Зміст

[Лабораторна робота № 1 2](#_Toc475603812)

[1.1. Створення програми «Моя перша форма» 2](#_Toc475603813)

[Самостійна робота 4](#_Toc475603814)

[1.2. Типи даних. Засоби введення та виведення 5](#_Toc475603815)

[Самостійна робота 7](#_Toc475603816)

[Висновки 8](#_Toc475603817)

[Лабораторна робота № 2 9](#_Toc475603818)

[2.1 Операції та вирази 9](#_Toc475603819)

[2.2 Умовні оператори 10](#_Toc475603820)

[Висновки 10](#_Toc475603821)

[Лабораторна робота № 3 11](#_Toc475603822)

[Множинний вибір. Оператор Сase 11](#_Toc475603823)

[Самостійна робота 12](#_Toc475603824)

[Лабораторна робота №4 13](#_Toc475603825)

[Оператори циклів. Цикл For. 13](#_Toc475603826)

[Самостійна робота 15](#_Toc475603827)

[Лабораторна робота № 5 16](#_Toc475603828)

[5.1. Оператори циклів. Оператор while 16](#_Toc475603829)

[5.2 Оператори циклів. Оператор Repeat 17](#_Toc475603830)

[Лабораторна робота № 6 19](#_Toc475603831)

[Строковий тип даних. 19](#_Toc475603832)

[Лабораторна робота № 7 21](#_Toc475603833)

[Об’єкти середовища PascalABC. Робота з меню. 21](#_Toc475603834)

[Лабораторна робота № 8 24](#_Toc475603835)

[8.1 Об’єкти середовища PascalABC. Смуги прокрутки. 24](#_Toc475603836)

[8.2 Об’єкти середовища PascalABC. Випадковий вибір. Використання списків. 25](#_Toc475603837)

# Лабораторна робота № 1

## Створення програми «Моя перша форма»

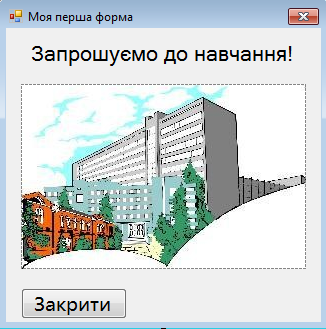
**Мета:** Модифікація форми. Робота з компонентами Label, PictureBox та Button. Програма закриття вікна.

**План**

* Об’єктно-орієнтоване програмування. Середовище програмування PascalABC.
* Створення проекту у PascalABC.

**Завдання:** Потрібно створити програму «Кольорове вікно з картинкою».

Приблизний вигляд вікна показано на малюнку 1.

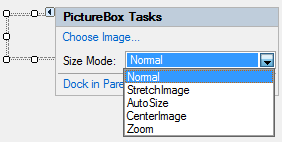


Процес створення програми відбувається за наступним алгоритмом:

1. Запустити PascalABC з пункту меню Windows або з «Рабочого столу»;
2. Зберегти проект в робочому каталозі командою «*Файл/Новий проект*».  **Рекомендація:** кожен проект краще зберігати в окремій папці; в іменах файлів можна використовувати тільки латинські букви без пробілів!;
3. Налаштування вікна форми:

* Заголовок вікна змінити на *Моя перша форма* 🡪 властивість **Text;**
* Змінити властивість **StartPosition** на *CenterScreen* для виводу вікна форми в центрі екрана;
* Змінити колір вікна на свій «улюблений колір», вибравши колір зі списку 🡪 властивість **BackColor;**
* Змінити іконки вікна програми 🡪 властивість **MaximizeBox** і **MinimizeBox**,Встановити всі підвластивості у значення *False* (в програмі відсутні усі системні кнопки вікна);

1. Помістити на форму картинку 🡪 об’єкт **Picture** на панелі Палітра *Стандартные элементы управления*, іконка значка . На формі з’явиться місце майбутньої картинки. Картинка може відображатися декількома способами: або займати місце, яке визначає користувач (рамка картинки) або змінити масштаб картинки під реальні розміри картинки. Для налаштування картинки потрібно змінити наступні властивості:



* Свойство **Choose Image** загружает «мастера» для выбора картинки из файла. Выберите любой файл с картинкой из имеющихся на жестком диске.
* Свойство **Size Mode –** при значении *Normal* значения установленные пользователем для картинки (свойства **Width** – ширина и **Height** – высота) будут изменены в соответствии с реальными размерами картинки;
* Свойство **Stretch Image** – при значении *true* размеры картинки будут адаптированы к размерам установленных пользователем (указанных в свойствах **Width** и **Height**);
* Свойство**AutoSize**приводит изображение к реальному размеру картинки (как в файле).

1. Поместить на форме кнопку для выхода из программы. Установить кнопку 🡪 объект **Button,** на панели Палитра *Стандартные элементы управления*, с иконкой объекта . Изменить свойства:
   * **Name** – изменить на *BQuit;*
   * **Text** – изменить на «Закрити»;
   * **Font –** шрифт текста, запускает мастера для настройки названия, цвета и стиля шрифта метки (установите любые параметры).
2. Поместить на форму метку для подписи к картинке 🡪 объект **Label**, на панели Палитра *Стандартные элементы управления*, значок . Изменить свойства:
   * **Text** – изменить на любой;
   * **TextAlign** – изменить на любой;
   * **Font –** шрифт текста, запускает мастера для настройки названия, цвета и стиля шрифта метки (установите любые параметры).
3. Запрограммировать выход при щелчке по кнопке:
   * Активный элемент, то есть элемент, свойства которого настраиваются, выделен рамкой с маркерами Ramka, если рамки нет ни на одном элементе, то активна форма. Выбрать элемент **кнопки** «Закрити» на форме (вокруг кнопки будет рамка);
   * Дважды щелкнуть по кнопке мышкой – будет выполнено событие *Click* на вкладке *Events* в **Инспектор Объектов** (это событие является событием «по умолчанием»). Станет активным окно **Редактора программ.**
   * В верхней части окна активной станет вкладка Код. Чтобы потом перейти снова к окну формы активируйте вкладку Дизайнер.



* + Автоматически будет создана процедура для программирования события *Click.* текст. *Пример*, если была использована кнопка *BQuit,* текст будет следующим:

**procedure** FMain.BQuit\_Click (sender: Object; e: EventArgs);

**begin**

**end;**

* + В теле созданной процедуры (между операторными скобками **begin** и **end)** вписать команду закрытия окна – **Close**. Так как у нас одно окно, то закрытие окна приведет к выходу из программы. Текст программы станет таким:

**procedure** FMain.BQuit\_Click (sender: Object; e: EventArgs);

**begin**

Close;

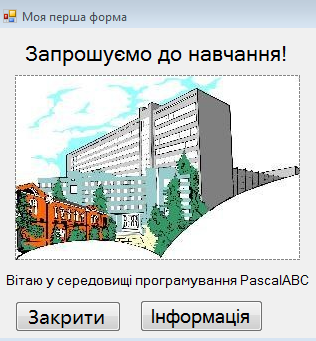
**end;**

**Обратите внимание!** После слова «Close» стоит знак «;», так как на языке Паскаль необходимо ставить знак «точка с запятой» – «;» после каждой команды.

1. Программа создана! Для запуска программы необходимо:
   * Сохранить программу 🡪 команда *Файл/Сохранить Все*, значок в меню **Главного окна PascalABC**;
   * Запустить программу 🡪 *Выполнить/Выполнить,* в меню **Главного окна PascalABC** или щелкнуть на значок  на *Панели инструментов* **в Главном окне PascalABC;**
2. Выход из оболочки **PascalABC** осуществляется выбором команды *Файл\Выход*.
3. Если при следующем запуске **PascalABC** потребуется загрузить проект, то это можно осуществить через пункт *Файл/Открыть проект*, значок Open ProjectM в **Главном окне PascalABC**.

### Самостійна робота

Додати до форми ще одну кнопку. При натиснені цієї кнопки на формі з’являється надпис – «Вітаю у середовищі програмування PascalABC».



**Вказівка:**

1. Встановити властивість **Visible** – False для напису; встановити ім’я напису – **Label2**;

2. Для кнопки «Інформація» при події *Click* змінити видимість текстового поля. Написати рядок програми для кнопки «Інформація»:

|  |
| --- |
| **if** Label2.Visible=True  **then** Label2.Visible:=False  **else** Label2.Visible:=True; |

1.2. Типи даних. Засоби введення та виведення

**Мета:** Робота з компонентами Label, textBox та Button.

**План**

* Поняття «тип даних». Основні типи даних середовища програмування PascalABC.
* Панель компонентів. Компоненти Label, textBox та Button.

**Практичне завдання:**

Написати програми з використанням розглянутих компонентів (Label, textBox та Button):

А) при запуску програми запускається форма 1 (з попереднього розділу) потім кнопкою відкривається форма 2.

Б) На формі 2 користувачеві пропонується ввести власне ім’я, вік та вагу. Крім цього на формі має бути кнопка «Введення». Після введення необхідної інформації користувач має натиснути на кнопку. Після чого праворуч від введеної інформації з’являються компоненти Label синього кольору із інформацією про типи даних (текст чи число), що відповідають змінним ім’я, вік та вага.

В) Удосконалити програму Б додавши на форму ще один компонент Button при натисненні на який виконувалося б очищення полів Ім’я та ін.

Приблизний вигляд вікна програми показано на рисунку 1.

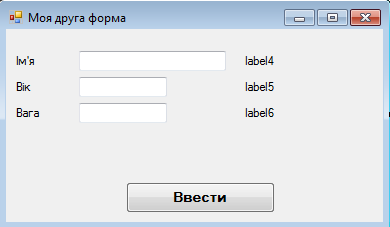
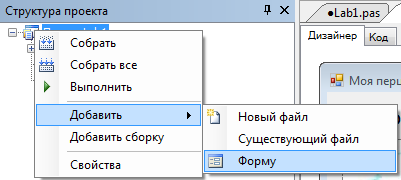


Рис. 1.

Створення програми відбувається за наступним алгоритмом:

1. Запустити PascalABC з пункту меню Windows або з «Рабочого столу»;
2. Відкрити проект1 в робочому каталозі командою «*Файл/Відкрити проект*».
3. Створення нового вікна форми:

* Викликати контекстне меню на назві проекту у вікні «Структура проекту». Вибрати команду Додати 🡪 Форму. З’явиться ще вікно ще однієї форми. Дати назву (або залишити за змовчуванням).

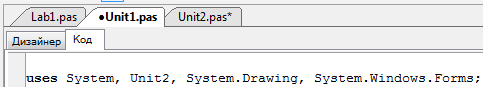


* Заголовок вікна змінити на *Моя друга форма* 🡪 властивість **Text;**
* Змінити властивіть **Name** форми на *Form2* , для звернення до неї в програмі.
* Змінити властивість **StartPosition** на *CenterScreen* для виводу вікна форми в центрі екрана;
* Змінити колір вікна на свій улюблений, властивість **BackColor;**
* Змінити іконки вікна програми **MaximizeBox** і **MinimizeBox**;

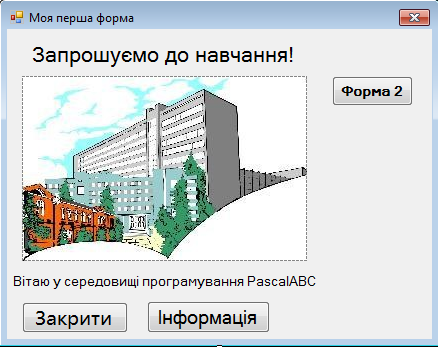
1. Активізувати вкладку основної програми (назва проекту.pas) і в рядок uses дописати назву нового юніта (нової форми).



1. Зробимо перехід з форми 1 у форму 2 по кнопці. Для цього під’єднаємо форму 2 у першу форму:
   * На вкладці Unit1 – Код допишемо у розділ Uses назву нової форми Unit2.



* + На вкладці Дизайнер поставити нову кнопку 🡪 об’єкт **Button**, на панелі Палітра *Стандартні елементи керування* . Змінити текст кнопки на «Форма 2».



* + Подвійним кліком на кнопці викликати обробку події *Click* і ввести текст програми:

|  |
| --- |
| **procedure** Form1.button2\_Click(sender: Object; e: EventArgs);  **begin**  Form(**new** Form2).show; // відкриття Form2  **end**; |

Створення форми 2.

1. Перейти до вкладки Unit2. Помістити на форму Підписи даних «Ім’я», «Вік» та «Вага» 🡪 объект Label, на панелі Палітра *Стандартні елементи керування*, значок . Змінити властивості: **Text; TextAlign; Font**.
2. Помістити на форму Вікна введення даних для «Ім’я», «Вік» та «Вага» 🡪 об’ект **TextBox** на панелі Палітра *Стандартні елементи керування*, іконка значка . На формі повинні з’явитися текстові поля.
3. Оскільки біля кожного з вікон має в майбутньому з’явитися напис, то для них потрібно підготувати місце: вставити напроти кожного з правої сторони ще по одному об’єкту типу **Label**. Встановити для них властивість **Visible** – False.



**Рекомендація**: можна створити одну трійку (вікна введення та написів до нього), встановити властивості, а потім виділити разом і утримуючи кнопку Ctrl перетягнути мишею по формі – скопіювати потрібну кількість разів. Змінити назви підписів.

1. На форму встановити кнопку «Введення».
2. Для кнопки «Введення» написати програму для події *Click*:

|  |
| --- |
| **procedure** Form2.button1\_Click(sender: Object; e: EventArgs);  **var**  s: string;  i: integer;  r: real;  **begin**  **try**  s:=(textBox1.Text);  i:=StrToInt(textBox2.Text); // перевід значення Box2 в ціле число  r:=StrToFloat(textBox3.Text); // перевід значення Box3 в дробове число  label4.Text:='Текст';  label5.Text:='Ціле число';  label6.Text:='Дробове число'  **except**  MessageBox.Show('Введіть інші дані', 'Помилка', MessageBoxButtons.OK);  **end**;  **end**; |

Команда **try ... except** робить спробу виконати усі команди, які стоять після слова **try**. Якщо це не вдається – формується виключення (**except**) і виконуються усі команди з цього розділу. Ця команда не дає програмі закінчитися в аварійному режимі, а дозволяє виправити дані.

### Самостійна робота

1. Зробіть компоненти Label, які з’являються на екрані, синього кольору.

2. Удосконалити програму, додавши на форму ще один компонент Button при натисненні на який виконувалося б очищення полів Ім’я та ін.

**Контрольні питання:**

1. Який тип даних ми називаємо *integer*?
2. Чи можна виконати операцію ділення над цілим типом даних?
3. До якого типу даних відносяться:

А) кількість книжок

Б) вартість товару

В) назва вулиці

Г) правдивість інформації

Д) об’єм кімнати

1. Які властивості компонента **Button** ви знаєте?

## Висновки

В першій темі були вивчені: о*болонка PascalABC; структура програми; команди редактора.*

При створенні програми були отримані навички роботи з наступними командами і візуальними компонентами: вікном форми **Form;** кнопкою **Button;** малюнком **Image;** міткою **Label;** командою закриття вікна **Close.**

# Лабораторна робота № 2

## 2.1 Операції та вирази

**Мета:** Ознайомлення з поняттям операцій та виразів. Вивчення операцій для роботи з простими типами даних. Закріплення знань з попередньої теми.

**План**

* Унарні та бінарні операції. Арифметичні операції. Приклади використання.
* Логічні операції. Приклади використання.
* Строкові операції. Приклади використання.

**Практичне завдання:**

Створити проект у середовищі програмування основним завданням якого є навчити учня складу числа 5.   
Назва проекту – «Склад числа».  
При виконанні роботи необхідно використовувати компоненти Label, TextBox та Button та компонент для виведення зображення на екран (Image). Крім цього підрахунок суми чисел необхідно виконувати введенням доданків у поле TextBox та виведенням у Label після натиснення на кнопку. В Image при запуску форми має з’явитися малюнок числа 5. На формі також має бути кнопка «Close» при натисненні якої робота програми завершиться.

Приблизний вигляд вікна програми показано на рисунку 1.

|  |  |
| --- | --- |
| Рис.1 | **Порада**. Приблизний програмний код буде виглядати так:  **var** a,b:integer;  **begin**  **try**  a:=StrToInt(textBox1.Text);  b:=StrToInt(textBox2.Text);  **if** a+b=5 **then**  **begin** pictureBox1.Visible:=True;  label4.Text:='Молодець!';  label4.Visible:=True;  **end**  **except**  **begin** pictureBox1.Visible:=False;  label4.Text:='Введіть інші дані';  label4.Visible:=True;  **end**;  **end**;  **end**; |

**Контрольні питання:**

1. Що таке вираз в мові програмування ОР?
2. З чого складається вираз?
3. Які операції існують?
4. Розкажіть про арифметичні операції.
5. Назвіть строкові операції, які ви запам’ятали.
6. З чого складаються логічні вирази?
7. Перерахуйте логічні операції.
8. Для чого потрібні функції перетворення?

## 2.2 Умовні оператори

**Мета:** Ознайомлення з поняттям умовного оператору. Розробка програмного продукту з використанням повної та скороченої форм запису оператору If.

**План**

* Поняття оператору. Умовний оператор.
* Скорочена форма запису умовного оператора.
* Повна форма запису умовного оператора.

**Практичне завдання:**

Створити проект, метою якого є навчити школяра поняттям «>», «<», «=». Розміщення компонентів на формі має бути наступним:

* 2 комненти TextBox, в які користувач буде вводити числа (від 1-го до 9-ти);
* 2 компонети Image для відображення введених чисел у малюнках;
* Компонент Button для виконання порівняння введених чисел;
* Компонент Label для відображення знаків «>», «<» та «=».

Приблизний вигляд виконаної роботи:

|  |  |
| --- | --- |
|  | **Порада**. Для висвітлення потрібної картинки скористайтесь рядком  pictureBox1.Image:=System.Drawing.Image.FromFile('шлях'+a+'.jpg');  де а – номер файлу (зчитується з TextBox),  'шлях' - це розташування файлу. Наприклад: D:\Lab3\Lr\_3\ |

**Контрольні питання:**

1. Що таке умовний оператор?
2. Які форми запису умовного оператору ви знаєте?

## Висновки

В другій темі ви отримали знання про: *структуру програми, типи даних, перетворення типів, умовні оператори, підключення нових форм в програму, звертання до різних файлів і виклик потрібного.*

При створенні програми були отримані навички роботи з наступними командами і візуальними компонентами: вікном форми **Form;** кнопкою **Button;** малюнком **Image;** міткою **Label;** текстовим полем **TextBox.**

# Лабораторна робота № 3

## Множинний вибір. Оператор Сase

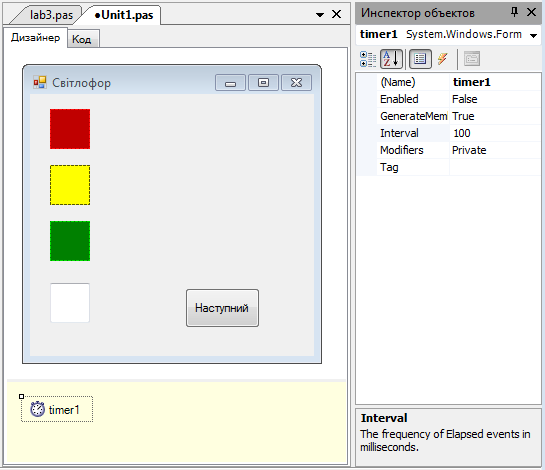
**Мета:** Ознайомити студентів з поняттям множинного вибору. Розглянути призначення оператора вибору. Закріпити отримані знання на практиці.

**План**

* Поняття множинного вибору. Оператор CASE.
* Форми запису оператора вибору.
* Випадки використання оператору CASE.

**Практичне завдання:**

Користувач вводить номер стану світлофора, а програма виводить кольори запалених у цей момент лампочок. Є 5 станів світлофора: червоний, червний+жовтий, зелений, мигаючий зелений, жовтий. Використати компонент PictureBox зі зміненою властивістю BackColor для зображення кольорів світлофора.



**Порада**.

Для того, щоб зелений мерехтів треба використати компонент «Таймер» з палітри елементів і кожну секунду змінювати видимість зеленого кольору. Щоб його запустити потрібно

перетягнути елемент з панелі «Компоненти» в Палітрі в поле форми (таймер з’явиться внизу екрана);

встановити властивість Enabled у стан False , а властивість Interval поміняти на 1000 (що відповідає 1 секунді);

двічі клікнути мишею по зображенню таймера – з’явиться процедура для таймера. Вписати наступний код:

|  |
| --- |
| **procedure** Form1.timer1\_Tick(sender: Object; e: EventArgs);  **var** i:integer;  **begin**  **for** i:=1 **to** 3 **do**  **if** pictureBox3.Visible=True  **then** pictureBox3.Visible:=False  **else** pictureBox3.Visible:=True;  **end**; |

Використайте оператор Case для зміни властивості Visible кожного PictureBox.

|  |
| --- |
| **case** a **of**  1: pictureBox1.Visible:=True;  2: **begin**  pictureBox1.Visible:=True;  pictureBox2.Visible:=True;  **end**;  3: pictureBox2.Visible:=True;  4: pictureBox3.Visible:=True;  5:timer1.Enabled:= True; //запуск таймера для мерехтіння зеленого |

### Самостійна робота

1. Модернізувати програму таким чином, щоб під 6-им станом світлофора знову розумівся 1-ий, під 7-им — 2-ий і т. д.
2. Поруч, крім кольору «автомобільного» світлофора, виводити одночасно колір світла, яке зараз світиться для пішоходів.

**Контрольні питання:**

1. Що таке оператор вибору?
2. Які форми запису оператору case ви знаєте?
3. Алгоритм роботи оператора case.
4. Що ми називаємо вкладеною умовою?

# Лабораторна робота №4

Оператори циклів. Цикл For.

**Мета:** Ознайомити студентів з поняттям циклу. Розглянути призначення циклу For. Закріпити отримані знання на практиці.

**План**

* Поняття циклу. Приклади використання.
* Цикл з лічильником.
* Інструкції to та down to.

**Практичне завдання**

|  |  |
| --- | --- |
| Створити комп'ютерну версію однієї з головоломок Самуеля Ллойда: із заданого набору чисел треба вибрати ті, сума яких складе 50. Числа, які обрав Ллойд для своєї головоломки: 25, 27, 3, 12, 6, 15, 9, 30, 21, 19. | **Приклад виконаної роботи** |

Виконати наступні дії:

* + 1. Після запуску програми у вікні зображується список чисел Ллойда.
    2. Вибирані за допомогою прапорців числа з’являються у правому вікні.
    3. Сума вибраних чисел представлена ​​у вигляді напису.
    4. Для виходу з програми необхідно натиснути мишею на закриваючою кнопку в рядку заголовка.

Опис плану розробки програми

1. Відкрити новий проект.

2. Розмістити на формі екземпляри компонентів: список прапорців CheckedListBox, напис Label, список ListBox.

3. Виконати наступні дії:

|  |  |  |  |
| --- | --- | --- | --- |
| **Виділений об’єкт** | **Вкладка вікна**  **Object Inspector** | **Ім’я властивості/**  **ім’я події** | **Дія** |
| Form1 | Properties | Text | Встановлюємо ім’я форми «Головоломка» |
| CheckedListBox1 | Properties | Items    або | Задаємо склад списку. клацнути на кнопці будівника. відкриється вікно Edit (Редактор списку рядків). Введіть у список задані числа через Enter. Натисніть кнопку OK. |
| Підібрати висоту списку так, щоб всі числа помістилися в список (без смуг прокручувань) | |
| Events | SelectedIndexChanged    або подвійний клік на CheckedListBox1 | Якщо числа немає в списку, то додати його. Інакше – число видалити з обраних. Після оновлення списку необхідно порахувати суму обраних чисел. Елементи списку виглядають як числа, але є текстовими рядками (Скористатися функцією StrToInt). |
| ListBox1 | Properties | Size | Підібрати значення так, що всі числа помістилися в список (без смуг прокручувань). спочатку список порожній. Заповнюватися він буде при роботі програми. |
| Label1 | Properties | Text | Введіть текст: «Сумма: 0» |
| TextAlignment | Задайте значення MiddleLeft |

Текст процедури checkedListBox1\_SelectedIndexChanged

|  |
| --- |
| **procedure** Form1.checkedListBox1\_SelectedIndexChanged(sender: Object; e: EventArgs);  **var**  i, r, s: Integer;  **begin**  r := 0; //для позначення що елемент був видалений  **for** i := 0 **to** ListBox1.Items.Count - 1 **do** //огляд вікна списку  **if** checkedListBox1.SelectedItem = ListBox1.Items[i] //Чи є виділений елемент у списку?  **then begin**  ListBox1.Items.Remove(ListBox1.Items[i]); //видалення елементу  r := 1;  **break**;  **end**;  **if** r <> 1 **then** ListBox1.Items.Insert(ListBox1.Items.Count, checkedListBox1.SelectedItem);//додавання елемента у список    {сума елементів}  **for** i := 0 **to** ListBox1.Items.Count - 1 **do**  s := s + StrToInt(ListBox1.Items[i].ToString());  Label1.Text := 'Сума: ' + S;  **end**; |

### Самостійна робота

1. Модернізувати програму таким чином, щоб в окремому елементі Label відображалась кількість зроблених «ходів».
2. Як тільки сума стане дорівювати п’ятидесяти – повинне з’явитися вікно з привітанням.

**Контрольні питання:**

1. Що таке цикл? Наведіть приклади.
2. Який процес називається циклічним?
3. Що таке змінна циклу?
4. Що таке тіло циклу? Його розміри в кількості інструкцій?
5. Чим відрізняються інструкції for з to, downto та як вони використовуються?
6. Що відбудеться при використанні службового слова to в циклі, якщо початковий вираз буде більшим за кінцевий?
7. Скільки операторів може стояти після службового слова do в циклі?

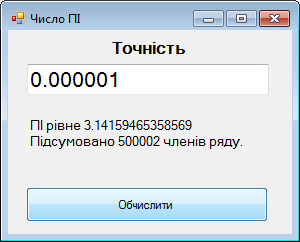
# Лабораторна робота № 5

5.1. Оператори циклів. Оператор while

**Мета:** Розглянути призначення циклу та структуру операторів циклу while та repeat. Закріпити отримані знання на практиці.

**Хід роботи**

1. Написати програму, яка обчислює значення числа ПІ з точністю, що задається користувачем під час роботи.



У основі алгоритму обчислення лежить той факт, що сума ряду:

1 - 1/3 + 1/5 -1/7 + 1/9 + ...

наближається до значення ПИ/4 при достатньо великій кількості членів ряду.

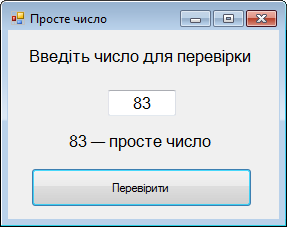
Кожен член ряду з номером n обчислюється за формулою: 1/(2\*n - 1) і множиться на мінус один, якщо n парне (визначити, чи є ПІ парним, можна перевіркою залишку від ділення ПІ на 2). Обчислення закінчується тоді, коли значення чергового члена ряду стає меншим, ніж задана точність обчислення.

**Лістинг обчислення числа ПІ**

|  |
| --- |
| **procedure** Form1.button1\_Click(sender: Object; e: EventArgs);  **var**  pi: real; // обчислюване значення ПІ  t: real; // точність обчислення  n: integer; // номер члена ряду  elem: real;// значення члена ряду  **begin**  pi := 0;  n := 1;  t := StrToFloat(TextBox1.text);  elem := 1; // щоб почати цикл  **while** elem >= t **do**  **begin**  elem := 1 / (2 \* n - 1);  **if** n **MOD** 2 = 0  **then** pi := pi - elem **else** pi := pi + elem; n := n + 1;  **end**;  pi := pi \* 4;  Label4.Text := 'ПІ рівне ' + FloatToStr(pi) + #13  + 'Підсумовано ' + IntTostr(n) + ' членів ряду.';  **end**; |

5.2 Оператори циклів. Оператор Repeat

Написати програму, яка перевіряє, чи є введене користувачем число простим (як відомо, число називається простим, якщо воно ділиться тільки на одиницю і саме на себе).



Наприклад, число 21 — звичайне (ділиться на 3), а число 17 — просте (ділиться тільки на 1 і на 17).

Перевірити, чи є число n простим, можна діленням числа n на два, на три і т.д. до n і перевірки залишку після кожного ділення. Якщо після чергового ділення залишок рівний нулю, то це означає, що знайдено число, на якому n ділиться без залишку. Порівнявши n і число, на якому n розділилося без залишку, можна визначити, чи є n простим числом.

**Лістинг програми**

|  |
| --- |
| **procedure** Form1.button1\_Click(sender: Object; e: EventArgs);  **var**  n: integer; // число n, що перевіряється:  d: integer; // дільник  r: integer;// залишок від ділення n на d  **begin**  n := StrToInt(TextBox1.text);  d := 2; // спочатку ділимо на два  **repeat**  r := n **mod** d;  **if** r <> 0 // n не розділилося без остачі на d  **then** d := d + 1;  **until** r = 0; // знайдене число, на яке n розділилося без залишку  label2.Text := TextBox1.text;  **if** d = n  **then** Label2.Text := Label2.Text + ' — просте число'  **else** Label2.Text := Label2.Text + ' — звичайне число';  **end**; |

###### **Контрольні питання**

1. Що таке цикл? Наведіть приклади.
2. Який процес називається циклічним?
3. Що таке змінна циклу? Приклади.
4. З яких складових утворюється алгоритм циклічної структури?
5. Що таке тіло циклу? Його розміри в кількості інструкцій?
6. Що таке модифікація циклу?
7. Як управляти циклом? Як виконується перехід до початку циклу?
8. Наведіть загальний вигляд циклічного алгоритму.
9. Скільки операторів може стояти після службового слова do в циклі?
10. Як зберегти останнє значення параметру циклу?
11. У я кому випадку слід використовувати оператор While?
12. Яка форма запису оператору циклу Repeat? У якому випадку використовується вказаний оператор?

# Лабораторна робота № 6

## Строковий тип даних.

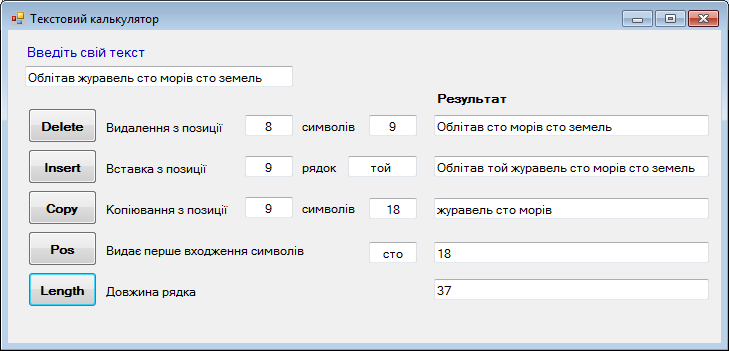
**Мета:** Ознайомити студентів зі строковим типом даних. Розглянути процедури та функції для роботи зі строковим типом даних.

**План**

1. Поняття строкового типу даних.
2. Основні процедури та функції обробки рядків.

**Практичне завдання**

Створити програму, яка виконуватиме функції строкового калькулятора.



Опис плану розробки програми  
1. Відкрити новий проект.  
2. Розмістити на формі екземпляри компонентів: Edit, Label, Button.

**Фрагменти програми**

|  |
| --- |
| **procedure** Form1.button1\_Click(sender: Object; e: EventArgs);  **begin**  s := TextBox1.Text;  k := StrToInt(TextBox2.Text);  t := StrToInt(TextBox5.Text);  Delete(s, k, t);  TextBox9.Text := s;  **end**; |
| **procedure** Form1.button2\_Click(sender: Object; e: EventArgs);  **begin**  s := TextBox1.Text;  q := StrToInt(TextBox3.Text);  w := TextBox6.Text;  Insert(w, s, q);  TextBox10.Text := s;  **end**; |
| **procedure** Form1.button3\_Click(sender: Object; e: EventArgs);  **begin**  s := TextBox1.Text;  p := StrToInt(TextBox4.Text);  l := StrToInt(TextBox7.Text);  Copy(s, p, l);  s1 := copy(s, p, l);  TextBox11.Text := s1;  **end**; |
| **procedure** Form1.button4\_Click(sender: Object; e: EventArgs);  **begin**  s := TextBox1.Text;  r := TextBox8.Text;  a := Pos(r, s);  TextBox12.Text := IntToStr(a);  **end**; |
| **procedure** Form1.button5\_Click(sender: Object; e: EventArgs);  **begin**  s := TextBox1.Text;  b := Length(s);  TextBox13.Text := IntToStr(b);  **end**; |

###### **Контрольні питання**

1. Які дії можна виконувати з рядками? Наведіть приклади.

# Лабораторна робота № 7

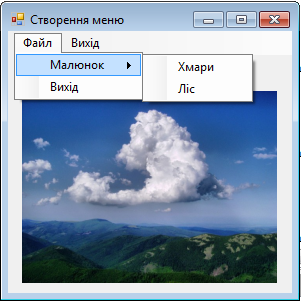
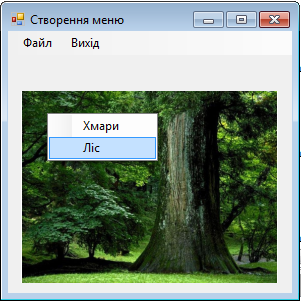
## Об’єкти середовища PascalABC. Робота з меню.

**Мета:** Ознайомити студентів з компонентами середовища PascalABC. Розглянути компонент MainMenu для роботи з меню. Закріпити отримані знання на практиці.

**Завдання для виконання**

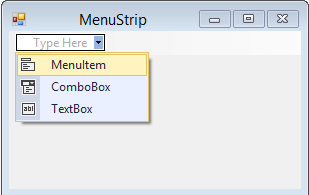
Створити програму, що виконує наступні дії:

1. Після запуску програми у вікні зображується рядок меню (Файл, Вихід).
2. При виборі пункту меню Файл з'являються пункти меню (Малюнки, Вихід).
3. При виборі пункту меню Малюнки з'являється вкладене меню, що складається з двох пунктів (Хмари, Ліс). Клікнувши на пункт меню ми вказуємо на вибір малюнка для відображення на формі.
4. По кліку правої кнопки миші з'являється контекстне меню. Вибрати за пунктом інший малюнок
5. Для виходу з програми необхідно натиснути мишею на закриваючу кнопку в рядку заголовка або  будь-який з пунктів меню "Вихід".

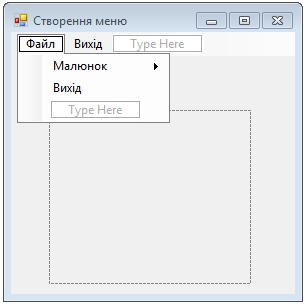
 

**Опис плану розробки програми**

1. Відкрити новий проект.
2. Розмістити на формі екземпляри компонентів: панель MenuStrip, малюнок Picture, контекстне меню Сontext MenuStrip.



1. MenuStrip виступає свого роду контейнером для окремих пунктів меню, які представлені об'єктом ToolStripMenuItem. Додати нові елементи в меню можна в режимі дизайнера.
2. Для додавання доступно три види елементів: MenuItem (об’єкт ToolStripMenuItem), ComboBox і TextBox. Таким чином, в меню ми можемо використовувати списки, що випадають і текстові поля, однак, як правило, ці елементи застосовуються в основному на панелі інструментів. Меню ж зазвичай містить набір об'єктів **ToolStripMenuItem**. Впишіть пункти меню в конструкторі (поля Type Here).

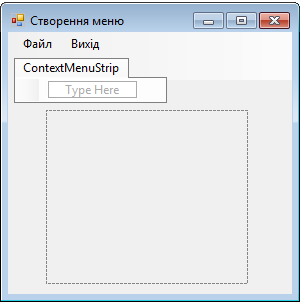
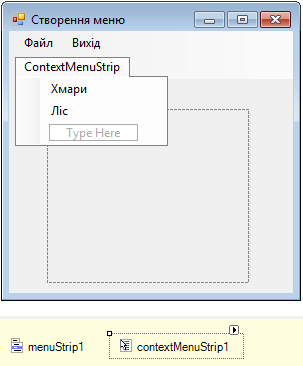


Кожен пункт меню буде автоматично надаватися ім’я (Name) з номером: toolStripMenuItem1, toolStripMenuItem2, ...

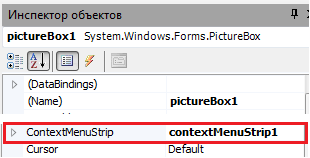
1. Подвійний клік миші на кожному з пунктів меню (Вихід, Вихід, Файл, Ліс, Хмари) створює процедуру Click для них.
2. Для усіх пунктів меню «Вихід» пропишіть процедуру Click з командою Close для закриття програми.
3. Для пунктів зміни малюнка пропишіть команду виклику потрібного малюнку. Наприклад:

pictureBox1.Image:=System.Drawing.Image.FromFile('D:\Lab7\Forest.jpg');

1. Контекстне меню Сontext MenuStrip створюється аналогічно MenuStrip. Аналогічно ToolStripMenuItem (меню Файл…) прописати для кожного з пунктів (ліс, хмари) команду виклику потрібного малюнку.

1. У інспекторі об’єктів для pictureBox1 вибрати в пункті ContextMenuStrip назву контекстного меню (асоціація контекстного меню з картинкою).



1. Збережіть проект, запустіть та протестуйте його

**Лістинг підпрограм**

|  |
| --- |
| **procedure** Form1.toolStripMenuItem1\_Click(sender: Object; e: EventArgs);  **begin**  Close; //закриття в меню Вихід  **end**; |
| **procedure** Form1.toolStripMenuItem2\_Click(sender: Object; e: EventArgs);  **begin**  Close; //закриття в меню Файл-Вихід  **end**; |
| **procedure** Form1.toolStripMenuItem3\_Click(sender: Object; e: EventArgs);  **begin**  //вибір в меню Файл-Малюнок-Хмари  pictureBox1.Image:=System.Drawing.Image.FromFile('D:\Lab7\Hmara.jpg');  **end**; |
| **procedure** Form1.toolStripMenuItem4\_Click(sender: Object; e: EventArgs);  **begin**  //вибір в меню Файл-Малюнок-Ліс  pictureBox1.Image:=System.Drawing.Image.FromFile('D:\Lab7\Forest.jpg');  **end**; |
| **procedure** Form1.toolStripMenuItem5\_Click(sender: Object; e: EventArgs);  **begin**  //вибір в контекстному меню - Хмари  pictureBox1.Image:=System.Drawing.Image.FromFile('D:\Lab7\Hmara.jpg');  **end**; |
| **procedure** Form1.toolStripMenuItem6\_Click(sender: Object; e: EventArgs);  **begin**  //вибір в контекстному меню - Ліс  pictureBox1.Image:=System.Drawing.Image.FromFile('D:\Lab7\Forest.jpg');  **end**; |

# Лабораторна робота № 8

## 8.1 Об’єкти середовища PascalABC. Смуги прокрутки.

**Мета:** Ознайомити студентів з компонентами середовища PascalABC. Розглянути компонент ScrollBar. Закріпити отримані знання на практиці.

**Завдання для виконання**

Створити програму, що виконує наступні дії:

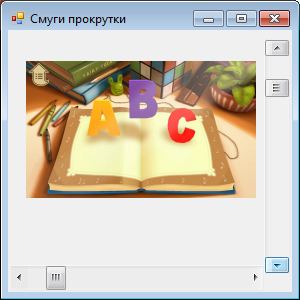
1. Після запуску програми у вікні зображується дві смуги прокрутки. Вертикальна смуга буде керувати рухом по вертикалі, горизонтальна - по горизонталі.

2. Наводячи курсор миші на одну з двох фігур (квадрат та круг), можна вибирати, яка з цих фігур пов'язана з смугами прокрутки.

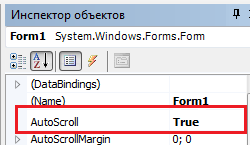
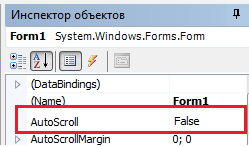
3. Потрібні додаткові об'єкти, за допомогою яких обмежується область руху фігур у вікні.

4. Якщо смуга прокрутки активна, то вона повинна реагувати на клавіші ВВЕРХ, ВНИЗ, ВЛІВО, ВПРАВО, PAGE UP, PAGE DOWN.

5. Для виходу з програми необхідно натиснути мишею на закриваючій кнопці в рядку заголовка.



**Опис плану розробки програми**

В формі смуги прокрутки з’являються автоматично, якщо виставити властивість форми AutoScroll в режим True.

Але ми будемо додавати у форму власні смуги прокрутки. Тому прослідкуйте, щоб ця властивість мала значення False.

1. Відкрити новий проект.

2. Розмістити на формі екземпляри компонентів: смуга прокрутки hScrollBar1, vScrollBar1, малюнок pictureBox1.

3. Виконати наступні дії:

* Встановити для pictureBox1 властивість Location (0;0).
* Встановіть для hScrollBar1 наступні властивості: Min(5), Max(148), SmallChange(2).
* Аналогічні властивості встановіть для vScrollBar1.
* Виконати подвійний клік на горизонтальній смузі прокрутки. В процедуру Scroll ввести наступний код:

**procedure** Form1.hScrollBar1\_Scroll(sender: Object; e: ScrollEventArgs);

**begin**

pictureBox1.Left := hScrollBar1.Value + hScrollBar1.SmallChange;

**end**;

* Аналогічно зробити процедуру Scroll для вертикальної смуги прокрутки:

pictureBox1.Top := vScrollBar1.Value + vScrollBar1.SmallChange;

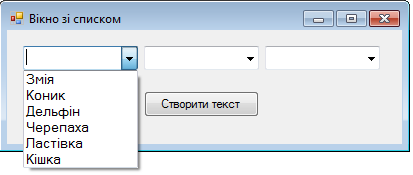
4. Збережіть проект, запустіть та протестуйте його.

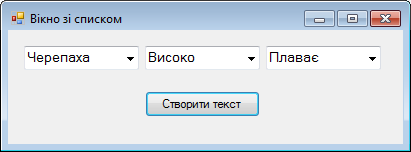
## 8.2 Об’єкти середовища PascalABC. Випадковий вибір. Використання списків.

**Мета:** Ознайомлення студентів з поняттям випадкового вибору. Вивчення основних властивостей компонента ComboBox. Створення програмного додатку з використанням ComboBox.

**Завдання для виконання**

1. Після запуску програми у вікні знаходяться три поля.
2. При натисненні кнопки «Створити текст» з трьох слів випадковим чином складається речення.
3. Для виходу з програми необхідно натиснути мишею на закриваючій кнопці в рядку заголовка.





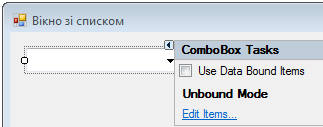
**Опис плану розробки програми**

1. Відкрити новий проект.

2. Розмістити на формі екземпляри компонентів: три вікна зі списком ComboBox, кнопка Button.

3. Виконати наступні дії:

* Змініть властивості форми – змініть назву.
* Для кожного з елементів ComboBox введіть список значень (Items) – іменники, прислівники, дієслова (див. таблицю).



|  |  |  |
| --- | --- | --- |
| **Список іменників** | **Список прислівників** | **Список дієслів** |
| Змія | Швидко | Плаває |
| Коник | Високо | Бігає |
| Дельфін | Повільно | Літає |
| Черепаха | Сильно | Плазує |
| Ластівка | Добре | Стрибає |
| Кішка | Погано | Ховається |

* Змініть напис для кнопки «Створити текст».
* Викличте процедуру button1\_Click . Введіть рядки коду.

|  |
| --- |
| **procedure** Form1.button1\_Click(sender: Object; e: EventArgs);  **var** n: integer;  rnd: random;  **begin**  rnd := **new** Random();  n:=ComboBox1.Items.Count;  ComboBox1.SelectedIndex := rnd.Next(0, n);  ComboBox2.SelectedIndex := rnd.Next(0, n);  ComboBox3.SelectedIndex := rnd.Next(0, n);  **end**; |

* Збережіть проект та протестуйте його