Цитується за виданням:
Програмування. Практикум / Укл.: Семенюк А.Д., Сопронюк Ф.О. – Чернівці: Рута, 2001.– 143 с.

Тематика завдань
для обчислювальної практики.
1 курс, КСМ 2015

Варіанти завдань

	Группа 142а **Прізвище, ім’я**	**Варіант** **№**
1	Андрощук Володимир	1
2	Блошко Юрій	2
3	Бузовський Денис	3
4	Гаврилюк Влас	4
5	Герман Сергій	5
6	Голіней Денис	6
7	Гордійчук Андрій	7
8	Гунько Юрій	8
9	Довгань Денис	9
10	Жмурко Анатолій	10
11	Заєць Богдан	11
12	Катеринчук Руслан	12
13	Кузьмін Михайло	13
14	Михайлюк Олексій	14
15	Пайлик Андрій	15
16	Присакарь Ілля	16
17	Псарюк Станіслав	17
18	Руснак Дмитро	18
19	Семованюк Дмитро	19
20	Сенчак Олександр	20
21	Триднівка Владислав	21
22	Чарковський Ігор	22
23	Якимчук Артем	23

	Группа 142б **Прізвище, ім’я**	**Варіант** **№**
1	Бежнар Анастасія	1
2	Богданов Назар	2
3	Булезюк Іван	3
4	Гарвасюк Руслан	4
5	Гнатюк Юрій	5
6	Гнідан Степан	6
7	Головайко Роман	7
8	Гуменюк Олександр	8
9	Гуцуляк Максим	9
10	Дуляк Віктор	10
11	Зубик Ярослав	11
12	Кіріл Андрій	12
13	Луцу Андріан	13
14	Махіборода Микола	14
15	Мензатюк Василь	15
16	Непийвода Віталій	16
17	Патралюк Іван	17
18	Продан Вадим	18
19	Рудик Василь	19
20	Сопівник Іван	20
21	Тарбай Богдан	21
22	Чепак Олег	22
23	Ясюлянец Владислав	23

Зміст

Побудова проекту, що складається з декількох форм

Розробити в середовищі візуального програмування Delphi проект програми під управлінням головного меню для виконання індивідуального комплексного завдання, яке складається з чотирьох пунктів (кожен пункт – відповідний варіант задачі з лабораторної роботи № 1,2,3,4).

Головна форма проекту повинна мати орієнтовно такий вигляд:

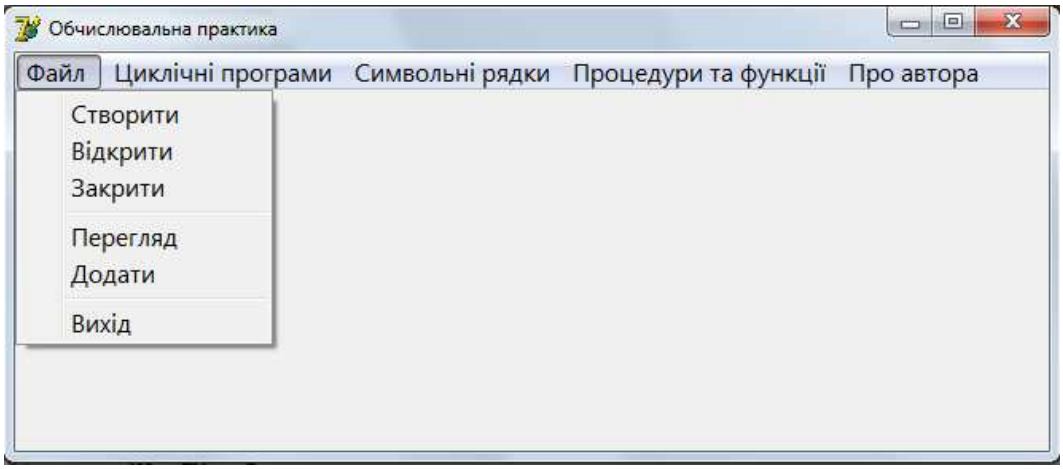

Рис. 1. Головна форма проекту

Головне меню міститиме пункти:

- Файл (Лабораторна робота №1)
- Циклічні програми (Лабораторна робота №2)
- Символьні рядки (Лабораторна робота №3)
- Процедури та функції (Лабораторна робота №4)
- Інформація про автора повинна містити такі відомості: прізвище, група, номери варіантів чотирьох завдань.

Пункт меню **Файл** в діалоговому режимі повинен надавати можливість:

а)створити файл, кожний запис якого містить дані, **тип яких заданий в конкретній умові задачі** (завдання лабораторної роботи №1);

б)відкрити існуючий файл;

в)закрити файл;

г)переглянути містиме типізованого файлу;

д)добавити запис в файл;

е)вийти з програми.

Для цього в пункті головного меню **Файл** створити таке підменю: Створити, Відкрити, Закрити, Переглянути, Добавити, Вихід.

Решта пунктів меню повинна реалізовувати підключення форм Delphi, на яких реалізовано виконання індивідуального завдання лабораторних робіт № 2,3,4.

Підключення додаткової форми до проекту здійснюється наступним чином:

I. Створення форми. Командою File – New – Form створимо нову форму. При створенні форма отримає назву Form2 (число означає номер створюваної форми, і пов'язаний з нею модуль Unit2).

II. Підключення модуля. При посиланні на іншу форму необхідно пам’ятати про взаємозв'язок між формами і модулями. Будь-яка форма має свій модуль. При створенні форми Delphi автоматично створює код модуля і в процесі роботи додаються різноманітні частини коду або програмістом або автоматично середовищем Delphi.

Модуль другої форми має бути включений за допомогою зарезервованого слова uses поточного модуля. Це можна зробити за допомогою команди File – Use Unit, вибравши потрібний модуль посилання на нього автоматично буде додано у поточний модуль. Можна, безпосередньо, дописати підключення модуля Unit2 у розділі uses поточного модуля. Це виглядатиме наступним чином:

```
uses Unit2;
```

Але якщо ми забудемо це зробити то при компіляції програми з'явитися діалогове вікно. В якому буде сказано, що перша форма використовує другу, але модуль другої форми відсутній в списку uses модуля першої форми. Достатньо відповісти Yes і необхідне посилання буде додано.

III. Виклик додаткової форми. Форми в Delphi можна викликати двома способами:

Модально - користувач має можливість працювати тільки на од-

ній формі і перш ніж перейти до іншої, необхідно закрити модальну форму (методом Close, напр., Form2.Close;).

Form2.ShowModal; // викликає форму в модальному режимі

Не модально - користувач може одночасно працювати з декількома формами.

Form2.Show; // викликає форму в немодальному режимі

Лабораторна робота № 1.
Робота з типізованими файлами

Мета роботи:

1. Засвоєння роботи з записами.
2. Отримання практичних навиків роботи з типізованими файлами.

Завдання:

1. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого містить інформацію про **співробітника підприємства** у вигляді (посада; прізвище_ім'я; рік_народження; заробітна плата, ідентифікаційний номер); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) реалізувати пошук співробітника за прізвищем.

2. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого містить інформацію про **інформацію про автомобіль** у вигляді (прізвище власника, марка, номер, колір); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) реалізувати пошук автомобіля за номером та за маркою.

3. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого містить інформацію **про студента** у вигляді (Прізвище Ім'я; група; іноземну мову, яку вивчав; середній бал); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) реалізувати формування вибірки студентів за іноземною мовою, яку вивчали.

4. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого містить інформацію **про абонента телефонного зв'язку** у вигляді (Прізвище Ім'я; номер телефону, адреса проживання); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) реалізувати пошук за прізвищем та номером телефону.

5. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого містить інформацію **про книгу** у вигляді (Прізвище автора, Назва, рік видання); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) реалізувати пошук книги за прізвищем автора та назвою.

6. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл,

кожний запис якого містить інформацію про інформацію **про товар** у вигляді (назва, ціна, код, постачальник, наявна кількість); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) реалізувати пошук товару за назвою та кодом.

7. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого містить інформацію **про розклад руху потягів** у вигляді (номер потягу, пункт відправлення, пункт призначення, дата та час відправлення, дата та час прибуття, кількість місць); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) отримати інформацію про потяги, що відправляються з заданого міста.

8. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого містить інформацію **про зареєстрованих користувачів** у вигляді (логін; пароль; прізвище та ім’я; дата реєстрації; електронна пошта); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) реалізувати пошук користувача за вказаним логіном.

9. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого містить відомості про **приміщення** у вигляді (адреса; площа; дата побудови; ринкова вартість; опис); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) отримати відомості про приміщення, вартість яких не перевищує вказаної.

10. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого містить інформацію про **авіаквитки** у вигляді (пункт відправлення, пункт призначення, дата та час відправлення, дата та час прибуття, ціна, кількість вільних місць); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) реалізувати пошук квитків за вказаними пунктами відправлення та прибуття.

11. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого містить інформацію про **пацієнта** у вигляді (прізвище та ініціали пацієнта, стать, вік, адреса проживання, діагноз); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) створити звіт про пацієнтів для заданого з клавіатури діагнозу.

12. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого містить інформацію про **номери у готелі** у вигляді (номер; кількість кімнат; площа; ціна; інформація про те зайнятий номер чи ні); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) отримати відомості про вільні номери з вказаною кількістю кім-

нат.

13. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого містить інформацію **про замовлення в інтернет** – магазині у вигляді (ім’я замовника; назва товару; кількість; вартість замовлення, інформація про наявність оплати); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) реалізувати пошук замовлень за вказаним товаром при умові відсутності оплати.

14. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, що міститиме **розклад занять студентів групи** у вигляді записів типу (день тижня, номер пари, дисципліна, аудиторія, викладач); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) вивести розклад занять попередньо згрупувавши заняття по дисциплінах.

15. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, що міститиме інформацію **про екзаменаційну сесію** у вигляді записів типу (група; дисципліна; викладач; дата); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) вивести розклад екзаменаційної сесії, попередньо згрупувавши записи по дисциплінах .

16. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, що міститиме **особисті дані студентів** у вигляді запису (Прізвище Ім'я; група; номер паспорта; домашня адреса); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) вивести перелік студентів, що навчаються у вказаній групі.

17. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого містить інформацію про **співробітника підприємства** у вигляді (прізвище_ім’я; посада; відділ; ідентифікаційний номер); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) вивести перелік всіх співробітників вказаного відділу.

18. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого містить інформацію **про постачальника товару** у вигляді запису (постачальник; товар; код товару; ціна); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) вивести список всіх постачальників вказаного товару.

19. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, що міститиме інформацію про **результати екзаменаційної сесії** у вигляді записів вигляду (прізвище_ім’я студента; група; оцінка з першої дисципліни, з дру-

гої, з третьої); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) вивести перелік студентів, що отримуватимуть стипендію за результатами сесії.

20. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, що міститиме інформацію про **клієнтів банку** у вигляді записів наступної структури (прізвище_ім'я; номер паспорту; номер рахунку, баланс рахунку); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл; д) додати новий запис у файл; ж) реалізувати пошук усіх рахунків клієнта за вказаним прізвищем.

21. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого описуватиме **пряму на площині** (пряма визначається координатами двох точок); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл із забезпеченням графічного зображення поточного елемента; д) додати новий запис у файл, забезпечивши можливість попереднього графічного його перегляду.

22. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого міститиме параметри, що визначають **коло на площині** – координати центру (x_0,y_0) та радіус (рівняння кола має вигляд $(x-x_0)^2+(y-y_0)^2=R^2$)); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл із забезпеченням графічного зображення поточного елемента; д) додати новий запис у файл, забезпечивши можливість попереднього графічного його перегляду.

23. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого міститиме параметри, що визначають **еліпс на площині** – координати центру (x_0,y_0) та параметри a,b (рівняння еліпсу має вигляд $\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}=1$); б) відкрити існуючий файл; в) закрити файл; г) переглянути файл із забезпеченням графічного зображення поточного елемента; д) додати новий запис у файл, забезпечивши можливість попереднього графічного його перегляду.

Приклад. Розробити програму, яка в діалоговому режимі дозволяє: а) створити файл, кожний запис якого – дані про точку на площині вигляду (координата *x*, координата *y*, назва точки); б) відкрити існуючий файл; в) закрити файл; г) переглянути вміст файлу; д) додати запис у файл; ж) закрити програму.

Командою File!New Application створимо новий проект.

Згідно з умовою завдання **створимо пункт меню Файл, що міститиме підпункти:** Створити, Відкрити, Закрити, Перегляд, Додати, Вихід.

На головній формі розмістимо меню. Для цього помістимо на форму невидимий компонент MainMenu із сторінки Standard. На формі з'являється значок, який відображається тільки на етапі розробки і дозволяє активізувати компонент MainMenu.

Створення меню здійснюється за допомогою Дизайнера Меню та Інспектора Об'єктів. Управління пунктами меню, їх створення і вилучення здійснюється у вікні Дизайнера Меню, а їх властивості (назва, програмне ім'я та інше) встановлюються в Інспекторі Об'єктів.

Дизайнер Меню можна викликати подвійним клацанням лівої кнопки мишки на значку компонента MainMenu або командою Menu Designer контекстного меню, яке викликається правою кнопкою мишки (Рис.1.1). На екрані з'являється вікно із заголовком Form1.MainMenu1 і одним порожнім пунктом меню (Рис.1.1). **Це і є Дизайнер Меню**.

Виберемо цей пункт, лівою кнопкою мишки, і присвоїмо йому назву Caption = &Файл та програмне ім'я Name = FF. Значок & означає, що символ, який слідує за ним, буде підкресленим і даний пункт можна вибирати як мишкою так і за допомогою клавіш ALT+<підкреслений символ>.

Аналогічно створюються горизонтальні пункти меню з назвами *Циклічні програми*, *Символьні рядки*, *Процедури та функції* та *Про автора*.

Тепер для пункту ***Файл*** створимо відповідне вертикальне меню. Як показано на (Рис.1.2) виберемо в Дизайнері Меню пункти, а в Інспекторі Об'єктів встановимо їм назви (властивість Caption): &Створити, &Відкрити, &Закрити, - знак мінус, &Перегляд, До&бавити, - знак мінус, Ви&хід. Використання в якості назви пункту знаку мінус дозволяє об'єднати деякі пункти у групи і розділити їх лініями.

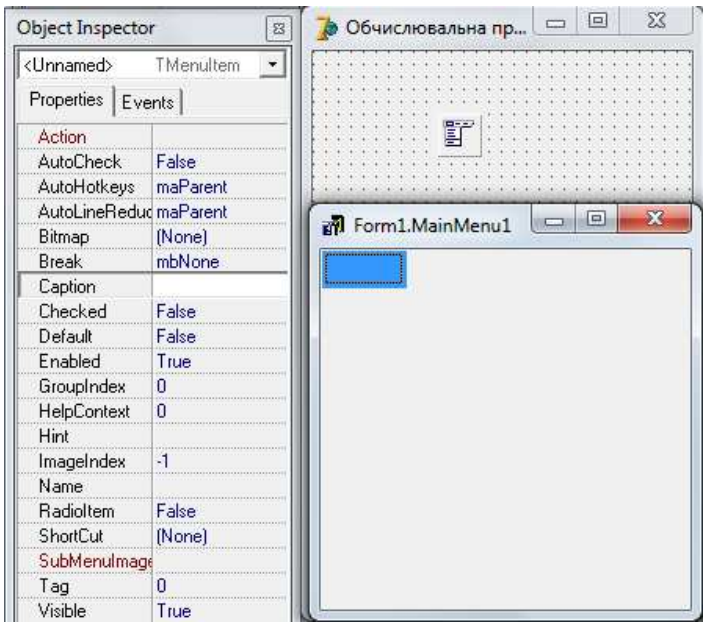

Рис.1.1. Головна форма, Інспектор Об’єктів і Дизайнер Меню.

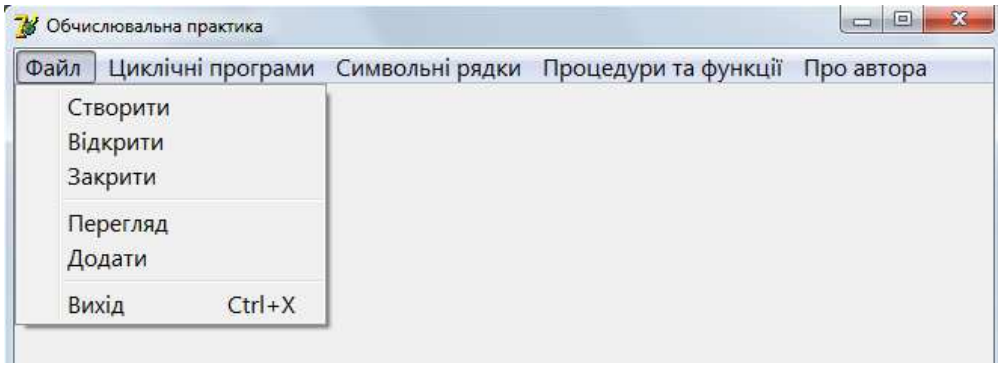

Рис. 1.2. Головне меню.

Пунктам меню можна призначати **клавіатурні скорочення** (властивість ShortCut), що дозволяє виконувати команди з допомогою клавіатури. Наприклад, для пункту Вихід призначимо клавіатурне скорочення Alt+X.

Деякі команди меню можуть бути недоступними користувачу в той або інший момент часу виконання програми. Такі команди потрібно зробити забороненими. Заборонені команди виглядають на екрані тьмяними, і їх вибір ігнорується (Рис.1.2). Стан команди визначається властивістю Enabled (Enabled=false - заборонена, Enabled = true - незаборонена), за замовчуванням Enabled =true.

У нашому випадку, якщо файл не відкритий, то команди Закрити, Перегляд, Додати потрібно зробити забороненими (Enabled =false), а команди Створити, Відкрити, Вихід - незабороненими (Enabled =true).

Після створення головного меню потрібно **написати обробники для всіх команд**. Для цього потрібно вибрати відповідний пункт меню перейти в Інспе-

кторі Об’єктів **на сторінку Evants** і двічі клацнути лівою кнопкою мишки в полі значення події **OnClick**. У програмному модулі з’явиться заготовка в яку потрібно вписати текст обробника.

Для написання обробників пунктів меню **Створити, Відкрити** використаємо компоненти OpenDialog та SaveDialog (вкладка Dialogs палітри компонетів), які попередньо розмістимо на формі.

// У розділі **type** опишемо тип – точка на площині:

```
to4ka = record
    x, y: integer;
    a: string[20];
end;
```

// Опис глобальних змінних у розділі **var** модуля головної форми:

```
  f: file of to4ka;
  FOpened: boolean = false;
  FileName: string = '';
```

// обробник пункту меню Файл

```
procedure TForm1.FileFClick(Sender: TObject);
begin
  if FOpened // файл відкритий
    then
      begin
          CloseF.Enabled:=true; // підпункт меню Закрити
          viewF.Enabled:=true;  // підпункт меню Перегляд
          AddF.Enabled:=true;   // підпункт меню Додати
      Else  // файл не відкритий
        begin
          CloseF.Enabled:=false; // підпункт меню Закрити
          viewF.Enabled:=false;  // підпункт меню Перегляд
          AddF.Enabled:= false;   // підпункт меню Додати
        end;
end;
```

// обробник пункту меню Створити

```
procedure TForm1.CreateFClick(Sender: TObject);
begin
  if Fopened
    then
      CloseFile(f);
      if SaveDialog1.Execute
        then
          begin
            FileName:=SaveDialog1.FileName;
            AssignFile(f, filename);
            rewrite(f);
            fopened:=true;
          end;
```

```
end;
```

// обробник пункту меню <u>Відкрити</u>

```
procedure TForm1.OpenFClick(Sender: TObject);
begin
  if fopened
    then
      CloseFile(f);
      if OpenDialog1.Execute
        then
          begin
            FileName:=OpenDialog1.FileName;
            AssignFile(f, filename);
            reset(f);
            fopened:=true;
          end;
end;
```

// обробник пункту меню <u>Закрити</u>

```
procedure TForm1.CloseFClick(Sender: TObject);
begin
  if fopened
    then
      begin
        closefile(f);
        fopened:=false;
      end;
end;
```

// обробник пункту меню <u>Перегляд</u>

```
procedure TForm1.viewFClick(Sender: TObject);
begin
  if FileSize(f) <> 0
    then
      viewForm.ShowModal // виклик допоміжної форми на якій
            //відображатимуться результати перегляду файлу
    else
      ShowMessage('File is empty!');
end;
```

// обробник пункту меню <u>Додати</u>

```
procedure TForm1.AddFClick(Sender: TObject);
begin
  AddForm.ShowModal; // виклик допоміжної форми для введення
                // даних що додаватимуться
end;
```

// обробник пункту меню <u>Вихід</u>

```
procedure TForm1.ExitFCloseFClick(Sender: TObject);
begin
  close;
```

```
end;
```

// обробник пункту меню <u>Циклічні програми</u>

```
procedure TForm1.CycleLabClick(Sender: TObject);
begin
  CycleForm.ShowModal; // виклик допоміжної форми для виконання
                // завдання ″Циклічні програми″


end;
```

// обробник пункту меню <u>Символьні рядки</u>

```
procedure TForm1.SymbolLabClick(Sender: TObject);
begin
  SymbolForm.ShowModal; //виклик допоміжної форми для виконання
                // завдання ″Символьні рядки″

 end;
```

// обробник пункту меню <u>Процедури та функції</u>

```
procedure TForm1.ProcLabClick(Sender: TObject);
begin
    ProcForm.ShowModal; //виклик допоміжної форми для виконання
                // завдання ″Процедури та функції″

end;
```

// обробник пункту меню <u>Про автора</u>

```
procedure TForm1.AuthorClick(Sender: TObject);
begin
 AuthorForm.ShowModal; //виклик форми з інформацією про автора
end;

end.
```

Лабораторна робота № 2.
Циклічні програми. Обчислення з заданою точністю

Мета роботи:

1. Засвоєння операторів циклу.
2. Отримання практичних навиків розробки ітераційних алгоритмів і програм для наближеного обчислення нескінченних сум та пошуку членів у нескінченних послідовностях, що задовольняють певні умови.

Завдання:

Задано дійсні величини x, a, ε $(x \neq 0, a \neq 0, \varepsilon > 0)$. Розробити програму, яка обчислює значення суми з заданою точністю ε і вказує кількість врахованих доданків.

1. $\sum_{k=1}^{\infty} \frac{(-1)^k \ln x^{2k}}{a^k + k!}$ **2.** $\sum_{k=0}^{\infty} \frac{\ln(a+x)^{2k}}{2^k + k!}$ **3.** $\sum_{k=1}^{\infty} \frac{(-1)^k x^{-k}}{a^k (2k)!}$

4. $\sum_{k=1}^{\infty} \frac{\sin(a^k + x^k)}{k!!}$ **5.** $\sum_{k=0}^{\infty} \frac{\sin a^k + \cos x^k}{(k^2)!}$ **6.** $\sum_{k=1}^{\infty} \frac{(-1)^k x^{-4k}}{a^4 + k!}$

7. $\sum_{k=1}^{\infty} \frac{\cos x^k + \sin a^k}{(2k-1)!}$ **8.** $\sum_{k=0}^{\infty} \frac{\cos(a^k + x^k)}{(k^2)!}$ **9.** $\sum_{k=0}^{\infty} \frac{(a+x)^{-k}}{a^{2k} + k!}$

10. $\sum_{k=1}^{\infty} \frac{e^{-k}}{a^{2k} + k!}$ **11.** $\sum_{k=1}^{\infty} \frac{\sin x^k}{a^{2k} + (2k)!}$ **12.** $\sum_{k=0}^{\infty} \frac{\cos^k x}{a^{4k} + k!}$

13. $\sum_{k=0}^{\infty} \frac{(a+x)^{-k}}{k!!}$ **14.** $\sum_{k=0}^{\infty} \frac{\cos^k (a+x)}{a^k k!}$ **15.** $\sum_{k=1}^{\infty} \frac{\sin x^k}{a^{2k} (2k)!}$

Задано дійсні величини x, ε $(\varepsilon > 0)$. Розробити програму, яка знаходить і друкує перший член a_n і його номер у заданій послідовності, для якого виконується умова $|a_n - a_{n-1}| < \varepsilon$. Обмежитись розглядом перших 10^2 членів послідовності.

16. $a_1 = x, \quad a_n = 2a_{n-1} + \frac{x}{4 + a_{n-1}^2}, \qquad n = 2,3,\ldots .$

17. $a_1 = x, \quad a_n = 2a_{n-1} + \dfrac{16+x}{4+\left|a_{n-1}^3\right|}, \qquad n = 2,3,\ldots .$

18. $a_1 = x, \quad a_n = \sqrt{\left|4a_{n-1}^2 - 2x\right|}, \qquad n = 2,3,\ldots .$

19. $a_1 = x, \quad a_n = 3 + \dfrac{1}{2^n} sin^2\left(a_{n-1} - x\right), \qquad n = 2,3,\ldots .$

20. $a_0 = x, \quad a_n = \dfrac{1}{2}\left(a_{n-1} + \dfrac{x}{a_{n-1}}\right), \qquad n = 1,2,\ldots .$

21. $a_0 = x, \quad a_n = \dfrac{a_{n-1}+1}{a_{n-1}+2}, \qquad n = 1,2,\ldots .$

22. $a_0 = x, \qquad a_n = \dfrac{4}{5}a_{n-1} + \dfrac{x}{5a_{n-1}^4}, \qquad n = 1,2,\ldots .$

23. $a_0 = x, \quad a_n = \dfrac{1}{4}\left(3a_{n-1} + \dfrac{x}{a_{n-1}^3}\right), \qquad n = 1,2,\ldots .$

24. $a_n = \left(1 - \dfrac{x}{2!}\right)\left(1 + \dfrac{x}{3!}\right)\cdots\left(1 - \dfrac{(-1)^n x}{(n+1)!}\right), \quad n = 1,2,\ldots .$

25. $a_n = \dfrac{(-1)^n x^{2n}}{n!!}, \qquad n = 1,2,\ldots .$

26. $a_n = \dfrac{x^{2n}\, sin\left(x^n\right)}{\left(n^2\right)!}, \qquad n = 1,2,\ldots .$

27. $a_n = \dfrac{(-1)^{n+1}}{(2n+1)!}\left(\dfrac{x}{3}\right)^{4n+2}, \qquad n = 1,2,\ldots .$

28. $a_n = \dfrac{1}{3^n} cos\left(x^{n-1}\right), \qquad n = 1,2,\ldots .$

29. $a_n = \dfrac{x^n}{2^n + (2n)!}, \qquad n = 1,2,\ldots .$

30. $a_n = \dfrac{(-1)^n x^{n+2}}{(n+1)(n+2)!}, \qquad n = 1,2,\ldots .$

Теоретичні відомості

I. Оператори повторень

Оператори циклу.

1. *Оператор циклу* WHILE має таку структуру

while <умова> do <оператор>;

де while, do (поки [виконується умова] виконувати])– зарезервовані слова; – вираз логічного типу; <оператор> –довільний оператор. При виконанні оператора while обчислюється вираз <умова> і якщо його

значення true, то виконується <оператор> і обчислення виразу повторюється знову. Якщо значення виразу <умова> дорівнює false, то виконання оператора while завершується.

2. Оператор циклу REPEAT…UNTIL має таку структуру

repeat <тіло циклу> until <умова>;

де repeat, until – зарезервовані слова (повторювати [до тих пір] поки [не буде виконана умова]); <умова> – вираз логічного типу; <тіло циклу> – довільна послідовність операторів. При виконанні оператора repeat …until while виконується хоча б один раз <тіло циклу>, після цього обчислюється вираз <умова> і якщо його значення false, то знову виконується <тіло циклу>. Якщо значення виразу <умова> дорівнює true, то виконання оператора repeat …until завершується.

3. *Оператор циклу* FOR має таку структуру

for <параметр циклу>:=<початкове значення> $\left\{ \begin{matrix} \text{to} \\ \text{downto} \end{matrix} \right\}$ <кінцеве значення> do <оператор>;

де for, to, downto, do – зарезервовані слова (для, до, донизу, виконати); <параметр циклу> – змінна порядкового типу; <початкове значення>, <кінцеве значення> – вирази того ж типу; <оператор> – довільний оператор.

Оператор for з зарезервованим словом to виконується за схемою:

1. Обчислюється вираз <початкове значення> і його значення присвоюється змінній <параметр циклу>.
2. Перевіряється умова <параметр циклу> ≤<кінцеве значення>, якщо умова виконана, то перехід до п.3, інакше перехід до п.5.
3. Виконання <оператора>.
4. Збільшення параметра циклу, <параметр циклу>:=<параметр циклу>+1. Перехід до п.2.
5. Завершення виконання оператора for.

При виконанні оператора for з зарезервованим словом downto параметр циклу зменшується на одиницю, а керуюча умова має вигляд <параметр циклу> ≥ <кінцеве значення>.

II. Обчислення нескінченних сум. При розробці алгоритмів обчислення нескінченних сум процес слід організувати так щоб для обчислення чергового доданку використовувалися результати обчислення попереднього доданку. Наприклад, для обчислення n-го доданку суми $\sum_{k=0}^{\infty}\frac{x^n}{n!}$ потрібно $(n-1)$- ий доданок помножити на множник $\frac{x}{n}$, тобто $\frac{x^n}{n!}=\frac{x^{n-1}}{(n-1)!}\times\frac{x}{n}$. Якщо такої закономірності немає, то можна цей підхід використати для обчислення окремих частин доданку. Наприклад, для обчислення доданків суми $\sum_{n=0}^{\infty}\frac{sin(a+x)^n}{3^n+n!}$ можна окремо обчислювати $(a+x)^n=(a+x)^{n-1}\times(a+x)$, $3^n=3^{n-1}\times 3$, $n!=(n-1)!\times n$, а потім обчислити доданок $\frac{sin(a+x)^n}{3^n+n!}$. Обчислення суми з заданою точністю ε означає, що процес накопичення доданків суми завершується, якщо деякий n-ий доданок буде $\left|\frac{sin(a+x)^n}{3^n+n!}\right|<\varepsilon$.

Програма для обчислення другої суми в консольному режимі має вигляд

```
Program LABR2_1;
{$APPTYPE CONSOLE}
uses Sysutils;
VAR     a, x, e: double;
       s, sn, st3, stax: double;
       n, nf: longint;
BEGIN
{Введення початкових даних}
        writeln('Введіть a, x, e');
        readln(a,x,e);
        sn:=sin(1)/2;
        s:=sn;
        n:=0; stax:=1; st3:=1; nf:=1;
        while abs(sn) > e do
         begin
           n:=n+1;
           stax:=stax*(a+x);
           st3:=st3*3;
           nf:=nf*n;
```

```
            sn:=sin(stax)/(st3+nf);
            s:=s+sn;
          end;
{Виведення результатів}
          writeln('Значення суми =', s);
          writeln('Враховано доданків =', n);
readln;
END.
```

III. Обчислення елементів нескінченної послідовності. При розробці алгоритмів знаходження елементів у нескінченній послідовності, які задовольняють певні умови, потрібно обчислювальний процес організувати так, щоб на кожному кроці обчислений елемент ставав попереднім для обчислення наступного елемента.

Приклад. Розробити проект програми, яка для заданих x, ε $(\varepsilon > 0)$ знаходить і друкує перший член a_n і його номер у послідовності $a_0 = x,\ a_n = \frac{1}{2}\left(a_{n-1} + \frac{x}{a_{n-1}}\right),\ n = 1,2,\ldots$, для якого виконується умова $|a_n - a_{n-1}| < \varepsilon$ і розглядається перших 10^3 членів послідовності.

Для розв'язку задачі командою File!New Application створимо новий проект. На екрані з'явиться чиста форма із заголовком Form1. Встановимо цій формі заголовок Caption = Елементи послідовності та присвоїмо їй програмне ім'я Name = FV. Тепер командою File!Save All запишемо програмний модуль під іменем ULABR2_2.pas, а проект – LABR2_2.dpr.

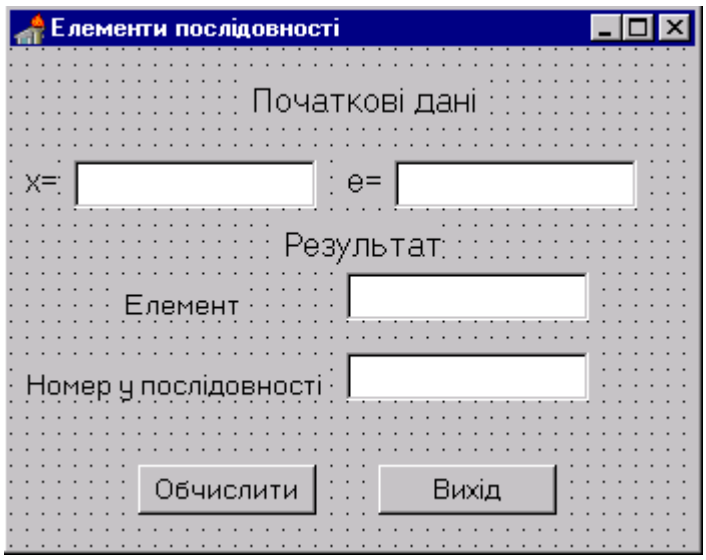

Рис. 2.1. Форма Елементи послідовності.

Розробимо форму для введення початкових даних і виведення результату. Розмістимо на формі два компоненти Edit для введення початкових даних x, e, і два для виведення результатів. Присвоїмо цим компонентам програмні імена Edit1, Edit2, Edit3, Edit4, встановлені за замовчуванням (властивість Name) і очистимо їм значення властивості Text. Пояснення до цих компонентів зробимо за допомогою компонента Label (властивість Caption).

Крім цього, розмістимо на формі дві керуючих кнопки (компонент Button) з написами Обчислити та Вихід (властивість Caption) і програмними іменами Button1, Button2 (властивість Name) (Рис. 2.1.).

Обробники кнопок Обчислити та Вихід містяться в програмному модулі ULABR2_2. і мають вигляд:

{Обробник кнопки Обчислити}

```
procedure TFP.Button1Click(Sender: TObject);
var     x, e: double;
    ap, an: double;
    n: integer;
begin
{Початкові дані}
  x:=StrToFloat(Edit1.Text);
  e:=StrToFloat(Edit2.Text);
{Обчислення елемента}
 an:=x; n:=0;
  repeat
```

```
        ap:=an;
        n:=n+1;
    an:=(ap+x/ap)/2;
  until (abs(an-ap) < e) or (n >= 1000);
 {Виведення результатів}
  Edit3.Text:=FloatToStr(an);
  Edit4.Text:=FloatToStr(n);
end;
{Обробник кнопки Вихід}
procedure TFP.Button2Click(Sender: TObject);
begin
Close;
end;
```

Лабораторна робота № 3.
Символьні рядки

Мета роботи:

1. Засвоєння структурованих даних типу символьний рядок і операцій над цими даними.
2. Отримання практичних навиків обробки і редагування текстів.

Завдання:

1. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка вилучає з цього тексту всі слова з подвоєнням літер і записує їх в окремий рядок, розділяючи пробілами. Друкує окремо вилучені слова і текст, що залишився після вилучення слів.

2. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка вилучає з цього тексту всі слова найбільшої довжини. (Слів найбільшої довжини може бути декілька). Друкує текст, що залишився після вилучення слів.

3. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка знаходить і друкує всі симетричні слова. (Симетричне слово це – абввба).

4. Задано два тексти, слова в яких розділені пробілами і розділовими знаками. Розробити програму, яка вилучає із першого тексту всі слова, що містяться у другому тексті.

5. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка вилучає в кожному слові цього тексту всі наступні входження першої літери.

6. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка вилучає в кожному слові цього тексту всі попередні входження останньої літери.

7. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка вилучає з цього тексту всі повторні входження слів.

8. Задано текст, слова в якому розділені пробілами і розділовими зна-

ками. Розробити програму, яка знаходить і вилучає всі слова, що входять в цей текст по одному разу.

9. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка в словах непарної довжини цього тексту вилучає середню літеру.

10. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка знаходить і друкує всі слова, що входять у заданий текст по одному разу.

11. Задано текст, слова в якому розділені пробілами і розділовими знаками, та два окремих слова. Розробити програму, яка замінює всі входження в заданий текст першого слова другим словом.

12. Задано два тексти, слова в яких розділені пробілами і розділовими знаками. Розробити програму, яка вилучає із другого тексту всі входження слів першого тексту.

13. Задано два тексти, слова в яких розділені пробілами і розділовими знаками. Розробити програму, яка створює третій текст із слів першого тексту, які не входять у другий текст, розділяючи їх пробілами.

14. Задано два тексти, слова в яких розділені пробілами і розділовими знаками, та окреме слово. Розробити програму, яка після кожного входження заданого слова в перший текст вставляє в нього другий текст.

15. Задано текст, слова в якому розділені пробілами і розділовими знаками, та окремий символ. Розробити програму, яка знаходить і друкує всі слова, в які входить заданий символ найбільшу кількість разів.

16. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка знаходить і друкує найдовший ланцюжок із слів однакової довжини.

17. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка вилучає із заданого тексту всі слова непарної довжини.

18. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка знаходить і друкує слово з найбі-

льшою кількістю однакових символів (якщо таких слів декілька, то взяти перше з них).

19. Задано текст із малих латинських літер, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка знаходить і друкує всі слова, літери в яких розміщені в лексикографічному порядку.

20. Задано два тексти, слова в яких розділені пробілами і розділовими знаками. Розробити програму, яка створює третій текст із слів першого тексту, які входять у другий текст, і розділяє їх пробілами.

21. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка вилучає із заданого тексту всі попередні входження останнього слова.

22. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка вилучає із кожного слова заданого тексту всі повторні входження кожної літери.

23. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка вилучає з цього тексту всі слова з повторенням літер.

24. Задано текст із малих латинських букв, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка знаходить і друкує всі слова, букви в яких розміщені в алфавітному порядку.

25. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка знаходить і вилучає всі слова, літери в яких розміщені в лексикографічному порядку.

26. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка вилучає із кожного слова заданого тексту всі попередні входження останньої літери.

27. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка знаходить і друкує всі слова, буква ’А‘ або а’ в яких зустрічається найбільшу кількість разів.

28. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка знаходить і друкує всі слова, букви в яких не повторюються.

29. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка вилучає у кожному слові цього тексту всі повторні входження кожної букви.

30. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка вилучає із цього тексту всі слова, які починаються з голосної букви.

Теоретичні відомості

Для обробки текстів в Object Pascal використовуються символьні рядки таких типів:

- ShortString або String{N}, де $N \leq 255$ – короткий рядок;
- String – довгий рядок;
- PChar – нуль-термінальний рядок,
- WideString –широкий рядок.

Приклади опису рядків у програмі:

```
Var Sk: ShortString;  {короткий рядок максимальної}
                      {довжини 255 символів}
       Sks: String[5];  {короткий рядок}
                        { довжиною до 5 символів}
       Ss1, Ss2: String;    {довгий рядок}
       Sp: Pchar;           {нуль-термінальний рядок}
       Sw: WideString;      {широкий рядок}
```

Довжина короткого рядка від 0 до 255 байтів, а довгого, нуль-термінального і широкого рядків від 0 до 2 Гбайтів.

Для описаного в програмі короткого рядка Sk компілятор виділить у статичній пам'яті 256 байтів з номерами від 0 до 255, а для Sks відповідно 6 байтів. У нульовому байті буде міститися біжуча довжина рядка, а починаючи з першого байта буде розміщатися ланцюжок символів. Так як у байт можна записати максимальне число 255, то довжина короткого рядка не може перевищувати 255. При виконанні оператора Sk:='Рядок символів'; у нульовий байт буде записана довжина – 14, а з першого байта розміститься ланцюжок символів. Якщо довжина рядка символів перевищує описану довжину, то лишні символи будуть відкидатися. Так, при виконанні оператора Sks:='Рядок

символів’; змінна Sks отримає значення – ’Рядок’

Описаний у програмі довгий рядок Ss1 буде розміщатися у динамічній пам’яті. При цьому, компілятор виділить для змінної Ss1 у статичній пам’яті 4 байти в яких буде міститися адреса виділеної динамічної пам’яті. Змінна Ss1 буде вказівником на цю пам’ять.

Динамічна пам’ять виділяється операційною системою під час виконання програми. У виділеній пам’яті, починаючи з першого байта, розміщується ланцюжок символів, термінальний нуль – #0, яким завершується рядок, і 4-байтовий лічильник посилань.

При виконанні оператора Ss1:=’Рядок символів’; програма визначить необхідний розмір пам’яті 14+5=19 байтів і звернеться до операційної системи. Операційна система виділить у динамічній пам’яті область із 19 байтів і у змінну Ss1 помістить її адресу. У виділеній пам’яті, починаючи з першого байта, розмістить символи – Рядок символів, термінальний нуль – #0, а в лічильник посилань занесе одиницю.

Лічильник посилань використовується для кешування пам’яті. Так, при виконанні оператора Ss2:=Ss1; пам’ять для розміщення змінної Ss2 не виділяється, а їй присвоюється значення вказівника Ss1 і лічильник посилань збільшується на одиницю. Таким чином, обидві змінні Ss1 і Ss2 будуть вказувати на одну і ту ж область пам’яті. При виконанні оператора Ss1:=’Це–’+’Рядок символів’; лічильник посилань зменшиться на одиницю, а операційна система виділить у динамічній пам’яті область із 19+5=24 байтів і у Ss1 помістить її адресу. У виділеній пам’яті, починаючи з першого байта, розмістить ланцюжок символів – ’Це – Рядок символів’, термінальний нуль – #0 і лічильник посилань із значенням – 1. Тепер змінні Ss1 і Ss2 будуть вказувати на різні області пам’яті, а лічильники посилань будуть містити по одиниці. Виділена для розміщення довгого рядка String область пам’яті звільняється, якщо лічильник посилань стає рівним нулю.

Подібним чином здійснюється робота з пам’яттю для нуль-термінальних рядків типу Pchar. Лічильник посилань тут не використовується, тому для кожного рядка виділяється окрема область пам’яті. Нуль-термінальні рядки використовуються при зверненні до API-функцій Windows (API – Application Program Interface – інтерфейс прикладних програм). Оскільки компоненти Delphi беруть на себе всі проблеми зв’язку з API-функціями, то програмісту рідко до-

водиться звертатися до нуль-термінальних рядків.

Однобайтові символи коду ANSI мають 256 можливих значень, достатніх для відображення будь-якої європейської мови, а для азіатських мов цього недостатньо. Тому міжнародна комісія виробила код Unicode, символи якого займають у пам'яті два байти і мають 65536 можливих значень. Цей код дозволяє представити всі символи всіх мов світу. Символи Unicode описуються стандартним типом WideChar, а складені з них рядки – типом WideString. Усі програми, що використовують OLE-технологію обміну рядками, повинні використовувати символи Unicode.

Звернення до окремих символів рядка здійснюється за допомогою індексу. При виконанні операторів

Ss1:='Рядок символів'; Ss1[4]:='к'; Ss1[5]:='и';

змінна Ss1 отримає значення – 'Рядки символів'. Нумерація символів у коротких і довгих рядках починається з одиниці, а в нуль-термінальних з нуля.

Над символьними рядками можуть виконуватися операції:

- конкатенація (зчіплення рядків) – (+);
- відношення – (=, <>, <, >, <=, >=).

Наприклад, при виконанні оператора

Ss1:='Це – ' + 'Рядок' + 'символів';

змінна Ss1 отримає значення – 'Це – Рядок символів'.

Операції відношення виконуються над двома рядками посимвольно, зліва направо з врахуванням внутрішнього кодування символів. Якщо один рядок коротший, то він доповнюється символами #0. Результат операції відношення бульового типу. Наприклад,

'A' < 'B' – true;
'A' < '1' – false;
'Object' < 'Object Pascal' – true;
'Пас' > 'Pascal' – true.

Всі останні дії над рядками і символами реалізуються за допомогою стандартних процедур і функцій, які наведені в додатку.

Приклад. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити проект програми, яка вилучає з цього тексту всі слова з подвоєнням літер.

Для розв'язку задачі командою File!New Application створимо новий проект. На екрані з'явиться чиста форма з заголовком Form1. Присвоїмо цій формі заголовок Caption = Символьні дані і програмне ім'я Name = FS. Тепер командою File!Save All запишемо програмний модуль під іменем ULABR7_1.pas, а проект – LABR7_1.dpr.

На формі розмістимо один компонент Edit для введення початкових даних і другий для виведення результатів. Присвоїмо цим компонентам програмні імена Edit1, Edit2, встановлені за замовчуванням (властивість Name) і очистимо їм значення властивості Text. Пояснення до цих компонентів зробимо за допомогою компонента Label (властивість Caption)

Крім цього, розмістимо на формі дві керуючих кнопки (компонент Button) з написами Виконати та Вихід (властивість Caption) і програмними іменами Button1, Button2 (властивість Name за замовчуванням) (Рис. 7.1).

Обробники кнопок Виконати та Вихід містяться у програмному модулі ULABR7_1 і мають вигляд:.

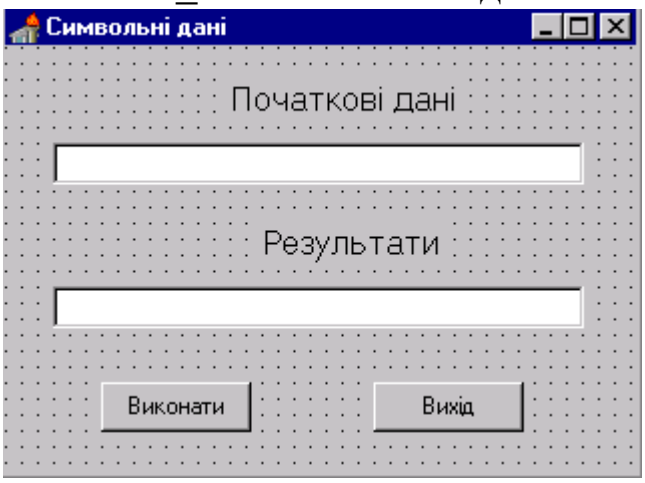

Рис. 7.1. Форма Символьні дані.

```
{ Обробник команди Виконати }
procedure TFS.Button1Click(Sender: TObject);
CONST m=[' ',',','.',';','.','!','?',':'];
VAR s: string;
    p, i, j: integer;
    f: boolean;
```

```
BEGIN
{Введення початкових даних}
s:=Edit1.Text;
{Пошук і вилучення слів з подвоєнням букв}
   i:=1;
   repeat
{Пошук початку слова}
       while s[i] in m do i:=i+1;
       p:=i;
{Пошук кінця слова}
       while not (s[i] in m) and (i <= length (s)) do
       i:=i+1;
{Перевірка умови подвоєння букв}
    j:=p; f:=false;
        while (j <= i-2) and (not f) do
        if s[j]=s[j+1] then  f:=true else j:=j+1;
{Вилучення слів з подвоєнням букв }
{Корекція параметра циклу і}
    if f then begin delete (s, p, i-p); i:=p; end;
   until i >= length (s);
{Виведення перетвореного тексту}
    Edit2.Text:=s;
end;
{ Обробник команди Вихід }
procedure TFS.Button2Click(Sender: TObject);
begin
Close;
end;
```

Розробка проекту завершена. Тепер командою Run!Run проект можна запустити на виконання. По завершенню компіляції потрібно ввести початкові дані і натиснути кнопку Виконати. Для завершення роботи програми потрібно натиснути кнопку Вихід.

Лабораторна робота № 4. Процедури та функції

Мета роботи:

1. Засвоєння структури процедур і функцій, звернення до процедур і функцій, типів параметрів та способів їх передачі.
2. Отримання практичних навиків розробки програмних засобів з використанням процедур і функцій.

Завдання:

1. Задано многочлен $P_n(x)$ степеня $n \le 100$, коефіцієнти якого містяться у дійсному масиві $A(n+1)$, та дійсні числа $x_1, x_2, ..., x_m$, $m \le 15$. Розробити програму обчислення коефіцієнтів многочлена $P_n'(x)$ і значень многочленів $P_n(x)$ та $P_n'(x)$ у точках x_i, $i = 1,2,...,m$. Написати і використати процедуру для обчислення коефіцієнтів і функцію для обчислення значення многочлена. Надрукувати обчислені коефіцієнти і таблицю значень многочленів, у кожному рядку якої розмістити значення x_i, $P_n(x_i)$, $P_n'(x_i)$.

2. Задано масиви чисел $A(n)$, $n \le 300$ і $B(m)$, $m \le 400$. Розробити програму побудови об'єднання масивів $A \cup B$ і обчислення суми його елементів. ($A \cup B$ – множина елементів A і B, взятих по одному разу). Написати і використати процедуру для побудови об'єднання і функцію для обчислення суми. Надрукувати елементи об'єднання та їх суму.

3. Задана матриця $X(n,n)$, $n \le 15$. Розробити програму перетворення заданої матриці так, щоб добутки елементів рядків утворювали неспадну послідовність. Написати процедуру для перетворення матриці та функцію для обчислення добутку елементів вектора і використати її для обчислення добутків елементів рядків. Надрукувати перетворену матрицю по рядках.

4. Задана матриця $X(n,n)$, $n \le 15$. Розробити програму побудови матриці $Y(n,n)$ за правилом: $Y(i,j)$ дорівнює скалярному добутку i-го рядка на j-ий стовпчик матриці X. Написати процедуру для побудови матриці та функцію для обчислення скалярного добутку векторів і використати її для обчислення скалярних добутків рядків і стовпчиків. Надрукувати отриману матрицю по рядках.

5. Задана матриця $A(n,n)$, $n \le 15$.. Розробити програму перетворення заданої матриці так, щоб суми елементів стовпців утворювали незростаючу послідовність. Написати процедуру для перетворення матриці та функцію для обчислення суми елементів вектора і використати її для обчислення сум елементів стовпців. Надрукувати перетворену матрицю по рядках.

6. Задана матриця $X(n,n)$, $n \le 15$. Розробити програму, яка будує вектори: $A(i)$ – сума елементів i -го рядка, $B(j)$ – сума елементів j -го стовпчика заданої матриці, $i, j = 1,2,...,n$. Написати процедуру для побудови векторів та функцію для обчислення суми елементів вектора і використати її для обчислення сум елементів рядків і стовпців. Надрукувати отримані вектори по п’ять елементів у рядку.

7. На площині задані множина n точок, $n <= 200$ і множина m точок, $m <= 100$. За означенням віддаль між множинами точок – це віддаль між найближче розміщеними точками цих множин. Розробити програму обчислення віддалі між заданими множинами і визначення координат найближче розміщених точок (якщо таких пар точок декілька, то взяти одну із них). Для обчислення віддалі між множинами та визначення координат найближче розміщених точок використати процедуру, а для обчислення віддалі між точками – функцію.

8. Задано два масиви чисел $A(n), (n \le 300)$ і $B(m), (m \le 100)$. Розробити програму обчислення суми $\sum_{x \in A \cap B} \sin x$ перерізу масивів $A \cap B$. ($A \cap B$ – множина елементів A, що входять у B і взятих по одному разу). Використати процедуру для побудови перерізу і функцію для обчислення суми. Надрукувати елементи перетину та їх суму.

9. На площині задано множину n точок, $n \le 300$, і окрему точку d. Розробити програму, яка підраховує кількість різних точок a,b,c із заданої множини таких, що чотирикутник $abcd$ є квадратом і обчислює площу найменшого з них. Використати процедуру для підрахунку кількості точок і обчислення площі найменшого квадрата і логічну функцію для перевірки умови, що точки $abcd$ утворюють квадрат.

10. Задана матриця $X(n,n)$, $n \le 15$. Розробити програму, яка будує вектор $B(k)$, $(k \le n)$ з номерів тих рядків матриці, елементи яких утворюють спадну послідовність. Якщо таких рядків немає, то друкує повідомлення про це. Використати процедуру для побудови вектора і ло-

гічну функцію для перевірки умови того, що послідовність спадна. Надрукувати вектор по п’ять елементів у рядок.

11. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка вилучає всі повторні входження кожного слова в цей текст. Використати процедуру для вилучення слів і логічну функцію для перевірки входження слова в текст. Надрукувати текст, що залишився після вилучення слів.

12. Задано масиви чисел $A(n)$, $n \le 300$ і $B(m)$, $m \le 400$. Розробити програму побудови симетричної різниці масивів $A \setminus B \cup B \setminus A$ і пошуку її мінімального елемента. ($A \setminus B \cup B \setminus A$ – множина елементів A, що не входять у B і множина елементів B, що не входять в A і взятих по одному разу)**.** Використати процедуру для побудови симетричної різниці і функцію для пошуку мінімального елемента. Надрукувати елементи симетричної різниці та її мінімальний елемент.

13. Задано два тексти, слова в яких розділені пробілами і розділовими знаками. Розробити програму побудови нового тексту, в який входять слова першого і другого текстів по одному разу, розділені пробілами. Використати процедуру для побудови нового тексту і функцію перевірки входження слова в текст. Надрукувати побудований текст.

14. Задана матриця $A(n,n)$, $n \le 15$. Розробити програму побудови вектора $B(m)$ $(m \le n)$ із номерів тих стовпців заданої матриці, які упорядковані за спаданням. Написати процедуру для побудови вектора та логічну функцію для перевірки умови упорядкованості послідовності і використати її для перевірки стовпців. Надрукувати вектор по десять елементів у рядку.

15. Задано дійсну матрицю $X(n,n)$, $n \le 20$. Розробити програму побудови вектора $Y(i)$, $i = 1,2,...,n$, за правилом: $Y(i)$ дорівнює добутку квадратів тих елементів i-го рядка матриці, модулі яких належать проміжку $[a,b]$, якщо таких елементів немає, то $Y(i) = 0$. Використати процедуру побудови вектора і функцію для обчислення добутку квадратів елементів. Надрукувати вектор по п’ять елементів у рядку.

16. Задана матриця $X(n,n)$, $n \le 10$. Розробити програму, яка упорядковує за зростанням ті рядки, в яких міститься максимальний елемент матриці (максимальних елементів може бути декілька). Написати процедуру упорядкування вектора і використати її для упорядкування

рядків та функцію для пошуку максимального елемента матриці. Надрукувати перетворену матрицю по рядках.

17. Задано два тексти, слова в яких розділені пробілами і розділовими знаками. Розробити програму побудови нового тексту, в який входять всі слова першого тексту, що не входять у другий текст, і всі слова другого тексту, що не входять в перший і розділені пробілами. Використати процедуру для побудови нового тексту і функцію перевірки входження слова в текст. Надрукувати побудований текст.

18. Задано масиви чисел $A(n)$, $n \le 300$ і $B(m)$, $m \le 400$. Розробити програму обчислення суми модулів елементів перерізу масивів $A \cap B$. ($A \cap B$ – множина елементів A, які містяться в B і взятих по одному разу). Використати процедуру для побудови перерізу і функцію для обчислення суми модулів його елементів. Надрукувати елементи перерізу та їх суму модулів.

19. Задана матриця $X(n,n)$, $n \le 20$. Назвемо слідом матриці суму елементів головної діагоналі. Розробити програму обчислення слідів матриць X, X^2. Написати і використати процедуру множення матриць та функцію обчислення сліду матриці. Надрукувати матриці X, X^2 по рядках та їх сліди.

20. Задана матриця $A(n,n)$, $n \le 15$. Характеристикою рядка матриці назвемо суму модулів його від'ємних елементів. Розробити програму перетворення цієї матриці перестановкою рядків так, щоб вони розміщувалися у порядку неспадання їх характеристик. Написати процедуру перетворення матриці і функцію для обчислення суми модулів від'ємних елементів вектора і використати її для обчислення характеристик рядків. Надрукувати перетворену матрицю по рядках.

21. Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка вилучає із заданого тексту слова, в яких повторюються букви. Використати процедуру перетворення тексту і функцію перевірки повторення букв у слові. Надрукувати текст, що залишився після вилучення слів.

22. Задана дійсна матриця $A(n,m)$, $n \le 15$, $m \le 10$ та дійсні числа $x_1, x_2, ..., x_r$, $r \le 5$. Розробити програму обчислення значень полінома $P_n(x) = b_1 x_j^{n-1} + b_2 x_j^{n-2} + \cdots + b_n$, у точках x_j, $j = 1,2,...,r$; де b_i – перший по порядку додатний елемент i-го рядка матриці. Якщо в i-му рядку немає

додатних елементів, то $b_i = 0.5$. Використати процедуру обчислення коефіцієнтів та функцію обчислення значень полінома. Надрукувати обчислені коефіцієнти і таблицю значень полінома, у кожному рядку якої розмістити значення $x_j, P_n(x)$.

23. Задана цілочисельна матриця $A(n,n)$, $n \le 20$. Розробити програму побудови вектора $B(i)$, $i = 1,2,...,n$, за правилом: $B(i)$ дорівнює кількості різних елементів i-го рядка матриці A. Написати процедуру побудови вектора та функцію для підрахунку кількості різних елементів вектора і використати її для підрахунку кількості різних елементів у рядку матриці. Надрукувати вектор по десять елементів у рядку.

24. Задана цілочисельна матриця $A(n,n)$, $n \le 15$. Розробити програму побудови вектора $B(k)$, $(k \le n)$ із номерів рядків матриці, елементи яких є симетричними послідовностями виду 1,2,3,3,2,1 або 1,2,3,5,3,2,1. Написати і використати процедуру побудови вектора і функцію для розпізнавання симетричних послідовностей. Надрукувати вектор по сім елементів у рядку.

25. Задано дійсну матрицю $A(n,n)$, $n \le 15$. Розробити програму побудови вектора $X(i)$, $i = 1,2,...,n$, за правилом: $X(i)$ дорівнює середньому арифметичному значенню елементів i-го рядка матриці. Використати процедуру побудови вектора і функцію для обчислення середнього арифметичного. Надрукувати вектор по п’ять елементів у рядку.

26. Задано дійсні матриці $A(n,n), B(n,n), (n \le 15)$. Назвемо слідом матриці суму діагональних елементів. Розробити програму обчислення слідів матриць AB і BA. Використати процедуру обчислення добутку матриць і функцію обчислення сліду. Надрукувати матриці по рядках і значення їх слідів.

27. Задано масиви чисел $A(n), (n \le 400)$ і $B(m), (m \le 200)$. Розробити програму обчислення добутку елементів симетричної різниці масивів $A \setminus B \cup B \setminus A$. ($A \setminus B \cup B \setminus A$ – множина елементів A, що не входять у B, і множина елементів B, що не входять в A, взятих по одному разу)**.** Використати процедуру для побудови симетричної різниці і функцію для обчислення добутку. Надрукувати елементи симетричної різниці та їх добуток.

28. Задана матриця $A(n,n)$, $n \le 20$. Назвемо характеристикою стовпчика

суму модулів його від'ємних непарних елементів. Розробити програму перетворення цієї матриці перестановкою стовпчиків так, щоб вони розміщувалися у порядку незростання їх характеристик. Використати процедуру перетворення матриці і функцію для обчислення характеристик. Надрукувати перетворену матрицю по рядках.

29. Задана дійсна матриця $A(n,n)$, $n \le 10$. Назвемо сусідами елемента матриці $A[i,j]$ елементи $A[k,l]$ з індексами $i-1 \le k \le i+1$, $j-1 \le l \le j+1$, $(i,j) \ne (k,l)$. Розробити програму, яка будує матрицю $B[i,j]$ за правилом: $B[i,j]$ є середнім арифметичним значенням усіх сусідів елемента $A[i,j]$, $i,j = 1,2,...,n$. Написати і використати процедуру для побудови матриці B і функцію для обчислення середнього арифметичного значення сусідів. Надрукувати побудовану матрицю по рядках.

30. Задана дійсна матриця $A(n,n)$, $n \le 15$. Розробити програму, яка міняє місцями діагональний елемент з мінімальним елементом у цьому рядку. Написати процедуру перетворення матриці та функцію пошуку номера мінімального елемента у векторі і використати її для пошуку номерів мінімальних елементів у рядках матриці. Надрукувати перетворену матрицю по рядках.

Теоретичні відомості

Для написання добре структурованих програм використовуються підпрограми. Підпрограма це іменована, спеціальним чином оформлена, логічно завершена група операторів, призначених для виконання певних дій. Підпрограму можна викликати за іменем з будь-якої точки програми будь-яку кількість разів. Концепція підпрограм у Object Pascal реалізована за допомогою процедур і функцій.

Структура процедур і функцій. Процедури і функції мають таку структуру:

Procedure <ім'я процедури > [(<формальні параметри >)]; [<директива >];

Label
Const
Type
Var
} Опис локальних міток, констант, типів і змінних

Procedure
Function } Опис внутрішніх процедур і функцій

Begin
. . .
End; } Оператори

Function <ім’я функції > [(<формальні параметри >)]; [<директива >];

Label
Const
Type
Var } Опис локальних міток, констант, типів і змінних

Procedure
Function } Опис внутрішніх процедур і функцій

Begin
. . .
[< ім'мя функції >:=< значення >;]
[Result :=< значення >;]
End; } Оператори

Процедури і функції складаються із заголовка, локальних даних, внутрішніх процедур та функції і операторів. У заголовку процедур і функцій описуються формальні параметри.

Відмінності в опису процедури і функції стосуються тільки заголовка і розділу операторів. У заголовку функції вказується тип результату, а в розділі операторів можуть бути присутніми оператор присвоєння значення імені функції або оператор присвоєння значення неявному параметру Result.

Звернення до процедури здійснюється оператором процедури. Це ім’я процедури за яким у круглих дужках слідують фактичні параметри. Звернення до процедури не може використовуватися у виразах. До функції також можна звертатися за допомогою оператора, але на відміну від процедури звернення до функції може ще й використовуватися у виразах. При зверненні до процедур і функцій формальним параметрам передаються значення фактичних параметрів. Механізм передачі параметрів забезпечує настроювання алгоритму на фактичні дані.

За заголовком підпрограми може йти одна із стандартних директив:

- ASSEMBLER – тіло підпрограми із команд вбудованого асемблера.
- EXTERNAL – зовнішня підпрограма.
- FAR – код підпрограми, розрахований на дальню модель виклику. За замовчуванням використовується ближня модель – NEAR.
- FORWARD – опис підпрограми з випередженням. Описується заголовок, а підпрограма буде йти далі за текстом програми.
- INLINE – тіло підпрограми із вбудованих машинних команд.
- INTERRUPT – процедура обробки переривань.

Параметри процедур і функцій. Параметри служать для передачі початкових даних у процедуру або функцію і повернення результатів у викликаючу програму. У залежності від призначення параметри описуються різними способами.

Параметри значення. Це список параметрів, після якого через двокрапку задається їх тип.

```
Procedure Proc (v,w: integer; x,y,z: double);
```

Передача цих параметрів здійснюється за значенням. При звернені до підпрограми

```
Proc (5, t, 2+sin (a), sqr (b), -17.2);
```

для формальних параметрів виділяється тимчасова пам'ять, обчислюються значення фактичних параметрів і поміщаються у цю пам'ять, так здійснюється настроювання алгоритму на конкретні дані. Зміна значення формального параметра не приводить до зміни значення фактичного параметра. Параметри значення використовуються тільки для передачі значень у підпрограму, тому фактичними параметрами можуть бути вирази.

Параметри змінні. Ці параметри описуються за допомогою ключового слова Var, за яким йде список параметрів і через двокрапку їх тип. Передача цих параметрів здійснюється за посиланням, тобто у підпрограму передається адреса фактичного параметра. Зміна значення формального параметра приводить до зміни значення відповідного фактичного параметра. Параметри змінні використовуються як для передачі значень у підпрограму, так і повернення результатів у

викликаючу програму, тому фактичними параметрами можуть бути тільки змінні.

Параметри константи. Ці параметри описуються за допомогою ключового слова Const, за яким йде список параметрів і через двокрапку їх тип. Передача цих параметрів здійснюється за посиланням. Параметри константи використовуються тільки для передачі значень у підпрограму. Зміна значення формальних параметрів заборонена. Фактичними параметрами можуть бути тільки змінні.

Нетипізовані параметри. Це параметри, які описуються за допомогою ключових слів Var або Const, за яким йде список параметрів, а тип даних не вказується. У підпрограмі тип таких параметрів невідомий, тому програміст повинен сам турбуватися про правильну інтерпретацію даних, що передаються. А так це звичайні параметри, що передаються за посиланням.

Параметри – відкриті масиви. Відкритий масив – це формальний параметр підпрограми, що описує базовий тип елементів масиву, але не визначає його розмірності та межі.

```
Procedure Proc (Mas: array of  integer);
```

У підпрограмі такий параметр трактується як одномірний масив з нульовою нижньою межею. Верхню межу відкритого масиву можна визначити за допомогою стандартної функції High. При зверненні до підпрограми в ролі фактичного параметра можна задавати як масив, так і конструктор масиву. Конструктор масиву – це список елементів масиву у квадратних дужках.

```
Proc ([5, 6, 9, -7, 14]);
```

Параметри за замовчуванням. Параметри за замовчуванням мають вигляд <ім’я>:<тип>=<значення> і розміщаються в кінці списку формальних параметрів.

```
Procedure Proc (mas: array of  integer; k: integer=1; s: string=’’; a:
                                real=0);
```

При зверненні до підпрограми параметри за замовчуванням можуть не задаватися. Встановлене за замовчуванням значення параметра можна змінити. Якщо змінюється значення деякого параметра за замовчуванням, то при зверненні до підпрограми потрібно також задавати всі параметри, що йому передують Proc (a, 1, ’Рядок’); тобто при зміні параметра s також задається значення параметра k.

Типом будь-якого параметра в списку формальних параметрів може бути тільки стандартний або раніше описаний тип. Тому при

передачі у підпрограму структурованих типів даних – масивів, множин, записів, коротких символьних рядків потрібно описувати їх тип. Наприклад:

```
TYPE Ta=array[1..10] of integer;
    Ts=string[15];
    Tf= string[30];
    Procedure Ks (var a: Ta);
    Function St ( s:Ts): Tf;
```

Приклад. Задано дійсні x і матриця $A(n,n)$, $n \le 100$. Розробити програму для обчислення значення $S = B_{n-1}(x) + C_{n-1}(x)$. Де $B_{n-1}(x) = b_1 + b_2 x^1 + b_3 x^2 + \ldots + b_n x^{n-1}$, b_i – мінімальний елемент i–го рядка матриці A; $C_{n-1}(x) = c_1 + c_2 x^1 + c_3 x^2 + \cdots + c_n x^{n-1}$, c_i – максимальний елемент i–го рядка матриці A, $i = 1,2,\ldots,n$. Для обчислення коефіцієнтів поліномів $B_{n-1}(x)$ і $C_{n-1}(x)$ написати і використати процедуру, а для обчислення значення поліномів – функцію.

Для розв'язку задачі командою File!New Application створимо новий проект. На екрані з'явиться чиста форма з заголовком Form1.

На формі розмістимо два компонента **Edit** для введення початкових даних (розмірність матриці, число *x*), компонент **StringGrid** для введення елементів матриці і третій **Edit** для виведення результатів. Присвоїмо цим компонентам програмні імена Edit1, Edit2, Edit3 встановлені за замовчуванням (властивість Name) і очистимо їм значення властивості Text. Пояснення до цих компонентів зробимо за допомогою компонента Label (властивість Caption).

Крім цього, розмістимо на формі дві керуючих кнопки (компонент Button) з написами ***Виконати*** та ***Закрити*** (властивість Caption) і програмними іменами Button1, Button2 (властивість Name за замовчуванням) (Рис. 7.2).

Зауваження. Для забезпечення можливості вводу даних користувачем в таблицю StringGrid необхідно встановити властивість goEditing групи Options в значення true.

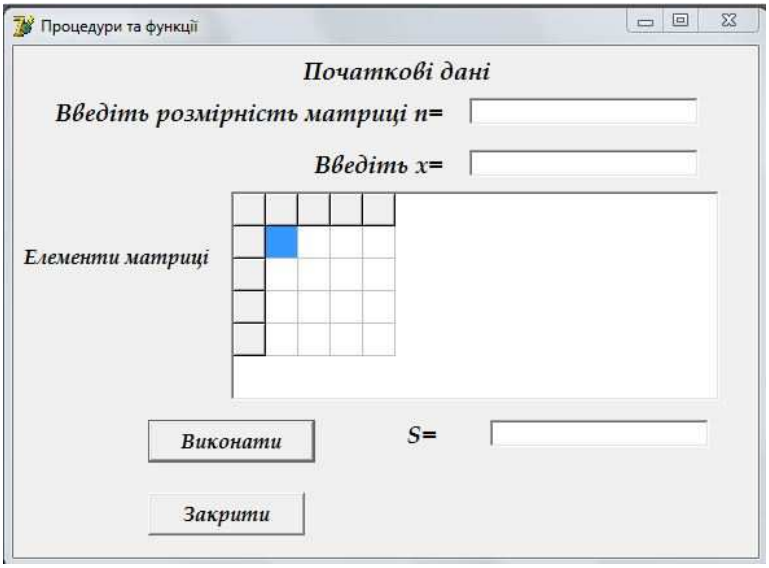

Рис. 7.2. Форма Процедури та функції.

```
{Обробник події Change компонента Edit1}

procedure TProcForm.Edit1Change(Sender: TObject);
 var i,n:integer;
begin
     //розмітка компонента StringGrid
      n:=StrToint(Edit1.Text);
      StringGrid1.RowCount:=n+1;
      StringGrid1.ColCount:=n+1;
      for i:=1 to n do
         begin
            StringGrid1.Cells[i,0]:=IntTostr(i);
            StringGrid1.Cells[0,i]:=IntTostr(i);
         end;
end;

{ Обробник командної кнопки Виконати }

procedure TProcForm.Button1Click(Sender: TObject);
{Ідентифікатори типу для опису формальних}
{ параметрів процедур і функції}
TYPE   Ta=array[1..15,1..15] of double;
       Tbc=array[1..15] of double;
VAR a: Ta;  b, c: Tbc;
    x, s: double;
    n, i, j: integer;
```

```
{Процедури обчислення коефіцієнтів поліномів}
Procedure Kfmin (n: integer; m: Ta; var k: Tbc);
    var    i, j: integer;
begin
    for i:=1 to n do
    begin  k[i]:=m[i,1];
       for j:=2 to n do
    if k[i] > m[i, j]  then  k[i]:=m[i,j];
     end;
end;
Procedure Kfmax (n: integer; m: Ta; var k: Tbc);
    var    i, j: integer;
begin
    for i:=1 to n do
    begin k[i]:=m[i,1];
       for j:=2 to n do
            if k[i] < m[i, j]  then  k[i]:=m[i,j];
    end;
end;

{Функція обчислення значення полінома}
Function  Pol  (  n:integer;  x:  double;  kf:  Tbc):
double;
    var    i: integer; p: double;
begin
         p:=kf[1];
             for i:=2 to n do
             begin
              p:=p+kf[i]*x;
              x:=x*x;
             end;
         Pol :=p;
end;

BEGIN  //TProcForm.Button1Click
try
    {Ввід розмірності матриці}
    n:=StrToint(Edit1.Text);
    {Ввід x}
```

```
    x:=StrToFloat(Edit2.Text);
except
      showmessage(Введене значення не є числом!!!');
       exit;
end;
    {Ввід елементів матриці}
    try
       for i:=1 to n do
        for j:=1 to n do
         a[i,j]:=StrToInt(StringGrid1.cells[j,i]);
    except
      showmessage(Введений елемент не є числом!');
      exit;
    end;

{Обчислення коефіцієнтів поліномів}
    Kfmin (n, a, b);
    Kfmax (n, a, c);
{Обчислення значення s}
    s:= Pol ( n, x, b)+ Pol ( n, x, c);
{Виведення результату}
    Edit3.Text:=FloatTostr(s);
end; //TProcForm.Button1Click

procedure TProcForm.Button2Click(Sender: TObject);
begin
   Close;
end;

END.
```