := bsSizeable;
    1 : fMain.BorderStyle := bsDialog;
…..
  End.
```

На формі створити діалоги TColorDialog та TFontDialog, які будуть виконуватись при натисканні на певну кнопку та змінювати колір та шрифт компонентів відповідно до індивідуального завдання.

6. Індивідуальні завдання

Номер завдання	Опис завдання
1	Створити компоненту TLabel, яка буде змінювати свій колір та шрифт.
2	Передбачити можливість зміни шрифту компоненти TMemo та кольору основної форми.
3	Передбачити можливість зміни кольору та шрифту всіх кнопок форми.
4	Передбачити можливість зміни кольору та шрифту всіх компонент TCheckBox форми.
5	Передбачити можливість зміни кольору та шрифту всіх компонент TRadioButton форми.
6	Створити компоненту TEdit, яка буде змінювати свій колір та шрифт.
7	Передбачити можливість зміни шрифту та кольору основної форми.
8	Передбачити можливість зміни кольору компонент TRadioButton та шрифту компоненти TMemo.
9	Передбачити можливість зміни кольору всіх кнопок форми та шрифту компоненти TMemo.
10	Передбачити можливість зміни кольору компонент TCheckBox та шрифту компонент TRadioButton.
11	Передбачити можливість зміни шрифту компонент TCheckBox та кольору основної форми.
12	Передбачити можливість зміни шрифту компонент TRadioButton та кольору основної форми.
13	Створити компоненту TLabel, яка буде змінювати свій колір. Передбачити можливість зміни шрифту компоненти TMemo.
14	Створити компоненту TEdit, яка буде змінювати свій шрифт. Передбачити можливість зміни шрифту компонент TRadioButton.
15	Передбачити можливість зміни шрифту всіх кнопок форми та кольору основної форми.
16	Передбачити можливість зміни шрифту всіх кнопок форми та кольору компоненти TMemo.

Запитання для контролю

1. Які основні властивості компонент TEdit, TMemo, TRadioButton та TCheckBox?
2. Пояснити принцип роботи конструкції if..else. Що таке умова і якого вона може бути типу?
3. Навести приклади вбудованих діалогів для зміни кольору та шрифту, пояснити основні етапи їх роботи.
4. Навести можливі варіанти значень властивості BorderIcons форми та пояснити їх вплив на її візуальний вигляд.
5. Навести можливі варіанти значень властивості BorderStyle форми та пояснити їх вплив на її візуальний вигляд.

Список рекомендованої літератури

1. А. Хомоненко, В. Гофман Delphi 7 (2-е издание), БХВ-Петербург · 2010.
2. Н. Культин. Основы программирования в Delphi, БХВ-Петербург · 2009.
3. А. Хомоненко, В. Гофман. Самоучитель Delphi, БХВ-Петербург · 2008.
4. Абрамян М.Э. Delphi 7. Карманный справочник с примерами, Кудиц-Образ · 2006.
5. Л. Климова Delphi 7. Основы программирования. Решение типовых задач. Самоучитель, Кудиц-Образ · 2004.
6. Ю.А. Шпак Delphi 7 на примерах, BHV – Санкт-Петербург, 1997. – 176 с.
7. Галисеев Г. В. Компоненты в Delphi 7. Профессиональная работа, МЭСИ,1998.
8. Кэнту М. Delphi 7. Для профессионалов, Питер • 2004.

4. Створення програми табулювання математичною функції з параметрами, що задає користувач

1. Мета роботи: вивчення циклів та основ роботи з текстовими файлами та TSaveDialog діалогу.

2. Завдання до роботи:

1. Створити форму, що містить необхідну кількість компонентів TGroupBox, TCheckBox, TEdit , TLabel, TMemo та TMainMenu.
2. Створити програму, що виконує табулювання функції (згідно з індивідуальним завданням) у заданих межах із заданим кроком і виводить результат у компоненту TMemo та/або файл у залежності від вибору користувача.
3. Для збереження файла використати діалог TSaveDialog.
4. У звіті навести:
   1) Зображення, що ілюструють роботу програми.
   2) Код програми.

3. Обладнання та матеріали: Персональний комп'ютер, середовище Delphi.

4. Теоретична частина

4.1. Діалог TSaveDialog

Діалоги в палітрі компонентів знаходяться на вкладці Dialogs. TSaveDialog - діалог збереження файла (рис.1).

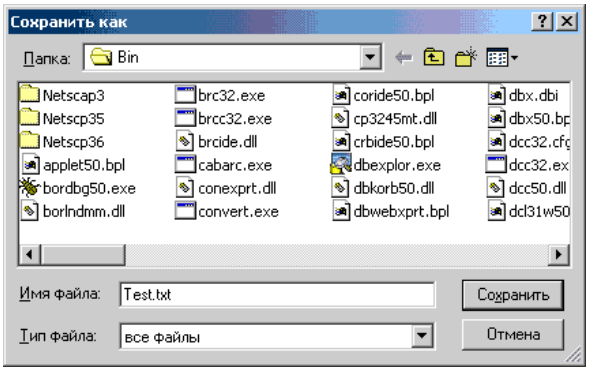


Рис.1. Діалог збереження файла

Основна властивість, в якій повертається у вигляді рядка вибраний користувачем файл, - FileName. Значення цієї властивості можна задати і перед зверненням до діалогу. Тоді воно з'явиться в діалозі як значення за замовчуванням у вікні "Ім'я файла" (див. рис.1).

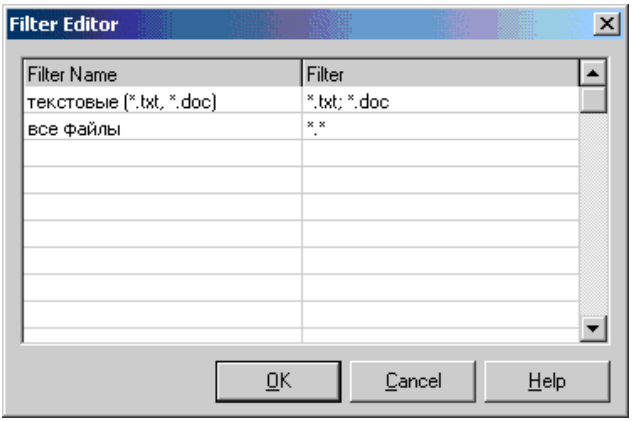


Рис.2. Властивість Filter

Типи шуканих файлів, що з'являються в діалозі у випадаючому списку "Тип файла" (рис. 1), задаються властивістю Filter. У процесі проектування цю властивість простіше відобразити з допомогою редактора фільтрів, який викликається натисненням кнопки з трьома крапками поряд із іменем цієї властивості в Інспекторові Об'єктів. При цьому відкривається вікно редактора (рис. 2). У його лівій панелі "Filter Name" ви записуєте той текст, який побачить користувач у випадаючому списку "Тип файла" діалогу. А в правій панелі "Filter" записуються розділені крапками з комою шаблони фільтра. У прикладі (рис. 2) задано два фільтри: текстових файлів із розширеннями. Txt і. Doc і будь-яких файлів із шаблоном *.*.

Після виходу з вікна редагування фільтрів задані вами шаблони з'являться у властивості Filter у вигляді рядка виду: текстові (*. txt, *. doc) | *. txt; *. doc | всі файли | *.*

У цьому рядку тексти і шаблони розділяються вертикальними лініями. В аналогічному вигляді, якщо потрібно, можна задавати властивість Filter програмно під час виконання програми.

Властивість FilterIndex визначає номер фільтра, який буде за замовчуванням показаний користувачеві в момент відкриття діалогу.

Наприклад, значення FilterIndex = 1 задає за замовчуванням перший фільтр.

Властивість InitialDir визначає початковий каталог, який буде відкритий у момент початку роботи користувача з діалогом. Якщо значення цієї властивості не задано, то відкривається поточний каталог чи той, який був відкритий при останньому зверненні користувача до відповідного діалогу в процесі виконання даної програми.

Властивість DefaultExt визначає значення розширення файла за замовчуванням. Якщо значення цієї властивості не задано, користувач повинен вказати в діалозі повне ім'я файла з розширенням. Якщо ж поставити значення DefaultExt, то користувач може писати в діалозі ім'я без розширення. У цьому випадку буде прийнято задане розширення.

Властивість Title дозволяє задати заголовок діалогового вікна. Якщо цю властивість не задано, вікно відкривається з заголовком, визначеним у системі (наприклад, «Збереження файла» у вікні на рис. 8.1). Але ви можете задати і свій заголовок, який підказує користувачеві очікувані дії. Наприклад, «Вкажіть ім'я файла».

Приклад використання:

```
if SaveDialog1.Execute then
 begin

  FName := SaveDialog1.FileName;
  ……
 End.
```

4.2. Запис масиву у файл

Якщо задано масив a:array[1..100] of real, то записати його в текстовий файл можна так:

1. Оголосити змінну типу текстовий файл

```
var
 f:TextFile.
```

2. Зв’язати змінну типу файл із заданим файлом

```
AssignFile(f, 'c:\file.txt').
```

3. Створити файл

```
 Rewrite(f).
```

4. За допомогою циклу виконати почерговий запис елементів масиву

```
For i:=1 to 100 do
   begin
Writeln(f,FloatToStr(a[i]));
   End.
```

5. Закрити файл

```
CloseFile(f).
```

5. Порядок виконання роботи

Створити форму, що наведено на рис. 3. Замість функції y=x*x*x*x+4 вкажіть функцію відповідно до індивідуального завдання.

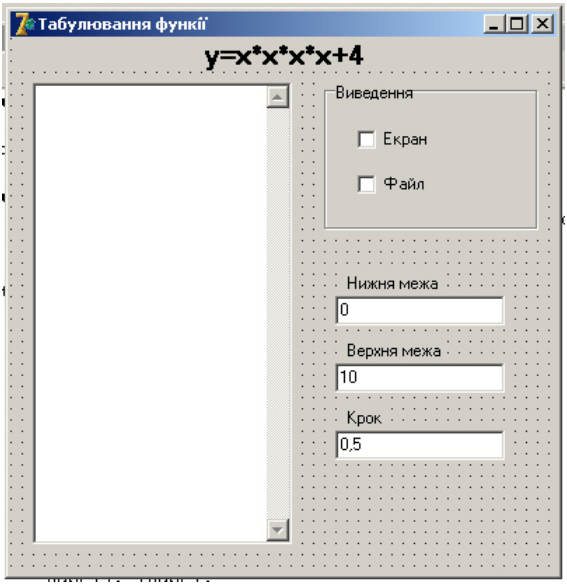


Рис.3. Зовнішній вигляд форми

Вставте у форму головне меню (об'єкт типу **MainMenu**). Для цього використайте компоненту **MainMenu** із закладки **Standard**. Введіть назви команд головного меню форми. Для цього виберіть об'єкт **MainMenul** і двічі клацніть на значенні його властивості **Items**. Інший шлях — двічі клацніть на самому об'єкті. У вікні, яке відкриється (**Form1**.**MainMenu1**), вибирайте мишею рамку команди і

записуйте назву команди, наприклад, "Дії" (рис.4), як значення властивості **Caption** у вікні *Object Ispector*. Закрийте вікно створення команд головного меню **Form1.MainMenu1**.

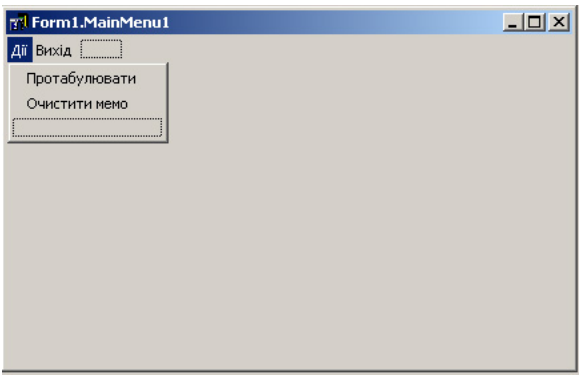


Рис.4. Головне меню програми

Запрограмуйте команду "Очистити поле виведення" головного меню, скориставшись методом **Clear** об'єкта **Memo1**.

У вікні форми в Delphi з'явилось головне меню. Виберіть пункт "Очистити поле виведення". З'явиться заготовка процедури реакції на подію виклику цієї команди. У ній запишіть оператор виклику методу Clear для очистки поля виведення об'єкта Memo1:

```
procedure TForm1.N4Click(Sender: TObject); {Тут N4 —ім'я команди "Очистити поле
виведення", у вас може бути інший номер, його не виправляйте}
begin
Memo1.Clear {Викликаємо метод об'єкта Memo1, який очищує багаторядкове поле
редагування}
end; {Тепер клацніть на формі}
```

Запрограмуйте команду "Вихід" головного меню, скориставшись процедурою Close.

```
procedure TForm1.N6Click(Sender: TObject); {Тут N6 —ім'я команди "Вихід"}
begin
Close {Закриваємо вікно програми}
end; {Тепер клацніть на формі}
```

Запрограмуйте команду "Протабулювати".

Результат роботи даної програми - це таблиця, що складається з декількох рядків. Щоб додати у поле **Memo1** новий рядок до таблиці, треба змінити значення комплексної властивості **Lines** (рядки) за допомогою її метода **Add** (додати) з одним аргументом — символьним рядком: **Memo1**.**Lines**.**Add** ('рядок символів'). Виконайте команду "Протабулювати" з головного меню, клацнувши на ній один раз. З'явиться заготовка до процедури, яку заповніть так:

```
procedure TForm1.N3Click(Sender: TObject);
var x, y, h, a, b : real;
str1, str2 : string;
begin
Memo1.Lines.Add(' X F(X)'); {В об'єкт Memo1 вставляємо рядок із підписами стовпців}
a:=StrToFloat(Edit1.Text); {Одержуємо числове значення лівої межі}
b:=StrToFloat(Edit2.Text); {Одержуємо числове значення правої межі}
h:=StrToFloat(Edit3.Text); {Одержуємо числове значення кроку}
x:=a; {Починаємо табулювати з лівої межі}
while x<=b+h/2 do {поки аргумент x не досягне правої межі з гарантією h/2}
begin
y:=x*x*x*x+4; {Обчислюємо значення функції – ФУНКЦІЮ ЗАМІНИТИ НА СВОЮ З ЗГІДНО ІНДЗ}
str1:=FloatToStr(x); {Формуємо символьні рядки з аргументу та значення функції}
str2:=FloatToStr(y);
if CheckBox1.Checked then Memo1.Lines.Add(str1+' ' +str2);
{Вставляємо рядок у поле
Memol}
x:=x+h; {Збільшуємо аргумент на величину кроку}
end
end.
```

Додати можливість запису у файл самостійно.

6. Індивідуальні завдання

Номер завдання	Вираз для обчислення
1	$\frac{\sin(x)}{(x-3)} + x$
2	$\frac{\sqrt{x}}{(x+5)}$
3	$\sqrt{x} + x^2$
4	$\cos(x)^2 + \sqrt{x}$
5	$\cos(x)\sin(x) + \sqrt{x}$
6	$\cos(\lvert x\rvert) + \sqrt{x}$
7	$x\sqrt{x+5} + x$
8	$\sqrt{x + \lvert\cos(x)\rvert}$
9	$x^4 + 5x + \frac{8}{x}$
10	$\cos(x^2) + \frac{\sin(x)}{x}$
11	$e^x + \sqrt{x}\,\frac{x+5}{x-5}$
12	$\sin(\sqrt{x}) + \frac{x}{x+2}$
13	$\lvert\sin(x) + \cos(x) + \sqrt{x}\rvert$
14	$\frac{x^5 + x^4}{x}$
15	$\frac{\sqrt{x+5}}{10} + x^2$
16	$\lvert\sin(x)\rvert + e^x + \sqrt{x}$

Запитання для контролю

1. Пояснити принцип роботи циклу while та побудувати на його основі цикл, який може замінити цикл for.
2. Як організувати запис тексту у файл?
3. Як працює діалог відкривання файла?
4. Опишіть методи компоненту TMemo.-
5. Додайте можливість автоматичної очистки поля TMemo при генерації результатів. Очистка повинна відбуватися лише якщо позначено відповідний прапорець.

Список рекомендованої літератури

1. А. Хомоненко, В. Гофман Delphi 7 (2-е издание), БХВ-Петербург · 2010.
2. Н. Культин. Основы программирования в Delphi, БХВ-Петербург · 2009.
3. А. Хомоненко, В. Гофман. Самоучитель Delphi, БХВ-Петербург · 2008.
4. Абрамян М.Э. Delphi 7. Карманный справочник с примерами, Кудиц-Образ · 2006.
5. Л. Климова Delphi 7. Основы программирования. Решение типовых задач. Самоучитель, Кудиц-Образ · 2004.
6. Ю.А. Шпак Delphi 7 на примерах, BHV – Санкт-Петербург, 1997. – 176 с.

5. Створення текстового редактора з вбудованими функціями обробки

1. Мета роботи: вивчення компоненти TRichEdit, умов, циклів та роботи з текстовими файлами.

2. Завдання до роботи:

1. Створити форму, що містить необхідну кількість компонентів TRichEdit, TButton, TLabel, TMainMenu .
2. Використати діалоги TColorDialog, TFontDialog для зміни кольору та шрифту виділеного фрагмента тексту. Для збереження та відкриття файла використати діалоги TSaveDialog та TOpenDialog.
3. Створити додаткові функції відповідно до індивідуального завдання.
4. У звіті навести:
    1) Зображення, що ілюструють роботу програми.
    2) Код програми.

3. Обладнання та матеріали: Персональний комп'ютер, середовище Delphi.

4. Теоретична частина

4.1. Діалог TOpenDialog

Діалоги в палітрі компонентів знаходяться на вкладці Dialogs.

TOpenDialog - діалог відкриття файла, що працює аналогічно до TSaveDialog.

Основна властивість TOpenDialog, яка видає у вигляді рядка назву вибраного користувачем файла, називається FileName. Значення цієї властивості можна задати і перед зверненням до діалогу. Тоді воно з'явиться в діалозі як значення за замовчуванням у вікні "Ім'я файла".

Типи шуканих файлів, що з'являються в діалозіув випадаючому списку "Тип файла", задаються властивістю Filter.

Властивість FilterIndex визначає номер фільтра, який буде за замовчуванням показаний користувачеві у момент відкриття діалогу. Наприклад, значення FilterIndex = 1 задає перший фільтр.

Властивість InitialDir визначає початковий каталог, який буде відкритий в момент початку роботи користувача з діалогом. Якщо

значення цієї властивості не задано, то відкривається поточний каталог чи той, який був відкритий при останньому зверненні користувача до відповідного діалогу в процесі виконання даної програми.

Властивість DefaultExt визначає значення розширення файла за замовчуванням. Якщо значення цієї властивості не задано, користувач повинен вказати в діалозі повне ім'я файла з розширенням. Якщо ж задати значення DefaultExt, то користувач може писати в діалозі ім'я без розширення. У цьому випадку буде прийнято задане розширення.

Властивість Title дозволяє вам задати заголовок діалогового вікна. Якщо цю властивість не задано, вікно відкривається з заголовком, визначеним у системі (наприклад «Відкриття файла» у вікні). Але ви можете задати і свій заголовок, який підказує користувачеві очікувані дії. Наприклад, «Вкажіть ім'я файла».

Приклад використання:

```
if OpenDialog1.Execute then
 begin
  FName := OpenDialog1.FileName;
  ……
 end;
```

4.2. Компонента TRichEdit

Компонента RichEdit працює з текстом у форматі RTF. При бажанні змінити атрибути виділеного фрагмента тексту ви можете задати властивість SelAttributes. Це властивість типу TTextAttributes, яка у свою чергу має підвластивості: Color (колір), Name (ім'я шрифту), Size (розмір), Style (стиль) та ряд інших. Наприклад, розмістіть на основній формі компоненту RichEdit, діалог вибору шрифту FontDialog і кнопку Button, яка дозволить користувачеві змінювати атрибути тексту. В обробник клацанням кнопки можна ввести код:

```
if FontDialog1.Execute then
 with RichEdit1.SelAttributes do
  begin
   Color: = FontDialog1.Font.Color;
```

```
    Name: = FontDialog1.Font.Name;
    Size: = FontDialog1.Font.Size;
    Style: = FontDialog1.Font.Style;
   end;
RichEdit1.SetFocus.
```

У наведеному коді присвоюється по черзі значення кожної властивості. Але цей текст можна кардинально скоротити, скориставшись тим, що об'єкти SelAttributes і Font сумісні за типом. Тому можна привласнити відразу всі властивості одного об'єкта іншому:

```
if FontDialog1.Execute then
 RichEdit1.SelAttributes.Assign (FontDialog1.Font);
RichEdit1.SetFocus.
```

Запустіть програму і побачите, що ви можете змінювати атрибути тексту, виконуючи окремі фрагменти різними шрифтами, розмірами, кольорами, стилями. Встановлювані атрибути впливають на виділений текст або, якщо нічого не виділено, на атрибути нового тексту, що вводиться, починаючи з поточної позиції курсору (позиція курсору визначається властивістю SelStart).

У компоненті є також властивість DefAttributes, що містить атрибути за замовчуванням. Ці атрибути діють до того моменту, коли змінюються атрибути у властивості SelAttributes.

Але значення атрибутів у DefAttributes зберігаються і в будь-який момент можуть бути методом Assign присвоєні атрибутам властивості SelAttributes, щоб повернутися до попереднього стилю.

Властивість DefAttributes доступна лише під час виконання. Тому її атрибути при необхідності можна задавати, наприклад, в обробнику події OnCreate.

За вирівнювання, відступи і т.д. в межах поточного абзацу відповідає властивість Paragraph типу TParaAttributes. Цей тип у свою чергу має ряд властивостей:

Alignment - визначає вирівнювання тексту. Може набувати значеннь taLeftJustify (ліворуч), taCenter (по центру) або taRightJustify (праворуч).

FirstIndent - число пікселів відступу червоного рядка.

Numbering - управляє вставкою маркерів, як у списках. Може набувати значення nsNone - відсутність маркерів, nsBullet - маркери виставляться.

LeftIndent - відступ у пікселях від лівого поля.

RightIndent - відступ у пікселях від правого поля.

TabCount - кількість позицій табуляції.

Tab - Значення позицій табуляції в пікселях.

Значення підвластивостей властивості Paragraph можна задавати тільки в процесі виконання програми, наприклад, у події створення форми або при натисненні якої-небудь кнопки. Значення і властивостей Paragraph розміщуються в тому абзаці, в якому знаходиться курсор. Наприклад, кожен із наступних операторів здійснить відповідне вирівнювання поточного абзацу:

```
RichEdit1.Paragraph.Alignment: = taLeftJustify; / / Вліво
RichEdit1.Paragraph.Alignment: = taCenter; / / По центру
RichEdit1.Paragraph.Alignment: = taRightJustify; / / Вправо.
```

Наступний оператор призведе до того, що поточний абзац буде відображатися як список, тобто з маркерами: RichEdit1.Paragraph.Numbering: = nsBullet;

Знищення списку в поточному абзаці здійснюється оператором RichEdit1.Paragraph.Numbering: = nsNone;

Властивості TabCount і Tab мають сенс при введенні тексту тільки при значенні властивості компонента WantTabs = true. Ця властивість дозволяє користувачеві вводити в текст символ табуляції. Якщо WantTabs = false, то натискання користувачем клавіші табуляції просто перемкне фокус на черговий компонент і символ табуляції в текст не буде введено.

Властивості Alignment і WordWrap мають той же зміст, що, наприклад, в позначках, і визначають вирівнювання тексту і допустимість перенесення довгих рядків. Встановлення властивості ReadOnly в true задає текст тільки для читання. Властивість MaxLength визначає максимальну довжину тексту, що вводиться. Якщо MaxLength = 0, то довжина тексту не обмежена. Властивості WantReturns і WantTab визначають допустимість введення користувачем у текст символів переведення рядка і табуляції.

Властивість ScrollBars визначає наявність смуг прокручування тексту у вікні. За замовчуванням ScrollBars = ssNone, що означає їх відсутність. Користувач може в цьому випадку переміщатися по тексту тільки за допомогою курсору. Можна задати властивості ScrollBars значення ssHorizontal, ssVertical або ssBoth, що буде відповідно означати наявність горизонтальної, вертикальної або обох смуг прокручування.

Розглянемо приклад текстового редактора, який використовує описані вище властивості компонента RichEdit (рис. 1). Текст у вікні редактора частково пояснює атрибути шрифту, використані при його написанні.

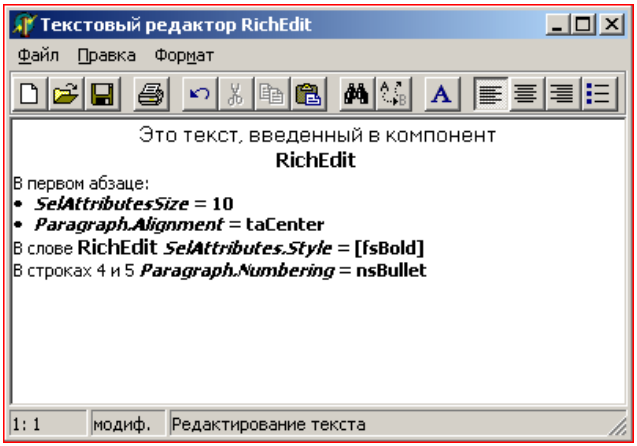


Рис. 1. Приклад текстового редактора

Основна властивість вікон RichEdit - Lines, що містить текст вікна у вигляді списку рядків і має тип TStrings. Початкове значення тексту можна встановити в процесі проектування, натиснувши кнопку з трьома крапками близько властивості Lines у вікні Інспектора Об'єктів. Перед вами відкриється вікно редагування списків рядків (рис. 2). Ви можете редагувати або вводити текст безпосередньо в цьому вікні або натиснути кнопку "CodeEditor" і працювати у звичайному вікні Редактора Коду. У цьому випадку, завершивши роботу з текстом, виберіть із контекстного меню, спливаючого при клацанні правою кнопкою миші, команду "Close Page" і дайте позитивну відповідь на запитання, чи хочете ви зберегти текст у відповідному властивості вікна редагування.

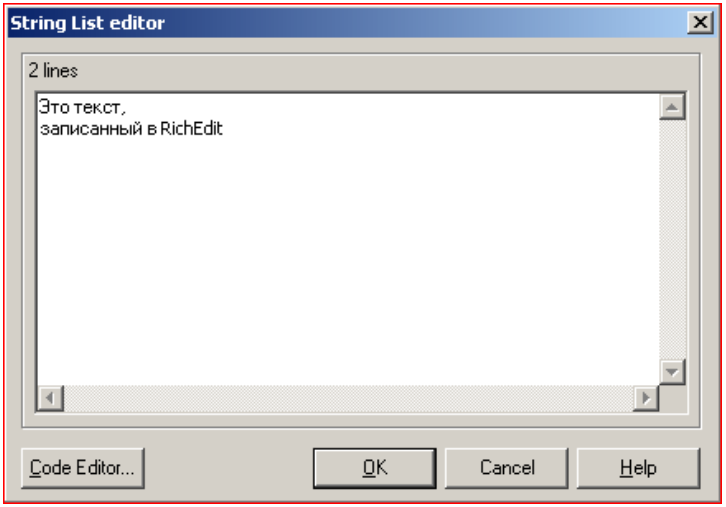


Рис. 2. Вікно редагування списків рядків

Під час виконання програми ви можете заносити текст до вікна редагування за допомогою методів властивості Lines типу TStrings. Весь текст, поданий одним рядком типу String, всередині якої використовуються роздільники типу символів повернення каретки та переведення рядка, міститься у властивості Text.

Доступ до окремого рядка тексту ви можете отримати за допомогою властивості Strings [Index: Integer]. Індекси, як і скрізь у Delphi, починаються з 0. Так що RichEdit1.Lines.Strings [0] - це текст першого рядка. Врахуйте, що якщо вікно редагування змінюється в розмірах при роботі з додатком і властивість WordWrap = true, то індекси рядків будуть змінюватися при перенесенні рядків, так що в цих випадках індекс мало про що говорить.

Властивість тільки для читання Count вказує кількість рядків у тексті.

Для очищення тексту у вікні треба виконати процедуру Clear. Цей метод стосується самого вікна, а не до його властивості Lines.

Для занесення нового рядка в кінець тексту вікна редагування можна скористатися методами Add або Append властивості Lines. Для завантаження тексту з файла застосовується метод LoadFromFile. Збереження тексту у файлі здійснюється методом SaveToFile.

Нехай, наприклад, у вашій програмі є вікно редагування Edit1, в якому користувач вводить ім'я працівника, і є кнопка, при натисканні на яку у вікно RichEdit1 повинна занестися шапка характеристики цього працівника, після чого користувач може заповнити текст характеристики.

Обробник клацання на кнопці може мати вигляд:

```
RichEdit1.Clear;
RichEdit1.Lines.Add ('Х А Р А К Т Е Р И С Т И К А');
RichEdit1.Lines.Add ('Співробітник' Edit1.Text);.
RichEdit1.SetFocus.
```

Завантаження у вікно RichEdit1 тексту з файла (наприклад, що зберігається у файлі характеристики працівника) може здійснюватися командою

```
RichEdit1.Lines.LoadFromFile ('text.txt');
```

Збереження тексту у файлі може здійснюватися командою

```
RichEdit1.Lines.SaveToFile ('text.txt').
```

Властивість SelStart компонентів RichEdit вказує позицію курсора в тексті або початок виділеного користувачем тексту. Властивість CaretPos вказує на запис, поле X якої містить індекс символу в рядку, перед яким розташований курсор, а поле Y - індекс рядка, в якій знаходиться курсор. Таким чином, враховуючи, що індекси починаються з 0, значення RichEdit1.CaretPos.Y 1 і RichEdit1.CaretPos.X 1 визначають відповідно номер рядка та символу в ній, перед яким розташований курсор.

4.3. Функції та процедури обробки рядків

У модулі StrUtils.pas містяться корисні функції для обробки рядкових змінних. Щоб ввести цей модуль до програми, потрібно додати його ім'я (StrUtils) у розділ Uses.

1) Функція **Length (Str: String)** - повертає довжину рядка (кількість символів). Приклад:

```
var
  Str: String; L: Integer;
{... }
Str: = 'Hello!';
L: = Length (Str); {L = 6}.
```

2) Функція **SetLength (Str: String; NewLength: Integer)** дозволяє змінити довжину рядка. Якщо рядок містив більшу кількість символів, ніж задано у функції, то "зайві" символи обрізаються. Приклад:

```
var Str: String;
{... }
Str: = 'Hello, world!';
SetLength (Str, 5); {Str = "Hello"}.
```

3) Функція **Pos (SubStr, Str: String)** - повертає позицію підрядка в рядку. Нумерація символів починається з одиниці (1). У разі відсутності підрядка в рядку повертається 0. Приклад:

```
var Str1, Str2: String; P: Integer;
{... }
Str1: = 'Hi! How do you do? ';
Str2: = 'do';
P: = Pos (Str2, Str1); {P = 9}.
```

4) Функція **Copy (Str: String; Start, Length: Integer)** - повертає частину рядка Str, починаючи з символу Start довжиною Length. Обмежень на Length немає - якщо воно перевищує кількість символів від Start до кінця рядка, то рядок буде скопійований до кінця. Приклад:

```
var Str1, Str2: String;
{... }
Str1: = 'This is a test for Copy () function.';
Str2: = Copy (Str1, 11, 4); {Str2 = "test"}.
```

5) Процедура **Delete (Str: String; Start, Length: Integer)** - видаляє з рядка Str символи, починаючи з позиції Start довжиною Length. Приклад:

```
var Str1: String;
{... }
Str1: = 'Hello, world!';
Delete (Str1, 6, 7); {Str1 = "Hello!" }.
```

6) Процедура **Insert (SubStr: String; Str: String; Pos: Integer)** - вставляє в рядок Str підрядок SubStr в позицію Pos. Приклад:

```
var Str: String;
{... }
Str: = 'Hello, world!';
Insert ('my', Str, 8); {Str1 = "Hello, my world!" }.
```

7) Функція **PosEx (SubStr, Str: String; Offset: Integer)** - функція аналогічна функції Pos (), але дозволяє задати відступ від початку рядка для пошуку. Якщо значення Offset задано (воно не є обов'язковим), то пошук починається з символу Offset в рядку. Якщо Offset більше довжини рядка Str, то функція повернення 0. Також 0 повертається, якщо підрядок не знайдений у рядку. Приклад:

```
uses StrUtils;
{... }
var Str1, Str2: String; P1, P2: Integer;
{... }
Str1: = 'Hello! How do you do? ';
Str2: = 'do';
P1: = PosEx (Str2, Str1, 1); {P1 = 12}
P2: = PosEx (Str2, Str1, 15); {P2 = 19}.
```

Рядки можна порівнювати один з одним стандартним способом:

```
var Str1, Str2, Str3: String; B1, B2: Boolean;
{... }
Str1: = '123 ';
Str2: = '456 ';
Str3: = '123 ';
B1: = (Str1 = Str2); {B1 = False}
B2: = (Str1 = Str3); {B2 = True}.
```

Якщо рядки повністю ідентичні, логічний вираз дорівнюватиме True.

5. Порядок виконання роботи

Для створення редактора використати компоненту RichEdit, (закладка **Win32**), оскільки він підтримує абзаци, нумерацію, колірне виділення тексту тощо. Компоненту необхідно розтягти, щоб вона займала всю робочу площу вікна. При зміні розмірів вікна користувачем вона теж повинна змінювати свої розміри. Для цього можна скористатись властивістю **Align** (**alClient**).

Очистити вікно редактора. Для цього в Інспекторі Об'єктів знайдіть властивість Lines об’єкта RichEdit. Натиснути на нього, а потім на кнопку з трьома крапками. З'явиться редактор тексту – треба стерти увесь вміст.

Створити панель інструментів, де будуть розташовуватись кнопки швидкого виклику команд (наприклад Відкрити, Зберегти тощо). Для цього треба розташувати на формі компоненту **Panel** (закладка Standart). Очистити її властивість **Caption**. Встановити властивість панелі - **Align** рівним **alTop**. Розташувати на створеній панелі дві кнопки (Button). Властивість Caption першої панелі відповідає Відкрити, другої - Зберегти. (Рис.3.)

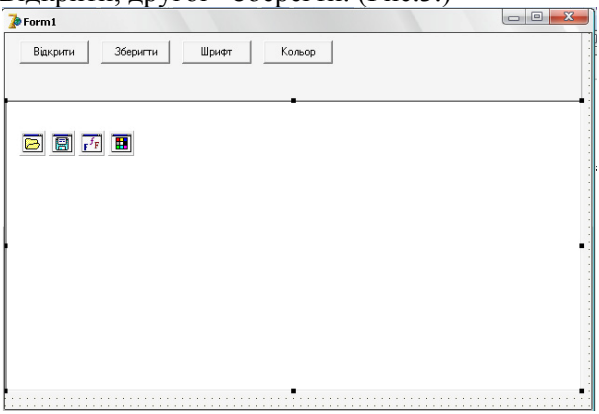


Рис.3. Зовнішній вигляд редактору

Створити компоненти відкриття і збереження файлів. Щоб відкривати і зберігати текстові файли, знадобляться ще дві компоненти, це **OpenDialog** і **SaveDialog**. Обидві знаходяться на вкладці **Dialogs**. Якщо запустити програму і при запуску діалогу відкриття файла вибрати нетекстовий файл, то відбудеться помилка. Щоб її уникнути, треба дозволити користувачу вибирати тільки текстові файли. Для цього можна скористатись властивістю **Filter** компоненти **OpenDialog**. При натисненні на кнопку з трьома крапками відкриється діалогове вікно "Filter Editor" (Редактор фільтра). Заповнити її за зразком: Filter Name (Ім'я фільтра) Filter (Фільтр) Текстові файли *.txt та Rich Text *.rtf У компонент SaveDialog і OpenDialog є властивість **DefaultExt** – розширення за замовчуванням. Бажано зробити його таким, що дорівнює *.txt в обох компонентах.

Створити процедуру відкривання файлів. Клацнути два рази на кнопці **Button1** на формі і написати процедуру відкриття файла:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
if OpenDialog1.Execute then
RichEdit1.Lines.LoadFromFile(OpenDialog1.Filename);
end.
```

Створити процедуру збереження файлів. Клацнути два рази на кнопці **Button2,** щоб одержати доступ до процедури:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
if SaveDialog1.Execute then
RichEdit1.Lines.SaveToFile(SaveDialog1.Filename);
end.
```

Створити смужки прокручування. За відображення смуг прокручування компонента **RichEdit** відповідає властивість **ScrollBars**. Вибрати **ssVertical** - це забезпечить появу вертикальної смуги прокручування при редагуванні великих текстів.

Створити можливість зміни шрифту. Компонент **FontDialog** (знаходиться на закладці **Dialogs**) реалізує стандартний діалог Windows настроювання шрифту – встановити його на форму. Крім нього знадобиться ще і кнопка (**Button3**) щоб викликати це діалогове вікно. Задати її властивість **Caption** рівним **Шрифт**. А в обробнику події OnCLick написати:

```
procedure TForm1.Button3Click(Sender:TObject);
begin
if FontDialog1.Execute then
RichEdit1.SelAttributes.Assign(FontDialog1.Font);
End.
```

Створити можливість зміни кольору шрифту. Компонент **ColorDialog** (знаходиться на закладці **Dialogs**) реалізує стандартний діалог Windows вибору кольору – встановити його на форму.

Створити кнопку (**Button4**), щоб викликати це діалогове вікно. Задати її властивість **Caption**, що відповідає **Кольор**. А в обробнику події OnCLick написати:

```
procedure TForm1.Button4Click(Sender:TObject);
begin
if FontDialog1.Execute then
RichEdit1.SelAttributes.Color:=ColorDialog1.Color;
End.
```

Самостійно реалізувати індивідуальне завдання, використовуючи цикли та функції роботи з рядками (див.4.3.).

6. Індивідуальні завдання

Номер завдання	Завдання
1	Організувати пошук слова, що задає користувач, у тексті. Виділити всі знайдені слова червоним кольором.
2	Організувати пошук слова, що задає користувач, у тексті. Виділити всі знайдені слова жирним шрифтом.
3	Організувати пошук слова, що задає користувач, у тексті. Виділити всі знайдені слова курсивом.
4	Організувати пошук слова, що задає користувач, у тексті. Видалити всі знайдені слова.
5	Організувати пошук слова, що задає користувач, у тексті. Замінити всі знайдені слова на слово, яке також задає користувач.
6	Виділити кожне друге слово в тексті курсивом.
7	Виділити кожне третє слово в тексті жирним шрифтом.
8	Виділити кожне четверте слово в тексті червоним кольором.
9	Організувати пошук слова, що задає користувач, у тексті. Вивести кількість знайдених слів.
10	Організувати підрахунок кількості слів у тексті. Вивести результат.
11	Організувати пошук та видалення тексту, що виділений курсивом.

12	Організувати пошук та видалення тексту, що виділений червоним кольором.
13	Організувати пошук слова, що задає користувач, у тексті. Замінити всі знайдені слова на слово, яке також задає користувач.
14	Організувати підрахунок кількості букв у тексті. Вивести результат.
15	Організувати підрахунок кількості букв, що написані курсивом, у тексті. Вивести результат.
16	Організувати підрахунок кількості голосних букв у тексті. Вивести результат.

Запитання для контролю

1. Які функції використовуються для роботи з рядками?
2. Як визначити властивості фрагменту тексту в RichEdit.
3. Як можна організувати збереження та відкривання тексту з компоненти RichEdit?
4. Як можна підрахувати кількість слів у тексті?
5. Як змінити властивості певного фрагменту тексту в RichEdit?

Список рекомендованої літератури

1. А. Хомоненко, В. Гофман Delphi 7 (2-е издание), БХВ-Петербург · 2010.
2. Н. Культин. Основы программирования в Delphi, БХВ-Петербург · 2009.
3. А. Хомоненко, В. Гофман. Самоучитель Delphi, БХВ-Петербург · 2008.
4. Абрамян М.Э. Delphi 7. Карманный справочник с примерами, Кудиц-Образ · 2006.
5. Л. Климова Delphi 7. Основы программирования. Решение типовых задач. Самоучитель, Кудиц-Образ · 2004.
6. Ю.А. Шпак Delphi 7 на примерах, BHV – Санкт-Петербург, 1997. – 176 с.

6. Створення програми побудови складних графічних фігур

1. Мета роботи: вивчення компонент TImage, TCanvas та побудова графічних примітивів.

2. Завдання до роботи:

1. Створити форму, що містить необхідну кількість компонент TImage, TButton та діалог TSaveDialog.
2. Створити процедуру, що будує зірку з кількістю променів відповідно до індивідуального завдання.
3. Створити процедуру, що будує заштриховану фігуру, яка складається не менш ніж із 7 базових фігур.
4. Передбачити можливість збереження створеного зображення.
5. У звіті навести:

1) Зображення, що ілюструють роботу програми,
2) Код програми.

3. Обладнання та матеріали: Персональний комп'ютер, середовище Delphi.

4. Теоретична частина

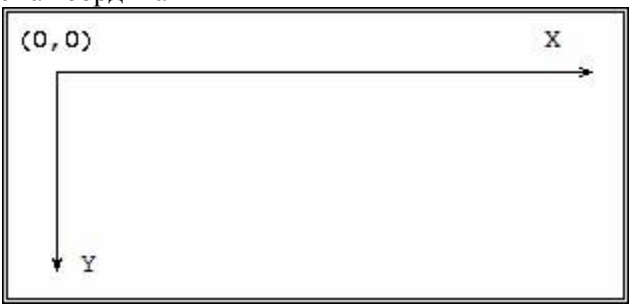
4.1. Графічні можливості Delphi

Canvas забезпечує простір (полотно, канву) для створення, зберігання та модифікації графічних об'єктів. Canvas є основою графічної підсистеми Delphi.

Полотно забезпечує:

- Завантаження і зберігання графічних зображень.
- Створення нових та зміна збережених зображень за допомогою пера, кисті, шрифту.
- Рисування та зафарбовування різних фігур, ліній, текстів.
- Комбінування різних зображень.

Система координат



4.1.1. Основні методи Canvas

Розглянемо частину методів, за допомогою яких можна створювати прості рисунки.

Arc

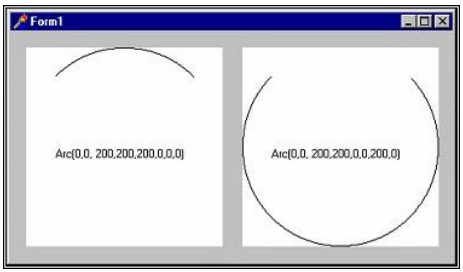
Рисує дугу кола або еліпса

Arc (x1, y1, x2, y2, x3, y3, x4, y4: Integer)

Метод Arc рисує дугу кола або еліпса за допомогою поточних параметрів пера Pen. Точки (x1, y1) та (x2, y2) визначають прямокутник, що описує еліпс. Початкова точка дуги визначається перетином еліпса з прямою, що проходить через його центр і точку (x3, y3). Кінцева точка дуги визначається перетином еліпса з прямою, що проходить через його центр і точку (x4, y4). Дуга рисується проти годинникової стрілки від початкової до кінцевої точки.

Приклад:

```
Image1.Canvas.Arc (0,0, 200,200, 200,0, 0,0);
Image2.Canvas.Arc (0,0, 200,200, 0,0, 200,0).
```



Chord

Рисує заповнену замкнену фігуру, обмежену дугою кола або еліпса і хордою

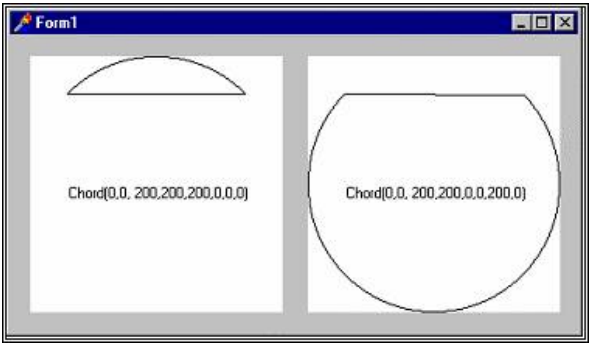
Chord (x1, y1, x2, y2, x3, y3, x4, y4: Integer).

Метод Chord рисує замкнену фігуру: дугу кола або еліпса, замкнену хордою, за допомогою поточних параметрів пера Pen. Фігура заповнюється поточним значенням Brush (розглянемо трохи нижче). Точки (x1, y1) та (x2, y2) визначають прямокутник, що описує еліпс. Початкова точка дуги визначається перетином еліпса з прямою, що проходить через його центр і точку (x3, y3). Кінцева точка дуги визначається перетином еліпса з прямою, що проходить через його центр і точку (x4, y4). Дуга рисується проти годинникової стрілки від початкової до кінцевої точки. Хорда з'єднує точки (x3, y3) і (x4, y4).

Приклад:

Image1.Canvas.Chord (0,0, 200,200, 200,0, 0,0);

Image2.Canvas.Chord (0,0, 200,200, 0,0, 200,0).



Ellipse

Рисує заповнене коло або еліпс

Ellipse (x1, y1, x2, y2: Integer);

Метод Ellipse рисує коло або еліпс за допомогою поточних параметрів пера Pen. Фігура заповнюється поточним значенням Brush. Точки (x1, y1) та (x2, y2) визначають прямокутник, що описує еліпс.

Приклад:

```
With Image1.Canvas do
 Begin
  Brush.Color: = clRed;
  Brush.Style: = bsDiagCross;
  Ellipse (0, 0, Image1.Width, Image1.Height);
 End.
```

FillRect

Заповнює вказаний прямокутник полотна, використовуючи поточне значення Brush.

FillRect (const: TRect).

Метод FillRect заповнює прямокутник полотна, вказаний параметром Rect, використовуючи поточне значення Brush. Заповнювана область містить верхню і ліву сторони прямокутника, але не включає праву і нижню сторони.

Приклад:

Image1.Canvas.FillRect (Rect (0,0, Width, Height)).

Очищує всю канву компонента Image1, заповнюючи її фоном, якщо він встановлений у властивості Brush.

FloodFill

Зафарбовує поточним пензлем закриту область полотна певним кольором.

Type TFillStyle = (fsSurfase, fsBorder);

Procedure FloodFill (x, y: Integer; Color: TColor; FillStyle: TFillStyle).

Метод FloodFill зафарбовує поточним пензлем Brush закриту область полотна, певним кольором і початковою точкою зафарбовування (x, y). Точка з координатами x і y є довільною внутрішньої точкою заповнюваної області, яка може мати довільну форму. Межа цієї області визначається поєднанням параметрів Color і FillStyle. Параметр Color вказує колір, який використовується при

визначенні межі області, а параметр FillStyle визначає, як саме по цьому кольору визначається межа. Якщо FillStyle = fsSurface, то заповнюється область, зафарбована кольором Color, а на інших кольорах метод зупиняється. Якщо FillStyle = fsBorder, то навпаки, заповнюється область, пофарбована будь-якими кольорами, що не дорівнюють Color, а на кольорі Color метод зупиняється.

Приклади:

1.

```
With Image1.Canvas do begin
 Brush.Color: = clWhite;
 FloodFill (X, Y, Pixels [X, Y], fsSurface);
End;
```

Наведені оператори зафарбовують білим кольором на полотні компонента Image1 всі пікселі, прилеглі до пікселя з координатами (x, y), і мають той же колір, що і цей піксель.

2.

```
With Image1.Canvas do begin
 Brush.Color: = clWhite;
 FloodFill (X, Y, clBlack, fsBorder);
End.
```

Наведені оператори зафарбовують білим кольором на полотні компонента Image1 всі пікселі, прилеглі до пікселя з координатами (x, y) і що мають колір, відмінний від чорного. При досягненні чорної межі області зафарбовування зупиняється.

FrameRect

Рисує на полотні поточним пензлем прямокутну рамку.

FrameRect (const Rect: TRect).

Метод FrameRect рисує на полотні прямокутну рамку навколо області Rect, використовуючи установку поточної кисті Brush. Товщина рамки - 1 піксель. Область всередині рамки пензлем не зафарбовується. Відрізняється від методу Rectangle тим, що рамка

рисується кольором кисті (у методі Rectangle - кольором пера Pen) і область не зафарбовується (в методі Rectangle зафарбовується).

Приклад:

```
With Form1.Canvas do
 Begin
 Brush.Color: = clBlack;
FrameRect (Rect (10,10,100,100));
End;
```

Рисує на полотні форми Form1 чорну рамку.

LineTo

Рисує на полотні пряму лінію, що починається з поточної позиції пера і закінчується зазначеної точкою.

```
LineTo (x, y: Integer);
```

Метод LineTo рисує на полотні пряму лінію, що починається з поточної позиції пера PenPos і закінчується точкою (x, y). Поточна позиція пера PenPos переміщується в точку (x, y). При рисуванні використовуються поточні установки пера Pen.

Приклад:

```
Form1.Canvas.MoveTo (x1, y1);
Form1.Canvas.LineTo (x2, y2);
Form1.Canvas.LineTo (x3, y3).
```

Рисує кусково-ламану пряму, що сполучає точки (x1, y1), (x2, y2) і (x3, y3).

MoveTo

Змінює поточну позицію пера на задану, нічого не рисуючи при цьому.

```
MoveTo (x, y: Integer);
```

Метод MoveTo змінює поточну позицію пера на задану точкою (x, y). При переміщенні пера методом MoveTo нічого не рисується.

Pie

Малює заповнену замкнуту фігуру - сегмент кола або еліпса.

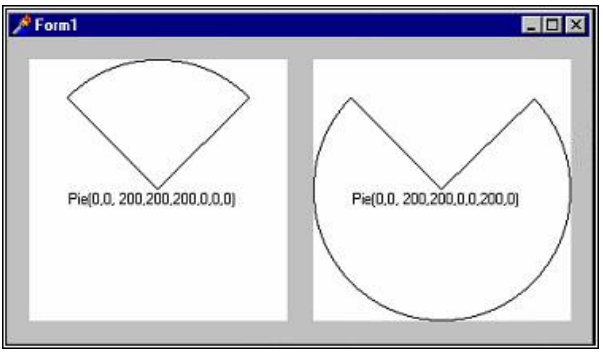
Pie (x1, y1, x2, y2, x3, y3, x4, y4: Longint);

Метод Pie рисує замкнену фігуру - сектор кола або еліпса, за допомогою поточних параметрів пера Pen. Фігура заповнюється поточним значенням Brush. Точки (x1, y1) та (x2, y2) визначають прямокутник, що описує еліпс. Початкова точка дуги визначається перетином еліпса з прямою, що проходить через його центр і точку (x3, y3). Кінцева точка дуги визначається перетином еліпса з прямою, що проходить через його центр і точку (x4, y4). Дуга рисується проти годинникової стрілки від початкової до кінцевої точки. Рисуються прямі, що обмежують сегмент і проходять через центр еліпса і точки (x3, y3) і (x4, y4).

Приклад:

Image1.Canvas.Pie (0,0, 200,200, 200,0, 0,0);

Image2.Canvas.Pie (0,0, 200,200, 0,0, 200,0).



Polygon

Рисує на полотні поточним пером замкнену фігуру (багатокутник) по заданій множині кутових точок, замикаючи першу і останню точки і зафарбовуючи внутрішню область фігури поточним пензлем.

Polygon (Points: array of TPoint).

Метод Polygon рисує на полотні замкнену фігуру (полігон, багатокутник) по безлічі кутових точок, заданих масивом Points. Перша із зазначених точок з'єднується прямою з останньою. Цим метод Polygon відрізняється від методу Polyline, який не замикає кінцеві точки. Рисування проводиться поточним пером Pen. Внутрішня область фігури зафарбовується поточним пензлем Brush.

Приклади:

1.

Form1.Canvas.Polygon ([Point (10,10), Point (30,10), Point (130,30), Point (240, 120)]).

Рисує на полотні форми чотирикутник по точках, заданих функціями Point.

2.

Form1.Canvas.Polygon (PointArray).

Рисує на полотні форми багатокутник по точках, що зберігаються в масиві PointArray, який може бути оголошений, наприклад, наступним чином:

Var PointArray: array [1 .. 100] of TPoint.

3.

Form1.Canvas.Polygon (Slice (PointArray, 10)).

Рисує на полотні форми багатокутника по перших 10 точках, що зберігаються в масиві PointArray з попереднього прикладу.

Polyline

Рисує на полотні поточним пером кусково-лінійну криву по заданій множині точок.

Polyline (Points: array of TPoint)

Метод Polyline Рисує на полотні кусково-лінійну криву по безлічі точок, заданих масивом Points. Відмінність методу Polyline від методу Polygon полягає в тому, що метод Polygon замикає кінцеві

точки, а метод Polyline - ні. Рисування виконується поточним пером Pen. Метод не змінює поточної позиції PenPos пера Pen.

Метод дозволяє рисувати кусково-лінійний графік функції, що зберігається в масиві типу TPoint. Якщо бажано використовувати для рисування тільки частину точок масиву, це можна зробити за допомогою функції Slice. Якщо треба нарисуватикриву усього по декількох точках, то передавати їх у метод Polyline зручно за допомогою функції Point.

Те, що робить метод Polyline, можна зробити і за допомогою методів MoveTo і LineTo, підвівши спочатку перо до першої точки, а потім послідовно виконуючи LineTo. Різниця буде полягати в тому, що метод Polyline не змінить поточну позицію пера, а методи MoveTo і LineTo змінять.

Приклад:

Form1.Canvas. Polyline ([Point (10,10), Point (30,10), Point (130,30), Point (240, 120)]).

Рисує кусково-лінійну криву по чотирьох точках, заданих функціями Point.

Rectangle

Рисує на полотні поточним пером прямокутник і зафарбовує його поточним пензлем.

Rectangle (x1, y1, x2, y2: Integer).

Метод Rectangle рисує на полотні поточним пером Pen прямокутник, верхній лівий кут якого має координати (x1, y1), а нижній правий - (x2, y2). Прямокутник зафарбовується поточної пензлем Brush. Рисування прямокутника без рамки можна здійснити методом FillRect. Прямокутник з заокругленими кутами рисується методом RoundRect. Прямокутник без внутрішнього зафарбовування рисується методом FrameRect.

Приклад:

Image1.Canvas.Rectangle (10,10,210,110).

RoundRect

Рисує на полотні прямокутну рамку з заокругленими кутами.

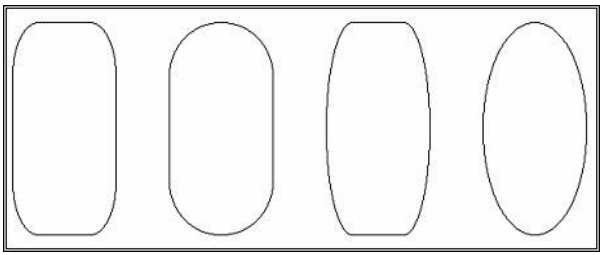
RoundRect (x1, y1, x2, y2, x3, y3: Integer);

Метод RoundRect рисує на полотні прямокутну рамку з заокругленими кутами, використовуючи поточні установки пера Pen і заповнюючи площу фігури поточним пензлем Brush. Рамка визначається прямокутником із координатами кутів (x1, y1) та (x2, y2). Кути заокруглюються за допомогою еліпсів із шириною x3 і висотою y3.

Якщо задати ширину еліпса x3 = x2-x1, то верхня і нижня межі рамки виявляться цілком заокругленими (без прямолінійної частини). Якщо y3 = y2-y1, те ж саме станеться з лівою і правою межами рамки. Якщо ж обидва вимірювання еліпса не менше розмірів рамки, то буде рисуватися просто еліпс. Але, звичайно, для рисування еліпса краще використовувати метод Ellipse. Якщо один із розмірів еліпса задати нульовим, то буде рисуватися прямокутна рамка, а для отримання такої рамки краще використовувати метод Rectangle.

Приклад:

```
with Image1.Canvas do
 begin
  RoundRect (10,10,110,210,50,100);
  RoundRect (160,10,260,210,100,100);
  RoundRect (310,10,410,210,50,200);
  RoundRect (460,10,560,210,100,200);
 End.
```



4.2. Процедура рисування ялинки

Наведений приклад працює в onPaint основної форми і рисує ялинки на її полотні.

```
procedure TForm1.FormPaint(Sender: TObject);
begin
With Form1.Canvas do // до кожного наступного рядка додавати Form1.Canvas.
 Begin
 Pen.Width := 1; // встановлення товщини пера
 Pen.Color := clGreen; // встановлення кольору пера
 Brush.Color:= clGreen; // встановлення кольору заливки
 {Рисування ялинки}
 PolyGon ([Point (350,90), Point (330,90), Point (400,160), Point (380,160),
        Point (470,250), Point (130,250), Point (220,160), Point (200,160),
        Point (270,90), Point (250,90), Point (300,40)]);
 Pen.Color := RGB (128,64,0); // встановленнякольору пера (відтінок коричневого)
 Brush.Color := RGB (128,64,0); // встановлення кольору заливки (відтінок коричневого)
 {рисування стовбура}
 PolyGon ([Point (350,251), Point (350,301), Point (250,301), Point (250,251)]);
 End;
end;
```

4.3. Процедура рисування незафарбованої зірки з 5 променями

Наведений приклад працює в onClick кнопки і рисує зірку на полотні компоненту TImage.

```
procedure TForm1.Button1Click(Sender: TObject);
var ang:real; i:integer;
   p:array[1..11] of TPoint;
begin
```

```
Image1.Picture.Bitmap.Height:=Image1.Height;
Image1.Picture.Bitmap.Width:=Image1.Width;
with Image1.Picture.Bitmap do
 begin
  Canvas.Pen.Color:=clBlack;
  for i:=1 to 11 do
    begin
      ang:=Pi*(90-36*(i))/180;
      if (i mod 2 = 0) then
        begin
          p[i].X := 250 + Trunc(50*cos(ang));
          p[i].Y := 250 + Trunc(50*sin(ang));
        end
        else
        begin
          p[i].X := 250 + Trunc(150*cos(ang));
          p[i].Y := 250 + Trunc(150*sin(ang));
        end;
    end;
 Canvas.Polyline(p);
 end;
end;
```

4.4. Збереження графіки з компонента TImage в bmp файлі.

Для збереження картинки з компонента у файлі використовується наступна процедура:

```
Image1.Picture.Bitmap.SaveToFile(`1.bmp`);
```

Дана процедура збереже картинку з компоненти з ім'ям Image1 в файл 1.bmp.

5. Порядок виконання роботи

Створити форму, що містить компонент Image (вкладка Additional) кнопки, при натисканні на яку буде відбуватися рисування фігур, кнопку при натисканні на яку буде відбуватися збереження рисунку та діалог SaveDialog.

6. Індивідуальні завдання

Номер завдання	Кількість променів зірки
1	3
2	4
3	6
4	8
5	9
6	10
7	11
8	12
9	13
10	14
11	15
12	16
13	17
14	18
15	19
16	20

Запитання для контролю

1. Для чого призначений об’єкт Canvas? Як розташована координатна система в Canvas.
2. Як можна встановити колір та стиль ліній?
3. Як встановити вид та колір заливки фігур?
4. Як можна нарисувати трикутник?
5. Як працює функція Ellipse?

Перелік рекомендованої літератури

1. А. Хомоненко, В. Гофман Delphi 7 (2-е издание), БХВ-Петербург · 2010.
2. Н. Культин. Основы программирования в Delphi, БХВ-Петербург · 2009.
3. А. Хомоненко, В. Гофман. Самоучитель Delphi, БХВ-Петербург · 2008.
4. Абрамян М.Э. Delphi 7. Карманный справочник с примерами, Кудиц-Образ · 2006.
5. Л. Климова Delphi 7. Основы программирования. Решение типовых задач. Самоучитель, Кудиц-Образ · 2004.
6. Ю.А. Шпак Delphi 7 на примерах, BHV – Санкт-Петербург, 1997. – 176 с.

7. Створення програми побудови графіків функцій

1. Мета роботи: вивчення компонент TImage, TCanvas, умов та циклів.

2. Завдання до роботи:

1. Створити форму, що містить необхідну кількість компонент TImage, TButton, TEdit та діалог TSaveDialog.
2. Створити програму яка будує графік функції (відповідно до індивідуального завдання) в довільних межах аргументу *x*.
3. Для збереження графіка використати діалог TSaveDialog.
4. У звіті навести:
    1) Зображення, що ілюструють роботу програми.
    2) Код програми.

3. Обладнання та матеріали: Персональний комп'ютер, середовище Delphi.

4. Теоретична частина

1) **Приклад процедури що будує графік функції *y=sin(x)***

у довільних межах, що задані користувачем.

```
procedure TForm1.Button1Click(Sender: TObject);
var x, y, i, vertical_start:integer;
   up_limit, down_limit, step,
   vertical_scale: real;
   data: array[0..2000] of real;
begin

//Зчитування меж аргументу

up_limit:=StrToFloat(Edit2.Text);
down_limit:=StrToFloat(Edit1.Text).
```

```
//Підготовка та очищення зображення
Image1.Picture.Bitmap.Height:=Image1.Height;
Image1.Picture.Bitmap.Width:=Image1.Width;
Image1.Picture.Bitmap.Canvas.Pen.Color:=clWhite;
Image1.Picture.Bitmap.Canvas.Brush.Color:=clWhite;
Image1.Picture.Bitmap.Canvas.Rectangle(0,0,Image1.Width,Image1.
Height);

Image1.Picture.Bitmap.Canvas.Pen.Color:=clBlack.

//Встановлення меж зміни значення функції та положення нуля
по вертикалі

vertical_scale:=(Image1.Height - 50) div 2;
vertical_start:=Image1.Height div 2;

Image1.Picture.Bitmap.Canvas.Pen.Width:=2;

//Рисування вісі у

Image1.Picture.Bitmap.Canvas.MoveTo(25, 0);
Image1.Picture.Bitmap.Canvas.LineTo(25, Image1.Height);
Image1.Picture.Bitmap.Canvas.MoveTo(25, 0);
Image1.Picture.Bitmap.Canvas.LineTo(15, 10);
Image1.Picture.Bitmap.Canvas.MoveTo(25, 0);
Image1.Picture.Bitmap.Canvas.LineTo(35, 10).

// Рисування вісі х

Image1.Picture.Bitmap.Canvas.MoveTo(0, vertical_start);
Image1.Picture.Bitmap.Canvas.LineTo(Image1.Width,
vertical_start);
Image1.Picture.Bitmap.Canvas.MoveTo(Image1.Width,
vertical_start);
```

```
    Image1.Picture.Bitmap.Canvas.LineTo(Image1.Width-10,
vertical_start+10);
    Image1.Picture.Bitmap.Canvas.MoveTo(Image1.Width,
vertical_start);
    Image1.Picture.Bitmap.Canvas.LineTo(Image1.Width-10,
vertical_start-10).

    //Табулювання функції в заданих межах

    step:=(up_limit-down_limit)/(Image1.Width-50);
    for i:=0 to (Image1.Width-25) do
     begin
     data[i]:=sin(i*step);
     end.

    //Встановлення графічного курсору в початкову точку графіка

    x:=25;
    y:=trunc(vertical_start + data[0]*vertical_scale);
    Image1.Picture.Bitmap.Canvas.MoveTo(x, y);

    //Побудова графіка

    Image1.Picture.Bitmap.Canvas.Pen.Width:=1;
    for i:=1 to (Image1.Width-25) do
     begin
     x:=i+25;
     y:=trunc(vertical_start + data[i]*vertical_scale);
     Image1.Picture.Bitmap.Canvas.LineTo(x, y);
     end;

    end.
```

5. Порядок виконання роботи

Побудувати форму відповідно до рис.1.

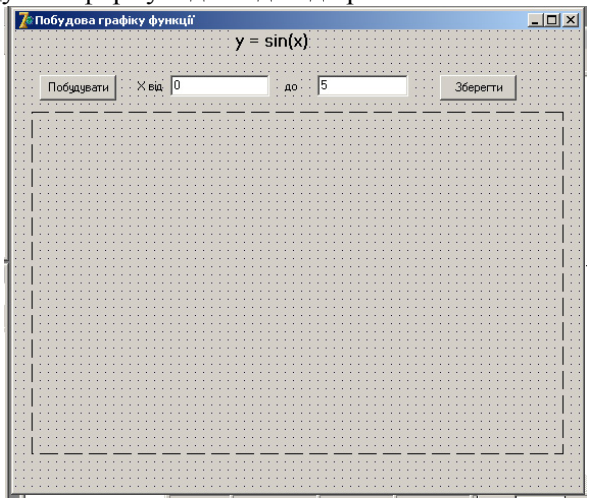


Рис.1. Форма програми для побудови графіка функції

Запрограмувати кнопку «Побудувати» на побудову графіка та осей координат на полотні компонента TImage. На рис.2. наведено приклад роботи програми.

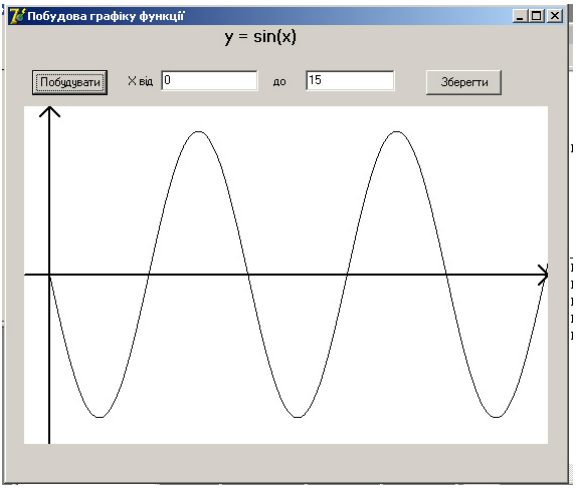


Рис.2. Приклад роботи програми

Кнопка «Зберегти» повинна виконувати збереження побудованого графіка в bmp-форматі за допомогою діалогу збереження файлів.

При виконанні роботи звернути увагу на правильне масштабування по осі у та допустимі значення аргументу функції.

6. Індивідуальні завдання

Номер завдання	Вираз для побудови графіка *y=f(x)*
1	$\frac{\sin(x)}{(x-3)}+x$
2	$\frac{\sqrt{x}}{(x+5)}$
3	$\sqrt{x}+x^2$
4	$\cos(x)^2+\sqrt{x}$
5	$\cos(x)\sin(x)+\sqrt{x}$
6	$\cos(\lvert x\rvert)+\sqrt{x}$
7	$x\sqrt{x+5}+x$
8	$\sqrt{x+\lvert\cos(x)\rvert}$
9	$x^4+5x+\frac{8}{x}$
10	$\cos(x^2)+\frac{\sin(x)}{x}$
11	$e^x+\sqrt{x}\,\frac{x+5}{x-5}$
12	$\sin(\sqrt{x})+\frac{x}{x+2}$
13	$\lvert\sin(x)+\cos(x)+\sqrt{x}\rvert$
14	$\frac{x^5+x^4}{x}$
15	$\frac{\sqrt{x+5}}{10}+x^2$
16	$\lvert\sin(x)\rvert+e^x+\sqrt{x}$

Запитання для контролю

1. Позначити на осі x кілька значень через рівний проміжок.
2. Позначити на осі y кілька значень через рівний проміжок.
3. Як працює збереження графіка у файлі?
4. Побудувати графік іншої функції іншим кольором поряд із існуючим графіком.
5. На якій компоненті і як відбувається рисування графіка?

Перелік рекомендованої літератури

1. А. Хомоненко, В. Гофман Delphi 7 (2-е издание), БХВ-Петербург · 2010.
2. Н. Культин. Основы программирования в Delphi, БХВ-Петербург · 2009.
3. А. Хомоненко, В. Гофман. Самоучитель Delphi, БХВ-Петербург · 2008.
4. Абрамян М.Э. Delphi 7. Карманный справочник с примерами, Кудиц-Образ · 2006.
5. Л. Климова Delphi 7. Основы программирования. Решение типовых задач. Самоучитель, Кудиц-Образ · 2004.
6. Ю.А. Шпак Delphi 7 на примерах, BHV – Санкт-Петербург, 1997. – 176 с.

8. Опрацювання GET-запитів за допомогою PHP

1. Мета роботи: вивчення основ роботи зі змінними, операцій виведення та обробки GET-запитів за допомогою PHP.

2. Завдання до роботи:

1. Вивчити процедури виведення та зчитування даних з GET-запитів.
2. Створити програму, яка здійснює обрахунок заданого виразу, підставляючи дані, отримані з GET-запитів. Згенерувати декілька гіперпосилань, які передають заданий набір чисел як параметр функції за допомогою GET-запиту.
3. У звіті навести:
   1) Зображення, що ілюструють роботу програми,
   2) Код програми.

3. Обладнання та матеріали: Персональний комп'ютер, сервер Apache +PHP.

4. Теоретична частина:

4.1. Внесення фрагментів PHP у HTML-код.

Для внесення в сторінки HTML коду PHP найчастіше використовуються стандартні дескриптори. Це пояснюється їхньою наочністю і зручністю у використанні. Наприклад,

```
<?php
  print "This is a simple test!";
?>
```

«<?php» та «?>» є відповідно дескрипторами початку і завершення коду.

4.2. Змінні

У мові PHP імена всіх змінних починаються із символу долара ($) і наступного за ним імені змінної. Імена змінних чуттєві до регістра. Тобто імена $username і $UserName є іменами двох різних змінних. Відповідно до правил іменування необхідно, щоб ім'я змінних починалося з букви або символу підкреслення, за яким може

йти будь-яка кількість буквено-цифрових символів або символів підкреслення. Наприклад,

```
<?php
    $username = ‘Barry’;
    $UserName = ‘White’;
    echo “$username $UserName”;
?>
```

Прикладами неправильних імен змінних є наступні: $4name, $&Test, $8+abc.

4.3. Деякі операції

Операція	Опис	Приклад
+	додавання	$a = 2 + $b
-	віднімання	$a = 2 - 3
*	множення	$b = $a*$c
/	ділення	$a = $b/2
.	додавання строк	$a = ‘Результат:’.$b

5. Порядок виконання роботи.

1. Відкрити порожню XTML-сторінку, створивши текстовий файл із наступним вмістом:

```
<! Doctype HTML public "- / / W3C / / DTD HTML 4.01 / / EN"
"Http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title> Моя перша PHP-програма </ title>
</ Head>
<body>
</ Body>
</ Html>
```

2. Зберегти файл у відповідну папку сервера. Файл назвати як ‘lab1.php’.

3. Додати дескриптори, які будуть обмежувати PHP-код:

```
<! Doctype HTML public "- / / W3C / / DTD HTML 4.01 / / EN"
"Http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title> Моя перша PHP-програма </ title>
</ Head>
<body>
<?php
?>
</ Body>
</ Html>
```

4. Організувати підрахунок виразу з індивідуального завдання використовуючи дані з GET-запиту.

Розглянемо наприклад підрахунок виразу a + b, де значення a і b передаються за допомогою GE- запиту:

Запит: «http://<>/<>/lab1.php?a=2&b=4»:

Приклад програми:

```
….
<?php
$a = $_GET[‘a’]; // Зчитування змінної a
$b = $_GET[‘b’]; // Зчитування змінної b
$rez = $a + $b; // Обрахунок виразу
echo ‘Результат:’.$rez; // Виведення результату
?>
….
```

5. Перевірити правильність роботи програми, відкривши її в браузері з передачею відповідних змінних.

6. Організувати гіперпосилання, яке буде передавати дані, задані відповідно до індивідуального завдання. Наприклад, для прикладу з п.4 таке гіперпосилання буде виглядати так:

```
<a href=”http://<сервер>/<папка>/lab1.php?a=2&b=4”>2 + 4</a>
```

Остаточний код прикладу буде мати вигляд:

```
<! Doctype HTML public "- / / W3C / / DTD HTML 4.01 / / EN"
"Http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title> Моя перша PHP-програма </ title>
</ Head>
<body>
<?php
$a = $_GET[‘a’]; // Зчитування змінної a
$b = $_GET[‘b’]; // Зчитування змінної b
$rez = $a + $b; // Обрахунок виразу
echo ‘Результат:’.$rez; // Виведення результату
?>
<br />
<a href=”http://<сервер>/<папка>/lab1.php?a=2&b=4”>2 + 4</a>
</ Body>
</ Html>
```

6. Індивідуальні завдання

Номер завдання	Опис завдання
1	(a+c)/b ; a=4; b=5; c=6
2	a*c*b ; a=2; b=7; c=5
3	a+c+b ; a=6; b=8; c=4
4	a+c/b ; a=7; b=3; c=3
5	a*c/b ; a=1; b=1; c=2
6	a/(b-c) ; a=9; b=3; c=1
7	a/c/b ; a=2; b=6; c=7
8	a-c-b ; a=5; b=8; c=6
9	a+c-b ; a=7; b=4; c=5
10	a-c*b ; a=3; b=2; c=2
11	a*c-b ; a=4; b=4; c=1
12	a*c/b ; a=8; b=6; c=6
13	a/b-c ; a=1; b=9; c=4
14	a*c-b ; a=2; b=1; c=3
15	c/b-a ; a=5; b=2; c=2
16	c+b*a ; a=7; b=5; c=1

Запитання для контролю

1. Як відрізнити фрагмент PHP-коду від HTML?
2. Як визначається змінна?
3. Змінити код таким чином, щоб у запиті ім’я першого параметра було test.
4. Змінити значення параметра в запиті за допомогою браузера.
5. Вивести результат жирним шрифтом.

Список рекомендованої літератури

1. PHP5 и MySQL. Библия пользователя.: пер. с англ. – М.: издательский дом «Вильямс», 2006. – 1216 с.: ил.
2. MySQL / Л. Ульман; пер. с англ. Слинкина А.А. – М: ДМК Пресс; Спб.: Питер, 2004 – 352 с. : ил.
3. PHP/ MySQL для начинающих . Пер. с англ.. – М. КУДИЦ-ОБРАЗ, 2005. – 384с.
4. Котеров Д. В. Самоучитель PHP 4. — СПб.: БХВ-Петербург, 2003. — 576 с.:
5. Аргерих Л. и др. Профессиональное PHP программирование, 2-е издание. - Пер. с англ. - СПб: Символ-Плюс, 2003. - 1048 с., ил.
6. Разработка Web-приложений на PHP и MySQL: Пер. с англ../ Томсон Л., Веллинг Л. – К.: Издательство «Диасофт», 2002. – 672 с.

9. Опрацювання POST запитів за допомогою PHP

1. Мета роботи: вивчення основ роботи зі змінними, операцій виведення та обробки POST-запитів за допомогою PHP.

2. Завдання до роботи:

1. Створити HTML-форму для генерування POST-запитів відповідно до індивідуального завдання.
2. Створити програму, яка виводить значення введене чи вибране користувачем в наступних елементах HTML-форм: текстовое поле, текстова область, радіокнопка, прапорець, список.
3. Створити програму, яка здійснює обрахунок виразу відповідно до індивідуального завдання та виводить результат у вигляді HTML-коду.
4. У звіті навести:
    1) Зображення, що ілюструють роботу програми,
    2) Код програми.

3. Обладнання та матеріали: Персональний комп'ютер, сервер Apache + PHP.

4. Теоретична частина

4.1. Елементи HTML-форми

При введенні даних у форму використовуються різні керуючі елементи. В одних елементах користувач вводить інформацію з клавіатури, в інших він вибирає потрібний варіант, клацаючи кнопкою миші. У формах можуть бути присутніми приховані поля, вміст яких не повинен змінюватися користувачем.

Одна сторінка може містити кілька форм, тому необхідні можливості, які дозволяли б відрізнити одну форму від іншої. Більше того, ви повинні якось повідомити формі, куди слід відіслати данні, коли користувач виконує дію з формою (як правило, натискає кнопку відправки даних). Обидва завдання розв'язуються наступним тегом HTML:

<Form action = "дія" method = "метод (POST/GET)">
….
елементи форми
….
</ form>

Як видно з наведеного фрагмента, в тегах форм вказуються два важливих елементи: дія і метод. Дія вказує, який сценарій повинен обробляти форму, а метод визначає спосіб передачі даних цим сценарієм.

Текстове поле

У текстових полях зазвичай вводиться коротка текстова інформація - скажімо, адреса електронної пошти, поштова адреса або ім'я. Синтаксис визначення текстового поля:

<input type="text" name="імя_змінної" size="N" maxlength="N" value=""> .

Визначення текстового поля містить п'ять атрибутів:

1. type - тип елемента (для текстових полів - text);
2. name - ім'я змінної, в якій зберігаються введені дані;
3. size - загальний розмір текстового поля в браузері;
4. maxlength - максимальна кількість символів, що вводяться в текстовому полі;
5. value - значення, що відображається в текстовому полі за замовчуванням.

Рис. 1. Текстове поле

Особливим різновидом текстових полів є поле для введення паролів. Воно працює так само, як звичайне текстове поле, проте введені символи замінюються зірочками. Щоб створити у формі поле для введення паролів, досить вказати type = "password" замість type = "text".

Текстова область

Текстова область (text area) використовується для введення кілька великих обсягів тексту, що не обмежуються простим ім'ям або адресою електронної пошти. Синтаксис визначення текстової області:

<textarea name=" імя_змінної " rows="N" cols="N" value=""> </ textarea>

Визначення текстового поля містить три атрибути:

1. name - ім'я змінної, в якій зберігаються введені дані;
2. rows - кількість рядків к текстовій області;
3. cols - кількість стовпців у текстовій області.

Рис. 2. Текстова область

Елементи форм, орієнтовані на введення з миші

В інших елементах форм користувач вибирає один із заздалегідь визначених варіантів за допомогою миші.

Прапорець

Прапорці (checkboxes) використовуються в ситуаціях, коли користувач вибирає один або кілька варіантів з готового набору - за аналогією з тим, як ставляться «галочки» в анкетах. Синтаксис визначення прапорця:

<input type="checkbox" name="імя_змінної " value="початкове_значення">

Визначення прапорця містить три атрибути:

1. type - тип елемента (для прапорців - checkbox);
2. name - ім'я змінної, в якій зберігаються введені дані (у даному випадку - стан елемента);
3. value - значення, що привласнюється змінній за замовчуванням. Якщо прапорець встановлений, саме це значення буде присвоєно змінній з вказаним ім'ям. Якщо прапорець не встановлений, значення атрибуту value не використовується.

Рис. 3. Прапорець

Перемикач

Перемикач (radio button) являє собою різновид прапорця, він працює практично так само за одним винятком - у будь-який момент часу в групі може бути встановлений лише один перемикач. Синтаксис визначення перемикача:

<input type="radio name="ім'я_змінної "
value="початкове_значення">

Як бачимо, синтаксис майже не відрізняється від визначення прапорця.

Визначення перемикача поля містить три атрибути:

1. type - тип елемента (для перемикачів - radio);
2. name - ім'я змінної, в якій зберігаються введені дані (у даному випадку - стан елемента);
3. value - значення, що привласнюється змінної за замовчуванням. Якщо перемикач встановлений, саме це значення буде присвоєно змінної з вказаним ім'ям. Якщо прапорець не встановлений, значення атрибуту value не використовується.

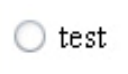


Рис. 4. Перемикач

Список, що розкривається

Списки, що розкриваються, особливо зручні в ситуації, коли у вас є довгий перелік допустимих варіантів, з якого користувач повинен вибрати один варіант. Як правило, списки, що розкриваються застосовуються при роботі з відносно великими наборами даних - наприклад, при перерахуванні американських штатів або країн. Синтаксис визначення списку:

<select name=" ім'я_змінної ">
<option value=" ім'я_змінної 1 "> назва елементу1</option>
<option value=" ім'я_змінної 2"> назва елементу2</option>
<option value=" ім'я_змінної 3"> назва елементу3</option>
<option value=" ім'я_змінної N"> назва елементу N</option>
</ Select>

Визначення перемикача поля містить три атрибути:

1. name - ім'я змінної, в якій зберігаються введені дані (у даному випадку - рядок, обрана у списку);
2. value - значення, яке відображається в списку за замовчуванням.

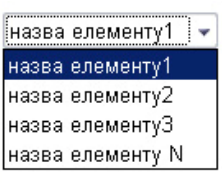


Рис. 5. Список, що розкривається

Приховані поля

Приховані поля не відображаються в браузері і зазвичай використовуються для передачі даних між сценаріями. Втім, приховані поля також використовуються в деяких ситуаціях і тому заслуговують на увагу.

Синтаксис визначення прихованого поля практично ідентичний синтаксису текстових полів, відрізняється тільки атрибут типу. Оскільки приховані поля не відображаються в браузері, навести приклад на сторінках книги неможливо. Синтаксис визначення прихованого поля:

<input type="hidden" name="ім'я змінної " value=" початкове значення">

Визначення прихованого поля містить три атрибути:

1. type - тип елемента (для прихованих полів - hidden);
2. name - ім'я змінної, в якій зберігаються приховані дані;

3. value - значення, за замовчування зберігається в прихованому полі.

Взагалі, назва цього елемента - приховане поле - дещо неточна. Хоча приховані поля не відображаються в браузерах,

користувач може просто виконати команду View Source і побачити, які приховані значення зберігаються у формі.

Кнопка відправки даних

Кнопка відправки даних ініціює дію, задану атрибутом action тега <form>. Синтаксис визначення:

<input type="submit" value=" текст_на_кнопці">

Визначення кнопки містить два атрибути:

1. type - тип елемента (для кнопки відправки даних - submit);
2. value - текст за замовчуванням відображається на кнопці.

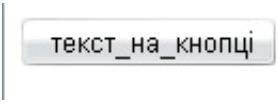


Рис. 6. Кнопка відправки даних

Кнопка скидання

Кнопка скидання скасовує всі зміни, внесені до елемента форми. Синтаксис визначення:

<input type="reset" value =" текст_на_кнопці ">

Визначення кнопки містить два атрибути:

1. type - тип елемента (для кнопки скидання - reset);
2. value - текст за замовчуванням відображається на кнопці.

Кнопка скидання виглядає точно так само, як і кнопка відправки даних, якщо не рахувати того, що на ній зазвичай виводиться слово «Reset» (рис. 6).

5. Порядок виконання роботи

7. Відкрити порожню XTML-сторінку, створивши текстовий файл із наступним вмістом:

```
<! Doctype HTML public "- / / W3C / / DTD HTML 4.01 / / EN"
"Http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title> Моя PHP-програма </ title>
</ Head>
<body>
</ Body>
</ Html>
```

8. Зберегти файл у відповідній папці сервера. Файл назвати ‘lab2.php’.

9. Додати дескриптори, які будуть обмежувати PHP-код та код необхідної форми. Наприклад, для додавання двох чисел форма буде виглядати наступним чином:

```
<! Doctype HTML public "- / / W3C / / DTD HTML 4.01 / / EN"
"Http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title> Моя PHP-програма </ title>
</ Head>
<body>
<form action=”http://<сервер>/<папка>/lab2.php” method=”POST”>
<input type=”text” value=”2” name=”zminna1”>
<input type=”text” value=”3” name=”zminna2”>
<input type="submit" value="Відправити">
</form>
<br />
<?php
?>
</ Body>
</ Html>.
```

10. Організувати підрахунок виразу з індивідуального завдання, використовуючи дані з POST-запиту. Розглянемо, наприклад, підрахунок виразу a + b, де значення a і b передаються за допомогою POST-запиту за допомогою створеної форми:

Приклад програми:

```
....
<?php
$a = $_POST[‘zminna1’]; // Зчитування змінної a
$b = $_POST[‘zminna2’]; // Зчитування змінної b
$rez = $a + $b; // Обрахунок виразу
echo ‘Результат:’.$rez; // Виведення результату
?>
….
```

11. Перевірити правильність роботи програми, відкривши її в браузері з передачею відповідних змінних.

Остаточний код прикладу буде мати вигляд:

```
<! Doctype HTML public "- / / W3C / / DTD HTML 4.01 / / EN"
"Http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title> Моя PHP-програма </ title>
</ Head>
<body>
<form action=”http://<сервер>/<папка>/lab2.php”
method=”POST”>
<input type=”text” value=”2” name=”zminna1”>
<input type=”text” value=”3” name=”zminna2”>
<input type="submit" value="Відправити">
</form>
<br />
<?php
$a = $_POST[‘zminna1’]; // Зчитування змінної a
$b = $_POST[‘zminna2’]; // Зчитування змінної b
$rez = $a + $b; // Обрахунок виразу
echo ‘Результат:’.$rez; // Виведення результату
?>
</ Body>
</ Html>
```

12. Додати до форми текстову область, радіокнопку, прапорець та список. Організувати виведення значень із даних елементів із відповідними поясненнями. Наприклад: «Значення списку: тестове_значення»

6. Індивідуальні завдання

Номер завдання	Опис завдання
1	(a+c)/b
2	a*c*b
3	a+c+b
4	a+c/b
5	a*c/b
6	a/(b-c)
7	a/c/b
8	a-c-b
9	a+c-b
10	a-c*b
11	a*c-b
12	a*c/b
13	a/b-c
14	a*c-b
15	c/b-a
16	c+b*a

Запитання для контролю

1. Як відрізняється зчитування параметрів POST- від GET-запиту?
2. Передбачити можливість збереження обраної опції елемента списку після відправлення запиту.
3. Передбачити можливість збереження введеного значення текстового поля після відправлення запиту.
4. Створити декілька радіокнопок та передбачити можливість збереження обраної кнопки після відправлення запиту.
5. Створити декілька прапорців та передбачити можливість збереження обраних прапорців після відправлення запиту.

Список рекомендованої літератури

1. PHP5 и MySQL. Библия пользователя.: пер. с англ. – М.: издательский дом «Вильямс», 2006. – 1216 с.: ил.
2. MySQL / Л. Ульман; пер. с англ. Слинкина А.А. – М: ДМК Пресс; Спб.: Питер, 2004 – 352 с. : ил.
3. PHP/ MySQL для начинающих . Пер. с англ.. – М. КУДИЦ-ОБРАЗ, 2005. – 384с.
4. Котеров Д. В. Самоучитель PHP 4. — СПб.: БХВ-Петербург, 2003. — 576 с.:
5. Аргерих Л. и др. Профессиональное PHP программирование, 2-е издание. - Пер. с англ. - СПб: Символ-Плюс, 2003. - 1048 с., ил.
6. Разработка Web-приложений на PHP и MySQL: Пер. с англ../ Томсон Л., Веллинг Л. – К.: Издательство «Диасофт», 2002. – 672 с.

10. Програма табулювання функцій на мові PHP”

1. Мета роботи: вивчення умовних операторів, циклів і генерування HTML-коду на мові PHP.

2. Завдання до роботи:

1. Створити форму, яка дозволяє користувачу ввести початкове і кінцеве значення та крок табулювання. Передбачити можливість вибору функції користувачем (кожне завдання складеться з трьох функцій згідно з індивідуальним завданням).
2. Створити програму, яка здійснює табулювання функції та виводить результат у вигляді HTML-таблиці. Табулювання проводиться для функції обраної користувачем.
3. У звіті навести:
    1) Зображення, що ілюструють роботу програми,
    2) Код програми.

3. Обладнання та матеріали: Персональний комп’ютер, сервер Apache +PHP.

4. Теоретична частина

4.1. Створення таблиць в HTML

Для опису таблиць використовується тег <TABLE>. Тег <TABLE>, як і багато інших, автоматично переводить рядок до і після таблиці.

Створення рядка таблиці - тег <TR> Тег <TR> (Table Row, рядок таблиці) створює рядок таблиці. Весь текст, інші теги і атрибути, які потрібно помістити в один рядок, повинні розміщатися між тегами <TR> </ TR>.

Визначення елементів таблиці - тег <TD> Всередині рядка таблиці зазвичай розміщуються осередки з даними. Кожна клітинка, яка містить текст або зображення, повинна бути оточена тегами <TD> </ TD>. Кількість тегів <TD> </ TD> у рядку визначає Кількістькомірок (відкрити)

<TABLE>
<TR>
<TD COLSPAN=3> Якщо в таблиці два теги TR, то в ній два рядки. </ TD>
</ TR>
<TR>
<TD> Якщо в рядку три теги TD, </ TD>
<TD> То в ній </ TD>
<TD> Три стовпці. </ TD>
</ TR>
</ TABLE>

Заголовки стовпців таблиці - тег <TH>

Заголовки для стовпців і рядків таблиці задаються за допомогою тега заголовка <TH> </ TH> (Table Header, заголовок таблиці). Ці теги подібні <TD> </ TD>. Відмінність полягає в тому, що текст, укладений між тегами <TH> </ TH>, автоматично записується жирним шрифтом і за замовчуванням розташовується посередині комірки. Центрування можна скасувати і вирівняти текст по лівому або правому краю. Якщо скористатися <TD> </ TD> з тегом <B> і атрибутом <ALIGN = center>, текст теж буде виглядати як заголовок. Проте слід мати на увазі, що не всі браузери підтримують у таблицях жирний шрифт, тому краще ставити заголовки таблиць за допомогою <TH>.

<TABLE>
<TR>
<TH> Тема центрований за замовчуванням </ TH>
<TH COLSPAN=2> Заголовок може об'єднувати стовпці </ TH>
</ TR>
<TR>
<TH> Заголовок може бути розташований перед стовпцями </ TH>
<TD> Текст або дані </ TD>
<TD> Текст або дані </ TD>
</ TR>
<TR>
<TH ROWSPAN=3> Заголовок може об'єднувати рядка </ TH>
<TD> Текст або дані </ TD>

<TD> Текст або дані </ TD>
</ TR>
<TR>
<TD> Текст або дані </ TD>
<TD> Текст або дані </ TD>
</ TR>
<TR>
<TD> Текст або дані </ TD>
<TD> Текст або дані </ TD>
</ TR>
</ TABLE>

Використання заголовків таблиці - тег <CAPTION>

Тег <CAPTION> дозволяє створювати заголовки таблиці. За замовчуванням заголовки центруються і розміщуються або над (<CAPTION ALIGN = top>), або під таблицею (<CAPTION ALIGN = bottom>). Заголовок може складатися з будь-якого тексту і зображень. Текст буде поділений на рядки, відповідні ширині таблиці. Іноді тег <CAPTION> використовується для підпису під рисунком. Для цього досить описати таблицю без меж.

<TABLE>
<CAPTION ALIGN=top> Тема над таблицею </ CAPTION>
<TR>
<TD> Текст або дані </ TD>
<TD> Текст або дані </ TD>
<TD> Текст або дані </ TD>
<TD> Текст або дані </ TD>
</ TR>
</ TABLE>

<TABLE>
<CAPTION ALIGN=bottom> Тема під таблицею </ CAPTION>
<TR>
<TD> Текст або дані </ TD>
<TD> Текст або дані </ TD>
<TD> Текст або дані </ TD>
</ TR>
</ TABLE>

4.2. Математичні функції PHP

Бібліотека математичних функцій PHP реалізує методи для тригонометричних обчислень, числових перетворень і числових операцій. Тригонометричні функції сприймають параметри в радіанах, але існують функції перетворення градусів у радіани і навпаки.

abs () - абсолютне значення числа.
echo abs (-0.7); / / Виводить 0.7

acos () - арккосинус, виражений у радіанах.
echo acos (-0.7); / / Виводить 2.3461938234056

asin () - арксинус, виражений у радіанах.
echo asin (-0.7); / / Виводить -0.77539749661075

atan () - арктангенс, виражений у радіанах.
echo atan (-0.7); / / Виводить -0.61072596438921

atan2 () - арктангенс для координат x і y, виражений у радіанах. Відмінність від виразу atan (y / x) полягає в тому, що знаки обох параметрів використовуються для визначення квадранта результату.

echo atan (-0.7 / 2); / / Виводить -0.33667481938673
echo atan2 (2, -0.7); / / Виводить 1.9074711461816

ceil () - заокруглення числа в більшу сторону.
echo ceil (2.35); / / Виводить 3

cos () - косинус аргументу, вираженого в радіанах.
echo cos (2.35); / / Виводить -0.70271307677355

deg2rad () - перетворює градуси в радіани.
echo deg2rad (90); / / Виводить 1.5707963267949

exp () - експонента числа.

echo exp (1); / / Виводить 2.718281828459

floor () - заокруглення числа в меншу сторону.
echo floor (2.99); / / Виводить 2

log () - натуральний логарифм.
echo log (exp (1)); / / Виводить 1

log10 () - десятковий логарифм.
echo log (1000); / / Виводить 3

max () - найбільше значення з списку параметрів. Можливе порівняння необмеженої кількостізначень. Як параметр може бути заданий масив.
echo max (12.23, 42.554, 58.234, 34.31); / / Виводить 58.234
$ A = array (12.23, 42.554, 58.234, 34.31);
echo max ($ a); / / Виводить 58.234

min () - найбільше значення з списку параметрів. Можливе порівняння необмеженої кількості значень. Як параметр може бути заданий масив.
echo min (12.23, 42.554, 58.234, 34.31); / / Виводить 12.23
$ A = array (12.23, 42.554, 58.234, 34.31);
echo min ($ a); / / Виводить 12.23

4.3. Цикл while PHP

Ось базова форма оператора while: while (оператор) {дії}

Значення оператора while виконується наступним чином. Значення виразу перевіряється щоразу на початку циклу і дії виконуються поки значення оператора TRUE.

Наступний приклад друкує числа від 1 до 10:

```
$ I = 1;
while ($ i <= 10) {
print $ i + + / * буде друкуватися значення
}
```

5. Порядок виконання роботи

1. Згідно з індивідуальним завданням запрограмувати табулювання функції, скориставшись прикладом, який наведено нижче.

```
<table>
<tr><td>Аргумент</td><td>Результат</td></tr>
<?php
$a=1;
$b=6;
$h=1;
$x=$a;
while ($x<=$b) {
$y = sin($x);
echo "<tr><td>$x</td><td>$y</td></tr>";
$x+=$h;
}
?>
</table>.
```

2. Додати до коду HTML-форму, за допомогою якої можна ввести крок табулювання, верхню та нижню межі.

3. Забезпечити передачу даних з HTML-форми у змінні $a (нижня межа), $b (верхня межа), $h (крок табулювання).

4. Поліпшити візуальне оформлення таблиці та форми.

6. Індивідуальні завдання

Номер завдання	Вираз для обчислення
1	$\frac{\sin(x)}{(x-3)} + x$
2	$\frac{\sqrt{x}}{(x+5)}$
3	$\sqrt{x} + x^2$

4	$\cos(x)^2 + \sqrt{x}$
5	$\cos(x)\sin(x) + \sqrt{x}$
6	$\cos(\lvert x \rvert) + \sqrt{x}$
7	$x\sqrt{x+5} + x$
8	$\sqrt{x + \lvert\cos(x)\rvert}$
9	$x^4 + 5x + \frac{8}{x}$
10	$\cos(x^2) + \frac{\sin(x)}{x}$
11	$e^x + \sqrt{x}\,\frac{x+5}{x-5}$
12	$\sin(\sqrt{x}) + \frac{x}{x+2}$
13	$\lvert\sin(x) + \cos(x) + \sqrt{x}\rvert$
14	$\frac{x^5 + x^4}{x}$
15	$\frac{\sqrt{x+5}}{10} + x^2$

Запитання для контролю

1. Забезпечити перевірку неправильно введених вхідних значень із виведенням відповідного повідомлення.
2. Вивести колонку-аргумент із відповідними значеннями жирним шрифтом.
3. Передбачити обрахунок за заздалегідь визначеними значеннями у випадку, якщо користувач не ввів жодних значень.
4. Додатково виводити суму значень аргументу чи функції.
5. Замінити текстове поле, яке відповідає за введення кроку списком з декількома значеннями.

Список рекомендованої літератури

1. PHP5 и MySQL. Библия пользователя.: пер. с англ. – М.: издательский дом «Вильямс», 2006. – 1216 с.: ил.
2. MySQL / Л. Ульман; пер. с англ. Слинкина А.А. – М: ДМК Пресс; Спб.: Питер, 2004 – 352 с. : ил.
3. PHP/ MySQL для начинающих . Пер. с англ.. – М. КУДИЦ-ОБРАЗ, 2005. – 384с.
4. Котеров Д. В. Самоучитель PHP 4. — СПб.: БХВ-Петербург, 2003. — 576 с.:
5. Аргерих Л. и др. Профессиональное PHP программирование, 2-е издание. - Пер. с англ. - СПб: Символ-Плюс, 2003. - 1048 с., ил.
6. Разработка Web-приложений на PHP и MySQL: Пер. с англ../ Томсон Л., Веллинг Л. – К.: Издательство «Диасофт», 2002. – 672 с.

11. Створення сторінки для голосування на мові PHP

1. Мета роботи: вивчення умовних операторів, циклів, масивів, функцій для роботи з рядками та файлами мови PHP.

2. Завдання до роботи:

1. Створити файл з формою для голосування на довільну тему, що містить кількість питань згідно з індивідуальним завданням.
2. Створити файл, який обробляє дані з форми, виводить результати голосування та зберігає їх у текстовому файлі, оновлюючи його зміст.
3. Вдосконалити програми, зробивши неможливим багатократне голосування з одного браузера, використовуючи сесії.
4. У звіті навести:
    1) Зображення, що ілюструють роботу програми,
    2) Код програми.

3. Обладнання та матеріали: Персональний комп'ютер, сервер Apache +PHP.

4. Теоретична частина

4.1. Функції, що використані в роботі

count - Підраховує кількість елементів масиву або кількість властивостей об'єкта

Синтаксис: int **count** (mixed var [, int mode])

Повертає кількість елементів змінної var, яка зазвичай є array , або будь-яким іншим об'єктом, який може містити хоча б один елемент.

file - Читає вміст файла filename і поміщає його в масив

Синтаксис: array **file** (string filename [, int use_include_path [, resource context]])

fopen - Відкриває файл або URL

Синтаксис: resource **fopen** (string filename, string mode [, bool use_include_path [, resource zcontext]])

fopen () закріплює ресурс, зазначений в аргументі filename, за потоком. Якщо filename переданий у формі "scheme ://...", він вважається URL'ом і PHP проведе пошук обробника протоколу для цієї схеми.

fwrite - Binary-safe запис файла.

Синтаксис: int **fwrite** (int fp, string string [, int length])

fwrite () записує вміст рядка string у потік файла, специфікованого дескриптором fp. Якщо аргумент length заданий, запис буде зупинено після запису length кількості байтів або досягнення кінця string. fwrite () повертає кількість записаних байтів, або -1 при помилці.

fclose - Закриває дескриптор файла.

Синтаксис: bool fclose (resource handle).

Функція закриває файл, на який вказує handle. Повертає TRUE в разі успішного завершення або FALSE в разі виникнення помилки.

5. Порядок виконання роботи

Голосування - це засіб дізнатися думку Ваших відвідувачів з різних питань. Для прикладу візьмемо просте питання - Ваша думка про сайт. Варіанти відповідей надамо наступні:

- Круто!
- Нормально.
- Мені все одно.
- Це щось страшне.

Для роботи нам знадобиться створити три файли. У першому будемо запитувати відвідувача про його думку, у другому зберігати результати і в третьому - виводити їх і обробляти. Перший файл буде мати ім'я index.php, другий - golos.txt і третій - golos.php.

index.php

Файл index.php містить потрібну форму.

```
<FORM METHOD="POST" action="golos. php ">
<TABLE BORDER=1> <TR> <TD> <TABLE BORDER=0>
<TR> <TD> Ваша думка про сайт? </ TD> </ TR>
<TR> <TD> <INPUT TYPE=radio NAME=answer VALUE=1>
Круто! </ TD> </ TR>
<TR> <TD> <INPUT TYPE=radio NAME=answer VALUE=2>
Нормально </ TD> </ TR>
<TR> <TD> <INPUT TYPE=radio NAME=answer VALUE=3>
Мені все рівно </ TD> </ TR>
<TR> <TD> <INPUT TYPE=radio NAME=answer VALUE=4> Це
щось страшне! </ TD> </ TR> <TR> <TD> <INPUT TYPE=Submit
NAME=vote VALUE=" відправити "> </ TD> </ TR>
<TR> <TD> <INPUT TYPE=Submit NAME=result
VALUE="смотреть результат ">
</ TD> </ TR> </ TABLE> </ TD> </ TR> </ TABLE> </ FORM>
```

Цей код можна легко інтегрувати в будь-які сторінки і правити відповідно до індивідуального завдання. Тільки потрібно звертати увагу на відповідність імен і значень змінних у формі. І що найголовніше - ці форми повинні передаватися в скрипт PHP з обробкою результатів, тому ім'я файла в action повинно відповідати імені файла зі скриптом.

golos.txt

Файл буде містити чотири рядки (по кількості варіантів відповідей) і вважати їх номери відповідними номером обраного варіанта відповіді. Це можна зчитувати з допомогою команди PHP введення файлу в масив, при цьому не забувайте, що масив завжди починається з індексу нуль. Відповідно - перший рядок нашого файлу не буде використаний, і ввести туди можна все що завгодно. Наприклад: Результати голосування. Файл буде заповнено автоматично.

golos.phtml

Файл golos.php буде містити код скрипта.

```
<? Php
$ File = "golos.txt";
$answer = $_POST['answer'];
$ A = file ($ file);
$ I = 1; $ fi = Count ($ a);
$ N = 0;
while ($ i <= $ fi):
$ A [$ i] = trim (str_replace ("\ n ","", $ a [$ i]));
$ N = $ n + $ a [$ i];
$ I + +;
endwhile;

if ($ answer! = "") {
echo "<br> Дякую, Ваша думка врахована:";
$ A [$ answer] + +; $ n + +;

$ Rez = "Результати голосування! \ N". $ A [1].
"\ N". $ A [2]. "\ N". $ A [3]. "\ N". $ A [4];
$ Fp = @ fopen ($ file, "w");
if ($fp) { $counter=fwrite($fp,$rez); fclose($fp); }
else {echo "Сталася помилка запису результатів!";}

} Else {echo "<br> Результати голосування";}
echo "<br> Круто! - <b>". $ a [1 ]."</ b> ";
echo "<br> Так собі - <b>". $ a [2 ]."</ b> ";
echo "<br> Зійде - <b>". $ a [3 ]."</ b> ";
echo "<br> Це щось страшне! - <b>". $ a [4 ]."</ b> ";
echo "<br> <br> Всього проголосувало:". $ n;
?>
```

У перших рядках ми визначаємо ім'ям файла результати і зчитуємо результати голосування в масив даних з ім'ям $ a. Далі йде цикл, в якому ми обробляємо отриманий масив так, щоб він не містив символів переведення каретки (введення рядка) та пробіли: $ a [$ i] = trim (str_replace ("\ n ","", $ a [$ i])); Паралельно ведемо підрахунок кількості виборців, що нескладно, тому що це просто сума значень нашого масиву. Видаляти символи введення із пропуску необхідно для перетворення даних з символьного рядка в

ціле число. Це можна зробити різними методами, але в цьому випадку просто видаляються символи "\ n" (що в PHP відповідають переводення рядка) і видаляються пробіли з початку і з кінця рядка функцією trim (). Результати заносяться назад у масив, але вже у вигляді цілочисельного значення, яке можна скласти, збільшити, ділити і т.д. Нас буде цікавити збільшення на одиницю певного елемента масиву, номер якого (його індекс) зберігається у змінній $ answer, яка у свою чергу прийшла до нас із форми.

Після обробки отриманого масиву скрипт повинен прийняти рішення щодо обраного відвідувачем режиму - або просто показати результати, або додати голос у відповідну позицію. Досягається це перевіркою змінної $ answer, в якій зберігається значення вибраного відвідувачем сайту варіанти голосування. Якщо ця змінна порожня, значить була натиснута кнопка показу результатів, і скрипт пропустить блок підрахунку голоси. Якщо змінна $ answer не порожня, в ній міститься номер обраного варіанта голосування, а значить, ми можемо просто збільшити на одиницю значення потрібної комірки масиву: $ a [$ answer] + +; Крім того потрібно збільшити значення кількості виборців для того, що б врахувати поточний голос.

Для запису результату в файл спочатку відкривається з'єднання з файлом: $ fp = fopen ($ file, "w"); Символ w вказує на необхідність очищення вмісту файлу перед записом. Якщо з'єднання не встановлено, виводиться повідомлення про помилку, якщо встановлено - попередньо відформатований значення змінної $ rez записується у файл. Змінна $ rez формується так: значення всіх комірок масиву (крім найпершої - нульовий, яка не використовується) склеюється так, щоб роздільником був символ нового рядка. Це дозволить надалі коректно вважати отриманий таки чином файл.

У кінці скрипту результати виводяться на екран.

5.1. Хід виконання роботи

1. Розробити питання до голосування в необхідній кількості.
2. Запрограмувати скрипт, використовуючи розроблені питання та наведений приклад.
3. Використовуючи сесії, унеможливити повторне голосування з одного і того ж браузера.

6. Індивідуальні завдання

Номер завдання	Кількість питань
1	5
2	6
3	8
4	6
5	5
6	8
7	7
8	5
9	6
10	8
11	5
12	6
13	8
14	6
15	5
16	8

Запитання для контролю

1. Як здійснюється запис/зчитування з файла?
2. Замінити кнопки для відповіді на одне з питань на прапорці.
3. Організувати поле для введення імені людини, що голосує, та відповідно виводити список тих, хто проголосував.
4. Вимагати від користувача надати відповіді на всі питання, у протилежному випадку результати не враховувати та виводити питання ще раз із відповідним повідомленням про помилку.
5. Пояснити структури текстового файла з відповідями.

Список рекомендованої літератури

1. PHP5 и MySQL. Библия пользователя.: пер. с англ. – М.: издательский дом «Вильямс», 2006. – 1216 с.: ил.
2. MySQL / Л. Ульман; пер. с англ. Слинкина А.А. – М: ДМК Пресс; Спб.: Питер, 2004 – 352 с. : ил.
3. PHP/ MySQL для начинающих . Пер. с англ.. – М. КУДИЦ-ОБРАЗ, 2005. – 384 с.
4. Котеров Д. В. Самоучитель PHP 4. — СПб.: БХВ-Петербург, 2003. — 576 с.:
5. Аргерих Л. и др. Профессиональное PHP программирование, 2-е издание. - Пер. с англ. - СПб: Символ-Плюс, 2003. – 1048 с., ил.
6. Разработка Web-приложений на PHP и MySQL: Пер. с англ../ Томсон Л., Веллинг Л. – К.: Издательство «Диасофт», 2002. – 672 с.

12. Створення бази даних MySQL з використанням програми PhpMyAdmin

1. Мета роботи: вивчення MySQL та PhpMyAdmin, оволодіння навиками проектування баз даних.

2. Завдання до роботи:

1. Створити базу даних згідно з індивідуальним завданням.
2. Заповнити створену базу даних тестовою інформацією.
3. У звіті навести:
   1) Схему спроектованої бази даних.
   2) Зображення, що ілюструють роботу програми PhpMyAdmin.
   3) SQL запити, згенеровані під час виконання завдання програмою PhpMyAdmin.

3. Обладнання та матеріали: Персональний комп'ютер, сервер Apache +PHP, PhpMyAdmin.

4. Теоретична частина

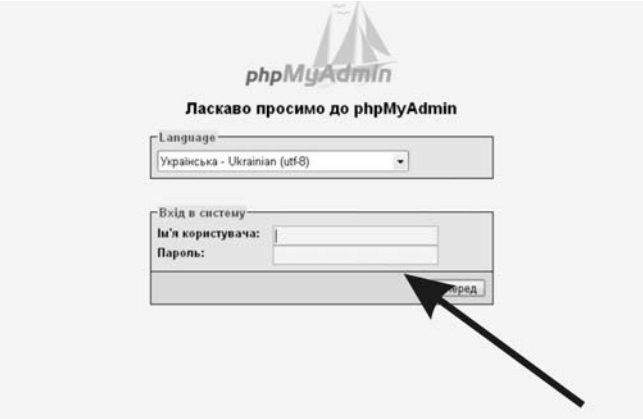
4.1. Програма PhpMyAdmin

PhpMyAdmin — веб-додаток з відкритим кодом, написаний на мові PHP, що являє собою веб-інтерфейс для адміністрування СКБД MySQL. PhpMyAdmin дозволяє через браузер здійснювати адміністрування серверу MySQL, запускати команди SQL та переглядати вміст таблиць й баз даних. Додаток користується великою популярністю у веб-розробників, оскільки дозволяє керувати СКБД MySQL без безпосереднього введення SQL-команд, надаючи дружній інтерфейс.

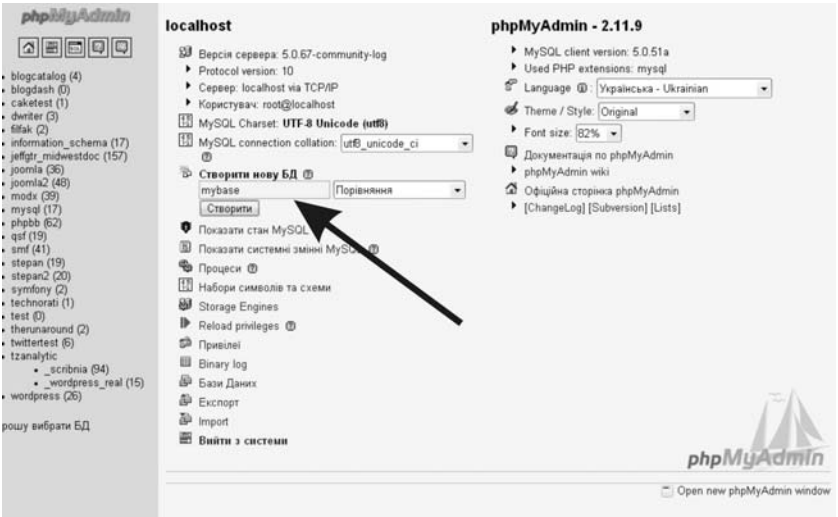
На сьогоднішній день PhpMyAdmin широко застосовується на практиці. Останнє пов'язано з тим, що розробники інтенсивно розвивають свій продукт з огляду на всі нововведення СКБД MySQL. Переважна більшість українських провайдерів використовують цей додаток як панель керування для того, щоб надати своїм клієнтам можливість адміністрування виділених їм баз даних.

5. Практична частина

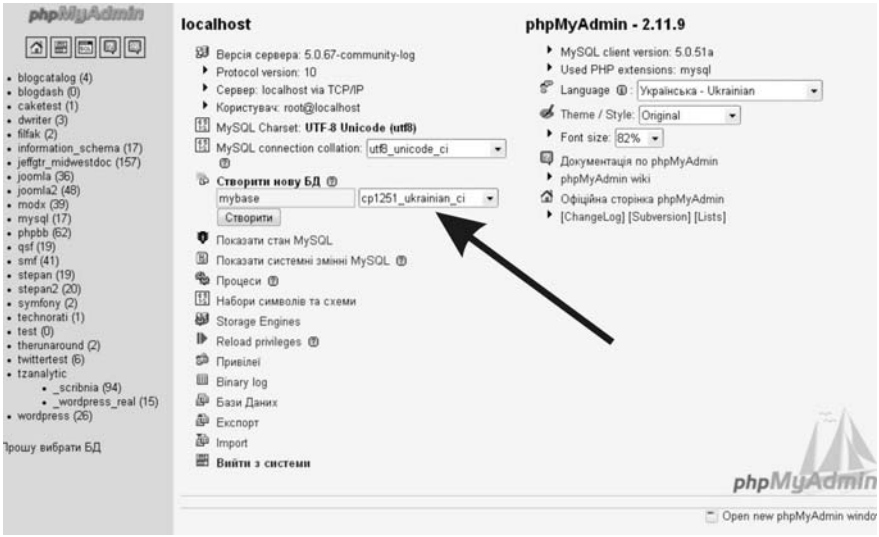
1. Розробіть логічну модель бази даних.
2. Увійдіть в PhpMyAdmin, використовуючи логін і пароль.



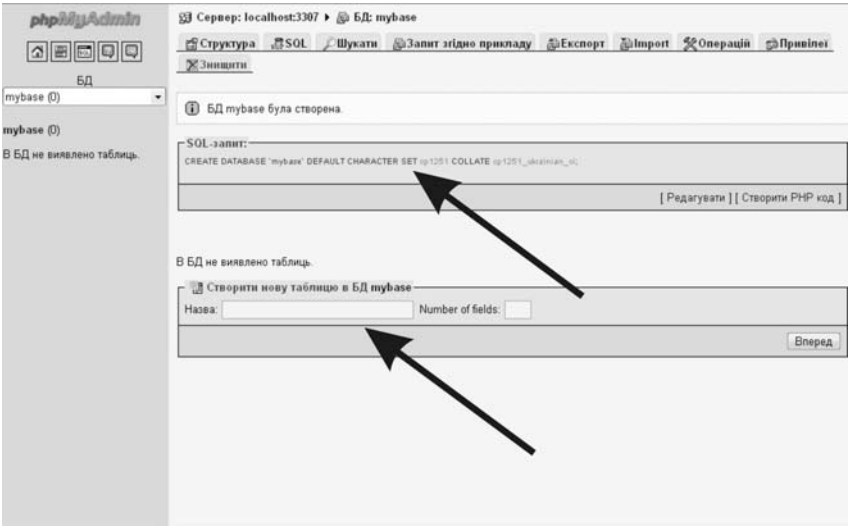
3. Введіть ім’я створюваної бази даних



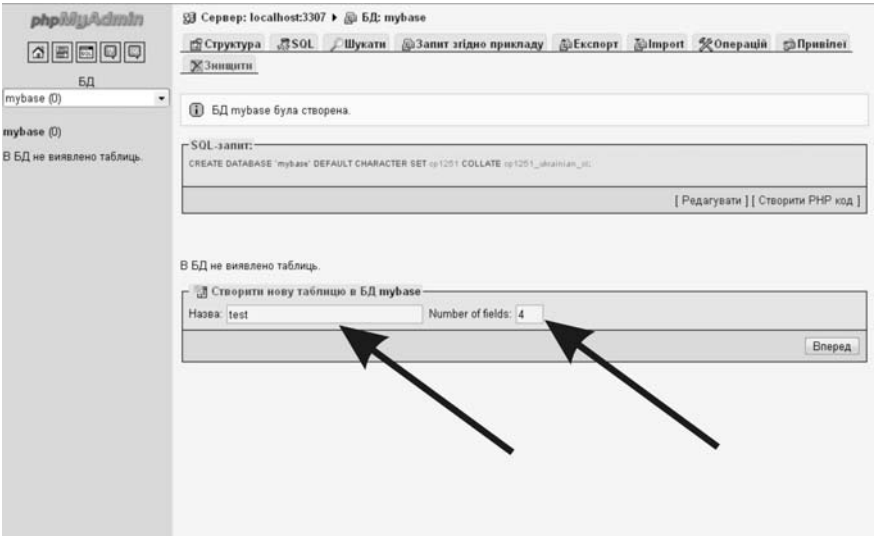
4. Виберіть україномовне кодування символів у створюваній базі даних та натисніть кнопку «Створити».



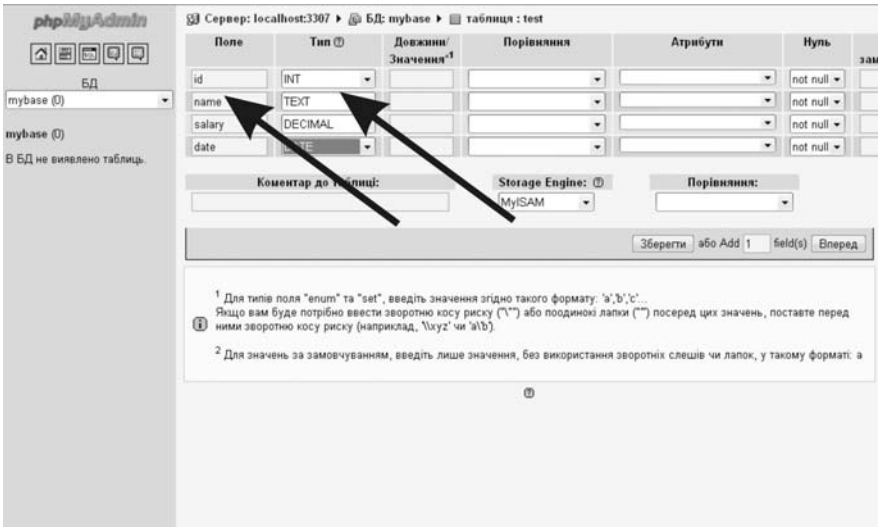
5. В наступному вікні ви побачите запит, який було виконано, та поле для створення таблиці.



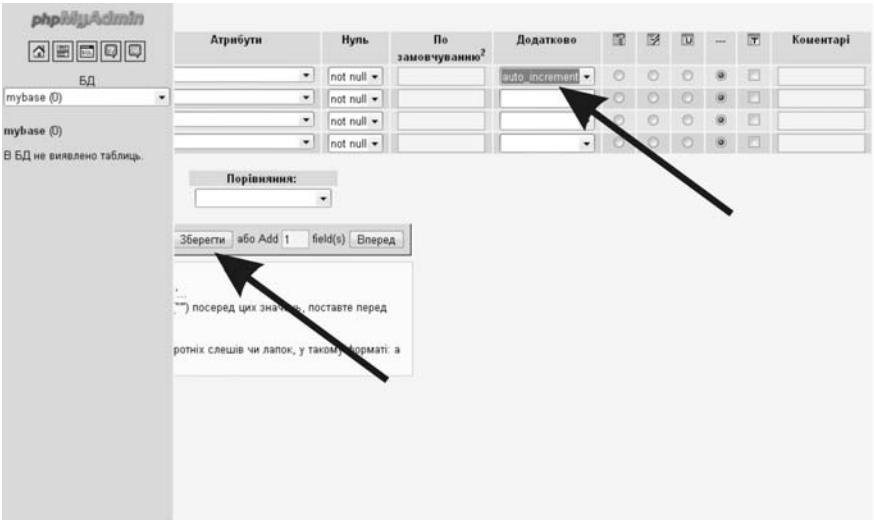
6. Введіть ім’я таблиці та кількість колонок у ній і натисніть кнопку «Створити».



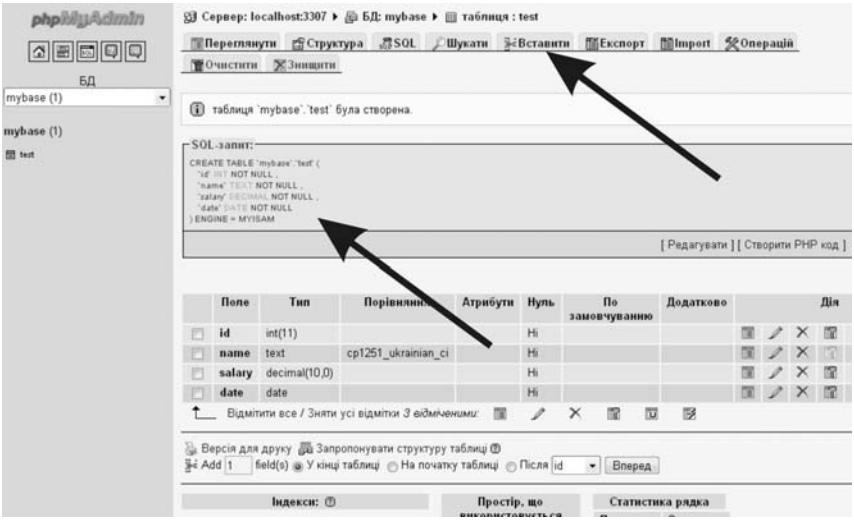
7. У наступному вікні ви побачите поля для введення назв колонок та їх типів, заповніть їх.



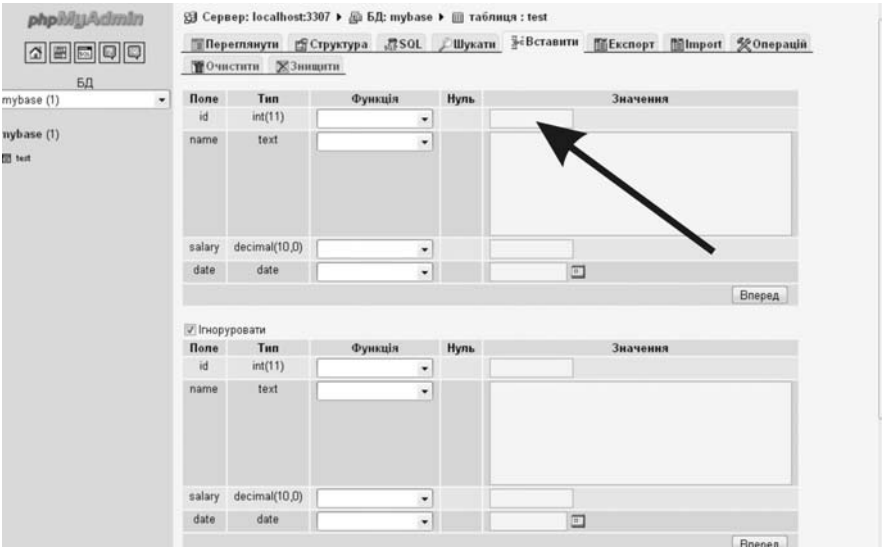
8. Для унікального ідентифікатора виберіть атрибут auto_increment та натисніть кнопку «Зберегти».



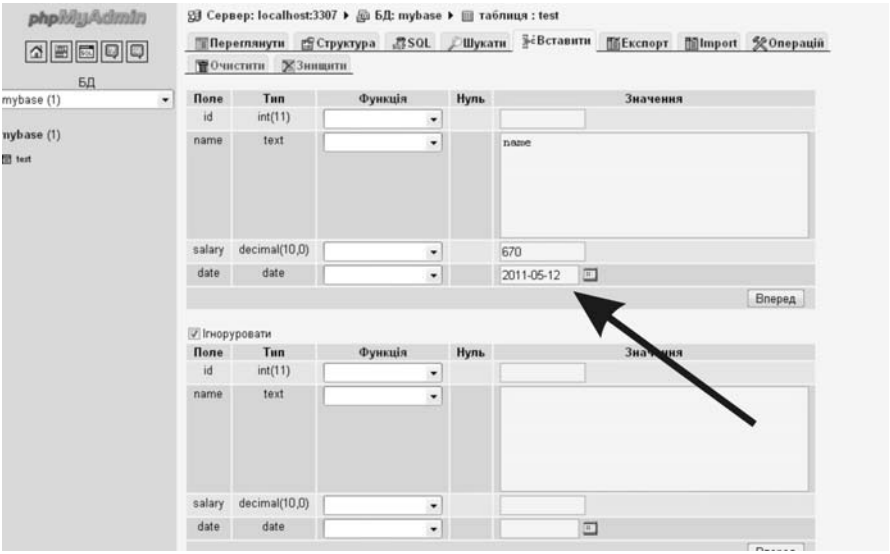
9. У наступному вікні ви побачите запит, що було виконано. Для вставки даних у таблицю натисніть кнопку «Вставити».



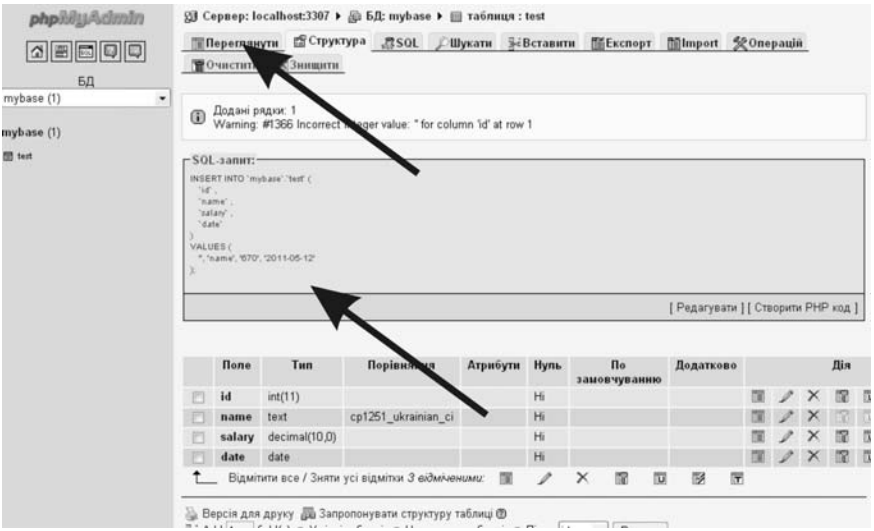
10. У наступному вікні ви побачите поля для введення даних. Поле унікального ідентифікатора можна не заповнювати, воно буде заповнене автоматично.



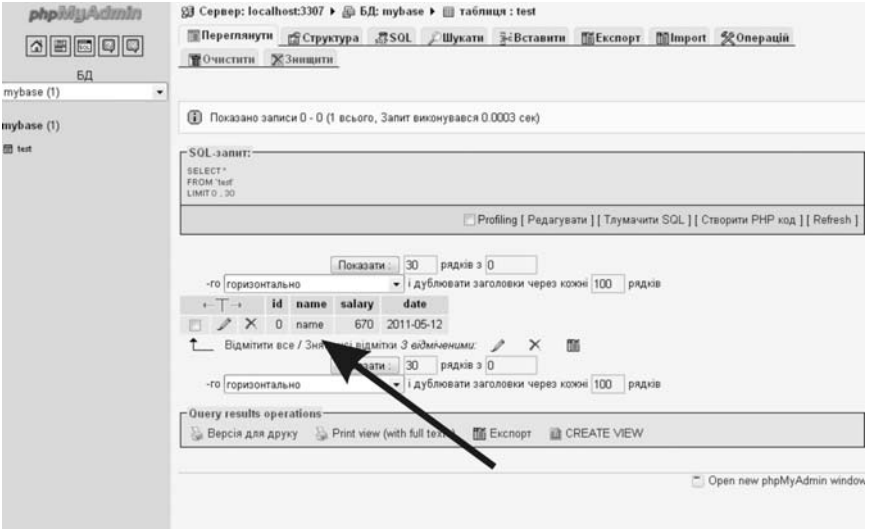
11. Заповніть інші поля та натисніть кнопку «Вперед».



12. В наступному вікні ви побачите виконаний запит. Для перегляду даних натисніть кнопку «Переглянути».



13. У наступному вікні ви побачите вміст таблиці.



14. Повторіть пункти 6-13 для інших таблиць.

6. Індивідуальні завдання

Номер завдання	Опис завдання
1	Розробити і створити базу даних, що містить інформацію про робітників та підрозділи фірми, належність робітників до певних підрозділів та хто з них є керівником якого підрозділу.
2	Розробити і створити базу даних, що містить інформацію про автомобілі, їх марки та моделі.
3	Розробити і створити базу даних, що містить інформацію про книжки, їх авторів та видавників.
4	Розробити і створити базу даних, що містить інформацію клієнтів комп'ютерної фірми, їх комп'ютери та встановлені на комп'ютерах операційні системи.
5	Розробити і створити базу даних, що містить інформацію про клієнтів фірми інтернет–провайдера, їх тарифні плани та операційні системи, встановлені на їх комп'ютерах.
6	Розробити і створити базу даних магазину, яка містить відомості про товар, його виробника та постачальника.
7	Розробити і створити базу даних мобільного оператора, яка містить відомості про клієнтів, моделі їх телефонів та тарифні плани.
8	Розробити і створити базу даних університету, яка містить відомості про студентів, факультет, де вони навчаються, та гуртки, які вони відвідують.

Запитання для контролю

1. Яка команда призначена для створення таблиці?
2. Яка команда призначена для вставки рядка в таблицю?
3. Яка команда призначена для зміни певних значень в рядку?
4. Яка команда призначена для видалення рядка?
5. Які типи даних можуть буди в таблиці БД?

Список рекомендованої літератури

1. PHP5 и MySQL. Библия пользователя.: пер. с англ. – М.: издательский дом «Вильямс», 2006. – 1216 с.: ил.
2. MySQL / Л. Ульман; пер. с англ. Слинкина А.А. – М: ДМК Пресс; Спб.: Питер, 2004 – 352 с. : ил.
3. PHP/ MySQL для начинающих . Пер. с англ.. – М. КУДИЦ-ОБРАЗ, 2005. – 384с.
4. Котеров Д. В. Самоучитель PHP 4. — СПб.: БХВ-Петербург, 2003. — 576 с.:
5. Аргерих Л. и др. Профессиональное PHP программирование, 2-е издание. - Пер. с англ. - СПб: Символ-Плюс, 2003. - 1048 с., ил.
6. Разработка Web-приложений на PHP и MySQL: Пер. с англ../ Томсон Л., Веллинг Л. – К.: Издательство «Диасофт», 2002. – 672 с.

13. Створення програми для тестування на мові PHP із використанням бази MySQL

1. Мета роботи: вивчення сесій мови PHP та організації зв'язку з БД MySQL.

2. Завдання до роботи:

5. Створити програму для тестування на довільну тему, що містить кількість питань та їх типи згідно з індивідуальним завданням.
6. У звіті навести:
   3) Зображення, що ілюструють роботу програми.
   4) Код програми.

3. Обладнання та матеріали: Персональний комп'ютер, сервер Apache +PHP.

4. Теоретична частина

4.1. Функції для роботи з MySQL

Для з'єднання нам потрібні:
1. логін користувача БД,
2. пароль користувача БД,
3. хост (зазвичай localhost) з БД,
4. база даних (попередньо створена в PhpMyadmin).

Створити зєднання дуже просто:

```
$user = "root";         // логін користувача БД
$password = "";         // пароль користувача БД
$host = "localhost";    //Хост
$db = "mysqli";         //БД
$sql = new mysqli($host,$user,$password,$db) .
```

Далі перевіряємо, чи з'єднання успішне. Якщо ні, то виводимо повідомлення про помилку та припиняємо виконання програми.

```
if (mysqli_connect_errno()) {
echo "Помилка з'єднання з БД: ". mysqli_connect_error();
exit.
}
```

Далі ми можемо виконати запит до БД за допомогою мови SQL. Наприклад, вибрати всі рядки з таблиці posts.

```
$result = $sql->query('select * from posts').
```

Далі кожен рядок у вигляді масиву записується у змінну $row. Всередині циклу здійснюються необхідні операції. Наприклад, виведення на екран.

```
while( $row = $result->fetch_assoc() ){
        var_dump($row); echo '<br />';
}
```

У кінці програми з'єднання з базою даних припиняється.

```
$sql->close();
```

5. Практична частина

Тестування складається з ряду запитань із варіантами відповідей або без них, на які користувач дає відповіді. Програма перевіряє, чи дав користувач правильні відповіді, і записує результат у базу даних. Запитання може мати одну правильну відповідь – тоді відповіді реалізуються за допомогою радіокнопок. Якщо у варіантах існує кілька правильних відповідей, то їх зручно реалізувати за допомогою прапорців. Також запитання може вимагати введення відповіді у вигляді слова. Тоді використовується текстове поле. На початку тестування програма повинна запитати ім'я користувача і мати можливість для перегляду всіх результатів, які зберігаються в БД.

Нехай треба створити тестування з двох запитань, одне з яких буде мати один правильний варіант, а інше - два правильних варіанти відповіді:

Якою функцією здійснюється зчитування файла в масив у PHP:

- fclose
- fopen
- fgets
- file

Якими PHP-тегами виділяються фрагменти коду:

- <?php ?>
- <# #>
- <? ?>
- <$ $>

Спочатку треба створити необхідну таблицю в БД. Нам потрібно зберегти ім'я та кількість правильних відповідей на перше та друге запитання. Разом з ідентифікатором це буде 4 стовпця – id(int), name(text,) answer_1(int), answer_2(int). Створення виконується за допомогою PhpMyAdmin.

Програма буде виглядати наступним чином:

- **Починаємо сесію:**

```
<?php session_start(); ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
```

```
<title></title>
</head>

<body>.
```

- **З'єднуємося з БД:**

```
<?php
$user = "root";         // логін користувача БД
$password = "1111";         // пароль користувача БД
$host = "localhost";    //Хост
$db = "caketest";         //БД

@$sql = new mysqli($host,$user,$password,$db,'3307');

if (mysqli_connect_errno()) {
echo "Помилка з'єднання з БД: ". mysqli_connect_error();
exit;
}.
```

- **Перевіряємо, чи треба показувати результат**

```
if ($_GET['result']=='true') {
```

- **Показуємо результати**

```
$_SESSION['name'] = '';
$_SESSION['step'] = '';

$result = $sql->query('select * from answers');
?>
<table border="1">
<tr><td>Ім'я</td><td>Балів за перше питання</td><td>Балів за друге питання</td></tr>
```

```
<?php
while( $row = $result->fetch_assoc() ){
        echo
'<tr><td>'.$row['name'].'</td><td>'.$row['answer_1'].'</td><td>'.$row['an
swer_2'].'</td></tr>';
}
?>
</table>
<?php

} else {
```

- **Зчитуємо і записуємо ім’я користувача в сесію та встановлюємо поточний номер запитання**

```
if ((bool)$_POST['name']) {
$_SESSION['name'] = $_POST['name'];
$_SESSION['step'] = 1;
}
```

- **Якщо в сесії немає імені - просимо його ввести**

```
if (!$_SESSION['name']) {?>
<form action="" method="post">
Введить ім'я:
<input name="name" type="text" />
<input name="" type="submit" />
</form><br />
<a href="?result=true"> Результати </a>
<?php } else {
```

- **Зчитуємо відповіді на перше запитання, записуємо їх у сесію та встановлюємо поточний номер запитання**

```
if ((bool)$_POST['answer_1']) {
$_SESSION['answer_1'] = $_POST['answer_1'];
$_SESSION['step'] = 2;
}
```

- **Зчитуємо відповіді на друге запитання, записуємо їх у сесію та встановлюємо номер, який відповідає запису даних**

```
if ((bool)$_POST['answer_2']) {
$_SESSION['answer_2'] = $_POST['answer_2'];
$_SESSION['step'] = 3;
}
?>
```

- **Виводимо перше запитання**

```
<?php if ($_SESSION['step'] == 1) { ?>
Якою функцією здійснюється зчитування файлу в масив у PHP?
<form action="" method="post">
<input name="answer_1" type="radio" value="1" />
-	fclose <br />
<input name="answer_1" type="radio" value="2" />
-	fopen <br />
<input name="answer_1" type="radio" value="3" />
-	fgets <br />
<input name="answer_1" type="radio" value="4" />
-	file <br />
<input name="" type="submit" />
</form>
```

- **Виводимо друге запитання**

```
<?php } else if ($_SESSION['step'] == 2) { ?>
```

Якими PHP тегами виділяються фрагменти коду?
<form action="" method="post">
<input name="answer_2[]" type="checkbox" value="1" />
- <?php ?>
<input name="answer_2[]" type="checkbox" value="2" />
- <# #>
<input name="answer_2[]" type="checkbox" value="3" />
- <? ?>
<input name="answer_2[]" type="checkbox" value="4" />
- <$ $>
<input name="" type="submit" />
</form>

- **Виконуємо запис даних з одночасною перевіркою правильності введених відповідей**

```
<?php } else if ($_SESSION['step'] == 3) { ?>
<?php
$answer_1 = 0;
if ($_SESSION['answer_1'] == 4) $answer_1 =1;

$answer_2 = 0;
foreach ($_SESSION['answer_2'] as $answer) {
 if ($answer == 1 || $answer == 3) $answer_2++;
}

$query = 'insert into answers (name, answer_1, answer_2) values ('".$_SESSION['name']."',".$answer_1."',".$answer_2."')';
$result = $sql->query($query);

$_SESSION['name'] = '';
$_SESSION['step'] = '';

?>
Ваші результати записані
```

```
<?php } ?>
<?php } ?>
<?php }
$sql->close();
?>
</body>
</html>
```

5.1. Хід виконання роботи

4. Розробити питання для тестування в необхідній кількості.
5. Створити програму, використовуючи розроблені запитання та наведений приклад.

6. Індивідуальні завдання

Номер завдання	Кількість питань
1	Два запитання з одним варіантом відповіді, одне з текстовою відповіддю та одне з кількома варіантами відповіді.
2	Одне запитання з одним варіантом відповіді, одне з текстовою відповіддю та два з кількома варіантами відповіді.
3	Два запитання з одним варіантом відповіді та два з кількома варіантами відповіді.
4	Одне запитання з одним варіантом відповіді та три з кількома варіантами відповіді.
5	Три запитання з одним варіантом відповіді та одне з кількома варіантами відповіді.
6	Одне запитання з одним варіантом відповіді, два з текстовою відповіддю та одне з кількома варіантами відповіді.
7	Одне з текстовою відповіддю та три з кількома варіантами відповіді.
8	Три з текстовою відповіддю та одне з кількома варіантами відповіді.

9	Три запитання з одним варіантом відповіді та одне з текстовою відповіддю.
10	Одне запитання з одним варіантом відповіді, одне з текстовою відповіддю та два з кількома варіантами відповіді.
11	Два запитання з одним варіантом відповіді та два з кількома варіантами відповіді.
12	Одне запитання з одним варіантом відповіді та три з кількома варіантами відповіді.
13	Три запитання з одним варіантом відповіді та одне з кількома варіантами відповіді.
14	Одне запитання з одним варіантом відповіді, два з текстовою відповіддю та одне з кількома варіантами відповіді.
15	Одне запитання з одним варіантом відповіді, одне з текстовою відповіддю та два з кількома варіантами відповіді.
16	Два запитання з одним варіантом відповіді та два з кількома варіантами відповіді.

Запитання для контролю

1. Як можна виконати запит в СКБД MySQL за допомогою PHP?
2. Як зчитати результати запиту до СКБД MySQL за допомогою PHP?
3. Як виконується зберігання проміжних результатів у програмі?
4. Додати можливість введення номеру академічної групи та виведення результату по групах.
5. Додати можливість сортування результатів за датою та прізвищем.

Список рекомендованої літератури

7. PHP5 и MySQL. Библия пользователя.: пер. с англ. – М.: издательский дом «Вильямс», 2006. – 1216 с.: ил.
8. MySQL / Л. Ульман; пер. с англ. Слинкина А.А. – М: ДМК Пресс; Спб.: Питер, 2004 – 352 с. : ил.
9. PHP/ MySQL для начинающих . Пер. с англ.. – М. КУДИЦ-ОБРАЗ, 2005. – 384с.
10. Котеров Д. В. Самоучитель PHP 4. — СПб.: БХВ-Петербург, 2003. — 576 с.:
11. Аргерих Л. и др. Профессиональное PHP программирование, 2-е издание. - Пер. с англ. - СПб: Символ-Плюс, 2003. - 1048 с., ил.
12. Разработка Web-приложений на PHP и MySQL: Пер. с англ../ Томсон Л., Веллинг Л. – К.: Издательство «Диасофт», 2002. – 672 с.

Навчальне видання

ПРИКЛАДНЕ ПРОГРАМУВАННЯ: ВІД ТЕОРІЇ ДО ПРАКТИКИ

Навчальний посібник

Укладачі ***М.П. Горський***
А.Л. Негрич
О.В. Олар

Відповідальний за випуск ***Ангельський О.В.***

Літературний редактор ***Макарова О.П.***

Технічний редактор ***Чораєва Г.К.***

Дизайн обкладинки ***Цвіра А.В.***

Підписано до друку 00.11.2021. Формат 60x84/16.
Папір офсетний. Друк різографічний. Умов.-друк. арк. 6,7.
Обл.-вид. арк. 7,0. Тираж 00. Зам. Н-000.
Видавництво та друкарня Чернівецького національного університету.
58012, Чернівці, вул. Коцюбинського, 2.
e-mail: ruta@chnu.edu.ua

Свідоцтво суб'єкта видавничої справи ДК № 891 від 08.04.2002.